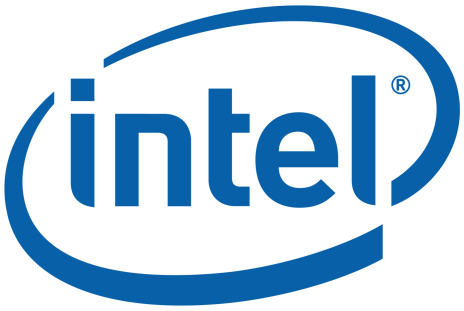
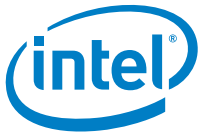


Intel®Atom™ Processor C3000 Product Family

Integrated 10 GbE LAN Controller
Programmer's Reference Manual (PRM)

November 2020

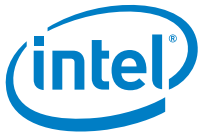
Document # 338653-003 **Revision 1.8**

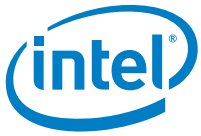


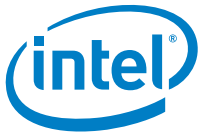
Revision History

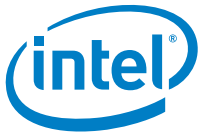
Rev	Date	Comments
1.8	November 2020	• Updated Table 8-4 (Device IDs per SKUs/Connection type). • Updated Section 7.2.2.16.7 (Management Control Register - MANC (0x00005820) RX_Filter).
1.7	August 2019	• Updated Table of Contents (TOC).
1.6	January 2019	• Added NVM Recovery Mode information.
1.5	February 2017	• Initial Release (Intel Public).
1.22	June 2016	• Updated section 1.3.
1.21	January 2016	• Updated section 2.1.1.4.3 (revised note). • Updated Table 2.1 Notes (step 11).
1.2	July 2015	• Updated Table 1-1.
1.1	May 2015	• Initial release (Intel Confidential)

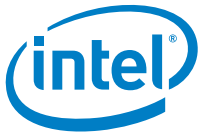
Contents

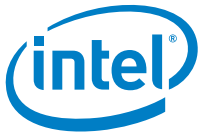


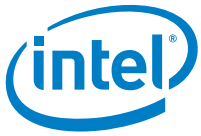












NOTE: ***This page intentionally left blank.***

1.0 Introduction

1.1 Scope

This document describes the external architecture for Intel® Ethernet Connection X553, a dual-port 10 GbE LAN controller integrated into the Intel® Atom™ Processor C3000 Product Family.

It 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 integrated 10 GbE LAN controller.

1.2 Product Overview

The integrated 10 GbE LAN controller contains four independent 10 GbE Media Access Control (MACs) that support an XGMII-like interface link to the following integrated Physical Layer (PHY) device interfaces (see Figure 1.1). Two integrated I/O interface blocks are associated with two 10 GbE ports.

Note: The integrated 10 GbE LAN controller only has one SMBus/NC-SI bus associated with all GbE ports. These sideband buses can be used at the same time.

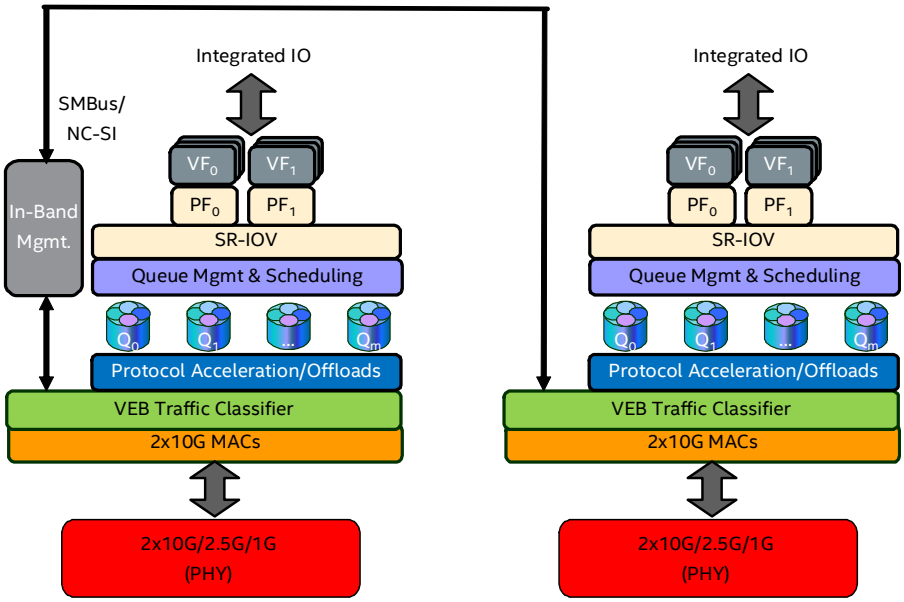
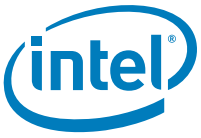


Figure 1.1. Integrated 10 GbE LAN Controller Block Diagram



1.3 Supported Modes of Operation

The integrated 10 GbE LAN controller LAN controllers support speeds of 10 GbE, 2.5 GbE[1] and 1 GbE.

There are two integrated 10 GbE LAN controllers (LAN0 and LAN1) with two ports each, providing a total of four ports.

For each port, there are options to use internal PHYs supporting:

- 10GBASE-KR
- SFI
- 1000BASE-KX
- 2500BASE-X

There are also options to connect to external PHYs via:

- 10GBASE-KR
- 1000-BASE-X
- SGMII is supported for full duplex 10 Mb/s, 100 Mb/s and 1 GbE

Each integrated I/O interface block provides the following physical interfaces and electrical modes:

- KR PHY supports:
  - 10GBASE-KR for GbE backplane applications (IEEE802.3 clause 72)
  - 10GBASE-KR FEC (IEEE 802.3 Clause 74)
  - 1000BASE-KX for GbE backplane applications (IEEE802.3 clause 70)
  - Auto-negotiation for backplane Ethernet (IEEE 802.3 Clause 73)
  - SFI compatible interface to SFP+
  - 10GBASE-KR to Inphi* CS4227/CS4223 re-driver PHY to SFP+[2]
  - 10GBASE-T PHY (Intel® X557-AT/AT2/AT4 10 GbE PHY)

No support for the following in any configuration:

- 1000BASE-T SFP modules
- Half duplex operation (10 Mb/s, 100 Mb/s, 1 GbE, 2.5 GbE or 10 GbE)

1. Auto-negotiate is not supported in 2.5 GbE. Auto-negotiate is supported in all other modes.
2. An Inphi* CS4227 re-driver PHY between the integrated 10 GbE LAN controller and the SFP+ module would be used to create longer channels allowing the SFP+ cage to be further away from the integrated 10 GbE LAN controller. See Table 1-1, “Supported System Configurations” on page 21 for supported options.

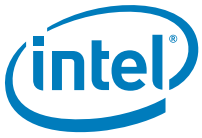


Table 1-1 lists all the supported system configurations, operating modes, and link partners for the integrated 10 GbE LAN controller. Selecting the desired PHY and system configuration is determined at power on through the use of the shared SPI Flash configuration.

Table 1-1. Supported System Configurations

Connection	Speed	Electrical Interface	Third Party PHYs/Switches
Backplane	1 GbE	1000BASE-KX	Intel® FM4000/FM5000/FM6000 Ethernet switches Broadcom* BCM5684x-series switches
Backplane	10 GbE	10GBASE-KR	Intel FM5000/FM6000 Ethernet switches Broadcom* BCM5684x-series switches
Backplane	2.5 GbE	2500BASE-X[1]	Intel FM4000/FM5000/FM6000 Ethernet switches Broadcom* BCM5684x-series switches
1000BASE-T	1 GbE	KX[2]	Marvell 88E1543
10BASE-T 100BASE-T 1000BASE-T	10 Mb/s 100 Mb/s 1 GbE	SGMII (no half duplex support)	Marvell* 88E1514/ Marvell 88E1543 Marvell 88E1512 Marvell 88E1112
10GBASE-T	10 GbE	SFI	Intel® X557-AT/AT2/AT4 10 GbE PHY
SFP +	10 GbE	SFI	Native SFI support
SFP+ SR/LR/ DAC	10 GbE	SFI	Inphi CS4227 (2 port) Inphi CS4223 (4 port)

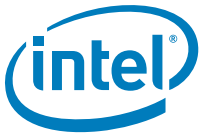
1. 2500BASE-X is not an IEEE standard.
2. No support for half duplex.

Note: The SFI interface is a limited voltage swing interface used for chip-to-chip communication. Refer to the *Intel® Atom™ Processor C3000 Product Family Platform Design Guide (PDG)* for more details.

1.4 Features and Signal Descriptions

Refer to the *Intel® Atom™ Processor C3000 Product Family External Design Specification (EDS)*, Volume 1, for more details.

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



2.0 Interconnects

2.1 Host Primary Interface

The integrated 10 GbE LAN controller communicates with the host CPUs and host memories over an integrated I/O interface. This interface is an Intel standard bus defined for SoCs and replaces the PCIe interface and functionality of the discrete LAN controller.

2.1.1 Host Interface Architecture, Transaction and Link Layer Properties

- All PCI functions are native PCIe functions
- Credit-based flow control
- Packet sizes/formats:
  - Maximum upstream (write) payload size of 256 bytes
    - Communicated as 128 bytes in PCIe registers
  - Maximum downstream (read) payload size of 256 bytes
    - Communicated as 128 bytes in PCIe registers
- Reset/initialization:
  - Credit negotiation performed by hardware
- Transaction layer mechanisms
  - 64-bit memory address spaces
  - Removal of I/O BAR (optional)
  - Relaxed ordering
  - Flow control update time out mechanism
  - ID-based ordering (IDO)
  - Function-Level Reset (FLR)
  - TLP Processing Hints (TPH)
  - Reliability
    - Advanced Error Reporting (AER)
- Power management
  - Wake capability
  - Latency Tolerance Reporting (LTR)
- DFT and DFM support for high-volume manufacturing
- The integrated 10 GbE LAN controller supports the following extended capabilities:
  - Device Serial Number
  - Alternative RID Interpretation (ARI)

  - — Single Root I/O Virtualization (SR-IOV)
  - — TPH Requester
  - — Access Control Services (ACS)
- Software configuration mechanism:
  - — Uses PCI configuration and bus enumeration model
  - — PCIe-specific configuration registers mapped via PCI extended capability mechanism

2.1.2 PCIe Transaction Layer

2.1.2.1 Transaction Types Accepted by the Integrated 10 GbE LAN Controller

Table 2.1 lists the transactions accepted by the device and their attributes.

Table 2.1. Transaction Types Accepted by the Transaction Layer

Transaction Type	Tx Layer Reaction	Hardware Should Keep Data From Original Packet
Configuration Read Request	CPLH + CPLD	Requester ID, TAG, attribute
Configuration Write Request	CPLH	Requester ID, TAG, attribute
Memory Read Request	CPLH + CPLD	Requester ID, TAG, attribute
Memory Write Request	-	-
I/O Read Request	CPLH + CPLD	Requester ID, TAG, attribute
I/O Write Request	CPLH	Requester ID, TAG, attribute
Read Completions	-	-
Message	-	-

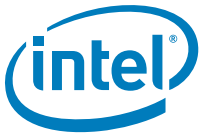
2.1.2.1.1 Size of Target Accesses

2.1.2.1.1.1 Memory accesses

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

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

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



- Read accesses
  - — Reads to Flash are 32-bit wide
  - — Partial reads with at least one byte disabled are handled internally as a full read. That is, any side effect of the full read (such as clear by read) is also applicable to partial reads. The completion on PCIe follows the specification rules regarding the number of bytes reported in the completion
  - — Larger CSR read requests are handled as completer abort - the completion includes a CA status and an error is generated per PCIe rules

2.1.2.1.1.2 I/O accesses

Rules for accesses to the I/O BAR:

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

2.1.2.1.2 Support for Dynamic Changes

The integrated 10 GbE LAN controller captures the bus number and device number per each configuration write request. However, a dynamic change of the bus number or device number is not supported. Rather, the PCIe link should be quiescent prior to such a change, including reception of all completion for previous requests.

2.1.2.2 Transaction Types Initiated by the Integrated 10 GbE LAN Controller

Table 2.2. Transaction Types Initiated by the Transaction Layer

Transaction Type	Payload Size	Tx Layer Reaction
Configuration Read Request Completion	Dword	CPLH + CPLD
Configuration Write Request Completion	-	CPLH
I/O Read Request Completion	Dword	CPLH + CPLD
I/O Write Request Completion	-	CPLH
Read Request Completion	Dword/Qword	CPLH + CPLD
Memory Read Request	-	NPH
Memory Write Request	<= MAX_PAYLOAD_SIZE	PH + PD
Message	64 bytes	PH

Note:

Configuration values:

- Max Payload Size - The value of the Max_Payload_Size Supported field in the Device Capabilities register is loaded from the NVM.
  - — Hardware default is 256B.
  - — System software then programs the actual value into the Max_Payload_Size field of the Device Control register.
    - Non-ARI mode: If not all functions are programmed with the same value, the max payload size used for all functions is the minimum value programmed among all functions.
    - ARI mode: Max_Payload_Size is determined solely by the setting in Function 0
- Max_Read_Request_Size - The integrated 10 GbE LAN controller supports read requests of up to 256 bytes.

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

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

2.1.2.2.1 Data Alignment

Requests must never specify an address/length combination that causes a memory space access to cross a 4 KB boundary. The integrated 10 GbE LAN controller therefore breaks requests into 4 KB-aligned requests (if needed). This does not place any requirement on software. However, if software allocates a buffer across a 4 KB boundary, hardware issues multiple requests for the buffer. Software should consider aligning buffers to a 4 KB boundary in cases where it improves performance. The maximum size of a read request is defined as the minimum (2 KB bytes, Max_Read_Request_Size).

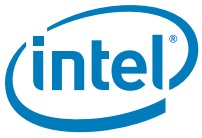
The general rules for packet alignment are as follows. Note that these apply to all Integrated 10 GbE LAN Controller requests (read/write):

- The length of a single request does not exceed the PCIe limit of MAX_PAYLOAD_SIZE (256 bytes) for write and MAX_READ_REQ (256 bytes) for read.
- The length of a single request does not exceed the integrated 10 GbE LAN controller internal limitations.
- A single request does not span across different memory pages as noted by the 4KB boundary alignment previously mentioned.

If a request can be sent as a single 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. 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.

For requests with data payload, if the payload size is larger than (MAX_PAYLOAD_SIZE - CACHELINE_SIZE), then the request is broken into multiple TLPs starting at the first cache line boundary following the (MAX_PAYLOAD_SIZE - CACHELINE_SIZE) bytes. For example, if MAX_PAYLOAD_SIZE = 256 bytes and CACHELINE_SIZE = 64 bytes, a 1 KB request starting at address 0x...10 is broken into TLPs such that the first TLP contains 240 bytes of payload (since 240 bytes+ 0x10 = 256 bytes is on cache line boundary).

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



2.1.2.3 Messages

2.1.2.3.1 Received Messages

Message packets are special packets that carry a message code. The upstream device transmits special messages to the integrated 10 GbE LAN controller by using this mechanism. The transaction layer decodes the message code and responds to the message accordingly.

Table 2.3. Supported Message in the Integrated 10 GbE LAN Controller (as a Receiver)

Message Code [7:0]	Routing r2r1r0	Message	Odem Later Response
0x40, 0x41, 0x43, 0x44, 0x45, 0x47, 0x48	100b	Ignored messages (used to be hot-plug messages)	Silently drop.
0x50	100b	Slot power limit support (has one Dword data)	Silently drop.
0x7E	000b, 010b, 011b, 100b	Vendor defined type 0	Drop and handle as an Unsupported request
0x7F	100b	Vendor defined type 1	Silently drop.
0x7F	010b, 011b, 000b	Vendor defined type 1	Silently drop.
0x00	011b	Unlock	Silently drop.

2.1.2.3.2 Transmitted Messages

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

2.1.2.4 Transaction Attributes

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

The integrated 10 GbE LAN controller only supports TC = 0b and VC = 0b (default).

2.1.2.5 Ordering Rules

The integrated 10 GbE LAN controller meets the PCIe ordering rules by following the PCI simple device model:

1. Deadlock Avoidance – The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller are allowed to proceed.
2. Descriptor/Data Ordering — The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller, 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 integrated 10 GbE LAN controller 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.

2.1.2.5.1 Relaxed Ordering

The integrated 10 GbE LAN controller takes advantage of the relaxed ordering rules in PCIe. By setting the relaxed ordering bit in the packet header, the integrated 10 GbE LAN controller enables the system to optimize performance in the following cases:

1. Relaxed ordering for descriptor and data reads — When the integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller cannot relax ordering for receive descriptor writes or an MSI write.

Relaxed ordering is enabled in the integrated 10 GbE LAN controller by clearing the *CTRL_EXT.RO_DIS* bit. Relaxed ordering is further controlled through the *Enable Relaxed Ordering* bit in the PCIe Device Control register.

2.1.2.5.2 ID-based Ordering (IDO)

IDO was introduced in the PCIe rev. 2.1 specification. When enabled, The integrated 10 GbE LAN controller sets IDO in all applicable TLPs defined in the PCIe specification.

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

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

- The NVM *PCI_CAPSUP.IDO Enable* bit is set (Section 4.4.4.1 and Section 8.2.2.4.5)
- The PCIe *IDO Request Enable* bit (for requests) or the *IDO Completion Enable* bit (for completions) in Device Control 2 register is set

2.1.3 Error Events and Error Reporting

2.1.3.1 General Description

PCIe defines three error reporting paradigms: the baseline capability, the Advanced Error Reporting (AER) capability, and a proprietary mechanism. The baseline error reporting capabilities are required of all PCIe devices and define the minimum error reporting requirements. The AER capability is defined for more robust error reporting and is implemented with a specific PCIe capability structure. Both mechanisms are supported by the integrated 10 GbE LAN controller. The proprietary error reporting mechanism used for error better handled by the software using internal CSRs is described in Section 2.1.3.7.

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

In a multi-function device, PCIe errors that are not related to any specific function within the device are logged in the corresponding status and logging registers of all functions in that device. Figure 2.1 shows, in detail, the flow of error reporting in the integrated 10 GbE LAN controller.

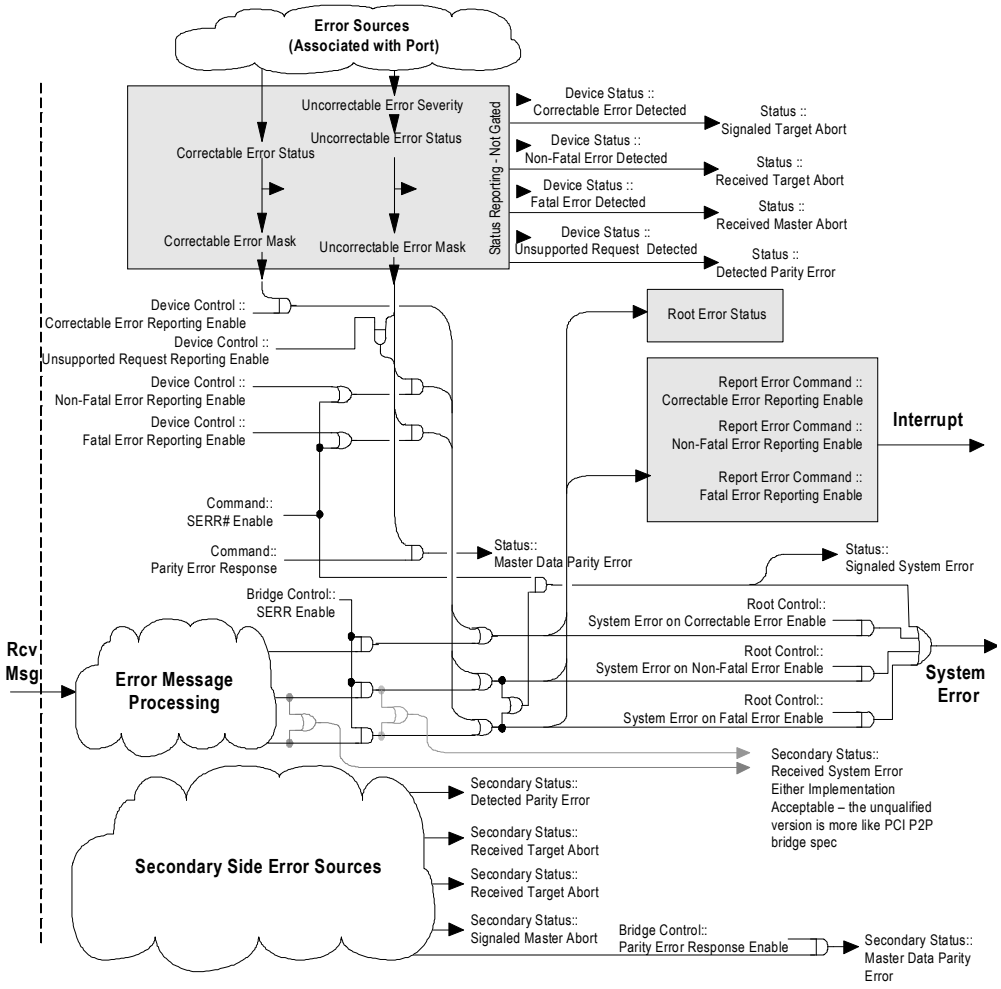


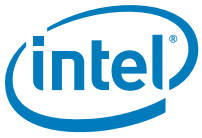
Figure 2.1. Error Reporting Mechanism

2.1.3.2 Error Events

Table 2.4 lists the error events identified by the integrated 10 GbE LAN controller 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 2.4. Response and Reporting of PCIe Error Events

Error Name	Error Events	Default Severity	Action
TLP Errors			
Poisoned TLP Received	• TLP With Error Forwarding (EP = 1b) • Data Parity	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)	• Receipt of TLP with unsupported Request Type • Receipt of an Unsupported Vendor Defined Type 0 Message • Invalid Message Code • Wrong Function Number • Received TLP Outside BAR Address Range • Receipt of a Request TLP during D3hot, other than Configuration and Message requests	Uncorrectable ERR_NONFATAL Log header	Send Completion With UR
Completion Timeout	• Completion Timeout Timer Expired	Uncorrectable ERR_NONFATAL	Error is non-fatal (default case) • Send error message if advisory Error is defined as fatal • Send uncorrectable error message
Completer Abort	• Received Target Access With illegal data size per Section 2.1.2.1.1	Uncorrectable. ERR_NONFATAL Log header	Send completion with CA
Unexpected Completion	• Received Completion Without a Request For It (Tag, ID, etc.)	Uncorrectable ERR_NONFATAL Log Header	Discard TLP
Malformed TLP (MP)	• Data Payload Exceed Max_Payload_Size Received • TD field value does not correspond with the observed size • Usage of Unsupported VC • Target request crosses a 4KB boundary	Uncorrectable ERR_FATAL Log Header	***Note:*** Drop the Packet, Free FC Credits
Completion with Unsuccessful Completion Status		No Action (already done by originator of completion)	Free FC Credits
Command Parity	Parity error is detected on the command	Uncorrectable ERR FATAL	



2.1.3.3 Completion Timeout Mechanism

The integrated 10 GbE LAN controller supports completion time out as defined in the PCIe specification.

The integrated 10 GbE LAN controller controls the following aspects of completion time out:

- Disabling or enabling completion timeout
  - The PCIe *Completion Timeout Disable Supported* bit in the Device Capabilities 2 register is hard wired to 1b to indicate that disabling completion timeout is supported
  - The PCIe *Completion Timeout Disable* bit in Device Control 2 register controls whether completion timeout is enabled
- A programmable range of timeout values
  - The integrated 10 GbE LAN controller supports all four ranges as programmed in the *Completion Time out Ranges Supported* field of the Device Capabilities 2 register. The actual completion time out value is written in the *Completion Time out Value* field of Device Control 2 register.

The following sequence takes place when completion timeout is detected:

- The appropriate message is sent on PCIe as listed in Table 2.4
- The affected queue or client takes action based on the nature of the original request.
- An interrupt is issued to the respective PF.

2.1.3.4 Error Forwarding (TLP Poisoning)

If a TLP is received with an error-forwarding trailer, the packet is dropped and is not delivered to its destination, the integrated 10 GbE LAN controller then reacts as listed in Table 2.4.

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

- The appropriate message is sent on PCIe as listed in Table 2.4.
- An interrupt is issued.
- If the TLP is a completion, a completion time out follows at some later time. Processing continues as described in Section 2.1.3.3.

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

2.1.3.5 Completion With Unsuccessful Completion Status

A completion arriving with an unsuccessful completion status (either UR or CA) is dropped and not delivered to its destination. A completion time out follows at some later time. Processing continues as described in Section 2.1.3.3.

2.1.3.6 Blocking on Upper Address

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

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

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

Processing a blocked transaction:

- Write transaction:
  - The transaction is dropped.
  - Set the Exceeded upper address limit (write requests) event in the PCIe errors register (see Section 2.1.3.7).
  - An interrupt is issued as described in Section 2.1.3.7.
- Read transaction:
  - The transaction is dropped.
  - Set the Exceeded upper address limit (read requests) event in the PCIe errors register (see Section 2.1.3.7).
  - The originating internal client is notified.
  - The affected queue or client takes action based on the nature of the original request. An interrupt is issued to the respective PF.

2.1.3.7 Proprietary Error Reporting

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

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

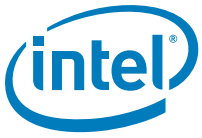
- The PCIe Errors Reported register (PCI_PCIERR - RO) indicates which errors are reported using this mechanism. It is shared by all PFs. It is loaded from the NVM. All the non-reserved errors are enabled.
- The PCIe Interrupt Cause register (PCI_ICAUSE - RW1C) indicates pending errors for errors set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

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

- The integrated 10 GbE LAN controller checks if the respective bit is set in the PCIe Errors Reported register. If cleared, done. Else, continue.
- The respective bit is set in the PCIe Interrupt Cause register.
- If the respective bit is set in the PCIe Interrupt Enable register, an interrupt is issued to the PCI function. The PCI_EXCEPTION cause is used (see the EICR register - Section 8.2.2.4.19).

PCIe Errors Reported to Device Software

Error Event	Index	Description and Comments	Function Association
Exceeded upper address limit (read requests)	00	See Section 2.1.3.6	Sent to PF
Exceeded upper address limit (write requests)	01	See Section 2.1.3.6	Sent to PF
Reserved	02	Reserved entries	N/A



PCIe Errors Reported to Device Software

Error Event	Index	Description and Comments	Function Association
Poisoned TLP received	03	See Section 2.1.3.4	Sent to PF
Reserved	04-05	Reserved entries	N/A
Unsupported Request - Request Type	07	Request causes an Unsupported Request due to receipt of TLP with unsupported Request Type	Sent to PF
Unsupported Request - Vendor Message	08	Request causes an Unsupported Request due to receipt of an Unsupported Vendor Defined Type 0 Message	Sent to PF unless r[2:0] = *Broadcast from Root Complex,* in which case sent to all PFs
Reserved	09	Reserved entries	N/A
Unsupported Request - Function Number	10	Request causes an Unsupported Request due to receipt of a not-supported Function Number	Sent to all PFs
Reserved	11 - 12	Reserved entries	N/A
Completer abort - target size	13	Received Target Access with illegal data size per Section 2.1.2.1.1 (CA)	Sent to PF
Reserved	14 - 31	Reserved entries	N/A

2.2 Management Interfaces

The integrated 10 GbE LAN controller contains three possible interfaces to an external BMC.

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

2.2.1 SMBus

SMBus is an optional interface for pass-through and/or configuration traffic between an external BMC and the integrated 10 GbE LAN controller. The SMBus channel behavior and the commands used to configure or read status from the integrated 10 GbE LAN controller are described in Section 9.5.

The integrated 10 GbE LAN controller also enables reporting and controlling the device using the MCTP protocol over SMBus. The MCTP interface is used by the BMC to control the NIC and for pass-through traffic. All network ports are mapped to a single MCTP endpoint on SMBus. For additional information, refer to Section 9.5.

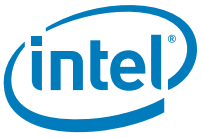
2.2.1.1 Channel Behavior

The SMBus specification defines a maximum frequency of 100 KHz. However, when acting as a slave, the integrated 10 GbE LAN controller can receive transaction with a clock running at up to 1 MHz. When acting as a master, it can toggle the clock at 100 KHz, 400 KHz or 1 MHz. The speed used is set by the *SMBus Connection Speed* field in the SMBus Notification Time out and Flags shared SPI Flash word.

2.3 Sideband Interface (NC-SI)

The NC-SI interface in the integrated 10 GbE LAN controller is a connection to an external MC. It operates as a single interface with an external BMC, where all traffic between the integrated 10 GbE LAN controller and the BMC flows through the interface.

The integrated 10 GbE LAN controller NC-SI interface meets the NC-SI version 1.0.0 specification as a PHY-side device.



2.3.1 Electrical Characteristics

The integrated 10 GbE LAN controller complies with the electrical characteristics defined in the NC-SI specification.

2.3.2 NC-SI Transactions

The NC-SI link supports both pass-through traffic between the BMC and the integrated 10 GbE LAN controller LAN functions, as well as configuration traffic between the BMC and the integrated 10 GbE LAN controller internal units as defined in the NC-SI protocol. Refer to Section 9.6.2 for information.

2.3.3 MCTP (over SMBus)

The integrated 10 GbE LAN controller supports MCTP protocol for management. MCTP runs over SMBus. The integrated 10 GbE LAN controller implements NC-SI over MCTP protocol for command and pass-through traffic. See Section 9.7 for details.

2.4 Non-Volatile Memory (NVM)

2.4.1 General Overview

The integrated 10 GbE LAN controller uses a Flash device to store product configuration information. The Flash is divided into a few general regions:

- Hardware Accessed — Loaded by the the integrated 10 GbE LAN controller 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 3.1.
- Firmware Area — Includes firmware code and structures used by the firmware for management configuration in its different modes.
- Software Accessed — This region is used by software entities such as LAN drivers, option ROM software and tools, PCIe bus drivers, VPD software, etc.

2.4.1.1 NVM Protection

To meet requirements previously described, the contents of several NVM modules must be protected via authentication.

The NVM protection method implemented in the integrated 10 GbE LAN controller relies on an authenticate on update concept. It means that protected modules are not authenticated after initialization, but prior to committing a module update operation only. NVM protection is guaranteed by an inductive authentication chain, that assumes an initial secured NVM image, and requires that any NVM update must be secure as well. This method mandates the following limitations and restricting working assumptions:

1. An initial good image is loaded into the Flash at the manufacturing site, which is assumed to be safe.
    a. It assumes customers (OEM and end-user) know the source of the installed components, the supply chain producing these components is not compromised during manufacturing, and that the NIC/LOM is physically protected from modification after deployment.

b. The possibility exists that unauthorized firmware might be loaded into the NVM via physical modification post manufacturing, as well as through supply chain vulnerabilities. However, firmware updates via programmatic (software) methods are enhanced to require authentication prior to updating NVM settings. Furthermore, host software can independently detect whether the firmware image has an invalid digital signature.

2. In a normal operating mode, NVM write accesses are controlled by the device (firmware) and cannot be performed by the host. Memory mapped NVM access remains available for NVM read accesses only. For simplicity and flexibility reasons, NVM write accesses from the host can be initiated via Software Host Interface commands (Section 9.8.3.3), VPD write interface, or via a BMC command, which are all handled by firmware. All other direct access write modes are blocked by hardware when the NVM is protected.
3. All the supported Flash parts share the same set of opcodes as described in Section 2.4.2. A blank Flash programming mode is provided (besides the normal programming mode previously mentioned in item 2), where the Flash can be programmed directly without firmware involvement via Flash BAR interfaces.

2.4.2 Shadow RAM

The first eight 4 KB sectors of the integrated 10 GbE LAN controller's Flash are allocated to create two 16 KB sections (section 0 and section 1), for storing the device configuration content. At least one of these two sections must be valid at any given time, otherwise the integrated 10 GbE LAN controller is configured based on its hardware defaults. Following a Power On Reset (POR), the integrated 10 GbE LAN controller copies the valid section of the Flash device into an internal shadow RAM. Modifications made to the shadow RAM contents are copied by the integrated 10 GbE LAN controller to the other 16 KB section of the NVM, circularly flipping the valid section between sections 0 and 1 in the NVM. This mechanism provides the following advantages:

1. A seamless backward-compatible interface for software/firmware to access the first 16 KB of the NVM as if an external EEPROM device was connected. This interface is referred as EEPROM-mode access to the Flash.
2. A way to protect the image-update procedure from power down events by establishing a double-image policy. See Section 2.4.8.1 for a description of the double-image policy. It relies on having pointers to NVM modules stored in the NVM section mirrored in the internal shadow RAM.

Figure 2.2 shows the shadow RAM mapping and interface.

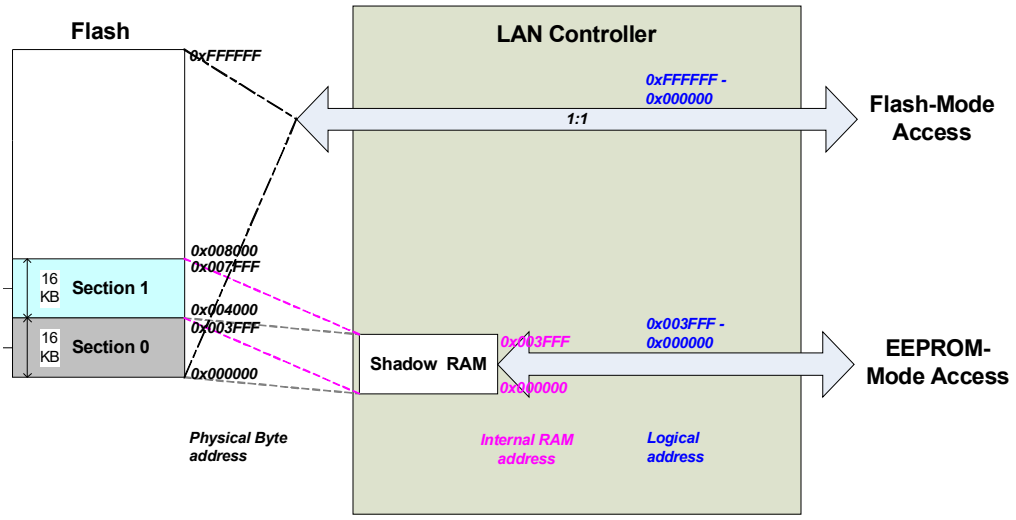


Figure 2.2. NVM Shadow RAM

2.4.2.1 Shadow RAM Update Flow

1. Following a write access by the software to modify the shadow RAM, the modified data should be updated in the Flash as well. The integrated 10 GbE LAN controllercommits the shadow RAM contents to the Flash when software explicitly requests an update using the Shadow RAM Dump Host Interface command (Section 9.8.3.3.8). To reduce Flash update operations, software is expected to issue this command only once its last shadow RAM write access completes. Once the shadow RAM dump command is issued, The integrated 10 GbE LAN controller copies the contents of the shadow RAM to the non-valid NVM configuration section and makes it the valid one.
2. Software should wait for the command completion to make sure the flow succeeded. Software should also be aware that this might takea while since the Flash update sequence is handled by firmware. The sector erase command by itself can last hundreds of milliseconds. The regular timeout of the host interface commands should be sufficient to cover this command as well.

2.4.3 NVM Clients and Interfaces

There are different software clients that can access the NVM: driver, tools, BIOS, VPD, etc. Table 2.5 lists the different NVM access methods.

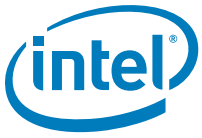
Table 2.5. Clients and Access Types to the NVM

Client	NVM Access Method (Data Width)	NVM Access Target	Logical Byte Address Range	NVM Access Interface	Protection and Enforcement
Host Software	Memory BAR (parallel 32-bit)	Flash	0x000000 - 0xFFFFFF	Memory mapped 32-bit read via BARs.	• Write access is not supported.
BIOS	Expansion ROM BAR (parallel 32-bit)	Flash	Expansion ROM Base Address register	Memory mapped 32-bit read via BARs.	• Write access is not supported.
Host Software	Host interface Shadow RAM Read/Write command	Shadow RAM	0x000000 - 0x003FFF	Access to shadow RAM through the Shadow RAM Read/Write command	Requires a valid firmware image. Write protection is enforced by firmware.
Host Software	Host interface Flash read command	Flash	0x000000 - 0xFFFFFF	Software read from Flash via Flash Read command.	Requires a valid firmware image.
Host Software	Host interface Flash write command/Flash sector erase command	Flash	0x008000 - 0xFFFFFF	Software writes and sector erase.	Requires a valid firmware image. Writes and erases to protected areas are dropped.
Host Software	VPD access (parallel 32-bit)	Shadow RAM	0x000000 - 0x0003FF	VPD Address and Data registers.	Write accesses are enabled to the R/W area of the VPD If the VPD structure is not valid, the entire 1024 bytes area becomes RO to VPD accesses.

2.4.4 Memory Mapped Host Interface

The Flash is accessed by the integrated 10 GbE LAN controller each time the host CPU performs a read operation to a memory location mapped to the Flash address space, or upon boot via accesses to 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 integrated 10 GbE LAN controller 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 (shadow RAM word address 0x05). This pointer is loaded by the integrated 10 GbE LAN controller from the NVM before enabling any access to the expansion ROM memory space.

The integrated 10 GbE LAN controller allows access to the Flash when it decodes a valid request. Attempting to read outside the address space allocated to the PCIe expansion ROM module beyond the range of the configured size returns the value of 0xDEADBEEF. All memory-mapped Flash write attempts are ignored by hardware.

Attempts to memory-mapped write accesses to the Flash when protection is enabled or via expansion ROM BAR are ignored.

2.4.5 Flash Access Contention

Flash accesses initiated through different LAN functions might occur concurrently. The integrated 10 GbE LAN controller does not synchronize between entities accessing the Flash, so a contention caused from one entity reading and another modifying the same location is possible.

To avoid such contention between software LANs or between software and firmware accessesall access to the Flash is done using the Host Interface Admin commands that are managed by the firmware.

However, two software entities cannot use this Host Interface Admin command mechanism: BIOS access through expansion ROM and VPD software.

- Since VPD software accesses only the VPD module, which is located in the configuration section of the NVM, VPD accesses are always performed against the shadow RAM. Firmware must take NVM ownership before dumping the VPD committing to the Flash. The Shadow RAM dump sequence is described in Section 2.4.2.1.
- No contention can occur between the BIOS access through expansion ROM and other software entities (including VPD) as it accesses the NVM while the operating system is down.
- Contentions between BIOS and firmware can however happen if a system reboot occurs while the MC is accessing the NVM.
  - If a system reboot is caused by a user pressing the standby button, it is required to route the wake-up signal from the standby button to the MC and not to the chipset. The MC issues a system reboot signal to the chipset only after the NVM write access completes. Firmware is responsible to respond with a busy error code to MC NC-SI commands while other NVM writes are in progress.
  - If a system reboot is issued by a local user on the host, there is no technical way to prevent NVM access contentions between the BIOS and the MC.

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

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

2.4.5.1 Flash Deadlock Avoidance

The Flash is a shared resource between the following clients:

3. LAN port 0 and LAN port 1 software accesses.
4. Manageability/firmware accesses.
5. Software tools.

All clients can access the Flash in parallel. Firmware implements the actual access to the Flash and is therefor responsible for arbitrating between the different clients and schedules these accesses, avoiding starvation of any client.

Note: An exception to the previous is the BAR READ interface. This however should be used only by BIOS at power on for reading the option ROM. Other software tools should be used by the Host Interface Command for read and write access.

2.4.6 Signature Field

The only way the integrated 10 GbE LAN controller can detect if a Flash is present is by trying to read from it. The integrated 10 GbE LAN controller first reads the Control words at section 0 (word address 0x0) and at section 1 (word address 0x2000). It then checks the signature value at bits 7 and 6 in both words.

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

If the signature bits are valid at both addresses, the integrated 10 GbE LAN controller assumes that section0 is the valid one.

2.4.7 VPD Support

The Flash image might contain an area for VPD. This area is managed by the OEM vendor and does not influence the behavior of hardware. The NVM header contains a pointer to the VPD area. A value of 0xFFFF means VPD is not supported and the *PCI_CAPCTRL.VPD_EN* bit in the PCI NVM section (Section 4.4.4) should be cleared. This prevents the VPD capability from appearing in the configuration space.

The maximum VPD area size is 1024 bytes but can be smaller. The VPD block is composed of a list of resources. A resource can be either large or small. The structures of these resources are listed in the following tables.

Table 2.6. Small Resource Structure

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

Table 2.7. Large Resource Structure

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

The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller assumes the following VPD fields with the limitations listed:

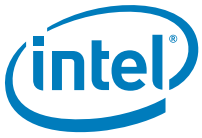


Table 2.8. VPD Structure

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.
0x78	N/A	N/A	End tag.

VPD structure limitations:

- The structure must start with the Tag = 0x82. If the integrated 10 GbE LAN controller does not detect a value of 0x82 in the first byte of the VPD area or if the structure does not follow the description listed in Table 2.8, it assumes the area is not programmed and the entire 1024 bytes area becomes read only to the VPD capability.
- The RO and RW areas 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 about the different tags.
- If a VPD-W tag is found, the area defined by its size is writable via the VPD capability.
- The structure must end with the Tag = 0x78.
- The VPD area can be accessed through the PCIe configuration space VPD capability structure listed in Table 2.8. Write accesses to a read only area or any access to an offset outside the VPD area via this structure are ignored.
- The VPD area must be mapped in the first 16 KB section of the Flash mapped to the shadow RAM.
- VPD software does not check the semaphores before attempting to access the VPDarea via the dedicated VPD registers. Even if the Flash is owned by another entity, VPD software read accesses to the VPD area might complete immediately since it is performed against the shadow RAM. However, VPD software write accesses might not complete immediately since the VPD changes are committed to the Flash device at the integrated 10 GbE LAN controller's initiative, once the other NVMentities release Flash ownership, which can take up to several seconds.

2.4.7.1 VPD Access Flows

2.4.7.1.1 First VPD Area Programming

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

The VPD contents and pointer can be written to a blank Flash without any limitation, similarto any other shared SPI Flash module when in blank Flash programming mode. After protection is enabled, if *VPD Write Enable* bit in NVM control word 1 is cleared, only the RW area of the VPD is writable and only via the VPD interface.

2.4.7.1.2 VPD Area Update Flow

1. The host performs a VPD write - it writes the offset/data into the VPD register set of the PCIe configuration space, and sets the VPD Flag (bit 15 in the VPD Address register - 0xE2).

Note: Firmware must not access the VPD registers before it receives the VPD access interrupt from the VPD mechanism.

2. Firmware checks that the VPD write is allowed - it verifies that the offset address falls within the VPD-W area, and neither within other VPD areas, nor RO areas of the shadow RAM.
3. writing is not allowed firmware firmware writes the changes to the VPD-W area and then re-arms a 10 ms VPD write timer.
4. VPD software might issue additional writes to the VPD-W area. Following the VPD write timer expiration, firmware dumps the shadow RAM contents into the Flash as described by the flow in Section 2.4.2.1.

Note: In case the Flash is busy by a previous sector erase operation, or if NVM ownership is held by software, the flow should not be restarted, for successive VPD write accesses, before the last step is completed.

2.4.8 Extended NVM Flows

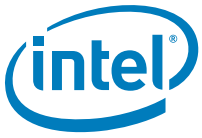
2.4.8.1 Flow for Updating Secured Modules

This section describes the flow used to update the firmware image or option ROM.

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

1. Take ownership over the shared SPI Flash via the semaphore bits. Refer to Section 9.8.4.
   a. If the SW_FW_SYNC.NVM_UPDATE_STARTED bit isn't set, set both the NVM ownership semaphore bit and the NVM_UPDATE_STARTED bit. This serves as an indication to other entities that an NVM update process, which might take several minutes, has started. During this time, other entities cannot perform a write access to the firmware module, but reading this module in between update write bursts is allowed using the Flash memory mapping. Shadow RAM mapped modules are not affected by this limitation.
   b. Otherwise, release NVM semaphore ownership and restart the update process later on.
2. Read the pointer to the free provisioning area (NVM word 0x40). Check that the free provisioning area size read from NVM word 0x41 is greater or equal to the size of the new firmware/option ROM module being written into the NVM.
   a. If not, release NVM semaphore ownership, clear the *SW_FW_SYNC.NVM_UPDATE_STARTED* bit and exit the flow.
3. Initiate sector erase instructions (Section 2.4.7.1.1) to the entire free provisioning area.
   a. In order to guaranty shared SPI Flash semaphore ownership time does not exceed the one second timeout, Intel recommends releasing the NVM ownership semaphore for 10 ms after every four consecutive sector erase operations. This way, other entities can insert shared SPI Flash read accesses in between erases without waiting for the entire update process to complete.
4. Write the new firmware/option ROM to the free provisioning area via Flash-mode access.
   a. Same as (3.a), it is recommended to write at this step no more than four sectors at once in a burst, releasing semaphore ownership for 10 ms in between.
5. Send the Flash module update module ID of the section to update.The encoding of the modules is:

Module	ID	Reference
Firmware code	0x1	
Reserved	0x2	
Reserved	0x3	
Reserved	0x4	
Reserved	0x5	



Module	ID	Reference
Reserved	0x6:0xFD	
Option ROM	0xFE	
Reserved (full shadow RAM)	0xFF	

6. Release the shared SPI Flash semaphore and clear the *SW_FW_SYNC.NVM_UPDATE_STARTED* bit.
   a. Software must avoid taking the Integrated 10 GbE LAN Controller semaphore again until the firmware command has completed. Any attempt to write the shared SPI Flash until then is ignored by Integrated 10 GbE LAN Controller.
7. Firmware swaps between the free provisioning area pointer (word 0x40) and either the firmware code module pointer or option ROM module pointer located at shadow RAM word addresses 0x3A/ 0x05, respectively. Firmware dumps the shadow RAM into the Flash.
8. Software waits for the command to complete.
   a. If the update process failed due to a security check failure or a Flash write fault, an Authentication Error (0x80) or Data Error (0x6), respectively, is returned. Software must then exit the flow, prior to attempting another update.

2.4.8.2 Flow for Updating One of the RW Legacy EEPROM Modules

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

1. Take ownership over the shared SPI Flash via semaphore bits. Refer to Section 9.8.4.
2. Invalidate the pointer to the module being modified by setting it to 0xFFFF using Shadow RAM Write command. This way, if an auto-load of the module is attempted, the associated register defaults are loaded instead. Do not invalidate pointers to firmware modules, only to hardware auto load modules.
3. Modify the contents of the module via Shadow RAM Write command.
4. Restore the pointers modified in step 2. via Shadow RAM Write command.
5. Compute and update the software checksum (word 0x3F) if the contents covered by the software checksum was modified.
6. Release the shared SPI Flash semaphore.
7. Send a Shadow RAM Dump command (Section 9.8.3.3.8) to instruct device firmware to write the shadow RAM contents into the Flash.

Note: Depending on the modified RO items, a system reset is generally required toload the changes into the device.

2.4.9 NVM Authentication Procedure

The NVM update integrity feature ensures that only Intel-approved firmware code (or other protected NVM module) is able to be updated on the integrated 10 GbE LAN controller devices after manufacturing. This procedure is performed each time there is an attempt to update one of the protected modules.

Integrity validation of NVM updates is provided by a digital signature. The digital signature is a SHA256 Hash computed over the protected content (long by 256-bits), which is then encrypted by a 2048-bits RSA encryption using an Intel private key. This digital signature is stored in the manifest in the NVM module image. Also stored in the manifest is the corresponding RSA modulus (public key) and RSA exponent to be used to decrypt the digital signature.

To verify the authenticity of the digital signature, firmware must first verify that the RSA Modulus and RSA Exponent fields in the new firmware image loaded are identical to those in the old firmware image. If the RSA Modulus and Exponent fields are the same, firmware decrypts the digital signature using the 2048-bit RSA Modulus and Exponent fields stored in the manifest of the old firmware image to extract the expected SHA256 Hash of content (stored hash). Firmware then performs an independent SHA256 Hash over the protected content (computed hash). If the stored hash matches the computed hash, the digital signature is accepted, and the NVM update is applied.

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

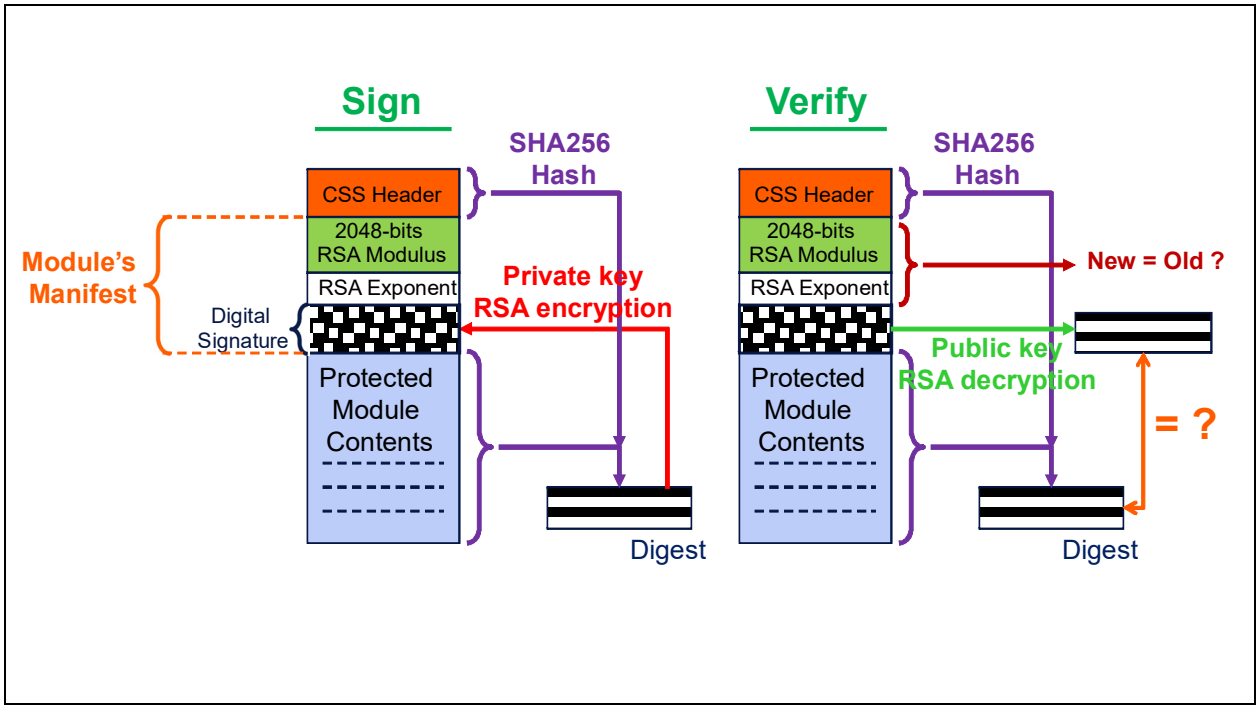


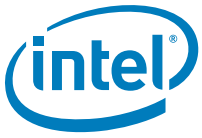
Figure 2-1 Sign and Verify Procedures for Authenticated NVM Modules

2.4.9.1 Digital Signature Algorithm Details

As described the digital signature generation is a hash computation followed by an RSA encryption. This is performed within Intel as part of the NVM update image generation process and not performed by Intel software in the field, nor by the integrated 10 GbE LAN controller.

The different algorithms used are described in the following locations:

- PKCS #1 v2.1: RSA Cryptography Standard, RSA Laboratories, June 14, 2002 - www.rsa.com
- SHA family definition - http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf



- SHA usage with digital signatures - http://csrc.nist.gov/publications/nistpubs/800-107/NIST-SP-800-107.pdf
- SHA validation vectors - http://csrc.nist.gov/groups/STM/cavp/documents/shs/SHAVS.pdf

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

The integrated 10 GbE LAN controller has two 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 shared SPI Flash.
- The SDP pins can also be configured for use as External Interrupt Sources (GPI). To act as GPI pins, the desired pins must be configured as inputs and enabled by the GPIE register. When enabled, an interrupt is asserted following a rising-edge detection of the input pin (rising-edge detection occurs by comparing values sampled at the internal clock rate, as opposed to an edge-detection circuit). When detected, a corresponding GPI interrupt is indicated in the EICR register.

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

The bit mappings are listed in the following table for clarity.

Table 2.9. GPI to SDP Bit Mappings

SDP Pin To Be Used As GPI	ESDP Field Settings		Resulting EICR Bit (GPI)
	Directionality	Enable as GPI interrupt	
1	SDP1_IODIR	SDP1_GPIEN	26
0	SDIPIODIR	SDP0_GPIEN	25

- The lowest SDP pins (SDP0_0) of port 0 can be used to encode the NC-SI package ID of the integrated 10 GbE LAN controller. This ability is enabled by setting bit 15 in NC-SI Configuration 2 word (offset 0x07) of the NVM. The 3-bit package ID is encoded as follows: Package ID = {0, SDP0_0, 0}.
- When the SDP pins are used as IEEE1588 auxiliary signals they can generate an interrupt on any transition (rising or falling edge).

All SDP pins can be allocated to hardware functions. See more details on IEEE1588 auxiliary functionality in Section 7.2 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.

The following table lists the setup required to achieve each of the possible SDP configurations:

Table 2.10. SDP Settings

SDP	Usage	NVM Setting	ESDP			
			SDPx_NATIVE	SDPx_IODIR	SDP1_Function	
0	SDP	N/A	0b	Input/Output	N/A	
	NC-SI package ID (Port #0 only)	bit 15 in NC-SI Configuration 2 NVM word	0b	Input		
	1588 functionality as defined by the TSSDP register	N/A	1b	Input/Output		
1	SDP	N/A	0b	Input/Output		
	1588 functionality as defined by the TSSDP register	N/A	1b	Input/Output	0b	

2.6 LEDs

The integrated 10 GbE LAN controller implements two output drivers intended for driving external LED circuits per port. Each of the 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 shared SPI Flash fields thereby supporting LED displays configurable to a particular OEM preference.

Each of the LED's can be configured to use one of a variety of sources for output indication. For more information on the MODE bits refer to the LEDCTL register description in Section 8.2.2.1.11.

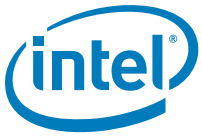
The *IVRT* bits enable the LED source to be inverted before being output or observed by the blink-control logic. LED outputs are assumed to normally be connected to the negative side (cathode) of an external LED.

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.

Note: The LINK/ACTIVITY mode ignores the BLINK value. The LED’s behavior in this mode 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[1]	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	Reserved.
0111b	LINK_2_5G	Asserted or blinking according to the LEDx_BLINK setting when a 2.5 Gb/s link is established and maintained.
1110b	LED_ON	Always asserted or blinking according to the LEDx_BLINK setting.
1111b	LED_OFF	Always de-asserted.

1. Undefined values are reserved.

2.7 Network Management Interface (MDIO or I^{2}C)

The integrated 10 GbE LAN controller supports MDIO (Section 2.7.4) and/or I^{2}C (Section 2.7.5) interfaces, per port, for control plane connection between the MAC (master side) and external PHY devices. The management interface enables both MAC and software access to the PHY for monitoring and controlling of the PHY's functionality and configurations. The integrated 10 GbE LAN controller is compliant with the IEEE Std 802.3 Clause 45 in 10 GbE and 1 GbE operation. The integrated 10 GbE LAN controller also supports IEEE Std 802.3 Clause 22 frame formats and register address space for accessing legacy PHY registers.

There are various connectivity options to external PHYs some use MDIO, such as Base-T PHYs, others might use I^{2}C, like SFP+ modules while in some special scenarios both might be needed such as when an SoC does not natively support SFI and requires the system designer to use both an external PHY for SFI translation, and an external SFP+ module. In such scenarios Denverton SoC LAN controller might need to use the MDIO to manage the external PHY and the I^{2}C to manage the external SFP+ module.

2.7.1 I^{2}C or MDIO Selection

Since the integrated 10 GbE LAN controller IP supports both MDC/MDIO and I^{2}C management interfaces the final board and device setup must be communicated to the software via an NVM loaded configuration.

The following settings are provided by the IP, in the NW Management Interface Select register, for the software to read. Based on these settings the software is able to understand if there is a companion device connected through either of the port's management interfaces. After power on, the software reads the identity of the device connected through the interface and learns the physical topology.

Table 2.11. NW Management Selection Settings - Per Port

Field Name	Description	"SFP+: SoC Without SFI PHY"	"SFP+: SoC With SFI PHY"	"10/1 GbE BASE-T PHY"	"QSFP+: SoC Without SFI PHY"	"QSFP+: SoC Without SFI PHY"
I2C ACT	A value of 1b indicates that the port's I^2C interface is active and connected to an external PHY or module.	1b	1b	0b	1b	1b
MDIO ACT	A value of 1b indicates that the port's MDIO interface is active and connected to an external PHY or module.	1b	0b	1b	1b	0b
MDIO IF MODE	Used to indicate what MDC/MDIO mode is used on Port 0 when NW Management Mode is MDC/MDIO. 1b = Use IEEE Clause 22 Frame Structure. 0b = Use IEEE Clause 45 Frame Structure.	0b	NA (0b)	0b	0b	NA (0b)
EN_SHARED_MDIO	When <MDIO_ACT> is set, then this bit indicates that the MDIO pins are shared among multiple drivers and require an arbitration request before access.	1b	0b	1b	1b	0b
EN_SHARED_I2C	When <I2C_ACT> is set, then this bit indicates that the I^2C pins are shared among multiple drivers and require an arbitration request before access.	0b	0b	0b	1b	1b

2.7.2 Recommended PHY Connectivity Modes

The following diagrams show various recommended PHY connectivity modes.

2.7.2.1 10 GbE SFP+

Figure 2.3 shows a mode where the SoC's Ethernet interface does not support SFI, or a scenario where system design limitations require adding an external PHY that translates from the SoC's supported interface to SFI.

The management interface is used to configure the external PHY as MDIO that is shared between the two ports using an on-die hardware arbitration mechanism (see Section 2.7.3).

The SFP+ modules are managed using an I^2C interface per port.

Figure 2.3. SFP+: SoC Without SFI PHY

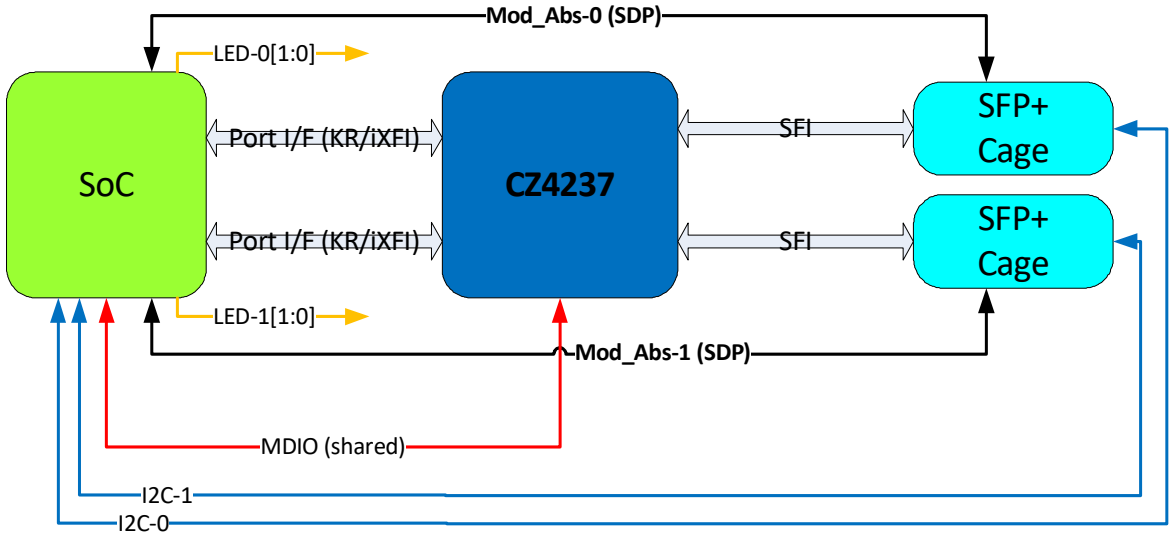
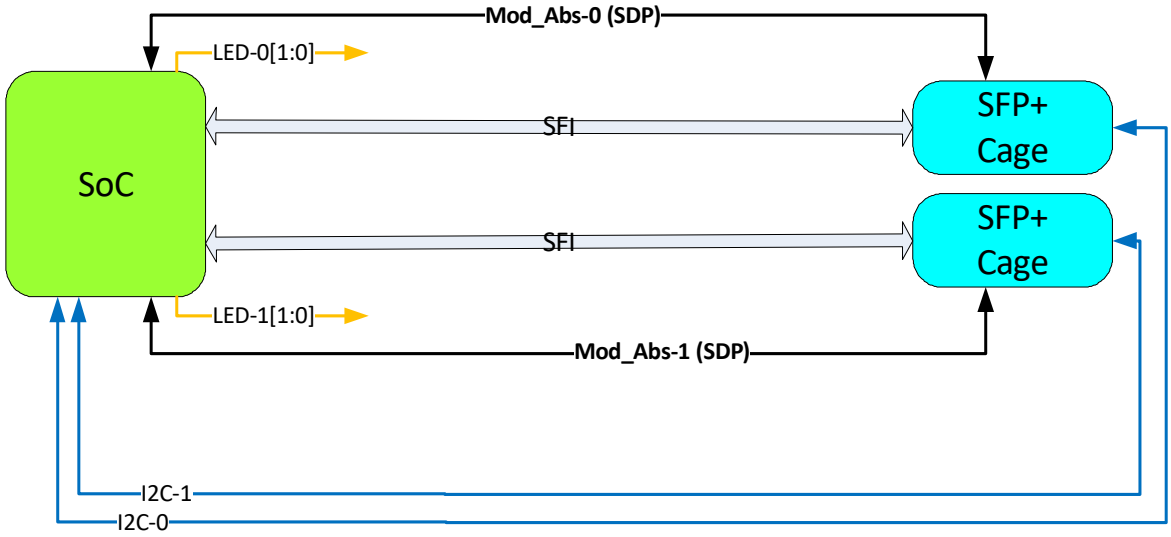


Figure 2.4 shows a mode where the SoC's Ethernet interface does support SFI. In such a setup, the SoC is connected directly to the SFP+ cages, which are managed using an I^2C interface per port.

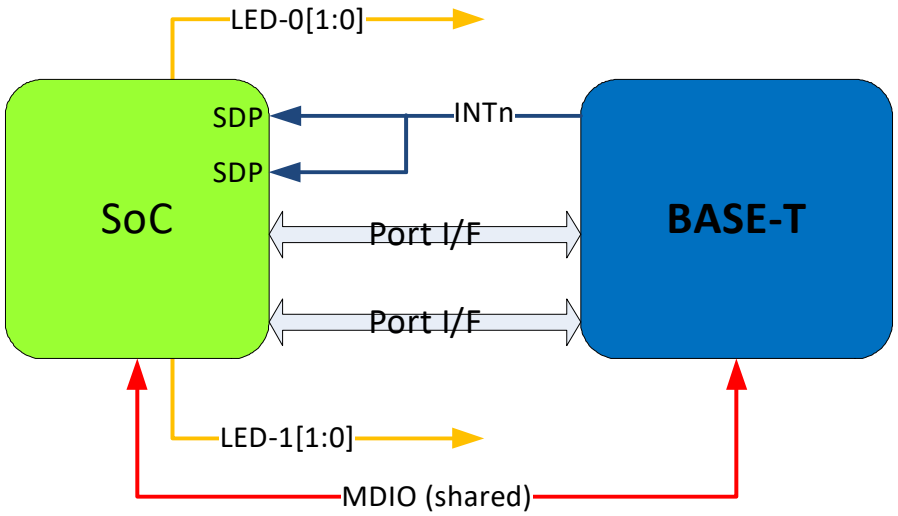
Figure 2.4. SFP+: SoC With SFI PHY



2.7.2.2 10 GbE/1 GbE BASE-T

Figure 2.5 shows a BASE-T solution. The management interface is used to configure the external BASE-T PHY as MDC/MDIO, which is shared between the two ports using an on-die hardware arbitration mechanism (see Section 2.7.3).

Figure 2.5. 10/1 GbE BASE-T PHY



2.7.2.3 QSFP+

Figure 2.6 shows a mode where the SoC has integrated dual Denverton SoC LAN controller IPs.

In Figure 2.6, the SoC's Ethernet interface does not support SFI or system design limitations require adding an external PHY to translate from the SoC's supported interface (KR/XAUI etc.) to SFI.

The management interface used to configure the external PHY is MDIO, which is shared between the ports using an on-die hardware arbitration mechanism.

The management interface used to configure the QSFP+ is I^2C, which is shared between the ports using an on-die hardware arbitration mechanism.

For details concerning the arbitration mechanism, see Section 2.7.3.

Figure 2.6. QSFP+: SoC Without SFI PHY

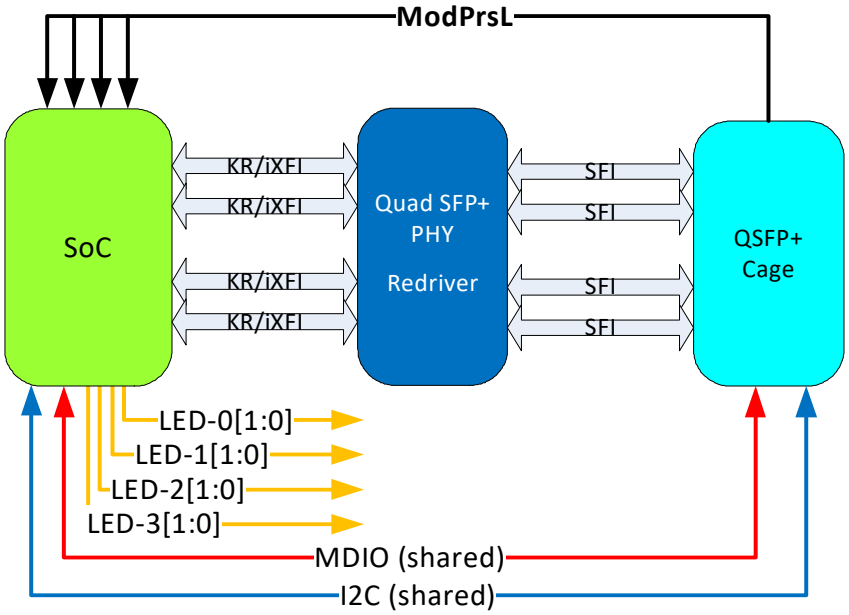
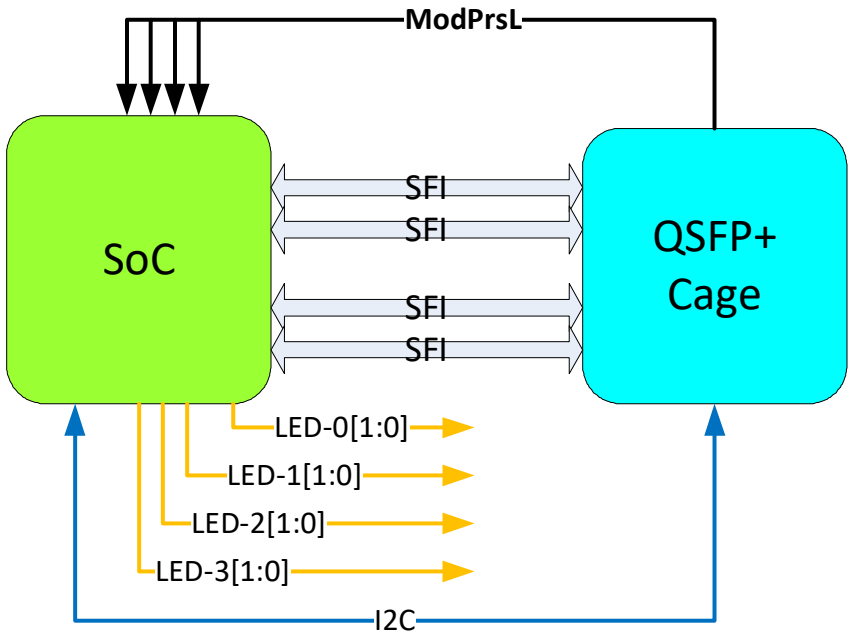


Figure 2.6 shows a mode where the SoC has integrated a dual Denverton SoC LAN controller and an SFI Hard IP. Together these can be used to connect to a QSFP+ cage for a quad 10 GbE setup.

The management interface used to configure the QSFP+ is I^2C, which is shared between the ports using an on-die hardware arbitration mechanism. See Section 2.7.3.

Figure 2.7. QSFP+: SoC With SFI PHY



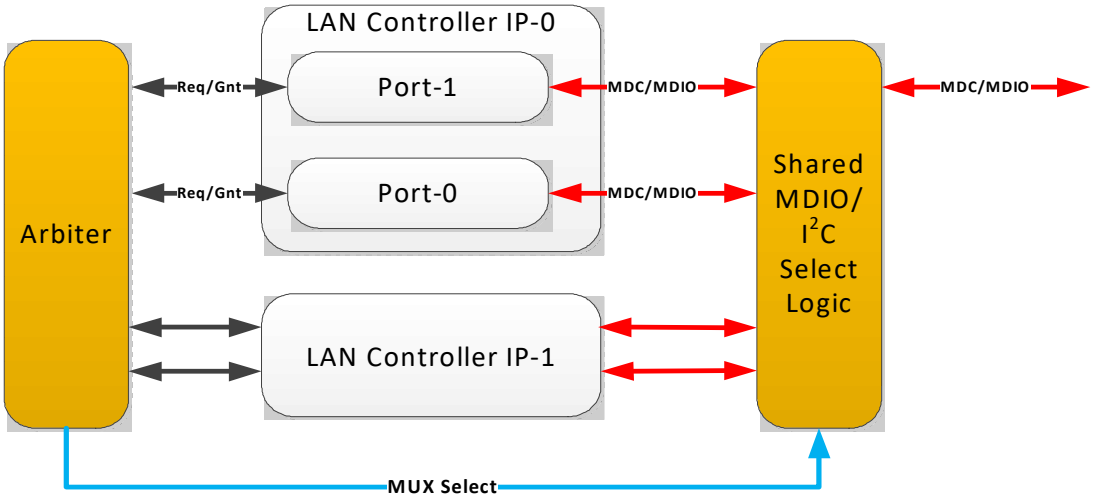
2.7.3 Management Interface Sharing

Since the registers that control the management interface can be used by either software or firmware in alternation, the ownership must be acquired/released via the semaphore ownership management flows described in Section 3.6.

2.7.3.1 Shared MDIO/I^{2}C Management Interface

There are cases where the physical interface being used for accessing the external PHY, MDIO or I^{2}C, is shared among several The integrated 10 GbE LAN controller instances. Figure 2.8 shows such a sample scenario where the MDIO interface is shared among several instances of the IP.

Figure 2.8. Shared MDIO/I²C Logic



The arbitration, for management interface access between the separate clients, is performed by an arbiter block that is external to the integrated 10 GbE LAN controller. This arbiter receives a request signal from each port client and asserts a grant signal to the selected client while at the same time controlling the MUX logic block to select the signals being driven by the selected client's MDIO/I²C and driving these to the external device interface.

Note: When firmware needs to use the shared NW Management I/F it can use either Port-0's or Port-1's control registers and physical interface.

The NW management interface sharing can be enabled for I²C or MDIO separately. To learn what is implemented on a specific board, the software device driver and firmware should read the NW Management Interface Select register.

- NW_MNG_IF_SEL<EN_SHARED_MDIO>
- NW_MNG_IF_SEL<EN_SHARED_I2C>

Each client, driver or firmware should perform the following flow before using the management interface access register when working in shared MNG interface mode (I²C or MDIO).

Note: The sections that follow describe the MDIO flow. The I²C flow is identical except for the signal names.

- Acquire the semaphore for the management interface.
- Read the physical port number, <x>, from *LAN_ID* field in the integrated 10 GbE LAN controller's Device Status register (0x00000008).
- Assert the physical port's request signal (SOC_GEN_CTRL_REG<SHARED_MDIO_REQx>).
- Poll the grant signal (SOC_GEN_CTRL_REG<SHARED_MDIO_GNTx>).
- When grant is asserted, use the standard MDIO registers to perform MDIO access.
- Once a transaction on MDIO completes, de-assert the request so that the arbiter can pass the control to another client.
- Release the semaphore.

2.7.4 Management Data I/O Interface (MDIO)

The integrated 10 GbE LAN controller supports the MDIO interface for a control plane connection between the MAC (master side) and PHY devices. The MDIO interface enables both MAC and software access to the PHY for monitoring and controlling of the PHY's functionality and configurations. The integrated 10 GbE LAN controller is compliant with the IEEE802.3 clause 45 in both 10 GbE and 1 GbE operation. The integrated 10 GbE LAN controller also supports IEEE 802.3 clause 22 frame formats and register address space for accessing legacy PHY registers.

Note: The MDIO interface uses LVTTL signaling as defined in Clause 22 of the IEEE802.3 standard. To access PHYs that support clause 45 1.2V electrical interface, level translators might be needed on board.

Figure 2.9 shows the basic connectivity between the PHY and MAC.

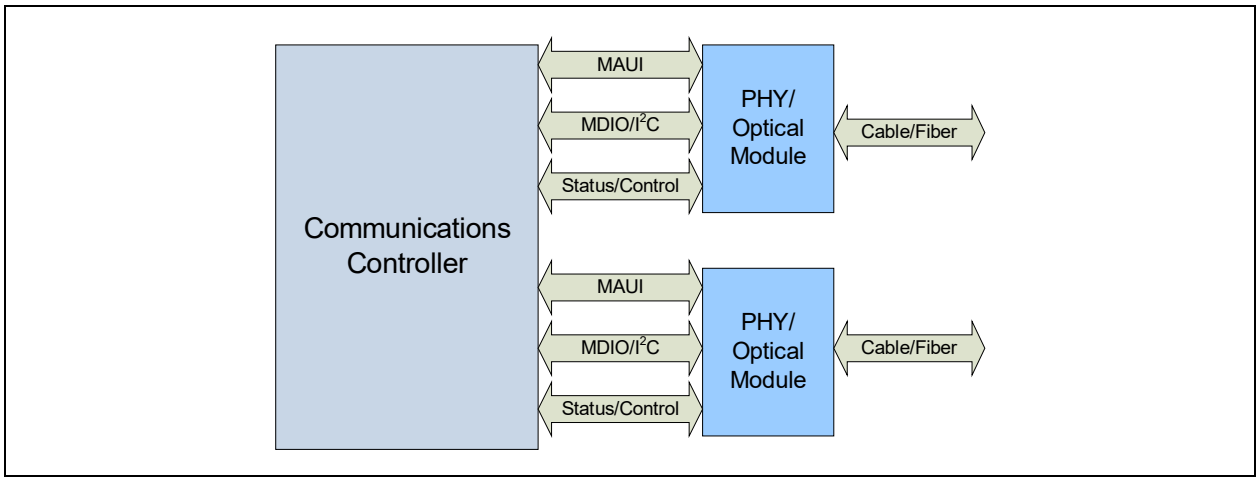


Figure 2.9 Basic PHY MAC Connectivity

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

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

2.7.4.1 MDIO Timing Relationship to MDC

The MDC clock toggles during a read/write operation at a frequency of 24 MHz, 2.4 MHz or 240 KHz depending on the link speed and register bit HLREG0.MDCSPD as listed in Table 2.12.

Table 2.12 MDC Frequency as Function of Link Speed and MDC Speed Bit

Link Speed	MDCSPD=1b	MDCSPD=0b
10 Gb/s	24 MHz	2.4 MHz
1 Gb/s	2.4 MHz	240 KHz

MDIO is a bidirectional signal that can be sourced by the Station Management Entity (STA) or the PHY. When the STA sources the MDIO signal, the STA must provide a minimum of 10 ns of setup time and a minimum of 10 ns of hold time referenced to the rising edge of MDC, as shown in Figure 2.10 (measured at the MII connector).

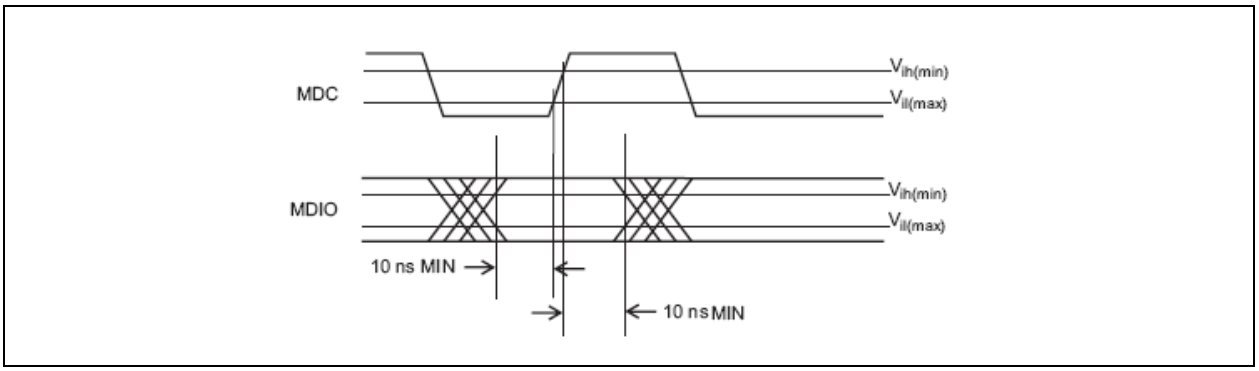


Figure 2.10 MDIO Timing Sourced by the MAC

When the MDIO signal is sourced by the PHY, it is sampled by the MAC (STA) synchronously with respect to the rising edge of MDC. The clock to output delay from the PHY, as measured at the MII connector, must be a minimum of 0 ns, and a maximum of 300 ns, as shown in Figure 2.11.

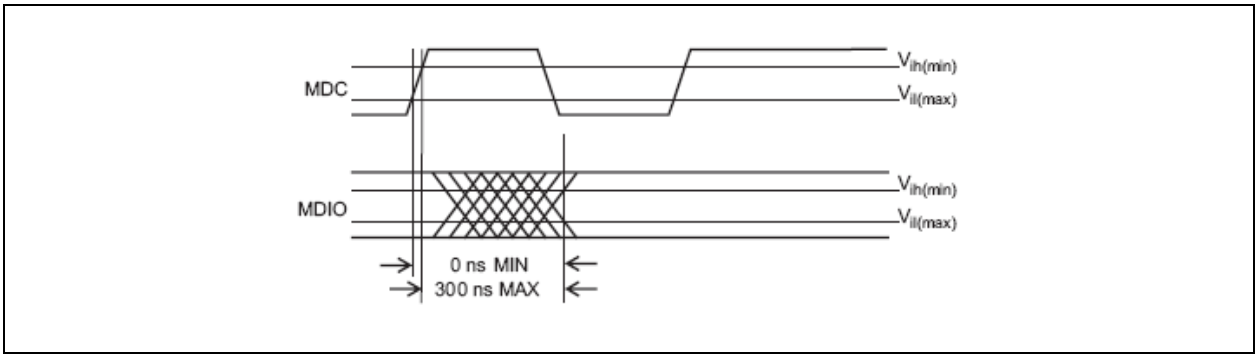


Figure 2.11 MDIO Timing Sourced by the PHY

2.7.4.2 IEEE802.3 Clause 22 and Clause 45 Differences

IEEE802.3 clause 45 provides the ability to access additional device registers while still retaining logical compatibility with interface defined in Clause 22. Clause 22 specifies the MDIO frame format and uses an ST code of 01 to access registers. In clause 45, additional registers are added to the address space by defining MDIO frames that use a ST code of 00.

Clause 45 (MDIO interface) major concepts:

a. Preserve management frame structure defined in IEEE 802.3 Clause 22.
b. Define mechanism to address more registers than specified in IEEE802.3 Clause 22.
c. Define ST and OP codes to identify and control the extended access functions.

2.7.4.3 MDIO Management Frame Structure

The MDIO interface frame structure defined in IEEE802.3 clause 22 and Clause 45 are compatible so that the two systems supporting different formats can co-exist on the same MDIO bus. The integrated 10 GbE LAN controller supports both frame structures to enable interfacing PHYs that support either protocol.

The basic frame format as defined in IEEE802.3 clause 22 can optionally be used for accessing legacy PHY registers is listed in Table 2.13.

Table 2.13 Clause 22 Basic MDIO Frame Format

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

The MDIO interface defined in clause 45 uses indirect addressing to create an extended address space enabling access to a large number of registers within each MDIO Managed Device (MMD). The MDIO management frame format is listed in Table 2.14.

Table 2.14 Clause 45 Indirect Addressing MDIO Frame Format

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

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

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

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

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

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

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

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

For Clause 45 compatible frames:

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

For Clause 22 compatible frames:

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

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

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

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

Turnaround (TA) — The TA time is a 2-bit time spacing between the *DEVAD* field and the *Data* field of a management frame. This is to avoid contention during a read transaction. For a read or post-read-increment-address transaction, both the STA and the PHY must remain in a high-impedance state for the first bit time of the TA. The PHY must drive a zero bit during the second bit time of the TA of a read or post-read-increment-address transaction. During a write or address transaction, the STA must drive a one bit for the first bit time of the TA and a zero bit for the second bit time of the TA. Figure 2.12 shows the behavior of the MDIO signal during the *TA* field of a read transaction.

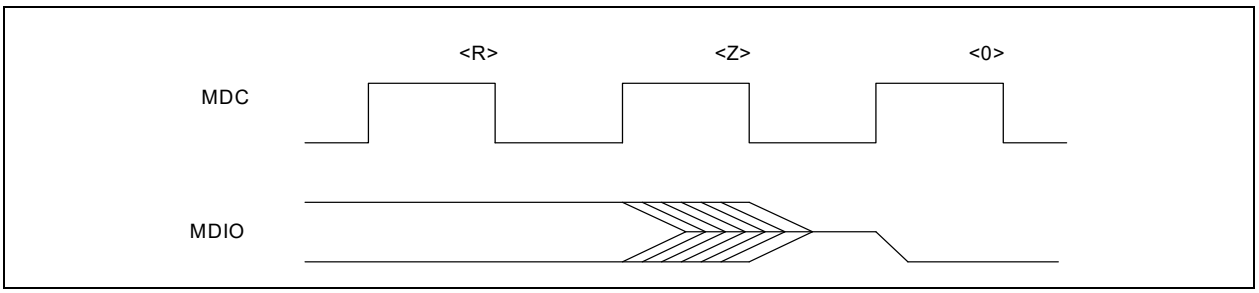


Figure 2.12 Behavior of MDIO During TA Field of a Read Transaction

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

2.7.4.4 MDIO Direct Access

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

Note: For most implementations, it is not possible or recommended that software use MSCA/MSRWD registers to enable LAN MDIO. All Intel-provided solutions use firmware-based LAN MDIO. Please consult your Intel field representative before attempting to use these registers.

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

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

2.7.5 I^2C

The integrated 10 GbE LAN controller supports 2-wire management interface (I^2C) for connectivity to external SFP+ modules. Optical and direct attached copper PHYs use 2-wire management interface (I^2C) as described in SFF8431 (SFP+) or SFF8636 (Direct attach Cu).

Note: Some external PHYs also support I^2C management interface instead of MDIO. It is up to the board designer to choose which interface to use. See Section 2.7.1.

The I^2C interface operates via the I2CCMD and I2CPARAMS register set. Since this register set can be used by either software or firmware in alternation, its ownership must be acquired/released via the semaphore ownership taking/release flows described in Section 3.6.

The I^2C interface can be used in two methods, a hardware based access, where the device initiates a transaction following a software request via the I2CCMD register or a software controlled bit banging using the I2CPARAMS register.

2.7.5.1 Hardware Based I^2C Access

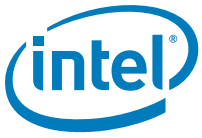
The following flows should be used to access an I^2C register.

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

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

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

1. Check that register is ready: Poll the *I2CCMD.R* bit until it is read as 1b.



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

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

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

Note: See Section 2.7.5.3 for the I^2C commands supported when using the built-in read and write commands. When using the bit bang method any command can be given to the I^2C device.

2.7.5.2 Bit Bang Based I^2C Access

In this mode, the software controls the I^2C interface directly using the I2CPARAMS register according to the following table:

Pad	Field Controlling The Output Value	Field Reflecting The Input Value	Field Controlling The Output Enable Value[1]
I2C clock	CLK_OUT	CLK_IN	CLK_OE_N
I2C data	DATA_OUT	DATA_IN	DATA_OE_N

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

2.7.5.3 Supported Commands

Note: The gray columns that follow denotes cycles driven by the I^2C device. White columns denotes cycles driven by the integrated 10 GbE LAN controller.

When a word Read command (*I2CPARAMS.ACCESS_WIDTH* = 1b, *I2CCMD.OP* =1b) is given, the following sequence is done by the integrated 10 GbE LAN controller:

Table 2.15. I^2C Read Transaction - Dummy Write

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

Table 2.16. I^2C Read Transaction - Word Read

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

When a byte read command (*I2CPARAMS.ACCESS_WIDTH* = 0b, *I2CCMD.OP* =1b) is given the following sequence is done by the integrated 10 GbE LAN controller:

Table 2.17. I²C Read Transaction - Dummy Write

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

Table 2.18. I²C Read Transaction - Byte Read

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

When a word Write command (*I2CPARAMS.ACCESS_WIDTH* = 1b, *I2CCMD.OP* =0b) is given the following sequence is done by the integrated 10 GbE LAN controller:

Table 2.19. I²C Write Transaction - Word Write

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

When a byte write command (*I2CPARAMS.ACCESS_WIDTH* = 0b, *I2CCMD.OP* =0b) is given the following sequence is done by the integrated 10 GbE LAN controller:

Table 2.20. I²C Write Transaction - Byte Write

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

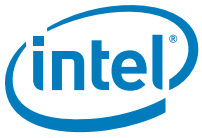
2.8 Network Interface

The integrated 10 GbE LAN controller connects, over XGMII, to two MDI interfaces with supported speeds of 10 GbE, 2.5 GbE, 1 GbE, 100 Mb/s and 10 Mb/s Ethernet. Each interface provides the following physical interfaces and electrical modes:

- A single lane running at up to 10.3125 GHz providing support for:
  - 10GBASE-KR for GbE backplane applications (IEEE802.3 clause 72)
  - 1000BASE-KX for GbE backplane applications (IEEE802.3 clause 70)
  - SGMII 1GbE, 100 Mb/s and 10 Mb/s

2.8.1 Integrated PHY Support

See Appendix B.



2.8.2 Ethernet Flow Control (FC)

The integrated 10 GbE LAN controller supports flow control as defined in 802.3x, as well as the specific operation of asymmetrical flow control defined by 802.3z. The integrated 10 GbE LAN controller also supports Priority Flow Control (PFC), known as Class Based Flow Control, as part of the DCB architecture.

Note: The integrated 10 GbE LAN controller can either be configured to receive regular FC packets or PFC packets. The integrated 10 GbE LAN controller 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 Source Address Low, High (RAL[0], RAH[0]).
- Flow Control Destination Address Low, High (*FCAMACL, FCAMACH)* — 6-byte FC multicast address.
- Priority Flow Control Type Opcode (PFCTOP) — Contains the type and OpCode values for PFC.
- Flow Control Receive Threshold High (FCRTH[7:0]) — A set of 13-bit high watermarks indicating receive buffer fullness. A single watermark is used in link FC mode and up to eight watermarks are used in PFC mode.
- Flow Control Receive Threshold Low (FCRTL[7:0]) — A set of 13-bit low watermarks indicating receive buffer emptiness. A single watermark is used in link FC mode and up to eight watermarks are used in PFC mode.
- Flow Control Transmit Timer Value (FCTTV[3:0]) — A set of 16-bit timer values to include in transmitted PAUSE frame. A single timer is used in link FC mode and up to eight timers are used in PFC mode.
- Flow Control Refresh Threshold Value (FCRTV) — 16-bit PAUSE refresh threshold value (in legacy FC FCRTV[0] must be smaller than FCTTV[0]).

2.8.2.1 MAC Control Frames and Reception of Flow Control Packets

2.8.2.1.1 MAC Control Frame — Other than FC

The IEEE specification reserved the Ethertype value of 0x8808 for MAC control frames, which are listed in Table 2.21.

Table 2.21. 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.

2.8.2.1.2 Structure of 802.3X FC Packets

802.3X FC packets are defined by the following three fields (see Table 2.22):

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 2.22. 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.

2.8.2.1.3 Priority Flow Control

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

PFC is implemented to prevent the possibility of receive packet buffers overflow. Receive packet buffers overflow results in the dropping of received packets for a specific TC. Implement PFC by sending a timer indication to the transmitting station TC (XOFF) of a nearly full receive buffer condition at the integrated 10 GbE LAN controller. 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 integrated 10 GbE LAN controller 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.

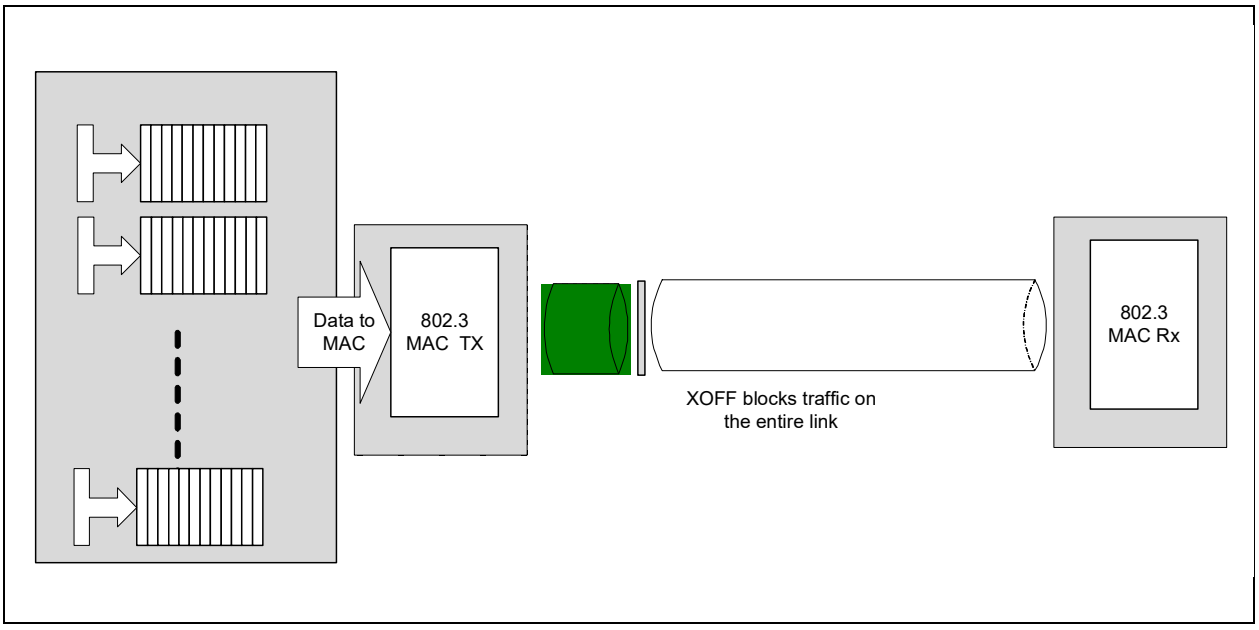


Figure 2.13. 802.3X Link Flow Control (PAUSE)

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 2.14.

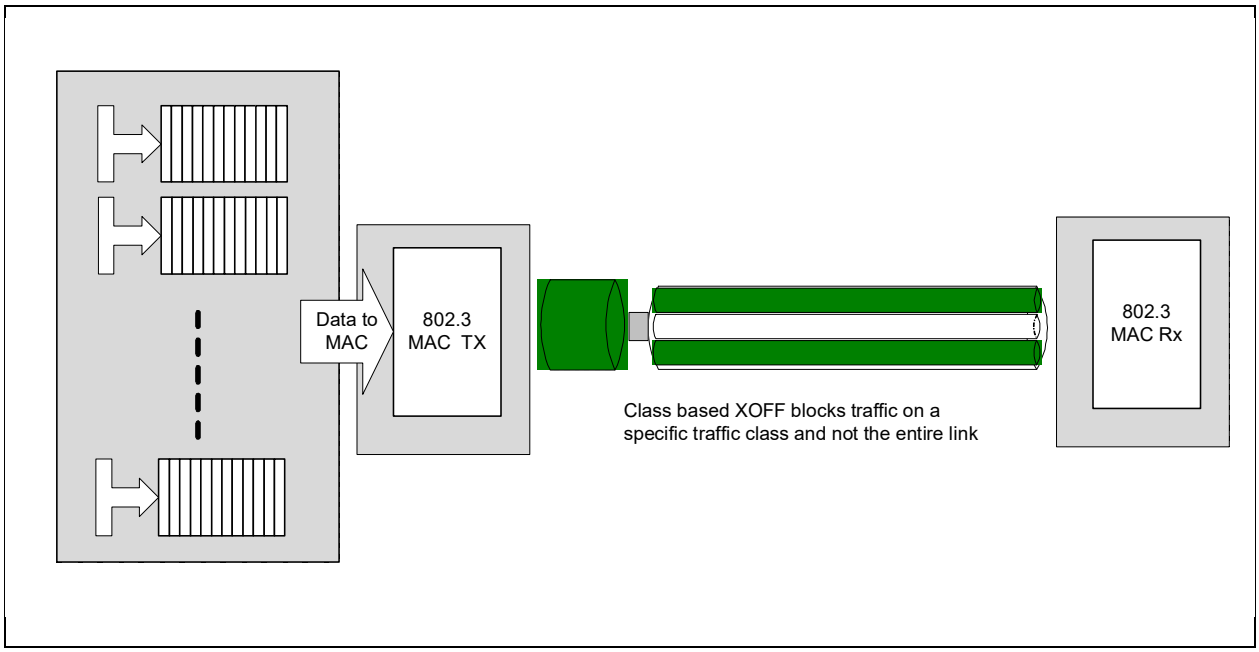


Figure 2.14. Priority Flow Control

Table 2.23. 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.

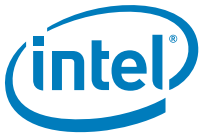


Table 2.24. Format of Priority Enable Vector

	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 integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller, 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.

2.8.2.1.4 Operation and Rules

The integrated 10 GbE LAN controller 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 software by the DCB management agent. It is the software device 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:

- Increments 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.

2.8.2.1.5 Timing Considerations

When operating at 10 GbE line speed, the integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller must not begin to transmit a (new) frame more than two pause quanta 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, or of two 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.

2.8.2.2 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 2.25. 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.

Table 2.25. 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).

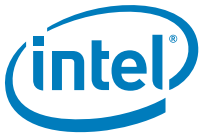


Table 2.25. Transfer of PAUSE Packet to Host (DPF Bit)

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.

2.8.2.3 Transmitting PAUSE Frames

The integrated 10 GbE LAN controller generates PAUSE packets to ensure there is enough space in its receive packet buffers to avoid packet drop. The integrated 10 GbE LAN controller monitors the fullness of its receive FIFOs and compares it with the contents of a programmable threshold. When the threshold is reached, the integrated 10 GbE LAN controller sends a PAUSE frame. The integrated 10 GbE LAN controller 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).

2.8.2.3.1 Priority Flow Control (PFC)

The integrated 10 GbE LAN controller operates in either a link 802.3X compliant mode or in a PFC mode, but not in both at the same time.

The same watermarks mechanism is used for PFC and for 802.3X FC to determine when to send XOFF and XON packets. When PFC is used in the receive path, priority PAUSE packets are sent instead of 802.3X PAUSE packets. The format of priority PAUSE packets is described in Section 2.8.2.1.3.

Specific considerations for generating PFC packets:

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

2.8.2.3.2 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 integrated 10 GbE LAN controller transmits the first PAUSE frame. The integrated 10 GbE LAN controller monitors the fullness of the receive FIFO and compares it with the contents of FCRTH. When the threshold is reached, the integrated 10 GbE LAN controller sends a PAUSE frame with its pause time field equal to FCTTV.

At this time, the integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller 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.

2.8.2.3.3 Flow Control High Threshold — FCRTH

The integrated 10 GbE LAN controller 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 TC for which FCRTH is computed. It is referred as MaxFrame(TC).
2. Maximum frame size over the link (all TCs altogether). It is referred as MaxFrame(link).

Three values are envisaged for MaxFrame:

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

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

Std DV = MaxFrame(TC) + MaxFrame(link) + PFC Frame + 2 x Cable Delay + 2 x Interface Delay + Higher Layer Delay x Sec Y Transmit Delay

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

MaxFrame(TC) term and MaxFrame(link) term included in Sec Y Transmit Delay correspond to worst case scenarios issued by the link partner. All other terms in Std DV formula must take in account worst case incoming traffic pattern which would lead to worst case buffer utilization as per the internal architecture of Rx packet buffer in the integrated 10 GbE LAN controller.

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 following revised formula for the integrated 10 GbE LAN controller:

The integrated 10 GbE LAN controller DV (bit time units) = 1.44 x [(MaxFrame(link) + 672 + 2 x 5,556 + 2 x (25,600 + 8,192 + 2 x 2,048) + 6,144 x (3,200)] + MaxFrame(TC) + MaxFrame(TC) x MaxFrame(link).

FCRTH must be set to the size of the Rx packet buffer allocated to the TC minus the integrated 10 GbE LAN controller DV.

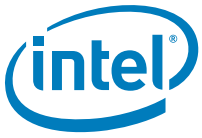


Table 2.26. Integrated 10 GbE LAN Controller Delay Values (DV) Used For FCRTH

9.5 KB Jumbo Enabled	Integrated 10 GbE LAN Controller DV
No	24 KB
No	25 KB
Yes	50 KB
Yes	35 KB
No	27 KB
No	27 KB
Yes	60 KB
Yes	45 KB

Note: 9.5 KB jumbo enabled/disabled is a global setting per port which concerns all TCs.

2.8.2.3.4 FC Low Threshold — FCRTL

The low threshold value is aimed to protect against wasted available host bandwidth. There is some latency from the time that the low threshold is crossed until the XON frame is sent and packets are received from the link partner. The low threshold must be set high enough so that the Rx packet buffer does not get empty before any new entire packets are received from the link partner. When considering data movement from the Rx packet buffer to host memory, 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 µs and it must cover for worst case incoming traffic pattern (buffer utilization by 1.44 than wire rate):

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

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

Table 2.27. Odem FCRTL

9.5 KB Jumbo Enabled	Integrated 10 GbE LAN Controller DV
No	5 KB
No	7 KB
Yes	21 KB
Yes	7 KB

2.8.2.3.5 Packet Buffer Size

When FC is enabled, the total size of a TC packet buffer must be large enough for the Low and high thresholds. 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 bytes)
  - Rx packet buffer size = FCRTH + the integrated 10 GbE LAN controller DV (see Section 2.8.2.3.3)
- For a best effort TC:
  - Rx packet buffer size = FCRTL plays 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 TCs are defined over the port, of which two are allocated for loss less traffic types like iSCSI, etc. The table lists the recommended settings for the supported combinations. When less than 4 PFC-enabled TCs are defined, and/or when less than 8 TCs are defined, it is recommended to refer to the setting rules described in this section, in Section 2.8.2.3.3. Note that reducing the number of TCs of a port to what is really needed, helps increasing the port's throughput.

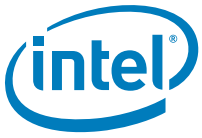


Table 2.28. Some Recommended Rx Packet Buffer Settings

Flow Director Table Size	9.5 KB Jumbo Enabled	Packet Buffer Size of Any of the 4 Best Effort TCs	Packet Buffer Size of Any of The Other 2 PFC-enabled TCs
No	No	33 KB	61 KB FCRTL = 6 KB FCRTH = 37 KB
No	Yes	27 KB	86 KB FCRTL = 21 KB FCRTH = 36 KB
No	No	32 KB	63 KB FCRTL = 6 KB FCRTH = 36 KB
No	Yes	22 KB	91 KB FCRTL = 21 KB FCRTH = 31 KB
64 KB	No	25 KB	53 KB FCRTL = 6 KB FCRTH = 29 KB
64 KB	Yes	19 KB	78 KB FCRTL = 20 KB FCRTH = 28 KB
64 KB	No	24 KB	55 KB FCRTL = 6 KB FCRTH = 28 KB
64 KB	Yes	14 KB	83 KB FCRTL = 18 KB FCRTH = 23 KB
128 KB	No	17 KB	45 KB FCRTL = 6 KB FCRTH = 21 KB
128 KB	No	16 KB	47 KB FCRTL = 6 KB FCRTH = 20 KB

Notes: In some of the previous 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.

In some other cases, after having applied all the rules there was an exceeding available Rx packet buffer left that has been used to extend the hysteresis budgets.

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

2.8.2.4 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 integrated 10 GbE LAN controller can also throttle the traffic according to incoming link FC notifications. Following is the required device setting and functionality.

- The integrated 10 GbE LAN controller 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.

2.8.3 Inter Packet Gap (IPG) Control and Pacing

The integrated 10 GbE LAN controller supports transmission pacing by extending the IPG (the gap between consecutive packets). The pacing mode enables the average data rate to be slowed in systems that cannot support the full link rate (10 GbE, 1 GbE or 100 Mb/s). As listed in Table 2.29, 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 2.29 lists the pacing configurations supported by the integrated 10 GbE LAN controller at link rates of 10 GbE. When operating at lower link speeds the pacing speed is proportional to the link speed.

Table 2.29. 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

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

3.0 Initialization

3.1 Reset Operation

The integrated 10 GbE LAN controller reset sources are described in the sections that follow.

3.1.1 Reset Sources/Reset Order

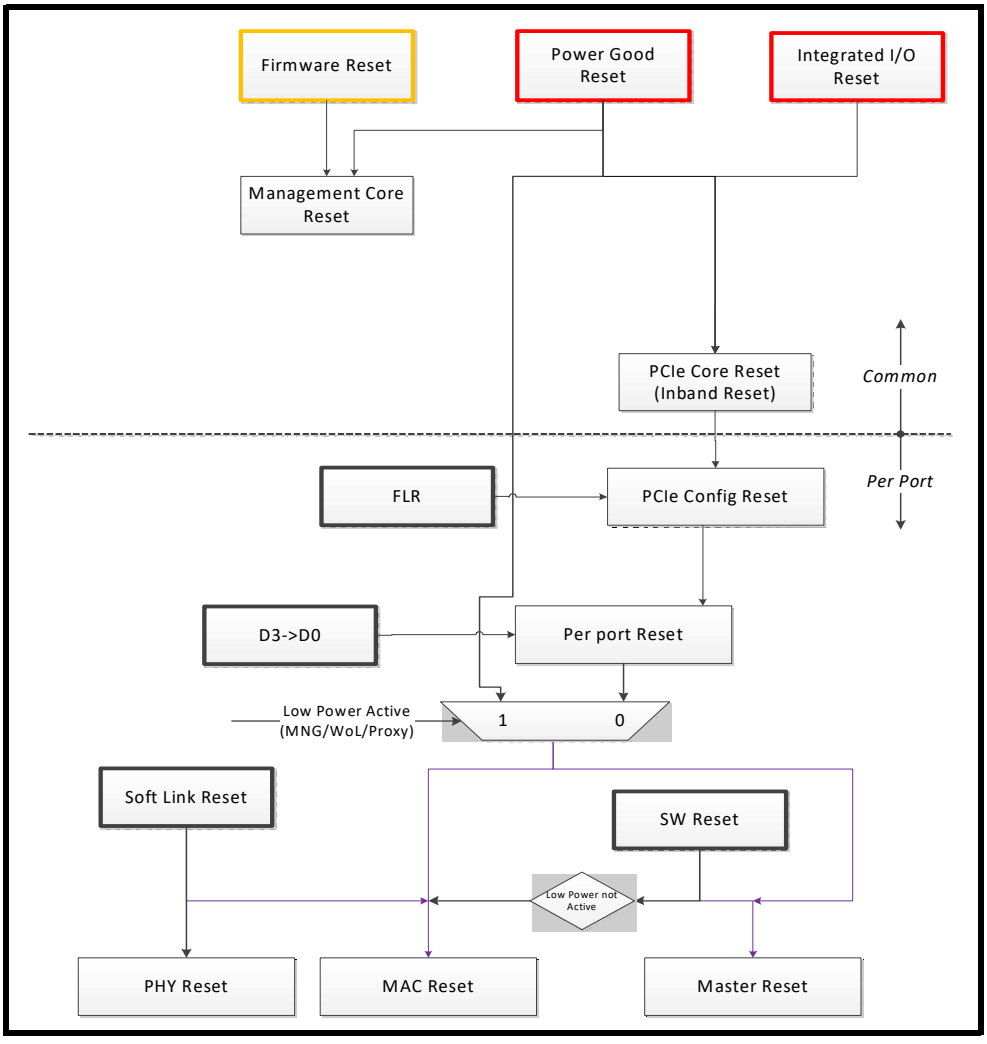


Figure 3.1. Reset Tree

The different reset sources are organized in the following order (high-to-low):

1. Power Good Reset
2. Integrated I/O Reset
3. Physical Function Reset (PFLR)
4. Virtual Function Reset (VFLR)

A higher-level reset is either followed by a lower-level reset or directly resets the information related with a lower-level reset.

3.1.1.1 Power Good Reset

This reset signal is the master reset of the integrated 10 GbE LAN controller. It is level sensitive, and while it is 0b, all of the registers are held in reset.

Power Good Reset changes state during system power up.

3.1.1.2 Integrated I/O Reset

This is the reset event that replaces the PCIe Reset (PE_RST_N) signal in a discrete LAN controller. Most of the internal blocks of the integrated 10 GbE LAN controller are reset on the rising edge of this reset.

This reset might also reset the integrated PHY depending on WoL and manageability requirements.

3.1.1.2.1 Integrated I/O Cold/Warm Reset

When the SoC is about to assert an integrated I/O reset, it first issues an internal message to enable a graceful reset flow. Upon receiving this message, the integrated 10 GbE LAN controller:

- Stops sending new requests out to the Integrated I/O interfaces.
- Waits for all pending completion to arrive.
- Responds to all pending non-posted requests.

Further, during the period between receiving this message and the actual reset event, the integrated 10 GbE LAN controller is expected to respond to new incoming requests either normally or with a Unsupported Request (UR). Once these previous steps complete, the integrated 10 GbE LAN controller sends an acknowledge to the SoC to indicate that it might perform:

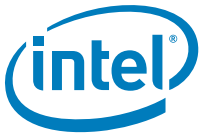
- A warm reset is equal to asserting the Integrated I/O reset

Or

- A cold reset. For example, a full power on

3.1.1.3 D3hot to D0 Transition

This is also known as an ACPI reset. The integrated 10 GbE LAN controller generates an internal reset on the transition from D3hot power state to D0 (caused after configuration writes from a D3-to-D0 power state). Note that this reset is per function and resets only the function that transitions from D3hot-to-D0 and does not reset the configuration space of the function. Its effect is equivalent to a software reset (Section 3.1.1.5.1).



3.1.1.4 Function Level Reset (FLR) Capability

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

3.1.1.4.1 FLR in Non-IOV Mode

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

3.1.1.4.2 Physical Function FLR (PFLR)

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

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

3.1.1.4.3 Virtual Function FLR (VFLR)

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

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

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

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

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

3.1.1.5 Soft Resets

3.1.1.5.1 Software Reset

A software reset is done by writing to the *Device Reset* bit of the Device Control (CTRL.RST) register. The integrated 10 GbE LAN controller re-reads the per-function shared SPI Flash fields after software reset. Bits that are not normally read from the shared SPI Flash 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 3.1.3).

Prior to issuing a software reset, the software device driver needs to execute the master disable algorithm as defined in Section 4.1.3.3.2.

1. Clear the Flow Control enablement in the MAC by clearing the MFLCN.RFCE (or just clear the entire register).
2. Software should wait ~10 μs.
3. Software polls the *TFCS.TC_XON*[0] = 0b (most of the time it is expected to be found at 0b while maximum poll time is always shorter than the maximum expected PAUSE time before the software reset was initiated).
4. Software maps the manageability transmit TC (setting the MNGTXMAP register) and then map the user priority of manageability traffic to the manageability TC (setting the RTRUP2TC and RTTUP2TC registers).
5. Software then waits ~10 μs.
6. Software can re-enable the FC as part of the rest of the initialization flow.

3.1.1.5.2 Physical Function (PF) Software Reset

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

The procedure for PF software reset is as follows:

1. The PF driver disables master accesses by the device through the master disable mechanism (see Section 4.1.3.3.2). Master disable affects all VFs traffic.
2. Execute the procedure described in Section 3.1.2 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.

3.1.1.5.3 VF Software Reset

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

3.1.1.5.4 Force TCO

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

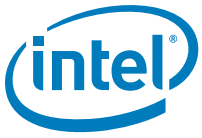
3.1.1.6 Link Reset

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 integrated 10 GbE LAN controller re-reads the per-function shared SPI Flash fields after link reset. Bits that are not normally read from the shared SPI Flash 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 device driver needs to execute the master disable algorithm as defined in Section 4.1.3.3.2.

3.1.1.7 PHY Resets

A PHY reset event causes a link down and restarts the auto-negotiation process. This can take a few seconds to complete, which might result in the drop of the TCP sessions with the host and/or with a Manageability Controller (MC). Because the PHY can be accessed by the MC (via internal firmware) and by the software device driver concurrently, the software device driver 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 device driver can skip the PHY reset (if it is not mandatory) or wait for this bit to be cleared by the MC. See Section 3.2 for more details on *MNG_VETO* bit.
2. Take ownership of the relevant PHY using the flow described in Section 9.8.4.
3. Reset the PHY using register configuration.

3.1.2 Reset in a PCI-IOV Environment

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

3.1.2.1 RSTI/RSTD

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

- One of the following reset cases takes place:
  - Power Good Reset
  - Integrated I/O Reset
  - D3hot-to-D0 transition
  - FLR
  - Software reset by the PF
- The integrated 10 GbE LAN controller sets the *RSTI* bits in all the VF Mailbox registers. Once the reset completes, each VF can read its VF Mailbox register to identify a reset in progress.
  - The VF can poll the *RSTI* bit to detect if the PF is in the process of configuring the device.
- Once the PF completes configuring the device, it sets the CTRL_EXT.PFRSTD bit. As a result, the integrated 10 GbE LAN controller clears the *RSTI* bits in all the VF Mailbox registers and sets the *Reset Done (RSTD)* bits in all the VFMailbox registers.
  - The VF might read the *RSTD* bit to detect that a reset has occurred. The *RSTD* bit is cleared on read.

3.1.2.2 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.

3.1.3 Reset Effects

The resets listed in Section 3.1.1 affect the following registers and logic:

Table 3.1. Reset Effects — Common Resets

Reset Activation	Power Good Reset	Integrated IO Reset	FW Reset	Force TCO	Notes
Shared SPI Flash read	See Section 4.4				
PCI configuration registers (RO)	X	X			8.
PCI configuration registers (RW)	X	X			8.
PCIe local registers	X				
Data path	X	X		X	2., 7.
MAC, TimeSync	X	X 6.		X	
Integrated PHY (state only reset)		X 6.		X	15.
Integrated PHY (full auto-load)	X				
Wake up (PM) Context	X	X1.			3.
Wake up/manageability control/status registers	X				4., 5.
Manageability unit	X		X		
LAN disable strapping pins	X	X			
All other strapping pins	X				
Shadow RAMs in MAC or PHY	X				

Table 3.2. Reset Effects — Per Function Resets

Reset Activation	D3 or Dr	FLR or PFLR	SW Reset	Exit from LAN Disable	Notes
Shared SPI Flash read	See Section 4.4				
PCI configuration registers (RO)					8.
PCI configuration registers (RW)		X			8.,9.
Data path and memory space, TimeSync	X	X	X	X	2., 7.
MAC	X 6.	X 15.	X 15.	X	
Integrated PHY (state only reset)	X 6.	X 15.		X	
Virtual function resources	X	X	X		10.
Wake up (PM) context					3.
Wake up/manageability control/status registers					4., 5.
Manageability unit					
Shadow RAM					

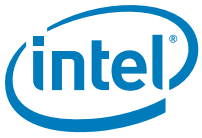


Table 3.3. Reset Effects -Virtual Function Resets

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 3.1 Through Table 3.3 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 integrated 10 GbE LAN controller does not support PME from D3cold).
2. The following register fields do not follow the general rules previously described:
   a. ESDP registers — Reset on Power Good Reset only.
   b. LED configuration registers — Reset on Power Good Reset and on software reset events.
   c. The *Aux Power Detected* bit in the PCIe Device Status register is reset on Power Good Reset and Integrated I/O Reset only.
   d. FLA — Reset on Power Good Reset only.
   e. RAH/RAL[n, where n>0], MTA[n], VFTA[n], WUPM[n], FFMT[n], FFVT[n], TDBAH/TDBAL, and RDBAH/RDVAL registers have no default value. If the functions associated with these registers are enabled they must be programmed by software. Once programmed, their value is preserved through all resets as long as power is applied.
   f. Statistic registers (physical function)
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).

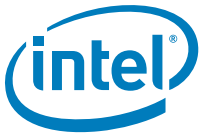
   The shadow copies of these bits in the Wake Up Control (WUC) register are treated identically.
4. Refers to bits in the WUC register that are not part of the wake up context (the *PME_En* and *PME_Status* bits). The WUFC register is not part of the wake up context and is reset as part of the data path.
5. The Wake Up Status (WUS) registers include the following:
   a. WUS register.
   b. Wake Up Packet Length (WUPL) register.
   c. Wake Up Packet Memory (WUPM) register.
6. The MAC cluster and the PHY are reset by the appropriate event only if the manageability unit is disabled and the host is in a low-power state with WoL disabled. WoL disabled means either AUX_PWR pin is cleared, or the *APM Enable* bit in shared SPI Flash 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.

8. Sticky bits and hardware initialization bits (indicated as HwInit) in the PCI Configuration registers are cleared only by Power Good Reset reset.
9. The following register fields are not affected by FLR or PFLR:

- Max_Payload_Size in the Device Control register

10. These registers include:
    a. VFEICS.
    b. VFEIMS.
    c. VFEIAC.
    d. VFEIAM.
    e. VFEITR 0-2.
    f. VFIVAR0.
    g. VFIVAR_MISC.
    h. VFPBACL.
    i. VFMailbox.
11. These registers include:
    a. VFEICS.
    b. VFEIMS.
    c. VFEIMC.
    d. VFEIAC.
    e. VFEIAM.
    f. VFEICR.
    g. EITR 0-2.
    h. VFIVAR0.
    i. VFIVAR_MISC.
    j. VFPBACL.
    k. VFMailbox.
    l. RSCINT
12. These registers include specific VF bits in the FVRE and FVTE registers are cleared as well.
13. These registers include:
    a. MSI/MSI-X enable bits.
    b. BME.
    c. Error indications.
14. Rx and Tx counters might miss proper counting due to VFLR indicating more packets than those ones actually transferred. It could happen if VFLR happened after counting occurred but before Tx or Rx completed.
15. Will not be reset in case the *VETO* bit is asserted by manageability.

Note: Unless specified otherwise the integrated 10 GbE LAN controller's on-die memories are reset together with the functional block(s) they belong to.



3.2 Queue Disable

See Section 3.5.7.1.1 for details on disabling and enabling an Rx queue.

See Section 3.5.8.1.1 for details on disabling and enabling a Tx queue.

3.3 Function Disable

3.3.1 General

The integrated 10 GbE LAN controller provides support for BIOS to selectively enable or disable one or both LAN device(s) in the system.

3.3.2 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 integrated 10 GbE LAN controller's LAN-disable mechanism is implemented in order to be compatible with such a solution. These straps might be controlled by BIOS using a dedicated register in the SoC. Refer to the BIOS Writer's Guide for more details.

LAN ports and/or PCIe functions can be disabled by:

- The LAN_DIS_n[x]straps (one strap per LAN port) are read by the integrated 10 GbE LAN controller on reset to determine the LAN enablement.
- One of the LAN ports can be disabled using a shared SPI Flash configuration.

Disabling a LAN port affects the PCI function it resides on. When function 0 is disabled (either LAN0 or LAN1) 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 (See Section 8.2.2.2), class code 0xFF0000). In addition, the function does not require any I/O space, and does not require an interrupt line. It requires a minimal memory space (4 KB) that is not mapped to internal registers.

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

Table 3.4. PCI Functions Mapping

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

The following rules apply to function disable:

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

- The disabled LAN port is still available for manageability purposes through the shared SPI Flash mechanism.
- The disabled LAN port is still available for manageability purposes if disabled through the shared SPI Flash. In this case, and if the *LPLU* bit is set, the PHY attempts to create a link at the lowest supported speed. The disabled LAN port is not available for manageability purposes if disabled through the LAN_DIS_N[x] strap mechanism.
- Dummy function mode should not be used in PCI IOV mode (since PF0 is required to support certain functionality).

The following shared SPI Flash Control Word 2 bits control function disable:

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

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.

3.3.3 Control Options

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

LAN0 and LAN 1 can be disabled by BIOS by driving the LAN_DIS_n[x] pins respectively to low. These pins are strapping options, sampled at Power Good Reset or at integrated I/O reset.

3.3.4 Event Flow for Enable/Disable Functions

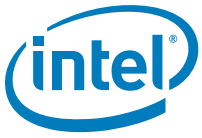
This section describes the driving levels and event sequence for device functionality.

Following a Power Good Reset or integrated I/O reset events the LAN_DIS_n[x] signals should be driven high for normal operation. If any of the LAN functions are not required statically, its associated disable strapping pin can be configured statically to low.

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

Assume that following a power-up sequence LAN_DIS_n[x] (strap_gbe_lan_dis_n <n> signals are driven high.

1. Integrated I/O is established following an Integrated I/O reset.
2. BIOS recognizes that a LAN function in the integrated 10 GbE LAN controller should be disabled.
3. The BIOS drives the LAN_DIS_n[x] signal to a low level.
4. BIOS issues an Integrated I/O reset.
5. As a result, the integrated 10 GbE LAN controller samples the LAN_DIS_n[x] 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 LAN_DIS_n[x] signal high and then requesting the end user to issue a warm boot to initialize new bus enumeration.

3.3.4.2 Multi-Function Advertisement

If one of the LAN devices is disabled and function 0 is the only active function, the integrated 10 GbE LAN controller is no longer a multi-function device. The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller reports 0x0 in this field to signify single-function capability.

3.3.4.3 Interrupt Use

When both LAN devices are enabled, the integrated 10 GbE LAN controller uses the PCI legacy interrupts of both ports for interrupt reporting. The shared SPI Flash 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 shared SPI Flash 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.

3.3.4.4 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 a shared SPI Flash 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 integrated 10 GbE LAN controller 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.

3.4 Device Disable

3.4.1 Overview

In order to disable the device, both LAN_DIS_n[x] signals should be tied statically to low. When sampled at a Power Good Reset or integrated I/O reset events, the integrated 10 GbE LAN controller is disabled. It is held in reset and power-down mode (some clocks are still running), and digital I/O pins are at High-Z if *DEV_FUNC_EN.DEV_OFF_EN* bit was set in the shared SPI Flash control word 1. As an example, digital I/O pins are in an electrical off state where pull-up/pull-down resistors are at their defined values. The manageability interface is also disabled in this state.

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

Assume that following a power-up sequence LAN_DIS_n[x] signals are driven high.

1. Integrated I/O is established following an Integrated I/O reset.

2. BIOS recognizes that the integrated 10 GbE LAN controller should be disabled.
3. The BIOS drives the LAN_DIS_n[x] signals to the low level.
4. BIOS issues an Integrated I/O reset.
5. As a result, the integrated 10 GbE LAN controller samples the LAN_DIS_n[x] signals and disables the LAN ports and the Integrated I/O connection.
6. Re-enable can be done by driving high at least one of the LAN_DIS_n[x] signals and then issuing an Integrated I/O reset to restart the device.

3.5 Software Initialization and Diagnostics

3.5.1 Introduction

This section discusses general software notes for the integrated 10 GbE LAN controller, especially initialization steps. These include:

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

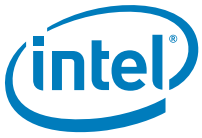
3.5.2 Power-Up State

When the integrated 10 GbE LAN controller powers up, it automatically reads the shared SPI Flash. The shared SPI Flash contains sufficient information to bring the link up and configure the integrated 10 GbE LAN controller for manageability and/or APM wake up. However, software initialization is required for normal operation.

3.5.3 Initialization Sequence

The following sequence of commands is typically issued to the device by the software device driver in order to initialize the integrated 10 GbE LAN controller to normal operation. The major initialization steps are:

1. Disable interrupts.
2. Issue a Power Good Reset and perform general configuration (see Section 3.5.3.2).
3. Wait for the shared SPI Flash auto-read completion.
4. Wait for manageability configuration done indication.
5. Wait until the DMA initialization completes (RDRXCTL.DMAIDONE).
6. Setup the PHY and the link — see Section B.
7. Initialize all statistical counters — see Section 3.5.5.



8. Initialize receive — see Section 3.5.7.
9. Initialize transmit — see Section 3.5.8.
10. Initialize Virtualization support — see Section 3.5.9
11. Enable interrupts — see Section 3.5.3.1.

3.5.3.1 Interrupts During Initialization

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

1. Disable interrupts.
2. Issue a Power Good Reset.
3. Disable interrupts (again).

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

3.5.3.2 Global Reset and General Configuration

Power Good Reset = software reset + link reset.

Device initialization typically starts with a software reset that puts the device into a known state and enables the software device driver to continue the initialization sequence. Following a global reset the software device 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 2.8.2.3.2 through Section 2.8.2.3.5 for recommended settings of the Rx packet buffer sizes and flow control thresholds.

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

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

3.5.4 Link Initialization

Refer to Section B.4.1 for the initialization and link setup steps.

3.5.5 Initialization of Statistics

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

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

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

3.5.6 Interrupt Initialization

3.5.6.1 Working with Legacy or MSI Interrupts

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

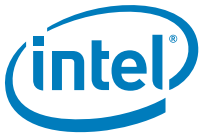
3.5.6.2 Operating with MSI-X

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

3.5.7 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 shared SPI Flash or from any other means (for example, it could be stored anywhere in the shared SPI Flash or even in the platform PROM for a LOM design).

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

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

Initialize the flexible filters 7:0 — 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 off loads via registers FCTRL, VLNCTRL, MCSTCTRL, RXCSUM, RQTC, RFCTL, MPSAR, RSSRK, RETA, SAQF, DAQF, SDPQF, FTQF, SYNQF, ETQF, ETQS, RDRXCTL, and RSCDBU.

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

Program RXPBSIZE, MRQC, PFQDE, RTRUP2TC, and MFLCN.RFCE registers according to virtualization mode.

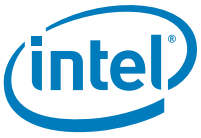
Enable receive jumbo frames by setting *HLREG0.JUMBOEN* in the following case:

1. Jumbo packets are expected. Set *MAXFRS.MFS* to the expected maximum packet size.
2. Enable receive coalescing if required as described in Section 3.5.7.2.

3.5.7.1 Receive Queues Enable

The following should be done for each receive queue:

1. Allocate a region of memory for the receive descriptor list.
2. Receive buffers of appropriate size should be allocated and pointers to these buffers should be stored in the descriptor ring.
3. Program the descriptor base address with the address of the region (registers RDBAH and 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 the *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 might 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 the 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.

3.5.7.1.1 Dynamic Enabling and Disabling of Receive Queues

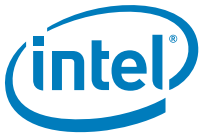
Receive queues can be enabled or disabled dynamically using the following procedure.

3.5.7.1.1.1 Enabling

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

3.5.7.1.1.2 Disabling

- Disable the routing of packets to this queue by re-configuring the Rx filters.
- 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 integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller completes the current buffer write. If the packet spreads over more than one data buffer, all subsequent buffers are not written.
- The integrated 10 GbE LAN controller clears the *RXDCTL.ENABLE* bit only after all pending memory accesses to the descriptor ring are done. The software device driver should poll this bit before releasing the memory allocated to this queue.
- Once the *RXDCTL.ENABLE* bit is cleared the software device driver should wait an additional amount of time (~100 μs) before releasing the memory allocated to this queue.



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

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

3.5.7.2 RSC Enablement

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

3.5.7.2.1 RSC Global Setting

- Enable global CRC stripping via the HLREG0 register (hardware default setting).
- Software device driver should set the *RDRXCTL.RSCACKC* bit that forces RSC completion on any change of the *ACK* bit in the Rx packet relative to the RSC context.
- The *SRRCTL[n].BSIZEHEADER* (header buffer size) bit must be larger than the packet header (even if header split is not enabled). A minimum size of 128 bytes for the header buffer addresses this requirement.

3.5.7.2.2 RSC Per Queue Setting

- Enable RSC and configure the maximum allowed descriptors per RSC by setting the *MAXDESC* and *RSCEN* fields in the RSCCTL[n] register.
- Use a non-legacy descriptor type by setting the *SRRCTL[n].DESCTYPE* bit to non-zero values.
- TCP header recognition: the *PSR_type4* bit in the PSRTYPE[n] registers should be set.
- The *SRRCTL[n].BSIZEPACKET* (packet buffer size) must be 2 KB at minimum.
- Interrupt setting:
  - Interrupt moderation must be enabled by setting the *EITR[n].ITR_INTERVAL* bit to a value greater than zero. The *ITR Interval* bit must be larger than the *RSC Delay* field described later. Note that if the *CNT_WDIS* bit is cleared (write enable), the ITR *Counter* bit should be set to 0b.
  - The *RSC_DELAY* field in the *GPIE* register should be set to the expected system latency descriptor write-back cycles. 4 to 8 μs should be sufficient in most cases. If software sees cases where RSC did not complete as expected (following EITR interrupt assertion), then the *RSC Delay* field might need to be increased.
  - Map the relevant Rx queues to an interrupt by setting the relevant IVAR registers.

3.5.7.3 Flow Director Initialization

Flow Director initialization is described in Section 3.5.7.3.

3.5.8 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, and DTXTCPFLGH.
- Refer to the *TIPG* description in Section 2.8.3 for more details.
- Set *RTTDCS.ARBDIS* to 1b.

- Program the DTXMXSZRQ, TXPBSIZE, TXPBTHRESH, MTQC, and MNGTXMAP registers according to virtualization mode.
- Clear *RTTDCS.ARBDIS* to 0b.
- Legacy software device drivers that uses queue zero and assumes it is enabled by default, should at this stage, set queue zero parameters as described in the section that follows.
- Enable the transmit path by setting the *DMATXCTL.TE* bit.

3.5.8.1 Transmit Queues Enable

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

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

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

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

3.5.8.1.1 Dynamic Enabling and Disabling of Transmit Queues

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

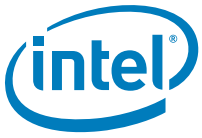
Enabling:

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

Disabling:

1. Stop storing packets for transmission in this queue.
2. The completion of the last transmit descriptor must be visible to software in order to guarantee that packets are not lost in step 5 (Section 3.5.8). 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.



3.5.9 Virtualization Initialization Flow

3.5.9.1 VMDq Mode

3.5.9.1.1 Global Filtering and Offload Capabilities

- Select one of the VMDQ pooling and queuing methods
  - Pool selection can be based on MAC/VLAN, and optionally E-tag. The filtering mode is defined using the *PFVTCTL.POOLING_MODE* field.
- A queue in a poll can be based on RSS. Potential values for *MRQC.Multiple Receive Queues Enable* to set VMDq modes are 1000b to 1111b.
- 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.Note that software could read these registers and then write them back with the same content.

3.5.9.1.2 Mirroring Rules

For each mirroring rule to be activated:

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

3.5.9.1.3 Security Features

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

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

3.5.9.1.4 Per-pool Settings

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

- Associate the unicast Ethernet MAC address of the VM by enabling the pool in the MPSAR registers.
- If all the Ethernet MAC addresses are used, the Unicast Hash Table (PFUTA) can be used. Pools servicing VMs whose address is in the hash table should be declared as so by setting the *PFVML2FLT.ROPE* bit. Packets received according to this method didn't pass perfect filtering and are indicated as such.
- Enable the pool in all RAH/RAL registers representing the multicast Ethernet MAC addresses this VM belongs to.
- If all the Ethernet MAC addresses are used, the Multicast Hash Table (MTA) can be used. Pools servicing VMs using multicast addresses in the hash table should be declared as so by setting the

*PFVML2FLT.ROMP*E 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 un-tagged packets via *PFVML2FLT.AUPE*.
- Enable the pool in each of the PFVLVF and PFVLVFB registers this VM belongs to.
- A VM might be set to receive it's own traffic in case the source and the destination are in the same pool via the *PFVMTXSW*.LLE bit.
- Whether VLAN header and CRC should be stripped from the packet. Note that even if the CRC is kept, it might not match the actual content of the forwarded packet, because of other offloading applications such as VLAN strip.
  - A striped VLAN might also be hidden from the VM.
- Set which header split is required via the PSRTYPE register.
- In RSS mode, define if the pool uses RSS via the proper *MRQC.MRQE* mode.
  - Enable the pool in the PFVFRE register to enable Rx filtering.
  - To enable multiple Tx queues, Set MTQC.
  - Enable the pool in the PFVFTE register to enable Tx filtering.
- Enable Rx and Tx queues as described in Section 3.5.7 and Section 3.5.8.
- 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 enable 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.

3.5.9.2 IOV Initialization

3.5.9.2.1 PF Driver Initialization

The PF driver is responsible for the link setup and handling of all the filtering and offload capabilities for all the VFs as described in Section 3.5.9.1.1 and the security features as described in Section 3.5.9.1.3. It should also set the bandwidth allocation per transmit queue for each VF as described in Section 3.5.9.

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

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.

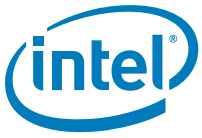
PF enables VF traffic via the PFVFTE and PFVFRE registers after all VF parameters are set as defined in Section 3.5.9.1.4.

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

3.5.9.2.1.1 VF Specific Reset Coordination

After the PF driver receives an indication of a VF FLR via the PFVFLREC register, it should enable the receive and transmit for the VF only once the device is programmed with the right parameters as defined in Section 3.5.9.1.4. The receive filtering is enabled using the PFVFRE register and the transmit filtering is enabled via the PFVFTE register.

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



3.5.9.2.2 VF Driver Initialization

At initialization, after the PF indicated that the global initialization was done via the VFMailbox.RSTD bit, the VF driver should communicate with the PF, either via the mailbox or via other software mechanisms to assure that the right parameters of the VF are programmed as described in Section 3.5.9.1.4. The PF driver might then send an acknowledge message with the actual setup done according to the VF request and the IT policy.

The VF driver should then setup the interrupts and the queues as described in Section 3.5.5, Section 3.5.7, and Section 3.5.8.

3.5.9.2.3 Full Reset Coordination

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

The procedure is as follows:

One of the following reset cases takes place:

- Power Good Reset
- Integrated I/O reset
- D3hot-to-D0 transition
- FLR
- Software reset by the PF

The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller 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.

3.5.10 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 shared SPI Flash 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 integrated 10 GbE LAN controller provides the Alternate Ethernet MAC Address structure in the shared SPI Flash to store the original MAC addresses. This structure is described in Section 4.3.7.

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.

The integrated 10 GbE LAN controller 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

3.5.10.1 LAN MAC Address Restore

The integrated 10 GbE LAN controller restores the LAN MAC addresses stored in the Alternate Ethernet MAC Address structure to the regular LAN MAC address location (see Section 4.4.6.2) if the following conditions are met:

1. The *restore MAC address* bit in the *Common Firmware Parameters* shared SPI Flash word is set.
2. The value in word 0x37 is not 0xFFFF.
3. The MAC address set in the regular MAC address location is different than the address stored in the Alternate Ethernet MAC Address structure.
4. The addresses stored in the Alternate Ethernet MAC Address structure are valid (not all zeros or all ones).

If the value at word 0x37 is valid, but the MAC addresses in the Alternate MAC structure are not valid (0xFFFFFFFF or all zeros) and *restore MAC address* shared SPI Flash bit is set, the regular MAC address is backed up in the Alternate MAC structure.

3.5.10.2 Restore Reporting

If after the power up cycle, the alternate LAN address equals the LAN Factory MAC addresses (either they were restored by the internal firmware or they were originally equal), the *FWSM.Factory MAC address restored* bit is set. This bit is common to all ports.

3.6 Access to Shared Resources

Part of the resources in the integrated 10 GbE LAN controller are shared between several software entities — namely the software device drivers of the two ports and internal firmware. In order to avoid contentions, a software device driver that needs to access one of these resources should use the *Gaining Control of Shared Resource by Software* flow described in Section 9.8.4 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:

- Shared SPI Flash
- PHY 0 and PHY 1 registers
- MAC CSRs.
- I^2C interface of each port
- MDIO interface of each port

4.0 Power Management and Delivery

This section defines how power management is implemented in the integrated 10 GbE LAN controller.

4.1 Power Management

4.1.1 Integrated 10 GbE LAN Controller Power States

The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller supports a Dr state that is entered when its integrated I/O reset signals are asserted (including the D3cold state).

Figure 4.1 shows the power states and transitions between them.

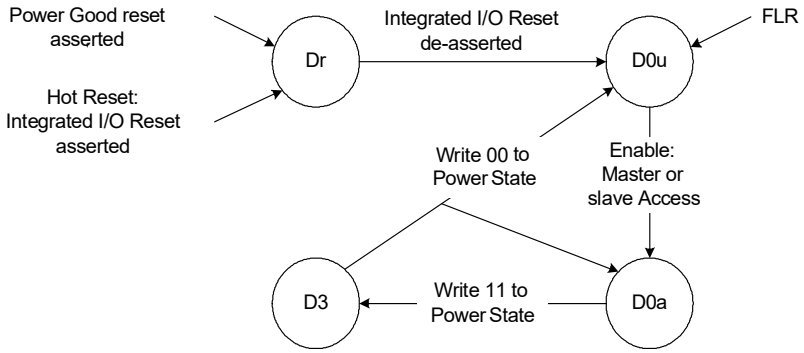


Figure 4.1. Power Management State Diagram

4.1.2 Auxiliary Power Usage

The integrated 10 GbE LAN controller uses the *AUX_PWR* indication that auxiliary power is available to the controller, and therefore advertises D3cold wake up support in the *PMC.PME_Support* field and sets the *Aux_Power_Detected* bit in the PCIe capability structure *Device Status* Register. The AUX_PWR pin is a soft strap that is loaded at Power On Reset (POR). The amount of power required for the function, which includes the entire Network Interface Card (NIC), is advertised in the Power Management Data register, which is loaded from the shared SPI Flash.

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 integrated 10 GbE LAN controller. 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 Good Reset.The integrated 10 GbE LAN controller 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).

According to the following settings, the integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller 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.

4.1.3 Power States

4.1.3.1 D0uninitialized State

The D0u state is a low-power state used after the integrated I/O interface is de-asserted following power-up (cold or warm) or on D3 exit.

When entering D0u, the integrated 10 GbE LAN controller disables wake ups.

4.1.3.1.1 Entry to a D0u State

D0u is reached from either the Dr state (on de-assertion of integrated I/O interface) or the D3hot state (by configuration software writing a value of 00b to the *Power State* field of the PCI PM registers).

De-asserting the integrated I/O interface means that the entire state of the device is cleared, other than sticky bits. State is re-loaded from the shared SPI Flash. Once this is done, configuration software can access the device.

On a transition from D3-to-D0u state, the integrated 10 GbE LAN controller requires that software perform a full re-initialization of the function excluding its PCI configuration space which is kept while in D3.

4.1.3.2 D0active State

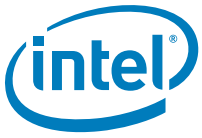
Once memory space is enabled, the integrated 10 GbE LAN controller enters an active state. It can transmit and receive packets if properly configured by the software device driver. The PHY is enabled or re-enabled by the software device driver to operate/auto-negotiate to full line speed/power if not already operating at full capability. Any APM wake up previously active remains active. The software device driver can deactivate ACPI wake up by writing to the Wake Up Control (WUC) register and APM wake up by writing to the General Receive Control (GRC) register, or activate other wake up filters by writing to the Wake Up Filter Control (WUFC) register.

4.1.3.2.1 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.

4.1.3.3 D3 State (PCI-PM D3hot)

The integrated 10 GbE LAN controller transitions to D3 when the system writes a 11b to the *Power State* field of the PMCSR. Any wake-up filter settings that were enabled before entering this reset state are maintained. Upon transition to D3 state, the integrated 10 GbE LAN controller clears the *Memory*



Access Enable and *I/O Access Enable* bits of the PCI Command register, which disables memory access decode. In D3, the integrated 10 GbE LAN controller 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 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 the integrated I/O interface assertion.

4.1.3.3.1 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 transitioning from D0 to the D3 state, the software device driver disables scheduling of further tasks to the integrated 10 GbE LAN controller; 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 4.1.3.3.2. If wake-up capability is needed, the software device driver should set up the appropriate wake-up registers and the system should write a 1b to the *PME_En* bit of the PMCSR prior to the transition to D3.

If all PCI functions are programmed into the D3 state the integrated 10 GbE LAN controller suspends scheduling of new TLPs and waits for the completion of all previous TLPs it has sent. The integrated 10 GbE LAN controller 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 software device driver might disable one of the LAN ports (LAN disable) and/or the PHY(s) auto-negotiates network link(s) to a lower speed (if supported by the network interface).

4.1.3.3.2 Master Disable

System software might disable master accesses on the integrated I/O interface link by either clearing the *PCI Bus Master* bit or by bringing the function into a D3 state. From that time on, the integrated 10 GbE LAN controller must not issue master accesses for this function. Due to the full-duplex nature of the integrated I/O interface and the pipelined design in the integrated 10 GbE LAN controller, 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 integrated I/O interface link after its *Master Enable* bit is cleared (or after entry to D3 state).

Two configuration bits are provided for the handshake between the device function and its software device driver:

- *PCIe Master Disable* bit in the Device Control Register (CTRL) register — When the *PCIe Master Disable* bit is set, the integrated 10 GbE LAN controller blocks new master requests by this function. The integrated 10 GbE LAN controller then proceeds to issue any pending requests by this function. This bit is cleared on master reset all the way to software reset to allow master accesses.
- *PCIe Master Enable Status* bits in the Device Status register — Cleared by the integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller clears the *PCIe Master Enable Status* bit:

— Master requests by the transmit and receive engines

— All pending completions to the integrated 10 GbE LAN controller are received.

Note: The software device driver disables any reception to the Rx queues as described in Section 3.5.7.1.1. Then the software device driver sets the *PCIe Master Disable* bit when notified of a pending master disable (or D3 entry). The integrated 10 GbE LAN controller then blocks new requests and proceeds to issue any pending requests by this function. The software device driver then reads the change made in the *PCIe Master Disable* bit and then polls the *PCIe Master Enable Status* bit. Once the bit is cleared, it is guaranteed that no requests are pending from this function.

The software device driver might time out if the *PCIe Master Enable Status* bit is not cleared within a given time. Examples for cases that the integrated 10 GbE LAN controller might not clear the *PCIe Master Enable Status* bit for a long time are cases of flow control, link down, or DMA completions not making it back to the DMA block. In these cases, the software device driver should check that the *Transaction Pending* bit (bit 5) in the Device Status register in the PCI configuration space is cleared before proceeding. Intel's device driver software times-out at ~800 µs.The *PCIe Master Disable* bit must be cleared to enable master request to the integrated I/O interface link. This bit should be cleared through reset.

4.1.3.4 Dr State (D3 Cold)

Transition to Dr state is initiated on several occasions:

- On system power up - Dr state begins with the assertion of Power Good Reset and ends with de-assertion of integrated I/O reset.
- On transition from a D0a state- During operation the system might assert the integrated I/O reset at any time. In an ACPI system, a system transition to the G2/S5 state causes a transition from D0a to Dr state. The transition can be orderly (such as user selected a shut down operating system option), in which case the software device driver might have a chance to intervene. Or, it might be an emergency transition (like power button override), in which case, the software device driver is not notified.
- On transition from a D3 state - The system transitions the device into the Dr state by asserting the Integrated I/O Reset.

Any wake-up filter settings that were enabled before entering this reset state are maintained.

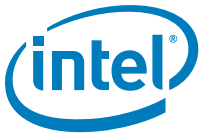
The system might maintain the Integrated I/O Reset asserted for an arbitrary time. The de-assertion (rising edge) of the Integrated I/O Reset causes a transition to D0u state.

While in Dr state, the integrated 10 GbE LAN controller 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.

4.1.3.4.1 Entry to Dr State at Power On

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

- Asserting Power Good Reset. Device power is kept to a minimum by keeping the PHYs in low power.
- The shared SPI Flash is then read and determines device configuration.
- If the *APM Enable* bit in the shared SPI Flash's Control Word 3 is set then APM wake up is enabled (for each port independently).
- If the *MNG Enable* bit in the shared SPI Flash word is set, pass-through manageability is not enabled.



- Each of the LAN ports can be enabled if required for WoL or manageability.In such a case, the PHY might auto-negotiate to a lower speed on Dr entry (see Section 4.2.2).
- The integrated I/O interface link is not enabled in Dr state following system power up (since the Integrated I/O Reset is asserted).

4.1.3.4.2 Transition to Dr (D3Cold) State

When the IP is in D3 or D0a state the transition to Dr state begins with a dedicated message.

Upon the reception of this message the integrated 10 GbE LAN controller performs the following operations:

- Suspends the integrated I/O interface scheduling of new TLPs and waits for the completion of all previous TLPs it has sent.
- Responds to all pending integrated I/O interface non-posted requests.

Once previous steps complete, the integrated 10 GbE LAN controller sends out a dedicated message to indicate to the SoC that it is ready for reset.

- After the response is sent, the integrated 10 GbE LAN controller is ready to transition to Dr state

To reduce power consumption, if any of manageability, APM wake or PCI-PM PME[1] is enabled, the PHY might auto-negotiate to a lower link speed on D0a to Dr transition (see Section 4.2.2).

At this point the IP is ready to have its integrated I/O reset asserted.

4.1.3.4.3 Dr Disable Mode

4.1.3.4.3.1 DR (D3Cold) to DR Disable

The integrated 10 GbE LAN controller enters Dr Disable mode on transition to D3cold when it does not need to maintain any functionality. The conditions to enter Dr Disable state are:

- The IP is in Dr state (all LAN functions are in Dr state)
  - Integrated I/O Reset was asserted -OR- PFs are disabled
  - APM WOL is inactive for all LAN functions
  - GRC<APME> = 0
  - ACPI PME is disabled for all PCI functions
  - PMCSR<PME_En> = 0
- Manageability is disabled for all LAN functions
  - Based on an NVM configuration read by firmware
    - Common Firmware Parameters<Manageability Mode> = 0x0 (None)

4.1.3.4.3.2 DR Disable at Power On

The integrated 10 GbE LAN controller may also enter Dr disable mode by reading the shared SPI Flash while already in Dr state. The usage model for this later case is on system power up when manageability and wake up are not required.

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.

Once the integrated 10 GbE LAN controller enters Dr state on power-up, the shared SPI Flash is read. If the shared SPI Flash contents determine that the conditions to enter Dr Disable are met, the integrated 10 GbE LAN controller then enters this mode (assuming that the Integrated I/O Reset is still asserted).

Exit from Dr disable is through de-assertion of the Integrated I/O Reset.

4.2 Network Interfaces Power Management

The integrated 10 GbE LAN controller transitions any of the network interfaces into a low-power state in the following cases:

- The respective LAN function is in LAN disable mode.
- The integrated 10 GbE LAN controller 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 this behavior:

- If manageability is disabled (*MNG Enable* bit in the shared SPI Flash is cleared), then LAN ports are not allocated for manageability.
- If manageability is enabled:
  - Power-up — Following an shared SPI Flash 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.

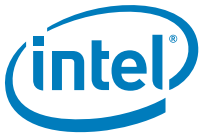
Enabling a port as a result of the previous behaviors cause an internal reset of the port including its associated PHY.

When a network interface is in low-power state, the integrated 10 GbE LAN controller MAC asserts an internal signals to notify the integrated PHY that it might power down as well. Alternatively, the integrated 10 GbE LAN controller might assert an internal signal to trigger LPLU to re-negotiation of link to a lower link speed. When working with an external PHY, the software device driver should power down the PHY using the dedicated SDPs or the side-band management interface.

4.2.1 PHY Power-Down State

Each Integrated 10 GbE LAN Controller 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 strap**: Each PHY is disabled if the associated LAN disable strap indicates that the port should be disabled.
3. ***LAN PCI Disable* bit in shared SPI Flash**: A single LAN port can also be disabled through shared SPI Flash settings. If the *LAN PCI Disable* bit is set in shared SPI Flash 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 shared SPI Flash 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 might be powered down.

When the integrated 10 GbE LAN controller is completely powered down (Dr state), the PHYs might reach a deeper power saving mode, however; they should retain all shared SPI Flash loaded settings. When the PHY exits power down, it re-initializes all analog functions.

4.2.2 Low Power Link Up (LPLU)

Normal PHY speed negotiation drives to establish a link at the highest possible speed. The PHY might support 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. In SGMII, electrical mode LPLU must be disabled.

4.2.3 Energy Efficient Ethernet (EEE)

The integrated 10 GbE LAN controller enters EEE Low Power Idle (LPI) mode on transmit each time the integrated 10 GbE LAN controller detects no data is scheduled for transmission.

EEE LPI mode, defined in IEEE802.3 Clause 78, enables power saving by switching off part of the PHY functionality when no data needs to be transmitted or/and received. A decision on whether the integrated 10 GbE LAN controller transmit path should enter LPI mode or exit LPI mode is done according to a need to transmit. Information on whether a link partner has entered LPI mode is detected by PHY and used for power saving in the receive circuitry.

When no data needs to be transmitted, a request to enter transmit LPI is issued on the internal XGMII Tx interface causing the PHY to transmit sleep symbols for a predefined period of time followed by a quite period. During LPI, the PHY periodically transmits refresh symbols that are used by the link partner to update adaptive filters and timing circuits in order to maintain link integrity. This quiet-refresh cycle continues until transmitting normal inter-frame encoding on the internal XGMII Tx interface. The PHY communicates to the link partner the move to link active state by sending wake symbols for a predefined period of time. The PHY then enters a normal operating state where data or idle symbols are transmitted.

In the receive direction, entering LPI mode is triggered by receiving sleep symbols from the link partner. This signals that the link partner is about to enter LPI mode. After sending the sleep symbols, the link partner ceases transmission. When the link partner initiates the move to LPI, the PHY indicates assert low power idle on the internal XGMII Rx interface and the PHY receiver disables functionality to reduce power consumption.

Figure 4.2 and Table 4.1 show the general principles of EEE LPI operation on the Ethernet link.

Figure 4.2. EEE Operation

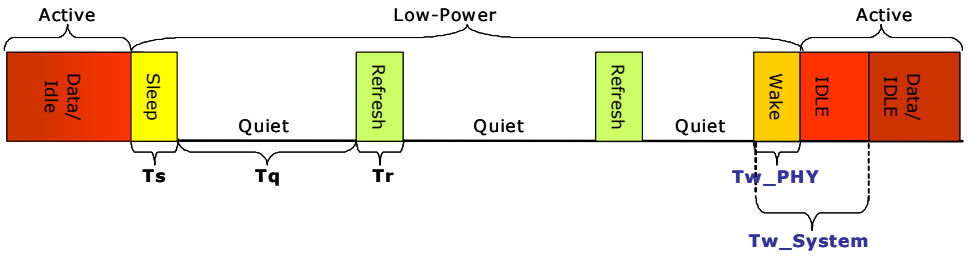


Table 4.1. EEE Parameters

Parameter	Description
Sleep Time (Ts)	Duration PHY sends sleep symbols before going quiet.
Quiet Duration (Tq)	Duration PHY remains quiet before it must wake for refresh period.
Refresh Duration (Tr)	Duration PHY sends refresh symbols for timing recovery and coefficient synchronization.
PHY Wake Time (Tw_PHY)	Minimum duration PHY takes to resume to active state after decision to wake.
Receive System Wake Time (Tw_System_rx)	Wait period where no data is expected to be received to give the local receiving system time to wake up.
Transmit System Wake Time (Tw_System_tx)	Wait period where no data is transmitted to give the remote receiving system time to wake up.

4.2.3.1 Conditions to Enter EEE Tx LPI

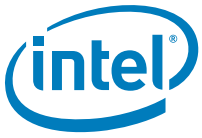
In the transmit direction entry into EEE LPI mode of operation is triggered when one of the following conditions exist:

1. No transmission is pending, management does not need to transmit, the internal transmit buffer is empty, and the *EEER.TX_LPI_EN* bit is set to 1b.
2. If the *EEER.TX_LPI_EN* bit is set to 1b and a XOFF flow control packet is received from the link partner, the integrated 10 GbE LAN controller moves the link into the Tx LPI state for the pause duration even if a transmission is pending.
3. When *EEER.Force_TLPI* is set (even if *EEER.TX_LPI_EN* is cleared).
   — If *EEER.Force_TLPI* is set in mid-packet, the integrated 10 GbE LAN controller completes the packet transmission and then moves Tx to LPI.

When one of the previous conditions to enter Tx LPI state are detected assert LPI is transmitted on the internal xxMII interface and the the integrated 10 GbE LAN controller PHY transmits sleep symbols on the network interface to communicate to the link partner entry into Tx LPI link state. After sleep symbols transmission, the PHY immediately enters the low power quiet mode. In this state the PHY periodically transitions between quiet link state, where link is idle, to sending refresh symbols until a request to transition link back to normal (active) mode is transmitted on the internal xxMII Tx interface (See Figure 4.2).

Note: Initial EEER.TX_LPI_EN configuration is loaded from shared SPI Flash.

Note: EEE should always be disabled while in SGMII link mode.



Note: IEEE802.3 Ethernet LPI enables each link direction to enter sleep, refresh or wake states asymmetric from the other direction.

Note: EEE LPI status of a Integrated 10 GbE LAN Controller port can be found in the EEE_STAT register.

4.2.3.2 Transition from Tx LPI to Active Link State

The integrated 10 GbE LAN controller exits Tx LPI link state and transitions the link into active link state when none of the conditions defined in Section 4.2.3.1 exist. To transition into active link state, the integrated 10 GbE LAN controller transmits:

1. Normal inter-frame encoding on the internal xxMII Tx interface for a pre-defined link rate dependent period time of Tw_sys_tx-min (defined by the *EEE_SU* register and IEEE802.3az clause 78.5). As a result, the PHY transmits wake symbols for a Tw_phy duration followed by idle symbols.
2. If the Tw_System_tx duration defined in the *EEER.Tw_system* field is longer than Tw_sys_tx-min, the integrated 10 GbE LAN controller continues transmitting the inter-frame encoding on the internal xxMII interface until the time defined in the *EEER.Tw_system* field has expired, before transmitting the actual data. During this period the PHY continues transmitting idle symbols.

Note: When moving out of Tx LPI to transmit a 802.3x flow control frame, the integrated 10 GbE LAN controller waits for the Tw_sys_tx-min duration before transmitting the flow control frame. It should be noted that even in this scenario actual data is transmitted only after the Tw_System_tx time defined in the *EEER.Tw_system* field has expired.

4.2.3.3 EEE Auto-Negotiation

Auto-negotiation provides the capability to negotiate EEE capabilities with the link partner using the next page mechanism defined in IEEE802.3 Annex 73A. IEEE802.3 auto-negotiation is performed at power up, on command from software, upon detection of a PHY error or following link re-connection.

During the link establishment process, both link partners indicate their EEE capabilities using IEEE802.3 auto-negotiation. If EEE is supported by both link partners for the negotiated PHY type then the EEE function can be used independently in either direction.

EEE capabilities advertised during auto-negotiation can be modified via the Auto-negotiation EEE Advertisement Register in the backplane PHY registers.

4.2.3.4 EEE Link Level (LLDP) Capabilities Discovery

The integrated 10 GbE LAN controller supports LLDP negotiation via software using the EEE IEEE802.1AB Link Layer Discovery Protocol (LLDP) Type, Length, Value (TLV) fields defined in IEEE802.3az clause 78 and clause 79. LLDP negotiation enables negotiation of increased system wake time (Transmit T_w and Receive T_w) to enable improving system energy efficiency.

4.2.3.4.1 LLDP Negotiation Actions

Following negotiation of a new system wake time via EEE LLDP negotiation, the following fields and registers should be updated:

The EEER.Tw_system field with the negotiated Transmit T_w time value, to increase the duration where idle symbols are transmitted following a move out of EEE Tx LPI state before actual data can be transmitted.

— A value placed in *EEER.Tw_system* field does not effect transmission of flow control packets. Depending on the technology flow control packet, transmission is delayed following a move out of EEE Tx LPI state only by the minimum Tw_sys_tx time as defined in IEEE802.3az clause

78.5. In the integrated 10 GbE LAN controller, the minimum Tw_sys_tx time value is defined in the *EEE_SU* register together with time defined in IEEE802.3az clause 78.5. Value varies as a function of link rate and technology.

4.2.3.5 EEE Statistics

The integrated 10 GbE LAN controller supports reporting a number of EEE LPI Tx and Rx events via the RLPIC and TLPIC registers.

4.3 Wake Up

4.3.1 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. The NIC asserts the signal for approximately 50 ms to signal a wake up.

At power up, the integrated 10 GbE LAN controller reads the *APM Enable* bit from the shared SPI Flash into the *APM Enable* (*APME*) bits of the GRC register. This bit control the enabling of *APM Wakeup*.

When *APM Wakeup* is enabled, the integrated 10 GbE LAN controller checks all incoming packets for Magic Packets.

Once the integrated 10 GbE LAN controller receives a matching magic packet, it:

- Asserts the WAKE signal so that clock and power is resumed to the CPU when it is in Sx or possibly reset when it is in S0[1].
- Sets the *PME_Status* bit in the *PMCSR*.
- Stores the first 128 bytes of the packet in the Wake Up Packet Memory (WUPM).
- Sets the Magic Packet *Received* bit in the Wake up Status (WUS) register.
- Sets the packet length in the Wake up Packet Length Register (WUPL).
- Once the CPU is powered on and the integrated I/O reset is de-asserted, send a PME message.

The integrated 10 GbE LAN controller maintains the first magic packet received in the Wake Up Packet Memory (WUPM) until the software device driver writes a 0b to the Magic Packet Received *MAG* bit in the Wake Up Status Register (WUS), hence the software device driver has to read the packet from WUPM prior to clearing the WUS indication.

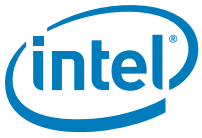
APM Wakeup event might be issued in any[2] power state. It is disabled if a subsequent shared SPI Flash read results in the *APM Wake Up* bit being cleared or if the software explicitly writes a 0b to the *APM Wake Up* (*APM*) bit of the GRC register.

4.3.2 ACPI Power Management Wake Up

The integrated 10 GbE LAN controller supports ACPI power management based wake up. It can generate system wake-up events from three sources, regardless of the power state:

- Receiving a Magic Packet.
- Receiving a network wake up packet.

1. Reset on LAN due to Magic Packet is a platform feature. The integrated 10 GbE LAN controller performs the same flow as Wake on LAN.
2. Wake on Lan in D0 (S0) might be used for Reset on LAN at the platform level.



- Detecting a link change of state.
- Firmware reset event.

Activating ACPI power management wake up requires the following steps:

- The operating system (at configuration time) writes a 1b to the *PME_En* bit of the PMCSR (PMCSR.8).
- The software device driver clears all pending wake-up status in the WUS register by writing 1b to all the status bits.
- The software device driver programs the WUFC register to indicate the packets it needs to wake up and supplies the necessary data to the IPv4/v6 Address Table (IP4AT, IP6AT), Flexible Host Filter Table (FHFT). It can also set the *Link Status Change Wake Up Enable* (*LNKC*) bit in the WUFC register to cause wake up when the link changes state.
- Once the integrated 10 GbE LAN controller wakes the system, the software device driver needs to clear the WUS and WUFC registers until the next time the system goes to a low-power state with wake up.

Normally, after enabling ACPI wake up, the operating system writes (11b) to the lower two bits of the PMCSR to put the integrated 10 GbE LAN controller into low-power mode, and once entering D0, OS disables ACPI wake up.

Once wake up is enabled, the integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller:

- Sets the *PME_Status* bit in the PMCSR.
- If the *PME_En* bit in the *PMCSR* is set and the device is in Dr state or *PCI_GLBL_CNF.WAKE_PIN_EN* is set in all states, asserts GbE wake pin.
- In non Dr state or when exiting Dr, a PM_PME message is also issued.
- Stores the first 128 bytes of the packet in the Wake Up Packet Memory (WUPM) register. Sets one or more of the *Received* bits in the WUS register. Note that the integrated 10 GbE LAN controller sets more than one bit if a packet matches more than one filter.
- Sets the packet length in the Wake Up Packet Length Register (WUPL).

If enabled, a link state change wake up causes similar results, setting *PME_Status*, asserting GbE wake pin and setting the *LNKC* bit in the WUS register when the link goes up or down.

The GbE wake pin 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.

If enabled, a firmware reset causes similar results, setting *PME_Status*, asserting the GbE wake pin and setting the *FW_RST_WK* bit in the *WUS* register when the firmware is reset

The GbE wake pin 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. the WUS register.

4.3.3 Wake-Up Packets

The integrated 10 GbE LAN controller 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.

4.3.3.1 Pre-Defined Filters

The following packets are supported by the integrated 10 GbE LAN controller'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 explanation of each filter includes a table listing 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.

4.3.3.1.1 Directed Packets

Unicast — The integrated 10 GbE LAN controller generates a wake-up event after receiving any packet whose destination address matches one of the 128 valid programmed receive addresses if the *Directed Exact Wake Up Enable* bit is set in the WUFC register (WUFC.EX).

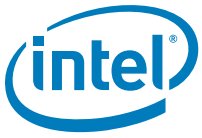
For multicast packets, the upper bits of the incoming packet's destination address index a bit vector, the Multicast Table Array (MTA) that indicates whether to accept the packet. If the *Directed Multicast Wake Up Enable* bit set in the WUFC register (WUFC.MC) and the indexed bit in the vector is one then the integrated 10 GbE LAN controller generates a wake-up event. The exact bits used in the comparison are programmed by software in the *Multicast Offset* field of the Multicast Control register (MCSTCTRL.MO).

If the *Broadcast Wake Up Enable* bit in the WUFC register (WUFC.BC) is set, the integrated 10 GbE LAN controller generates a wake-up event when it receives a broadcast packet.

4.3.3.1.2 Magic Packet

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 shared SPI Flash, the integrated 10 GbE LAN controller starts up with the Receive Address Register 0 (RAH0, RAL0) loaded from the shared SPI Flash. This enables the integrated 10 GbE LAN controller to accept packets with the matching IEEE address before the software device 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).



4.3.3.1.3 ARP/IPv4 Request Packet

The integrated 10 GbE LAN controller supports receiving ARP request packets for wake up if the *ARP* bit is set in the WUFC register. Four IPv4 addresses are supported, which are programmed in the IPv4 Address Table (IP4AT). A successfully matched packet must pass L2 address filtering, a Protocol Type of 0x0806, an ARP OpCode of 0x01, and one of the four programmed IPv4 addresses. The integrated 10 GbE LAN controller 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.

4.3.3.1.4 Directed IPv4 Packet

The integrated 10 GbE LAN controller supports receiving directed IPv4 packets for wake up if the *IPV4* bit is set in the WUFC register. Four IPv4 addresses are supported that are programmed in the IPv4 Address Table (IP4AT). A successfully matched packet must pass L2 address filtering, a Protocol Type of 0x0800, and one of the four programmed IPv4 addresses. The integrated 10 GbE LAN controller also handles directed IPv4 packets that have VLAN tagging on both Ethernet II and Ethernet SNAP types.

4.3.3.1.5 Directed IPv6 Packet

The integrated 10 GbE LAN controller supports receiving directed IPv6 packets for wake up if the *IPV6* bit is set in the WUFC register. One IPv6 address is supported and it is programmed in the IPv6 Address Table (IP6AT). A successfully matched packet must pass L2 address filtering, a Protocol Type of 0x0800, and the programmed IPv6 address. The integrated 10 GbE LAN controller also handles directed IPv6 packets that have VLAN tagging on both Ethernet II and Ethernet SNAP types.

4.3.3.2 Flexible Filter

The integrated 10 GbE LAN controller supports a total of eight host flexible filters. Each filter can be configured to recognize any arbitrary pattern within the first 128 bytes of the packet. To configure the flexible filter, software programs the required values into the Flexible Host Filter Table (FHFT_FILTER).

These contain separate values for each filter. Software must also enable the filter in the WUFC register, and enable the overall wake-up functionality must be enabled by setting *PME_En* in the PMCS register or the WUC register.

Structure of the Flexible Host Filter Table:

31 0	31 8	7 0	31 0	31 0
Reserved	Reserved	Mask [7:0]	DW 1	DW 0
Reserved	Reserved	Mask [15:8]	DW 3	DW 2
Reserved	Reserved	Mask [23:16]	DW 5	DW 4
Reserved	Reserved	Mask [31:24]	DW 7	DW 6

. . .

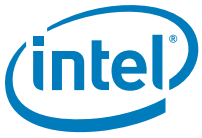
31 7	6 0	31 8	7 0	31 0	31 0
Reserved	Reserved	Reserved	Mask [127:120]	DW 29	DW 28
Reserved	Length	Reserved	Mask [127:120]	DW 31	DW 30

Each of the filters is allocated addresses as follows:

- Filter 0 — 0x09000 — 0x090FF
- Filter 1 — 0x09100 — 0x091FF
- Filter 2 — 0x09200 — 0x092FF
- Filter 3 — 0x09300 — 0x093FF
- Filter 4 — 0x09600 — 0x096FF
- Filter 5 — 0x09700 — 0x097FF
- Filter 6 — 0x09800 — 0x098FF
- Filter 7 — 0x09900 — 0x099FF

The following table lists the addresses used for filter 0.

Field	Dword	Address	Bit(s)	Initial Value
Filter 0 DW0	0	0x09000	31:0	X
Filter 0 DW1	1	0x09004	31:0	X
Filter 0 Mask[7:0]	2	0x09008	7:0	X
Reserved	3	0x0900C		X
Filter 0 DW2	4	0x09010	31:0	X
…				
Filter 0 DW30	60	0x090F0	31:0	X
Filter 0 DW31	61	0x090F4	31:0	X
Filter 0 Mask[127:120]	62	0x090F8	7:0	X
Length	63	0x090FC	6:0	X



Accessing the FHFT_FILTER registers during filter operation might result in a packet being mis-classified if the write operation collides with packet reception. It is therefore advised that the flex filters are disabled prior to changing their setup.

Once enabled, the flexible filters scan incoming packets for a match. If the filter encounters any byte in the packet where the mask bit is one and the byte 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 integrated 10 GbE LAN controller (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.

4.3.3.3 Wake-Up Packet Storage

The integrated 10 GbE LAN controller saves the first 128 bytes of the wake-up packet in its internal buffer, which can be read through the Wake Up Packet Memory (WUPM) after the system wakes up.

4.3.4 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 software device driver controls wake up and none of the Virtual Machines (VMs) has direct access to the wake-up registers. The wake-up registers are not replicated.

4.4 Protocol Offload (Proxying)

In prior operating system releases, ARP and IPv6 neighbor discovery messages were one of the possible wake-up types for the platform. ARP and IPv6 neighbor discovery packets are required to enable other network devices to discover the link layer address used by the system. Supporting these protocols while the host is in low-power state is fundamental to maintain remote network accessibility to the sleeping host. If the host does not respond, other devices in the network eventually are not able to send routable network traffic (such as IPv4 and IPv6) to the sleeping host.

1. Implementing ARP offload as defined in the Power Management specification on the NDIS Program Connect site. Specifically, the offload must respond to an ARP Request (operation = 1) by responding with an ARP Reply (operation = 2) as defined in RFC 826.
2. Implementing IPv6 NS offload as defined in Power Management specification on the NDIS Program, Connect site. Specifically, the offload must respond to an Neighbor Solicitation (operation = 135) by responding with an NS Advertisement (operation = 136) as defined in RFC 4861. Devices must support at least 2 NS offloads, each with up to 2 target IPv6 addresses
   a. When Neighbor Solicitation (NS) Protocol off load is enabled, the integrated 10 GbE LAN controller also supports protocol off load of up to two IPv6 Multicast-Address-Specific Multicast Listener Discovery (MLD) Queries (either MLDv1 or MLDv2) per function. In addition, the integrated 10 GbE LAN controller also responds to General MLD queries, used to learn which IPv6 multicast addresses have listeners on an attached link.
   — MLD protocol offload is supported when NS protocol offload is enabled so that IPv6 routers discover the presence of multicast listeners (that is, nodes wanting to receive multicast packets), for packets with the IPv6 NS solicited-node multicast address and continue forwarding these NS requests on the link.

— MLD protocol off load is supported for either MLD Multicast Listener Query packets or MLD Multicast Address and Source Specific Query packets that check for IPv6 Multicast Listeners with the solicited-node multicast address placed in the IPv6 destination address field of the IPv6 NS packets that are off-loaded by the integrated 10 GbE LAN controller.

— Responds to the IPv6 MLD queries, with the solicited-node multicast address placed in the IPv6 destination address field of the IPv6 NS packets that are off-loaded by the integrated 10 GbE LAN controller (as defined in RFC 2710 and RFC 3810).

4.4.1 Proxying Filters

The integrated 10 GbE LAN controller supports protocol offload (proxying) of packets using the embedded management controller. The relevant packet is directed to the MC using the pre-defined filters. Proxy filters are enabled if the corresponding bit in the *Proxying Filter Control* (*PROXYFC*) register is set to 1b.

The following packet types are detected per PF by the integrated 10 GbE LAN controller's pre-defined filters:

- General and directed ARP/IPv4 request packet
- General and directed NS IPv6 packet
- MLD IPv6 packet

Each of these packet type filters is enabled if the corresponding bit in the *Proxying Filter Control* (*PROXYFC*) register is set to 1b.

All the proxy filters are applied only to traffic that passes the L2 filtering described in Section 4.1.3.

Note: Packets for which the IP or TCP/UDP checksum is wrong are not considered as candidates for proxy.

4.4.1.1 Directed Packets

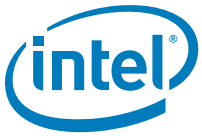
The integrated 10 GbE LAN controller forwards to proxying any packet whose destination address matches one of the 128 valid programmed receive addresses if the Directed Exact PROXY Enable bit is set in the PROXYFC register (*PROXYFC.EX*).

4.4.1.2 ARP/IPv4 Request Packet

The integrated 10 GbE LAN controller supports receiving ARP request packets for proxying if the ARP_*Directed* bit or the *ARP* bit is set in the *Proxying Filter Control* (*PROXYFC*) register.

- If the *Directed ARP* bit is set, a successfully matched packet must contain a broadcast/unicast MAC address, a protocol type of 0x0806, an ARP OpCode of 0x01 and the target IP address matches one of the four IPv4 addresses programmed in the *IPv4 Address Table* (*IP4AT*), in addition, the corresponding IPAV.V40, IPAV.V41, IPAV.V42 or IPAV.V43 bit should be set.
- If the *ARP* bit is set, a successfully matched packet must contain a broadcast MAC address or one of the unicast addresses configured for the host, a protocol type of 0x0806 and an ARP OpCode of 0x01.

The integrated 10 GbE LAN controller also handles ARP request packets that have VLAN tagging or double VLAN tagging on both Ethernet II and Ethernet SNAP types.



4.4.1.2.1 ARP Request Packet

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	D=(0/4/6/8)	Outer Tag (Outer VLAN or E-tag)	0x8100 **** 0x893F ******* *****	Skip	
12 + D	S = (0/4)	Possible VLAN Tag		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	Match IP4AT values or zero.
42 + S + D	18 - S - D	Padding	0x00	Ignore	Padding to 64 bytes.
60	4	CRC		Check	

4.4.1.2.2 ARP Response Packet

Offset	# of Bytes	Field	Value	Action
0	6	Destination Address		Copy from ARP request source address.
6	6	Source Address		Station address.
12	D=(0/4/6/8)	Outer Tag (Outer VLAN or E-tag)	0x8100 **** 0x893F ****** ******	Skip.
12	S = (0/4)	Possible VLAN Tag		Copy from ARP request.
12 + S + D	2	Type	0x0806	Constant (Copy from ARP request).
14 + S + D	2	Hardware Type	0x0001	Constant (Copy from ARP request).
16 + S + D	2	Protocol Type	0x0800	Constant (Copy from ARP request).
18 + S + D	1	Hardware Size	0x06	Constant (Copy from ARP request).
19 + S + D	1	Protocol Address Length	0x04	Constant (Copy from ARP request).
20 + S + D	2	Operation	0x0002	Constant.
22 + S + D	6	Sender Hardware Address		Station address.

Offset	# of Bytes	Field	Value	Action
28 + S + D	4	Sender IP Address		Target IP address from ARP request or valid IP address if target IP was zero.
32 + S + D	6	Target Hardware Address		Sender MAC address from ARP request.
38 + S + D	4	Target IP Address		Sender IP address from ARP request.
42 + S + D	18 - S - D	Padding	0x00	Padding to 64 bytes.
60	4	CRC		Calculate.

4.4.1.3 NS IPv6 Packet

In IPv6 networks, ICMPv6 neighbor solicitation and neighbor advertisement provides the address mapping of the IP address to a corresponding MAC address.

Machines that operate in IPv6 networks are sent an ICMPv6 neighbor solicitation and must respond with their link-layer (MAC) address in their ICMPv6 neighbor advertisement response. The solicitation can be for either the link-local, global, or a temporary IPv6 addresses.

Neighbor discovery messages have both an IPv6 header and the ICMPv6 header. The IPv6 header is a standard one, including the source and destination IP addresses. The network proxy off load does not support IPv6 neighbor discovery messages that also have IPv6 header extensions, these packets are silently discarded with no reply. The integrated 10 GbE LAN controller supports receiving IPv6 NS packets sent by a node to determine the link-layer address of a neighbor, or to verify that a neighbor is still reachable for wake up or proxying.

If the *NS* or *NS_DIRECTED* bits are set in the Proxying Filter Control (PROXYFC) register packet is sent to the MC for protocol offload.

- If the *NS directed* bit is set, a successfully matched packet must contain the station's MAC address (unicast or multicast), an Ethernet type of 0x86DD, a IPv6 header type of ICMPv6 (0x3A), a ICMPv6 type of 0x87 (NS), correct ICMPv6 checksum and the programmed IPv6 address in the IPv6 Address Table (IP6AT) must match the target IPv6 address. In addition, the corresponding *IPAV.V60, IPAV.V61, IPAV.V62* or *IPAV.V63* bit should be set.
- If the *NS* bit is set, a successfully matched packet must contain the station's MAC address (unicast or multicast), an Ethernet type of 0x86DD, a IPv6 header hype of ICMPv6 (0x3A), a ICMPv6 type of 0x87 (NS) and a correct ICMPv6 checksum.
- If the packet has an ICMPv6 checksum error, then it is considered as a proxy packet only if the *RXCSUM.ICMPV6XSUM* bit is cleared.

The integrated 10 GbE LAN controller also handles NS IPv6 packets that have VLAN tagging or double VLAN tagging on both Ethernet II and Ethernet SNAP types.

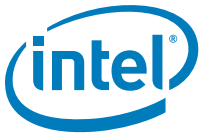
Figure 4.3. NS IPv6 Packet Flow

4.4.1.3.1 IPv6 Neighbor Solicitation Packet

Offset	# of bytes	Field	Value (hex)	Action	Comment
0	6	Destination Address		Compare	MAC Header – processed by main address filter.
6	6	Source Address		Skip	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN or E-tag)	0x8100 **** 0x893F ******* *****	Skip	
12+D	S=(0/4)	Possible VLAN Tag		Skip	
IPv6 header					
12+D+S	2	Type	0x86DD	Compare	IPv6
14+D+S	1	Version/ Traffic Class	0x6	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	0x3A	Check	ICMPv6
21+D+S	1	Hop Limit	0xFF	Compare	
22+D+S	16	Source Address		Ignore	Check if source address is undefined.
38+D+S	16	Destination Address		Ignore	
ICMPv6 header					
54+D+S	1	Type	0x87	Compare	Neighbors solicitation.
55+D+S	1	Code	0x0	Compare	
56+D+S	2	Checksum		Check	
58+D+S	4	Reserved	0x0000	Ignore	
62+D+S	16	Target IP Address	IP6AT	Compare	
78+D+S	1	Type	0x1	Compare	Possible source link layer address option (Should not appear if source address is undefined).
79+D+S	1	Length	0x1	Compare	
80+D+S	6	Link Layer Address		Skip	
86+D+S	4	CRC		Check	

4.4.1.3.2 IPv6 Neighbor Advertisement Packet

Offset	# of bytes	Field	Value (hex)	Action
0	6	Destination Address		Copy from ND packet.
6	6	Source Address		Station address.
12	D=(0/4/6/8)	Outer Tag (Outer VLAN or E-tag)	0x8100 **** 0x893F ******* *****	Skip



Offset	# of bytes	Field	Value (hex)	Action
12+D	S=(0/4)	Possible VLAN Tag		Copy from ND packet.
IPv6 header				
12+D+S	2	Type	0x86DD	Constant (Copy from ND packet).
14+D+S	1	Version/ Traffic Class	0x6	Constant (Copy from ND packet).
15+D+S	3	Traffic Class/Flow Label		Constant (Copy from ND packet).
18+D+S	2	Payload Length		
20+D+S	1	Next Header	0x3A	Constant.
21+D+S	1	Hop Limit	0xFF	Constant.
22+D+S	16	Source Address		relevant IPv6AT entry (ND target address).
38+D+S	16	Destination Address		Copy from ND packet source address If source address was undefined - send to All Nodes Multicast (FF02::1).
ICMPv6 header				
54+D+S	1	Type	0x88	Constant.
55+D+S	1	Code	0x0	
56+D+S	2	Checksum		Calculate.
58+D+S	4	Flags	0x60000000	Constant (solicited and override) if the source address was defined.
			0x20000000	Constant (override) if the source address was undefined.
62+D+S	16	Target IP Address	IP6AT	Same as source address.
78+D+S	1	Type	0x2	Target link layer address option.
79+D+S	1	Length	0x1	
80+D+S	6	Link Layer Address	From ND	
86+D+S	4	CRC		Calculate.

4.4.1.4 MLD IPv6 Packet

The integrated 10 GbE LAN controller supports receiving IPv6 MLD packets sent by an IPv6 router to discover the presence of multicast listeners (that is, nodes needs to receive multicast packets) on its directly attached links, and to discover specifically which multicast addresses are of interest to those neighboring nodes.

After receiving the following MLD packets (either MLDv1 as defined in RFC 2710 or MLDv2 as defined in RFC 3810):

1. Multicast Listener Query (ICMPv6 Type = decimal 130). Defined in MLDv1 and MLDv2.
2. Multicast Listener Report (ICMPv6 Type = decimal 131). Defined in MLDv1 and MLDv2.
3. Version 2 Multicast Listener Report Message (ICMPv6 Type = decimal 143). Defined in MLDv2 only.

If the *MLD* bit is set in the Proxying Filter Control (PFPM_PROXYFC) register packet is sent to firmware for protocol offload.

If the *MLD* bit is set, a successfully matched packet must contain the station's MAC address (unicast or multicast), a Ethernet type of 0x86DD, a IPv6 header type of ICMPv6 (0x3A), a ICMPv6 type of either 0x82 (130 decimal), 0x83 (131 decimal) or 0x8F (143 decimal) and a correct ICMPv6 checksum.

The integrated 10 GbE LAN controller also handles MLD IPv6 packets that have VLAN tagging on both Ethernet II and Ethernet SNAP types.

If the packet has an ICMPv6 checksum error, then it is considered as an MLD proxy packet only if the *RXCSUM.ICMPV6XSUM* bit is cleared.

4.4.2 Proxying and Virtualization

When operating in a virtualized environment, all proxying capabilities are managed by a single entity (such as the VMM or an IOVM). In an IOV architecture, the physical (PF) driver controls proxying and none of the VMs has direct access to the Proxying registers. The Proxying registers are not replicated per VF.

4.4.3 Protocol Offload (Proxying) Flow

The Proxying register set consists of the following Proxying registers:

1. *PROXYFC* - Proxying Filter Control Register.
2. *PROXYS* - Proxying Status Register.

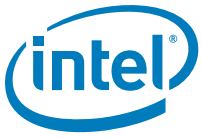
Proxying address filters:

1. MAC - (RAL/H) L2 address Table.
2. *IP4* - (IP4AT) IPv4 Address Table.
3. *IP6* - (IP6AT) IPv6 Address Table.

4.4.3.1 Protocol Offload Activation

To enable protocol offload, the software device driver should implement the following steps:

1. Clear all pending proxy status bits in the Proxying Status (PROXYS) register.
2. Program the Proxying Filter Control (PROXYFC) register to indicate type of packets that should be forwarded to manageability for proxying and program the necessary data to the IPv4/v6 Address Table (IP4AT, IP6AT).
3. Set the *WUFC.FW_RST_WK* bit to 1b to initiate a wake if firmware reset was issued and proxying information was lost.
4. Take ownership of the management host interface semaphore (*SW_FW_SYNC.SW_MNG_SM* register bit) using the flow defined in Section 3.6 to send protocol offload information to firmware.
5. Read and clear the *FWSTS.FWRI* firmware reset indication bit.
   — If a firmware reset was issued as reported in the *FWSTS.FWRI* bit, the software device driver should clear the bit and then re-initialize the protocol offload list.
6. Verify that the *HICR.En* bit is set to 1b, which indicates that the shared RAM interface is available.
7. Write proxying information in the shared RAM interface located in addresses 0x15800-0x15EFF using the format defined in Section 9.7.4. All addresses should be placed in networking order.
8. Once information is written into the shared RAM software should set the *HICR.C* bit to 1b.
9. Poll the *HICR.C* bit until bit is cleared by firmware indicating that command was processed and verify that command completed successfully by checking that *HICR.SV* bit was set.
10. Read firmware response from the shared RAM to verify that data was received correctly.
11. Return to 7. if additional commands need to be sent to firmware.
12. Release management host interface semaphore (*SW_FW_SYNC.SW_MNG_SM* register bit) using the flow defined in Section 3.6.



13. Verify that a firmware reset was not initiated during the proxying configuration process by reading the *FWSTS.FWRI* firmware reset indication bit. If a firmware reset was initiated return to 1.
14. Set *PROXYFC.PPROXYE* bit to 1b and enable proxy.
15. The software device driver might clear the PROXYFC.PPROXYE bit, PROXYS and PROXYFC registers until the next time the system moves to a low power state with proxying enabled.

Once proxying is enabled by setting the *PROXYFC.PPROXYE* bit to 1b, the integrated 10 GbE LAN controller monitors incoming packets, first filtering them according to its standard address filtering method, then filtering them with all of the proxying filters enabled in the PROXYFC register. If a packet passes both the standard address filtering and at least one of the enabled proxying filters and does not pass any of the enabled wake-up filters, the integrated 10 GbE LAN controller:

1. Executes the relevant protocol offload for the packet and not forward the packet to the host.
2. Sets one or more bits in the Proxying Status (PROXYS) register according to the proxying filters matched.
   — Note that the integrated 10 GbE LAN controller sets more than one bit in the PROXYS register if a packet matches more than one filter.
3. Wakes the system and forwards a packet that matches the proxying filters but can't be supported to the host for further processing if configured to do so by the software device driver via the No Match command in the Set Firmware Proxying Configuration command using the shared RAM interface.

Notes:

1. When the integrated 10 GbE LAN controller is in D3 a packet that matches both one of the enabled proxying filters as defined in the PROXYFC register and one of the enabled wake-up filters as defined in the WUFC register only wakes up the system and protocol off load (proxying) does not occur.
2. Protocol offload is not executed for illegal packets with CRC errors or checksum errors and the packets are silently discarded.

4.4.3.2 Disabling Protocol Offload

3. When the integrated 10 GbE LAN controller transitions from D3-to-D0, the protocol offload is disabled. It can also be disabled by the software device driver using the following steps. It programs the proxying control and status registers:
   a. Clears PROXYS
   b. Clears the relevant bits in PROXYFC to disable the offload

4.5 DMA Coalescing

The integrated 10 GbE LAN controller supports DMA coalescing that enables synchronizing port activity and optimizes power consumption of memory and CPU usage. DMA coalescing is a device-based feature and when the conditions to enter DMA coalescing operating mode exists (as defined in Section 4.5.2.1), the integrated 10 GbE LAN controller:

- Stops initiation of any activity on the integrated I/O interface link. This includes all DMA transactions (if *DMACR.EN_MNG_IND* is set).
- Buffers data received from the Ethernet link in internal Rx buffer until the conditions defined in Section 4.5.2.2 to exit DMA coalescing exist.

4.5.1 DMA Coalescing Activation

Activating DMA coalescing functionality is done by the software device driver. The software device driver can enable or disable the device DMA coalescing functionality by clearing the *DMACR.DMAC_EN* bit.

To activate DMA coalescing functionality the following fields need to be programmed:

1. *DMCTH.DMACRXT* fields to set the per Rx packet buffer receive threshold that causes the integrated 10 GbE LAN controller to move out of DMA coalescing operating mode. When the amount of data in the receive packet buffer exceeds the receive threshold, the integrated 10 GbE LAN controller moves out of DMA coalescing operating mode. A programmed receive watermark should take into account the actual link speed and Latency Tolerance Reported (LTR) to avoid receive buffer overflow when DMA coalescing is enabled. See the LTR description in Section 4.6 for the*DMACR.DMACWT* field to define a maximum time-out value for:
   a. A receive packet to be stored in the internal receive buffer before the integrated 10 GbE LAN controller moves a packet to host memory.
   — Each time the integrated 10 GbE LAN controller enters DMA coalescing, an internal DMA coalescing watchdog timer is re-armed with the value placed in *DMACR.DMACWT*. When in DMA coalescing, the internal watchdog timer starts to count when one of the following conditions occurs:
      - An incoming Rx packet is received.
      - An interrupt is pending.
   b. Once an interval defined in the *DMACR.DMACWT* field has passed the integrated 10 GbE LAN controller exits DMA coalescing internal buffers are flushed, RSC flows are closed and interrupts are flushed.
2. *DMCTLX.TTLX* timer field to define the time between detection of DMA idle condition and entry into DMA coalescing state. To limit entry into DMA coalescing state when packet rate is high.
3. *DMACR.DMAC_EN* bit should be set to 1b to enable activating DMA coalescing operating mode.
4. *DMACR.Lx Coalescing Indication* bit defines whether to move in/out of DMA coalescing when the integrated I/O interface moves in/out of IDLE state., when set to 1b, DMA coalescing conditions are met only when the integrated I/O interface is in of IDLE state, when set to 0b DMA coalescing can also start when the integrated I/O interface is not in IDLE state. In addition, when the bit is set to 0b, DMA coalescing stops when any TLP transactions are executed on the integrated I/O interface.
5. *DMACR.EN_MNG_IND* bit should be set to 1b to enable a management indication impact on DMA coalescing mode.
6. *DMCMNGTH.DMCMNGTHR* field to set the threshold for the management data buffer that causes a move out of DMA coalescing operating mode. The manageability indications are ignored if *DMACR.EN_MNG_IND* is cleared.

Note: Any change to a DMA coalescing configuration should be done while the feature is disabled by setting *DMAR.DMAC_EN* to 0b.

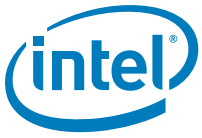
4.5.2 DMA Coalescing Operating Mode

Enabling DMA coalescing operation by setting the *DMACR.DMAC_EN* bit to 1b, enables aligning bus master traffic and interrupts from all ports. Power saving is achieved since synchronizing the integrated I/O interface accesses between ports increases the occurrence of idle intervals on the integrated I/O interface bus and also increases the duration of these idle intervals. The Power Management Unit (PMU) on a platform can use these idle intervals to reduce system power.

4.5.2.1 Conditions to Enter DMA Coalescing

The integrated 10 GbE LAN controller enters DMA coalescing when all of the following conditions exist:

1. DMA coalescing is enabled (*DMACR.DMAC_EN* = 1b).
2. Internal receive buffers are empty.
3. There are no pending DMA operations.
4. None of the conditions defined in Section 4.5.2.2 to move out of DMA coalescing exist.



4.5.2.2 Conditions to Exit DMA Coalescing

When the integrated 10 GbE LAN controller is in DMA coalescing operating mode, DMA coalescing mode is exited when one of the following events occurs:

1. Amount of data in the internal receive buffers passed the *DMCTH.DMACRXT* threshold.
2. An interrupt associated to a high priority vector defined by *EITR.HIGH_PRIORITY* was detected.
3. A received packet associated to a high priority traffic class defined by *DMACR.HIGH_PRIORITY* was detected.
4. DMA coalescing watchdog timer expires as a result of the following occurrences not being serviced for the duration defined in the *DMACWT timer* field:
   a. An incoming Rx packet is received.
   b. An interrupt is pending.
   c. An RSC flow was closed due to an ITR expiration.
5. DMA coalescing is disabled (*DMACR.DMAC_EN* = 0b).
6. Integrated I/O interface moves out of idle state.

Notes: Management indications are enabled through *DMACR.EN_MNG_IND* and the amount of data buffered in the management buffer exceeds *DMCMNGTH.DMCMNGTHR*.

4.5.3 DMA Coalescing Recommended Settings

These are the recommended DMA coalescing settings for the integrated 10 GbE LAN controller:

1. DMA coalescing receive threshold per traffic class (*DMCTH.DMACRXT[TC]*) = MAX{(RPB_Size[TC] - 70 µs of Rx data), Maximum Frame Size (MFS)}.

Note: The receive threshold must be set so that its distance from the effective Rx packet buffer size enables buffering of all Rx traffic that could be received at full blown during 70 µs, which corresponds to the worst case observed exit time from L1/L0s to L0. It means that the threshold setting depends on the link speed. A minimum of MFS must be set for the DMA coalescing threshold.

2. DMA coalescing watchdog timer (*DMACR.DMACWT*) must be set to get a worst case packet delay of ~ 20ms.
   a. If TTLX >= max ITR delay over all Qs: DMACWT = 0x262 ~ 20 ms. Flush the entire RPB to the host before re-entering a DMA coalescing state. The longest time a packet can wait in a packet buffer is DMACWT.
   b. If TTLX < max ITR delay over all Qs: DMACWT = (20 ms - Max ITR delay) / 2. The device re-enters the DMA coalescing state as soon as the first packet which is waiting in packet buffer is served. Then, when the ITR timer for the second packet expires, the device starts the DMACWT timer (again). It comes out that in this case, the longest time a packet can wait in packet buffer is ITR delay + 2 x DMACWT.
3. Time to LX request (*DMCTLX.TTLX*) = 0x20 ~ 40 µs, so that it is smaller than worst case L1/L0s max exit time (70 µs), taking advantage of the fact coalescing occurs in L0 too.
4. DMACR.Lx_Coalescing_Indication = 0b. For example, coalescing occurs when in L0.
5. DMA coalescing management threshold (*DMCMNGTH.DMCMNGTHR*) = 0x100 (equivalent to 4 KB).

4.6 Latency Tolerance Reported (LTR)

The integrated 10 GbE LAN controller generates LTR messages to report service latency requirements for memory reads and writes to the root complex for system power management.

The integrated 10 GbE LAN controller reports either latency tolerance or no latency tolerance requirements as a function of link, LAN port and function status. The reported latency tolerance value is set to optimize platform power consumption without incurring packet loss due to receive buffer overflow.

4.6.1 LTR Algorithm

The integrated 10 GbE LAN controller sends LTR messages according to the following algorithm when the capability is enabled in the LTR capability structure of Function 0 located in the PCIe configuration space:

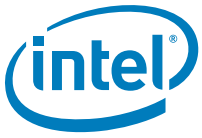
1. When links on all ports are disconnected or all LAN ports are disabled, the software device driver instructs the integrated 10 GbE LAN controller to send an LTR message with LTR requirement bits cleared, to indicate that no latency tolerance requirements exist.
2. If the integrated 10 GbE LAN controller reported latency tolerance requirements with any requirement bit set in the PCIe LTR message and all enabled functions where placed in D3 low power state via the PMCSR register, the integrated 10 GbE LAN controller sends a new LTR message with all the requirement bits clear.
3. If the integrated 10 GbE LAN controller reported latency tolerance requirements with any requirement bit set and the LTR *Mechanism Enable* bit in the PCIe configuration space is cleared, the software device driver instructs the integrated 10 GbE LAN controller to send an LTR PCIe message with all the requirement bits clear.
4. The integrated 10 GbE LAN controller holds the LTR value to send defined in LTR Control (LTRC) controlled and configured by the software device driver. The integrated 10 GbE LAN controller sends a new LTR message with the minimum latency tolerance requirement value upon an explicit request from the software device driver (issued by setting *LTRC.Send* bit). The following event can cause a the integrated 10 GbE LAN controller's software device driver to change its latency tolerance requirement:
   a. Link speed changed.
   b. Link was disconnected.

The integrated 10 GbE LAN controller conglomerates latency requirements from the different functions and sends a single LTR message in the following manner:

- The acceptable latency values for the message sent upstream by the integrated 10 GbE LAN controller must reflect the lowest latency tolerance values associated with any function.
  — It is permitted that the snooped and non-snooped values reported in the conglomerated message are associated with different functions.
  — If none of the functions have a latency requirement for a certain type of traffic (snoop/non-snoop), the message sent by the integrated 10 GbE LAN controller does not have the *Requirement* bit corresponding to that type of traffic set.
  — The scale parameter should be set to zero by the software device driver if minimal latency is required for a specific type of traffic (latency = 0 and requirement is set).

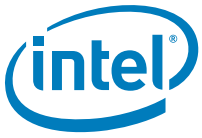
4.6.2 LTR Initialization Flow

- The BIOS/operating system configures the maximum snoop/non-snoop platform latency in the PCIe (function 0) configuration LTR capability registers.
- As part of software device driver initialization, the software device driver:
  — Reads the LTR capability to get the worst case latency tolerance requirement for the platform that the integrated 10 GbE LAN controller is connected to.



  - — Auto negotiates extended LPI exit latencies.
  - — The software device driver configures receive buffer watermarks for DMAC as aggressively as possible for the active target latency tolerance value (pre defined value).
  - — The software device driver configures the content of the LTR snoop/non-snoop requirements to LTRC.
- The software device driver initiates sending a LTR message from the integrated 10 GbE LAN controller by setting *LTRC.Send*.
- When a software device driver unloads, it should clear its latency tolerance requirements in the LTRC register and set the *LTRC.Send* bit, this could cause a new LTR message as defined in Section 4.6.1.

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



5.0 Shared SPI Flash Map

5.1 Shared SPI Flash Organization

The integrated 10 GbE LAN controller shared SPI Flash contains the following high-level modules:

- **Legacy EEPROM Modules**. These modules are mapped over one of the first two 16 KB sections of the Flash device, and cannot be extended beyond them. It is composed of all the shared SPI Flash modules used by MAC hardware or by manageability firmware, not including the manageability and PHY firmware code images. The sections included in this block are:
  - **Shared SPI Flash Pointers and Generic Words:** This section, described in Section 5.1, starts at the beginning of the valid section. It contains basic device information, pointers to other shared SPI Flash modules and several software configurations.
  - **PHY Module**: This module, described in Section 5.4.3, is pointed to by shared SPI Flash word 0x3. It contains the configuration of the PHY link layer (PCS + PMD) block and is loaded at each global reset.
  - **PCIe Modules:** These modules, described in Section 5.4.4, contain the parameters required to configure the PCIe configuration space. The pointers to these modules are located at shared SPI Flash words 0x6 (generic), 0x7 (port 0), and 0x8 (port 1).
  - **LAN Core Modules:** These two modules, described in Section 5.4.6, contain parameters required to configure the LAN port. These include the MAC address, LED, and SDP configuration. The pointers to these modules are located at shared SPI Flash words 0x9 (port 0) and 0xA (port 1). In addition, each LAN port has a module that enables loading specific CSR values after reset. The pointers to these modules are located at shared SPI Flash words 0xD (port 0) and 0xE (port 1). These modules are loaded following a software or hardware reset.
  - **Firmware Parameters Module**: Pointed to by the firmware module pointer located at shared SPI Flash word 0x0F. The firmware extension module starts with a list of firmware sub-modules pointers.
  - **Boot Configuration:** This module, pointed to by shared SPI Flash word 0x17, contains the configuration parameters used by the PXE and iSCSI boot code.
  - **VPD:** This module, pointed to by shared SPI Flash word 0x2F, contains the VPD module exposed via the VPD PCIe capability as described in Section 5.3.5.
- **Firmware Image**. This module contains the main code of the firmware loaded to the internal RAM. This image must fit within 512 KB and start at a 4 KB boundary. This module is authenticated upon update.
- **PCIe Expansion/Option ROM**. This module includes the PXE driver (61 KB), iSCSI boot image (116 KB), UEFI network driver (37 KB for x64, 67 KB for IA64), and can also include a CLP module (60 KB). It must fit within 512 KB and start at a 4 KB boundary. It is pointed by the PCIe expansion/option ROM pointer located at shared SPI Flash word 0x05.
- **Free Space Provisioning Segment.** The shared SPI Flash structure includes a space used to update the PXE code, firmware image, and PHY image modules via a double image policy. This space is referred to as the free space provisioning module or segment. It must be large enough to contain the largest of these three high-level modules. It is pointed to by the free space provisioning segment pointer located at shared SPI Flash word 0x40. See section 2.4.8.1 for the usage model of this Flash area.

Figure 5.1 shows a general shared SPI Flash structure and not a required order.

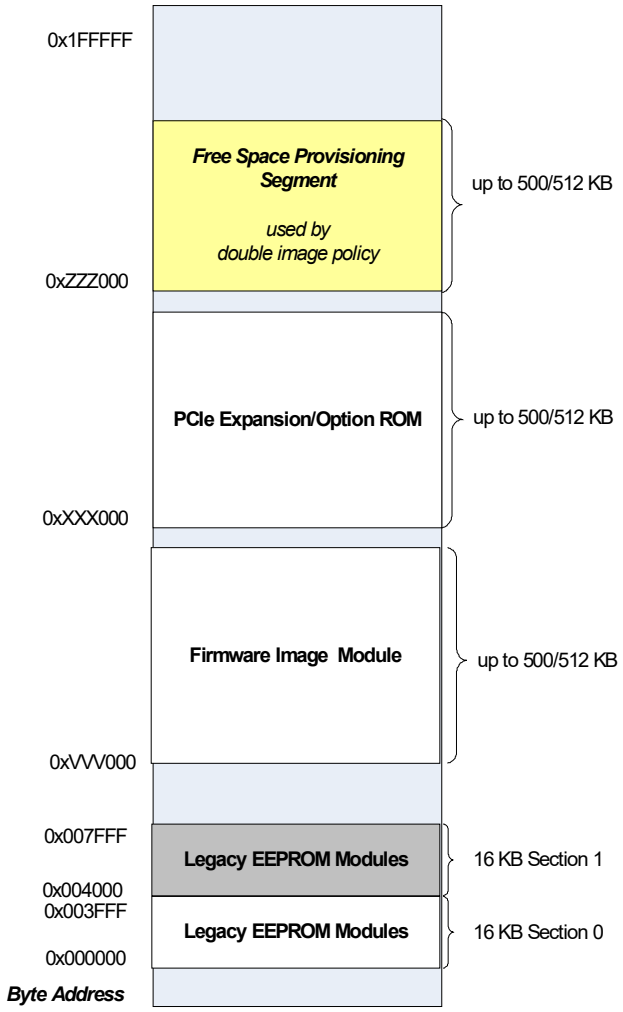


Figure 5.1. Shared SPI Flash Structure

5.1.1 Protected Areas

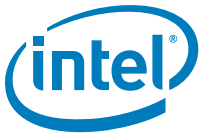
The following areas are protected from host writes:

- The firmware code area
- The pointers to the different modules.
- The mini loader
- Shared SPI Flash control words 1/2/3

The list of words and modules that are read only to software is listed in Table 5.1.

The firmware code area and option ROM can be updated using the flow described in Section 2.4.8.1.

Read/write words can be updated using the flow described in Section 2.4.2.1 and Section 2.4.8.1.



5.2 Shared SPI Flash Header

Note: Intel configures the reserved shared SPI Flash fields and they are not intended to be changed beyond the default image provided by Intel.

The following table lists the fixed part of the Legacy EEPROM modules used by the integrated 10 GbE LAN controller. This table lists common modules for the shared SPI Flash including: hardware pointers, software and firmware. Blocks pointed in this section are detailed in the following sections. All addresses in this table are absolute in word units.

Pointers can be in word units or in 4 KB units. If bit 15 of the pointer is set, then bits [14:0] points to a 4 KB sector and the word address is {pointer[14:0], 000,0000,0000b}. If bit 15 is cleared, then the word address is pointer[14:0].

Table 5.1. Shared SPI Flash Header Description

Word Address	Used By	Field Name	LAN 0 / 1	End User	RO to Host
0x00	HW	Shared SPI Flash Control Word 1 — Section 5.4.2.1	Shared Logic	Yes	RO word
0x01	HW	Shared SPI Flash Control Word 2 — Section 5.4.2.2	Shared Logic	Yes	RO word
0x02	HW	Reserved 0xFFFF	Shared Logic	No	RO pointer RO module
0x03	HW	PHY Link Configuration Module Pointer — Section 5.4.3	Shared Logic	No	RO pointer RO module
0x04	HW	Reserved	Shared Logic	No	RO pointer RO module
0x05	HW	PCIe Expansion/Option ROM Pointer - Section 5.5	SW + Shared Logic	No	RO pointer[1]
0x06	HW	PCIe General Configuration Module Pointer — Section 5.4.4	Shared Logic	No	
0x07	HW	PCIe Configuration Space 0 Module Pointer — Section 5.4.5	Function 0	No	
0x08	HW	PCIe Configuration Space 1 Module Pointer — Section 5.4.5	Function 1	No	
0x09	HW	LAN Core 0 Module Pointer — Section 5.4.6	Port 0	No	
0x0A	HW	LAN Core 1 Module Pointer — Section 5.4.6	Port 1	No	
0x0B	HW	Reserved	Port 0	No	
0x0C	HW	Reserved	Port 1	No	
0x0D	HW	CSR 0 Auto Configuration Module Pointer —	Port 0	No	RO pointer RO module
0x0E	HW	CSR 1 Auto Configuration Module Pointer —	Port 1	No	RO pointer RO module
0x0F	FW	Firmware Module Pointer — Section 5.3	FW	No	
0x10 – 0x14	SW	Software Compatibility Module — Section 5.3.1	SW	No	
0x15 - 0x16	SW	PBA Bytes 1...4 — Section 5.3.2	SW	No	
0x17	SW + FW	Boot Configuration Start Address — Section 5.3.3	SW+FW	No	
0x18	SW + FW	Shared SPI Flash Image Revision - Section 5.3.4	SW+FW	No	
0x19	SW + FW	Reserved	SW+FW	No	
0x1A – 0x2E	SW	Software Reserved — Section 5.3.4	SW	No	
0x2F	OEM	VPD Pointer — Section 5.3.5	Shared Logic	No	RO pointer[1] RO section
0x30 – 0x36	PXE	PXE Configuration Words — Section 5.3.6	SW	No	
0x37	SW + FW	Alternate Ethernet MAC Addresses Pointer — Section 5.3.7	SW+FW	No	

Table 5.1. Shared SPI Flash Header Description

Word Address	Used By	Field Name	LAN 0 / 1	End User	RO to Host
0x38	HW	Shared SPI Flash Control Word 3 — Section 5.4.2.3	Shared Logic	Yes	
0x39	SW	Reserved 0xFFFF	SW	No	
0x3A	FW	Firmware Code Pointer	FW	No	RO pointer[2] RO module
0x3B – 0x3E	HW	Hardware Reserved	Reserved	No	
0x3F	SW + FW	Software Checksum, Words 0x00 — 0x3F	Shared Logic	No	
0x40	FW	Free Space Provisioning Segment Pointer	FW	No	RO pointer
0x41	FW	Free Provisioning Area Size, expressed in 4 KB sectors. Default is 0x7A.	FW	No	RO word
0x42	FW	0xFFFF (Reserved)	FW	No	RO pointer RO section
0x43	FW	External PHY Default Configuration	FW	Yes	
0x44-0x4F		Reserved (0xFFFF)		No	
0x50	FW	RO Updates Version - This field is copied by firmware from the *RO Updates Version* field present in the trailer of the firmware secured module.	FW	No	RO word

1. VPD area is updated via the VPD PCIe capability or via shadow RAM update host I/F command if the VPD *Write Enable* bit in the shared SPI Flash control word 1 is set.
2. Updated via the secured module update flow(Section 2.4.8.1).

Note: All pointers refer to word addresses with the exception of pointers to the PCIe Expansion/ Option ROM (0x5), Firmware Code (0x3A), and Free Space Provisioning Segment (0x41), which are expressed in 4 KB sector units.

5.3 Software Sections

This section describes both words in the shared SPI Flash header and additional software sections pointed from the header.

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

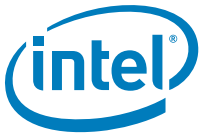
Five words in the shared SPI Flash image are reserved for compatibility information. New bits within these fields are defined as the need arises for determining software compatibility between various hardware revisions.

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

5.3.1.1 Software Compatibility Word 1 — Word Address 0x10

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

Bits	Name	Default	Description
7:0	Reserved	0x0	Reserved.
8	OEM/Retail	0b	Legacy, not currently used. 0 = Retail. 1 = OEM.
9	Reserved	0b	Reserved.



Bits	Name	Default	Description
10	Server	1b	Legacy, not currently used. 0 = Client. 1 = Server.
11	LOM	1b	Indicates whether the shared SPI Flash attached to LAN silicon contains a dedicated module for option ROM. Used by option ROM update applications. 0 = NIC (Attached flash contains module for option ROM). 1 = LOM (Attached flash has no module for option ROM).
15:12	Reserved	0x0	Reserved.

5.3.1.2 Software Compatibility Word 2-5 - Reserved

5.3.2 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 shared SPI Flash.

Note: Through the course of hardware ECOs, the suffix field is increased. 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
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.

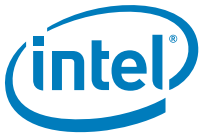
PBA Number	Byte 1	Byte 2	Byte 3	Byte 4
123456-003	12	34	56	03

Note: The PBA module (a length of 12 bytes) must be mapped in the first valid 16 KB sector of the Flash.

5.3.3 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.

Configuration Item	Offset (Bytes)	Size in Bytes	Comments
Shared Words			
Boot Signature	0x1:0x0	2	'i', 'S' (0x3569).
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.
Reserved	0x127:0x106	34	Reserved for future use. Should be set to zero.
Port 0 Configuration			
iSCSI Flags	0x129:0x128	2	Bit 0x00: Enable DHCP. 0 = Use static configurations from this structure. 1 = Overrides configurations retrieved from DHCP. Bit 0x01: Enable DHCP for getting iSCSI target information. 0 = Use static target configuration. 1 = Use DHCP to get target information by the option 17 root path. Bit 0x02 – 0x03: Authentication type. 00b = None. 01b = One way CHAP. 10b = Mutual CHAP. 11b = Reserved. Bit 0x04 – 0x05: Ctrl-D setup menu. 00b = Enabled. 01b = Reserved. 10b = Reserved. 11b = 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.



Configuration Item	Offset (Bytes)	Size in Bytes	Comments
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.
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.
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.
Reserved	0x383:0x2E4	160	Reserved for future use. Should be set to zero.
Port 1 Configuration			
Same configuration as port 0. Add to each offset 0x384.			

5.3.4 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.

5.3.4.1 Shared SPI Flash Image Revision — Word 0x18

Bits	Name	Default	Description
3:0	ID		Shared SPI Flash image ID.
11:4	Minor		shared SPI Flash minor version.
15:12	Major		shared SPI Flash major version.

5.3.4.2 Software Reserved Word 16— Word 0x19

Bits	Name	Default	Description
15:0	Reserved		Reserved.

5.3.4.3 Software Reserved Word 18 — Word Address 0x29

Word 0x29 is for the map version.

Bits	Name	Default	Description
15:12	Major		Shared SPI Flash major version (0-9).
11:8	Zero		Represents the decimal point.
7:0	ID		Shared SPI Flash map minor version (Hexadecimal value represent the decimal minor version. For example a value of 0x58 represent a decimal minor version of 58).

5.3.4.4 Software Reserved Word 19 — Word Address OEM 0x2A Image Revision

Optional field that enables an OEM to write a version and identifies it in the shared SPI Flash image. Used only for OEM shared SPI Flash images.

Bits	Name	Default	Description
15:12	Major		OEM shared SPI Flash major version (0-9).
11:8	Zero		Represents the decimal point.
7:0	ID		Shared SPI Flash map minor version (Hexadecimal value represent the decimal minor version. For example a value of 0x58 represent a decimal minor version of 58)

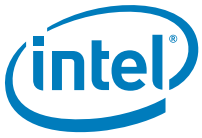
5.3.4.5 Software Reserved Word 21 — Word Address 0x2C

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

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

5.3.5 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 up to 1 KB, 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.



5.3.6 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/UEFI codes).

Word Address	Description
0x30	PXE Setup Options PCI Function 0 - Section 5.3.6.1
0x31	PXE Configuration Customization Options PCI Function 0 - Section 5.3.6.2
0x32	PXE Version - Section 5.3.6.3
0x33	Flash Capabilities - Section 5.3.6.4
0x34	PXE Setup Options PCI Function 1 - Section 5.3.6.5
0x35	PXE Configuration Customization Options PCI Function 1 - Section 5.3.6.6
0x36	iSCSI Option ROM Version - Section 5.3.6.7

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

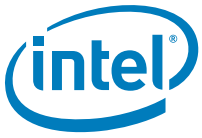
The main setup options for port 0 are stored in this word. These options are those that can be changed by the user using the Control-S setup menu.

Bits	Name	Default	Description
4:3	DBS	00b	Default Boot Selection. These bits select which device is the default boot device. These bits are only used if the agent detects that the BIOS does not support boot order selection or if the *MODE* field of word 0x31 is set to MODE_LEGACY. 00b = Network boot, then local boot (default). 01b = Local boot, then network boot. 10b = Network boot only. 11b = Local boot only.
5	Reserved	0b	Reserved. Must be 0b.
7:6	PT	00b	Prompt Time. These bits control how long the Press Control-S setup prompt message is displayed, if enabled by DIM. 00b = 2 seconds (default). 01b = 3 seconds. 10b = 5 seconds. 11b = 0 seconds. ***Note:*** The Ctrl-S message is not displayed if zero seconds prompt time is selected.
8	DSM	1b	Display Setup Message. If the bit is set to 1b, the Press Control-S message is displayed after the title message.
9	LWS	0b	Legacy OS Wakeup Support (for the integrated 10 GbE LAN controller-based adapters only). If set to 1b, the agent enables PME in the adapter's PCI configuration space during initialization. This enables 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. 0 = Disabled (default). 1 = Enabled.

Bits	Name	Default	Description
12:10	FSD	000b	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.
15:13	Reserved	000b	Reserved. Must be 0x0.

Bits 2:0 are defined as follows:

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



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

Word 0x31 of the shared SPI Flash 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
0	DSM	0b	Disable Setup Menu. If this bit is set to 1b, the user is not allowed to invoke the setup menu by pressing Control-S. In this case, the shared SPI Flash can only be changed via an external program.
1	DTM	0b	Disable Title Message. If this bit is set to 1b, the title message displaying the version of the Boot Agent is suppressed; the Control-S message is also suppressed. This is for OEMs who do not wish the boot agent to display any messages at system boot.
2	DPS	0b	Disable Protocol Select. If set to 1b, the user is not allowed to change the boot protocol.
3	DBS	0b	Disable Boot Selection. If this bit is set to 1b, the user is not allowed to change the boot order menu option.
4	DLWS	0b	Disable Legacy Wakeup Support. If this bit is set to 1b, the user is not allowed to change the Legacy OS Wakeup Support menu option.
5	DFU	0b	Disable Flash Update. If this bit is set to 1b, the user is not allowed to update the Flash image using PROSet.
7:6	RFU	00b	Reserved. Must be 00b.
10:8	MODE	000b	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.
11	RETRY	0b	Selects Continuous Retry operation. If this bit is set, IBA does NOT transfer control back to the BIOS if it fails to boot due to a network error (such as failure to receive DHCP replies). Instead, it restarts the PXE boot process again. If this bit is set, the only way to cancel PXE boot is for the user to press ESC on the keyboard. Retry is not attempted in case of hardware error conditions such as an invalid shared SPI Flash checksum or failing to establish link.

Bits	Name	Default	Description
13:12	RFU	00b	Reserved. Must be 00b.
15:14	SIG	01b	Signature. Must be set to 01b to indicate that this word has been programmed by the agent or other configuration software.

5.3.6.3 PXE Version — Word Address 0x32

Word 0x32 of the shared SPI Flash 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
7:0	BLD	0x0	PXE Build Number.
11:8	MIN	0x0	PXE Minor Version.
15:12	MAJ	0x0	PXE Major Version.

5.3.6.4 Flash Capabilities — Word Address 0x33

Word 0x33 of the shared SPI Flash 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.

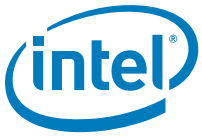
Bits	Name	Default	Description
0	BC		PXE Base Code is present if set to 1b.
1	UNDI		PXE UNDI driver is present if set to 1b.
2	RPL		RPL module is present if set to 1b. Reserved bit for devices.
3	EFI		EFI UNDI driver is present if set to 1b.
4	ISCSI		iSCSI boot code is present if set to 1b.
5	Reserved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

5.3.6.5 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.

5.3.6.6 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.



5.3.6.7 iSCSI Option ROM Version — Word Address 0x36

Word 0x36 of the shared SPI Flash 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 shared SPI Flash 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.

5.3.7 Alternate Ethernet MAC Address Pointer — Word Address 0x37

The alternate MAC address does not function if the pointer value is set to 0xFFFF.

Word offset for PCIe function 0 MAC address is: value in word 0x37 + 0 - 3 words.

Word offset for PCIe function 1 MAC address is: value in word 0x37 + 3 - Next 3 words.

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 3.5.10 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
0x0... 0x2	Alternate Ethernet MAC Address 1 (function 0).
0x3... 0x5	Alternate Ethernet MAC Address 2 (function 1).

5.4 Hardware Sections

Note: This module contains address control words and hardware pointers indicated as hardware in Table 5.1. The process of loading this module (or any of it sub-modules) into the integrated 10 GbE LAN controller is referred as the MAC auto-load process. This module must be mapped in the first valid 16 KB section of the Flash.

5.4.1 Hardware Section — Auto-Load Sequence

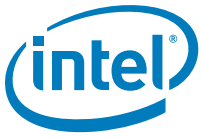
Table 5.2 lists sections of auto-read following device reset events or specific commands from registers. Auto-read is performed from the internal shadow RAM (or from internal memory for PHY module) and not from the shared SPI Flash device, except following Global Reset.

Table 5.2. Shared SPI Flash Section Auto-Read

	Global Reset	Integrated I/O Reset	D3 to D0 Transit (per port)	FLR (per port)	SW Reset (per port)	Link Reset (per port)	FW Reset	Force TCO
Integrated PHY Link Configuration (HLCB)	X							
PCIe General Configuration		X						
PCIe Function 0/1 Config Space (for each function)		X						
LAN Core and CSRs (for each LAN port)	X	X	X	X	X	X		X
Manageability Firmware Module	X						X	

5.4.2 Shared SPI Flash Init Module

The Init section (shared SPI Flash Control Word 1, 2, and 3) are read after a global reset and PCIe reset.



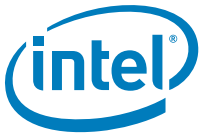
5.4.2.1 Shared SPI Flash Control Word 1 — Address 0x00

Bits	Name	Default	Description
3:0	Reserved	0x0	Reserved.
4	Disable Shared SPI Flash Protection	0b	If set, the Firmware will not enable shared SPI Flash security. This bit is not used by hardware and is not reflected in any register. Firmware should read it directly from the shadow RAM.
5	VPD Write Enable	1b	When set to 1b, the host can modify the entire VPD area via the Flash update host I/F command. When set to 0b, any host attempt to modify the VPD area via the Flash update host I/F command is completed with an error. ***Note:*** RW-VPD area can be modified from the config space register set regardless of the setting of this field.
7:6	Signature	01b	Signature. Indicates to the integrated 10 GbE LAN controller that there is a valid shared SPI Flash present. If the *Signature* field is not 01b, the other bits in this word are ignored, no further shared SPI Flash read is performed, and the default values are used for the configuration space IDs.
10:8	Reserved	101b	Reserved.
11	Reserved	01b	Reserved.
12	Host Debug Write	0b	When set, host can access restricted registers.
13	Disable FW reset fix	0b	FW reset has an effect on SW write access to shared SPI Flash via EEPROM-mode although it should not have any effect on it. Normal operating mode (default).
14	NC-SI HW ARB Disable	0b/1b	NC-SI Hardware Arbitration Enable. 1 = NCSI_ARB_IN and NCSI_ARB_OUT pads are not used. NCSI_ARB_IN is pulled up internally to provide stable input. 0 = NCSI_ARB_IN and NCSI_ARB_OUT pads are used.
15	Reserved	0b	Reserved.

5.4.2.2 Shared SPI Flash Control Word 2 — Address 0x01

Bits	Name	Default	Description
0	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.
3:2	Reserved	0x0	Reserved.
7:4	Reserved	0x0	Reserved. Formally power management controls for memories.
8	LAN PCI Disable	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. If the disabled port is used by MC, only the DMA block and PCIe interface of the port is powered down. Otherwise, the port is powered down up to the PHY included. If the LAN PCI Disable bit is set for a port, the port's respective APM bit in shared SPI Flash Control Word 3 must be cleared as well. Readable as DEV_FUNC_EN[0]
9	LAN Disable Select	0b	LAN Disable Select. 0 = LAN 0 is disabled. 1 = LAN 1 is disabled. Readable as DEV_FUNC_EN[1]

Bits	Name	Default	Description
10	Device OFF enable	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. 0 = Legacy mode (default) — Though the device is disabled, the digital I/O pins are not moved to an electrical off state. 1 = 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. Readable as DEV_FUNC_EN[2]
11	Reserved	0b	Reserved.
12	Reserved	0b	Reserved.
14:13	Reserved	0x0	Reserved/
15	LAN Function Select	0b	LAN Function Select. 0 = LAN 0 is routed to PCI function 0 and LAN 1 is routed to PCI function 1. 1 = LAN 0 is routed to PCI function 1 and LAN 1 is routed to PCI function 0. This bit is mapped to FACTPS[30].



5.4.2.3 Shared SPI Flash Control Word 3 — Address 0x38

Bits	Name	Default	Description
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 shared SPI Flash is set for *Port 0*, then the *APM* bit must be cleared.
1	APM Enable Port 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 shared SPI Flash is set for *Port 1*, then the *APM* bit must be cleared[1].
2	Keep_PHY_Link_Up_En	1b	When set to 1b and *MMNGC.MNG_VETO = 1b* changes in the power state. For example, D0 to D3/Dr, does not impact the link state,the link remains unchanged even if the device enters a low power state. When cleared, the *MMNGC.MNG_VETO* bit is meaningless
3	Enable LPLU	1b	Enable the Low Power Link Up Feature. When this field is set the device may trigger a link speed change to a lower supported speed.
4	Reserved	0b	Reserved.
5	Reserved	0b	Reserved. Must be zero. Formally D1GMP Port 0.
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).
7	Reserved	0b	Reserved. Must be 0b. Formally D1GMP Port 1.
8	D10GMP Port 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).
9	HW-Reserved	0b	Reserved.
15:10	Reserved	000b	Reserved

1. 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.

5.4.3 SoC Indirect Configuration Module

This section contains the link configuration information of the integrated PHY, PCS plus PMD. The information is stored as address plus data pairs that the integrated 10 GbE LAN controller translates to register write messages.

Note: This section is loaded only at power on following Global Reset.

Note: This section might contain configuration information for multiple integrated PHY modules, depending on the target device the integrated 10 GbE LAN controller is being integrated into.

Shared SPI Flash Word 0x3 is the pointer for this section. The structure of this section is listed in the following table.

Word offset	Description
0x0	Section Length (in words)
0x1-0x2	Sideband Integrated I/O Indirect Control register content - 1
0x3-0x4	Integrated PHY Data 1
0x5-0x6	Sideband Integrated I/O Indirect Control register content - 2
0x7-0x8	Integrated PHY Data 2
0x9 - 0xK	Reaming Integrated PHY registers.
0xK+1 - 0xK+2	Sideband Integrated I/O Indirect Control register content - X - for setting the IO muxing.
0xK+3 - 0xK+4	I/O MUX Control Data.
0xK+5 - 0xSection Length - (0xK+5)	Remaining I/O muxing registers.

5.4.3.1 Section Length — Offset 0x00

The section length word contains the length of the section in words. Note that section length does not include the section length word itself. The 2 LS bits must be zero. The maximum allowed value is 8K - 4 (8188)

5.4.3.2 Sideband Integrated I/O Indirect Control register content

The two first words in a Qword set in the integrated PHY LCB section contains the control word to be written into the sideband integrated I/O Indirect Control register. This register is used to generate a sideband integrated I/O write message to the relevant integrated PHY block. For detailed information about this flow see Section 3.8.1.

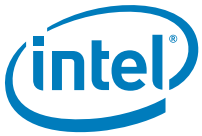
5.4.3.3 Integrated PHY Data

The two last words in a Qword set in the integrated PHY LCB section contains the data to be written.

5.4.4 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 shared SPI Flash (full-byte address; must be word aligned).

This section contains initial values for a predefined list of CSRs described as follows.



5.4.4.1 CSR Data — Offset 0x0- 0x23

Even words.

Bits	Name	Default	Description
15:0	CSR_Data_LSB	0x0	CSR Data LSB.

Odd Words.

Bits	Name	Default	Description
15:0	CSR_Data_MSB	0x0	CSR Data MSB.

The following registers are read through this section:

Offset[1]	CSR Name	Default	Description	Section (CSR Reference)
0x0 - 0x1	PCI_CNF2	0x400060FC	MSI-x and cache line configuration.	438
0x2 - 0x3	PCI_LBARCTRL	0x00001949	PF BAR registers configuration.	439
0x4 - 0x5	PCI_SERL	N/A	PCIe Serial Number Low.	442
0x6 - 0x7	PCI_SERH	N/A	PCIe Serial Number High.	443
0x8 - 0x9	PCI_CAPCTRL	N/A	VPD Enable.	442
0xA - 0xB	PCI_CAPSUP	0x001F007D	PCIe Capabilities exposed.	440
0xC - 0xD	PCI_DBGCTL	0x0	Allow access to VF config space.	402
0xE - 0xF	PCI_UPADD	0x00000000	Upper Address accessible.	447
0x10 - 0x11	PCI_SUBSYSID	N/A	PCIe Subsystem ID[2].	446
0x12 - 0x13	PCI_PWRDATA	TBD	PCIe Power Data Register.	445
0x14 - 0x15	PCI_REVID	0x0	Revision ID.	428
0x16 - 0x17	PCI_VFSUP	0x2	PF BAR registers configuration.	428
0x1A - 0x1B	PCI_LINKCAP	0x00000E80	PCIe Link Capabilities.	441
0x1C - 0x1D	PCI_PMSUP	0x00007397	PCIe Power Management Support.	444
0x1E - 0x1F	PCI_GLBL_CNF	0x0000100C[3]	Wake# enable.	440
0x20 - 0x21	PCI_VENDORID[4]	0x00008086	PCIe Vendor ID.	445
0x22 - 0x23	PCI_PCIERR	0x00003FCB	Defines the PCI errors to be reported to the device driver.	431

1. For each register, the first word contains the Least Significant Bytes and the second word describes the Most Significant Bytes.
2. This register is loaded only if the PCI_CAPSUP.*LOAD_SUBSYS_ID* bit is set.
3. Default assumes WAKE_PIN_EN = 0b, which may be modified in different systems.
4. This register is loaded only if the PCI_CAPSUP.*LOAD_DEV_ID* bit is set.

5.4.5 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. The structures of both functions are identical and contains initial values for a predefined list of CSRs as listed below.

5.4.5.1 CSR Data — Offset 0x0- 0x7

Even words.

Bits	Name	Default	Description
15:0	CSR_Data_LSB	0x0	CSR Data LSB.

Odd Words.

Bits	Name	Default	Description
15:0	CSR_Data_MSB	0x0	CSR Data MSB

The following registers are read through this section:

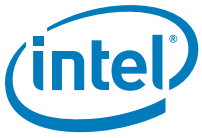
Offset	CSR Name	Default	Description	Section
0x0 - 0x1	PCI_PFDEVID	See Section 8.2.2.2	PCIe PF Device ID[1]	446
0x2 - 0x3	PCI_CNF	Function 0: 0x00000000 Function 1: 0x00000020	PCIe PF Configuration space settings	443
0x4 - 0x5	PCI_VFDEVID	See Section 8.2.2.2	PCIe VF Device ID	446
0x6 - 0x7	PCI_CLASS	0x0	PCIe Storage Class	443

1. This register is loaded only if the PCI_CAPSUP.*LOAD_DEV_ID* bit is set.

5.4.6 LLAN 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 5.4.6.1
0x1	Ethernet MAC Address Byte 2	Ethernet MAC Address Byte 1	Section 5.4.6.2.1
0x2	Ethernet MAC Address Byte 4	Ethernet MAC Address Byte 3	Section 5.4.6.2.2
0x3	Ethernet MAC Address Byte 6	Ethernet MAC Address Byte 5	Section 5.4.6.2.3
0x4	LED 1 configuration	LED 0 Configuration	
0x5	LED 3 Configuration	LED 2 Configuration	
0x6	SDP Control		
0x7	Filter Control		



5.4.6.1 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
15:0	Section Length	0x7	Section length in words.

5.4.6.2 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 shared SPI Flash 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

5.4.6.2.1 Ethernet MAC Address Register1 – Offset 0x01

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

5.4.6.2.2 Ethernet MAC Address Register2 – Offset 0x02

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

5.4.6.2.3 Ethernet MAC Address Register3 – Offset 0x03

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

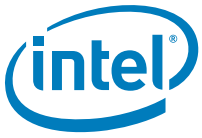
5.4.7 External PHY Default Configuration

This section holds a list of configurations to be executed by firmware at power on for initializing an external PHY or module.

The format is as follows:

Word Offset	Description
0x0	Relative pointer to MDIO accessible list of registers for Port-1 (PTR Value-0).

Word Offset	Description
0x1	Relative pointer to I2C accessible list of registers for Port-0 (PTR Value-1).
0x2	Relative pointer to I2C accessible list of registers for Port-1 (PTR Value-2).
0x3	Reserved.
0x4: Port-0 List of MDIO accessible registers (offset shown is a relative to the beginning of the list, such as 0x4)	
0x0	Number of registers in List <n>.
0x1	Value to write into MSCA register.
0x2	Value to write into MSRWD.MDIWRDATA.
Body of List	
<n-1>*4+2	Value to write into MSCA register.
<n-1>*4+4	Value to write into MSRWD.MDIWRDATA.
0xPTR Value-0: Port-1 List of MDIO accessible registers (offset shown is a relative to the beginning of the list)	
List of MDIO registers for Port-1. Format is identical to the Port-0 list.	
0xPTR Value-1: Port-0 List of I2C accessible registers (offset shown is a relative to the beginning of the list)	
0x0	Number of registers in List <n>.
0x1	Value to write into I2CCMD register.
0x2	Value to write into I2CCMD register.
Body of List	
<n-1>*4+2	Value to write into I2CCMD register.
<n-1>*4+4	Value to write into I2CCMD register.
0xPTR Value-2: Port-1 List of I2C accessible registers (offset shown is a relative to the beginning of the list)	
List of I2C registers for Port-1. Format is identical to the Port-0 list.	



5.5 PCIe Expansion/Option ROM

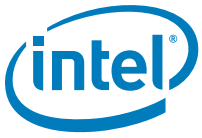
This module might include the PXE driver, iSCSI boot image, UEFI network driver, and a Command Line Protocol (CLP) module. It is made of a single module (no pointers to sub-sections) and must fit into 512 KB.

It is not required for LOM systems where it is stored on the BIOS Flash device.

The module is pointed by the PCIe expansion/option ROM pointer at the shared SPI Flash word 0x05, expressed in 4 KB sector units. Whenever modifying this pointer in the shared SPI Flash, it is required to issue a PCIe reset before any new access is performed to the expansion ROM. Otherwise, the integrated 10 GbE LAN controller would continue to use the old pointer each time it maps internally accesses to the expansion ROM.

The first 330 words listed in the header of firmware and option ROM images are mapped at the end of the area allocated to the module (a trailer), though for the sake of the authentication, these words are mapped at the module's header, as listed in the previous table.

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



6.0 Inline Functions

6.1 Receive Functionality

Packet reception consists of:

- Recognizing the presence of a packet on the wire (Section 6.1.1)
- Parsing of the packet header (Section 6.1.2)
- Performing address filtering (Section 6.1.2)
- Checksum off-loads (Section 6.1.6)
- DMA queue assignment (Section 6.1.3)
- Storing the packet in the receive data FIFO (Section 6.3.1)
- Transferring the data to assigned receive queues in host memory (Section 6.1.4)
- Optional Receive Side Coalescing (Section 6.8)
- Updating the state of a receive descriptor (Section 6.1.5).

6.1.1 MAC Layer - Receive

6.1.1.1 Packet Acceptance Criteria

In addition to the filtering rules described in the next sections, a packet must also meet the following criteria in order to be accepted by the device:

1. Normally, only good packets are received (packets with none of the following errors: Under Size Error, Over Size Error, Packet Error, Length Error and CRC Error). However, if the store-bad-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. In this case, bad packets are queued according to the same rules as regular packets.
2. Min. Packet Size (Runt packets) — Rx packets, smaller than 48 bytes, cannot be posted to host memory regardless of save bad frame setting.
3. Max Packet Size — Any Rx packet posted from the MAC unit to the DMA unit cannot exceed 15.5 KB. Further restrictions per queue are described in Section 6.1.4.
4. 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).

6.1.1.2 CRC Strip

The integrated 10 GbE LAN controller potentially strips the L2 CRC on incoming packets.

CRC strip is enabled by the *HLREG0.RXCRCSTRP* bit. When set, CRC is stripped from all received packets.

The global CRC strip bit (*HLREG0.RXCRCSTRP*) must be set in the following cases were the packet changes before being handled to the driver:

- RSC is enabled in any queue.
- VLAN is hidden (*PFQDE.HIDE_VLAN* = 1b) in any queue.
- E-tag is removed in any queue (*PFQDE.STRIP_TAG* is set)
- L2 tags are stripped from packets (*PFQDE.STRIP_TAG = 1*) in any queue.
- Time stamp is added to the packets in any TC (any bit of *TSYNCRXCTL.TSIP_UT_EN* or *TSYNCRXCTL.TSIP_UP_EN* is set).
- Short received packets are padded (*RDRXCTL.PSP* is set).

6.1.2 Packet Filtering

A received packet goes through up to three stages of filtering as depicted in Figure 6.1. The figure describes a switch-like structure that is used in virtualization mode to route packets between the network port (top of drawing) and one of many virtual ports (bottom of drawing), where each virtual port might be associated with a Virtual Machine (VM), a Virtual Machine Monitor (VMM), or the like. The three stages are:

1. **First stage — Admission Control:** Ensure that the packet should be received by the port. This is done by a set of L2 filters and is described in detail in this section.
2. **Second stage — Pooling:** This stage is specific to virtualization environments and defines the virtual ports (called pools in this document) that are the targets for the Rx packet. A packet can be associated with any number of ports/pools and the selection process is described in Section 6.1.3.2. In non virtualization mode, this stage is skipped and all the queues used in the next stage are considered as part of the same default pool.

Note: A pool is equivalent to a VSI as defined in IEEE 802.1Qbg specification.

3. **Third stage — Queueing:** A receive packet that successfully passed the Rx filters is associated with one of many receive descriptor queues as described in Section 6.1.3.

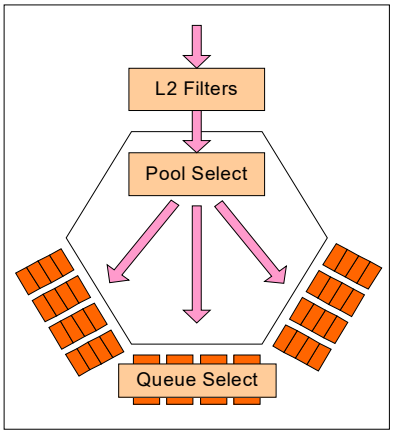


Figure 6.1. Stages in Packet Filtering (Virtualization Mode)

The receive packet filtering role is to determine which of the incoming packets are allowed to pass to the local machine and which of the incoming packets should be dropped since they are not targeted to the local machine. Received packets that are targeted for the local machine can be destined to the host, to a manageability controller, or to both. This section describes how host filtering is done, and the interaction with management filtering.

As depicted in Figure 6.2, host filtering is done in two stages:

1. Packets are filtered by L2 filters (Ethernet MAC address, unicast/multicast/broadcast). See Section 6.1.2.1.
2. Packets are filtered by VLAN if a VLAN tag is present. See Section 6.1.2.2.

A packet is not forwarded to the host if any of the following occurs:

- The packet does not pass L2 filters, as described in Section 6.1.2.1.
- The packet does not pass VLAN filtering, as described in Section 6.1.2.2.
- The packet passes manageability filtering and the manageability filters determine that the packet should not pass to the host (see MNGONLY register).

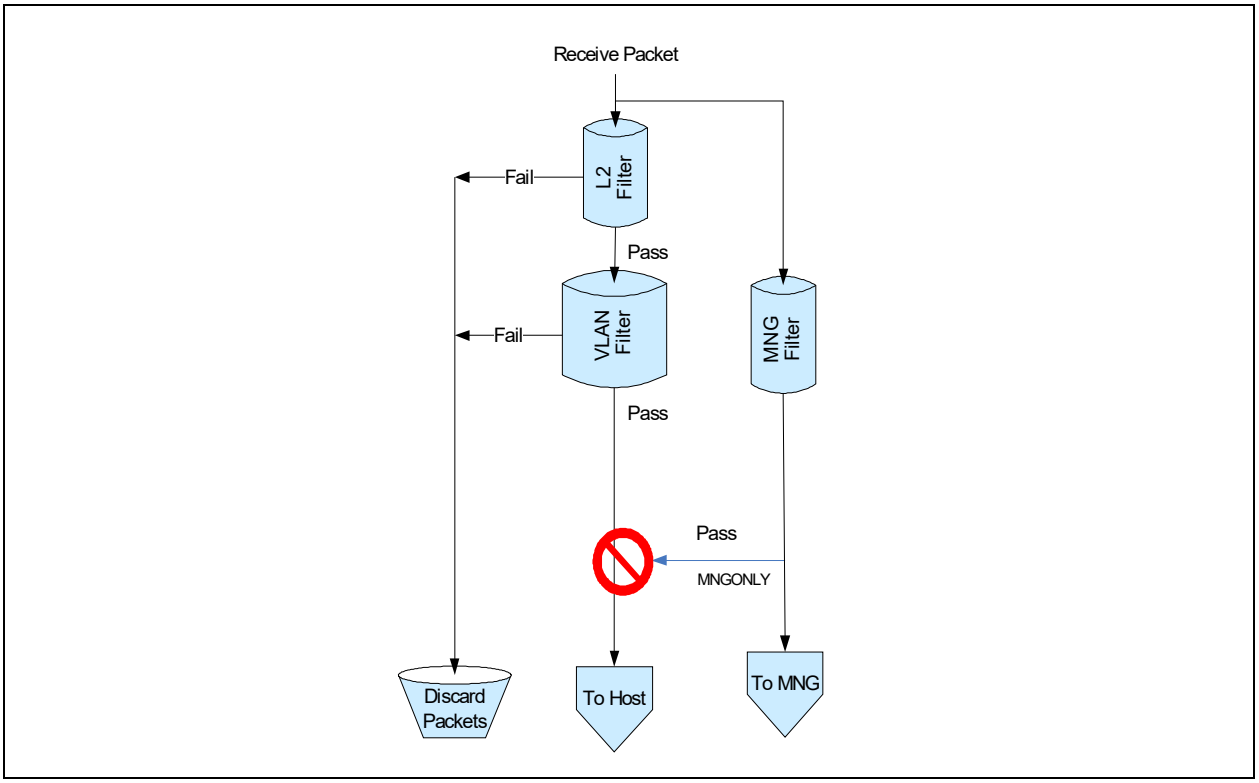


Figure 6.2. Rx Filtering Flow Chart

6.1.2.1 L2 Filtering

A packet passes successfully through L2 Ethernet MAC address filtering if any of the following conditions are met:

- Unicast packet filtering — Promiscuous unicast filtering is enabled (FCTRL.UPE=1b) or the packet passes unicast MAC filters (host).

- Multicast packet filtering — Promiscuous multicast filtering is enabled (FCTRL.MPE=1b) or the packet matches one of the multicast filters.
- Broadcast packet filtering to host — Promiscuous multicast filtering is enabled (FCTRL.MPE=1b) or Broadcast Accept Mode is enabled (*FCTRL.BAM* = 1b).

6.1.2.1.1 Unicast Filter

The Ethernet MAC address is checked against the 128 host unicast addresses and 4 KB hash-based unicast address filters (if enabled). The host unicast addresses are controlled by the host interface. The destination address of an incoming packet must exactly match one of the pre-configured host address filters. These addresses can be unicast or multicast. Those filters are configured through Receive Address Low (RAL), Receive Address High (RAH), In addition, there are 4 KB unicast hash filters used for host defined by the PFUTA registers. The unicast hash filters are useful mainly for virtualization settings in those cases that more than 128 filters might be required.

Promiscuous Unicast — Receive all unicasts. Promiscuous unicast mode is usually used when the LAN device is used as a sniffer.

6.1.2.1.2 Multicast Filter (Partial)

The 12-bit portion of the incoming packet multicast address must be set in the multicast filter address table (*MTA*) in order to pass the partial multicast filter. The bits (out of 48 bits of the destination address) used to index the MTA table can be selected by the *MO* field in the MCSTCTRL register.

Promiscuous Multicast — Receive all multicasts. Promiscuous multicast mode is usually used when the LAN device is used as a sniffer.

6.1.2.2 VLAN Filtering

The integrated 10 GbE LAN controller provides exact VLAN filtering as follows:

- Host VLAN filters are programmed by the VFTA[n] registers.
- A VLAN match might relate to the *DEI* bit in the VLAN header. It is enabled for host filtering only by the *VLNCTRL.DEIEN* while the expected value is defined by the *VLNCTRL.DEI*.

If double VLAN is enabled (see Section 6.4.5), filtering is done on the second (internal) VLAN tag. All the filtering functions of the integrated 10 GbE LAN controller ignore the first (external) VLAN in this mode.

A receive packet that passes MAC Address filtering successfully is subjected to VLAN header filtering as illustrated in Figure 6.3:

1. If the packet does not have a VLAN header, it passes to the next filtering stage.
2. Else, if VLAN filters are not enabled (*VLNCTRL.VFE* = 0b), the packet is forwarded to the next filtering stage.
3. Else, if the packet matches an enabled VLAN filter and DEI checking (if enabled), the packet is forwarded to the next filtering stage.
4. Otherwise, the packet is dropped.

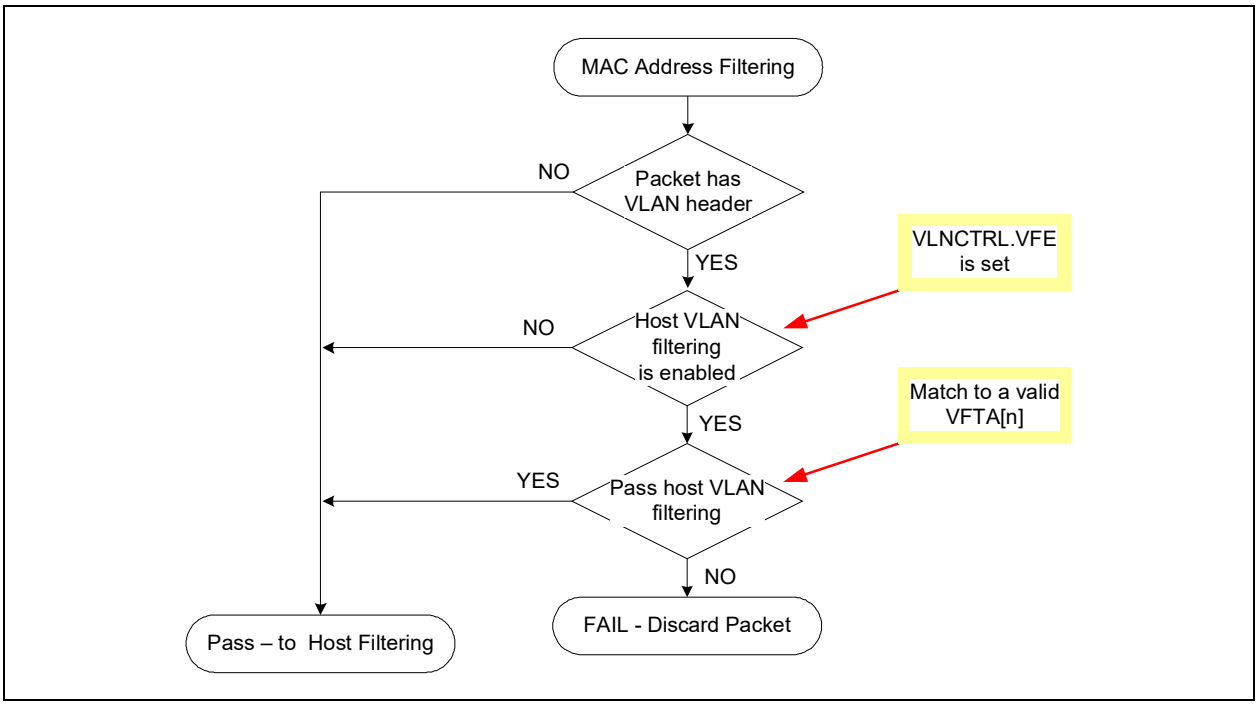


Figure 6.3. VLAN Filtering

6.1.2.3 E-tag filtering

If the *PFVTCTL.POOLING_MODE* is E-tag (01b), then special L2 filtering rules are applied to tagged packets.If the VT_mode is E-tag (01b), then special L2 filtering rules are applied to packets with E-tag.

- If the tag matches one of the RAH registers used for tag filtering (*RAH.ADTYPE* = 1b), then it is considered as a packet that passed L2.
- If the *FCTRL.TPE* bit is set (Tag Promiscuous Enable), all the tagged packets passes L2 filtering.
- Otherwise, packets with E-tag [if *PFVTCTL.POOLING_MODE*is E-tag (01b)]are dropped.

Note: E-tag packets are not expected when double VLAN are use, thus *PFVTCTL.POOLING_MODE* should be cleared if *CTRL_EXT.EXTENDED_VLAN* is set.

6.1.2.4 Manageability / Host Filtering

The host and manageability filtering process are mostly independent. Each entity defines which packet it should receive. The only exception is that the manageability filters can define a packet as exclusive and thus prevent it from reaching the host. Figure 6.4 describes this flow.

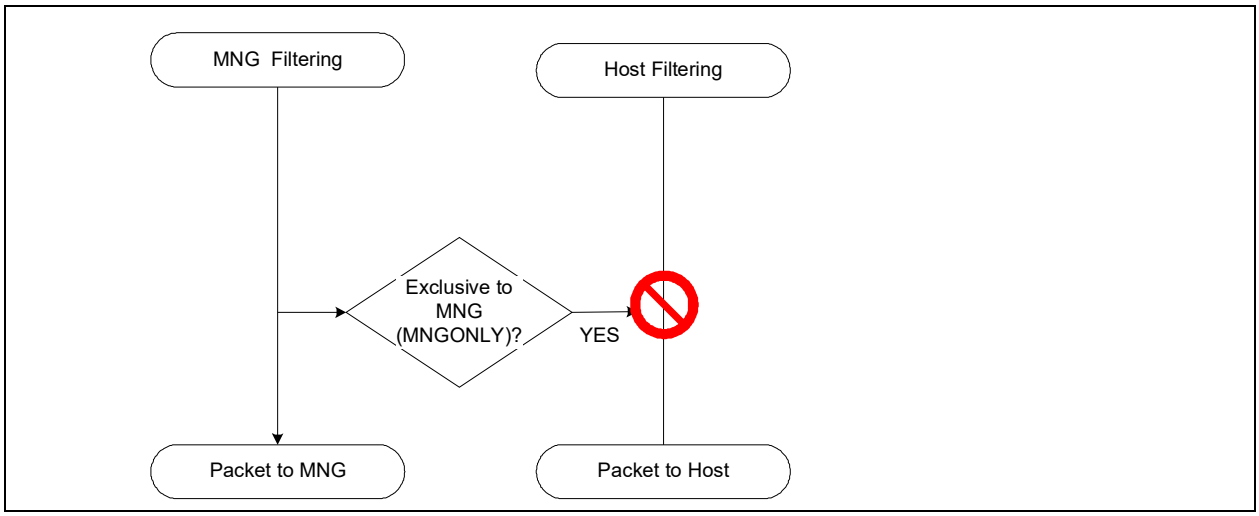


Figure 6.4. Manageability / Host Filtering

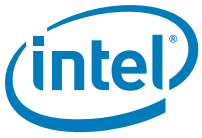
6.1.3 Rx Queues Assignment

The following filters/mechanisms determine the destination of a received packet. These filters are described briefly while more detailed descriptions are provided in the following sections:

- Virtualization — In a virtual environment, DMA resources are shared between more than one software entity (operating system and/or device driver). This is done by allocating receive descriptor queues to virtual partitions (VMM, IOVM, VMs, or VFs). Allocating queues to virtual partitions is done in sets, each with the same number of queues, called queue pools, or pools. Virtualization assigns to each received packet one or more pool indices. Packets are routed to a pool based on their pool index and other considerations such as RSS. See Section 6.1.3.2 for more on routing for virtualization.
- Receive Side Scaling (RSS) — RSS distributes packet processing between several processor cores by assigning packets into different descriptor queues. RSS assigns to each received packet an RSS index. Packets are routed to one queue from a set of Rx queues based on their RSS index and other considerations such as virtualization. See Section 6.1.3.6 for details.
- L2 Ethertype Filters — These filters identify packets by their L2 Ethertype and assigns them to receive queues. Examples of possible uses are LLDP packets, and 802.1X packets. See Section 6.1.3.3 for details. The integrated 10 GbE LAN controller incorporates eight Ethertype filters.
- Flow Director Filters — These filters provide up to additional 32 K filters. See Section 6.1.3.5 for details.
- TCP SYN Filters — The integrated 10 GbE LAN controller 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 6.1.3.4 for details.

A received packet is allocated to a queue based on the above criteria and the following order:

- Queue by L2 Ethertype filters (if match)



- Queue by SYN filter (if match)
- Queue by flow director filters
- Queue (in case of virtualization, within a pool) by RSS as described in Section 6.1.3.1 and Section 6.1.3.2.
- Send to queue zero.

6.1.3.1 Queuing in a Non-virtualized Environment

Table 6.1 lists the queuing schemes for packets that do not match any special filters (L2 Ethertype, SYN and flow director filters). Table 6.2 shows the queue indexing. Selecting a scheme is done via the *Multiple Receive Queues Enable* (*MRQE*) field in MRQC register.

Table 6.1. Rx Queuing Schemes Supported (No Virtualization)

RSS	RSS Queues	Special Filters (1)
No	1 queue Rx queue 0	Supported
Yes	64 RSS queues	Supported
No	8 TCs x 1 queue 4 TCs x 1 queue Assign Rx queue 0 of each TC	Supported
Yes	8 TCs x 16 RSS 4 TCs x 32 RSS	Supported

Table 6.2. Queue Indexing Illustration in Non-virtualization Mode

MRQC.MRQE	Queue Index bits	6	5	4	3	2	1	0
0000b	Default only	0						
0001b	RSS		RSS					
0101b	Reserved	TC		RSS[1]				
0011b	Reserved	TC		0				
0100b	Reserved	TC			RSS[1]			
0010b	Reserved	TC			0			

1. The number of bits used for each TC is set according to the *RQTC.RQTCx* fields

A received packet is assigned to a queue according to the ordering shown in Figure 6.5:

- RSS filters — Packets that do not meet any of the previous filtering conditions described in Section 6.1.3 are assigned to one of 128 queues as listed in Table 6.1. The following modes are supported:
  - — No RSS — Queue 0 is used for all packets.
  - — RSS — A set of 64 queues is allocated for RSS. The queue is identified through the RSS index. Note that it is possible to use a subset of these queues.
- 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 6.5 depicts the flow of allocation of Rx queues by the various queue filters previously described:

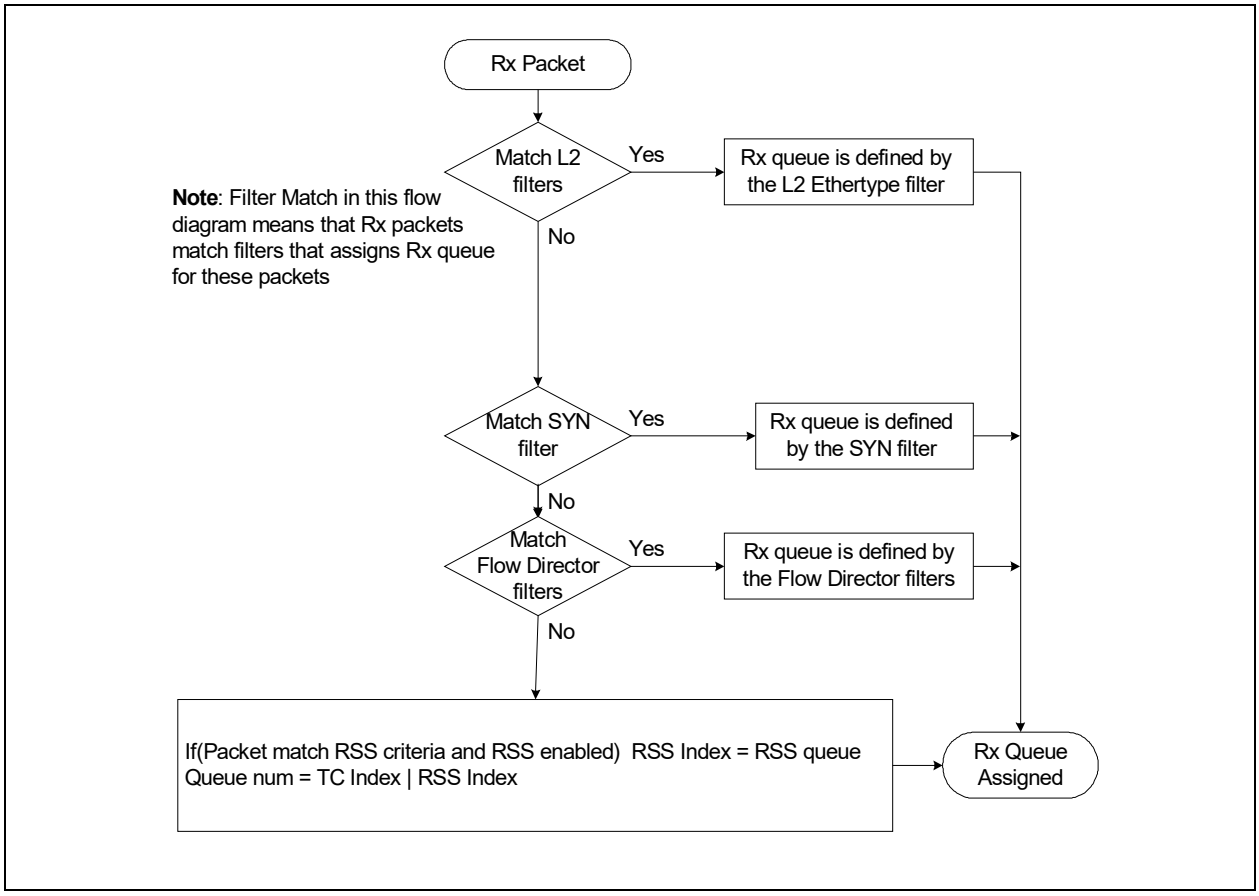


Figure 6.5. Rx Queuing Flow (Non-virtualized)

6.1.3.2 Queuing in a Virtualized Environment

The 128 Rx queues are allocated to a pre-configured number of queue sets, called pools. In non-IOV mode, system software allocates the pools to the VMM or to VMs. In IOV mode, each pool may be associated with a VF.

Incoming packets are associated with pools based on their L2 characteristics as described in Section 6.6.3. 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.

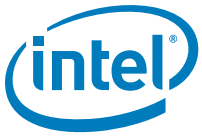


Table 6.3 lists the queuing schemes supported with virtualization. Table 6.4 lists the queue indexing.

Table 6.3. Rx Queuing Schemes Supported with Virtualization

RSS	RSS Queues	Special Filters (1)
No	16 pools x 1 queue 32 pools x 1 queue 64 pools x 1 queue - Rx queue 0 of each pool	Supported
Yes	32 pools x 4 RSS 64 pools x 2 RSS	Supported
No	16 pools x 8 TCs 32 pools x 4 TCs	Supported
Yes	N/A	N/A

Table 6.4. Queue Indexing Illustration in Virtualization Mode

MRQC.MRQE	Queue Index bits	6	5	4	3	2	1	0
1000b		Pool Index						0
1011b	VT(64) + RSS	Pool Index						RSS
1001b/1111b	VT(64) + RSS + Mega Pool	Pool Index					VF Index/ RSS[1]	RSS
1010b	VT(32) + RSS	Pool Index					RSS	
N/A	VT(16) + RSS	Not Supported						
1101b	Reserved	Pool Index					TC	
1100b	Reserved	Pool Index				TC		

1. When *MRQC.MRQE* = 0xF, VF index for pools 0-59, part of RSS for pool 60, 62 and pools 61 and 63 are disabled.When *MRQC.MRQE* = 0x0, VF index for pools 0-61, part of RSS for pool 62 and pool 63 is disabled.

Selecting a scheme is done in the following manner:

- Non-IOV mode
  - Select one of the above schemes via the *Multiple Receive Queues Enable* field in the *MRQC* register.
- IOV mode
  - Determine the number of pools: the number must support the value configured by the operating system in the PCIe NumVFs field. Therefore, the number of pools is min. of {16, 32, 64} that is still >= NumVFs.

A received packet is assigned to an absolute queue index according to the ordering shown in Figure 6.6). It is software responsibility to define a queue that belongs to the matched pool:

- RSS filters — The supported modes are listed in Table 6.3 and detailed as follows. The associated queue indexes are listed in Table 6.4.
  - No RSS — A single queue is used as default queue of the pool with either 16, 32, or 64 pools enabled. In a 64 pools setting, queues '2xN'...'2xN+1' are allocated to pool 'N'; In a 32 pools

setting, queues '4xN'...'4xN+3' are allocated to pool 'N'. The queues not used as default may be directed to by other filters.

— RSS — All 128 queues are allocated to pools. Two configurations supported:

- 32 pools with 4 RSS queues each
- 64 pools with 2 RSS queues each.
- 63 pools. The first 62 with 2 queues each and the last one with 4 RSS queues.In this mode the last pool is 62 and pool 63 is disabled.
- 62 pools. The first 60 with 2 queues each and the last two with 4 RSS queues each. In this mode the last two pools are 60 and 62 and pools 61/63 are disabled.

Notes: It is possible to use a subset of the RSS queues in each pool. The LS bits of the queue indexes are defined by the RSS index, and the pool index is used as the MS bits.See description below.

In the 62 and 63 pools modes, the pools with 4 queues should be assigned to the PF and can not be assigned to a VF.

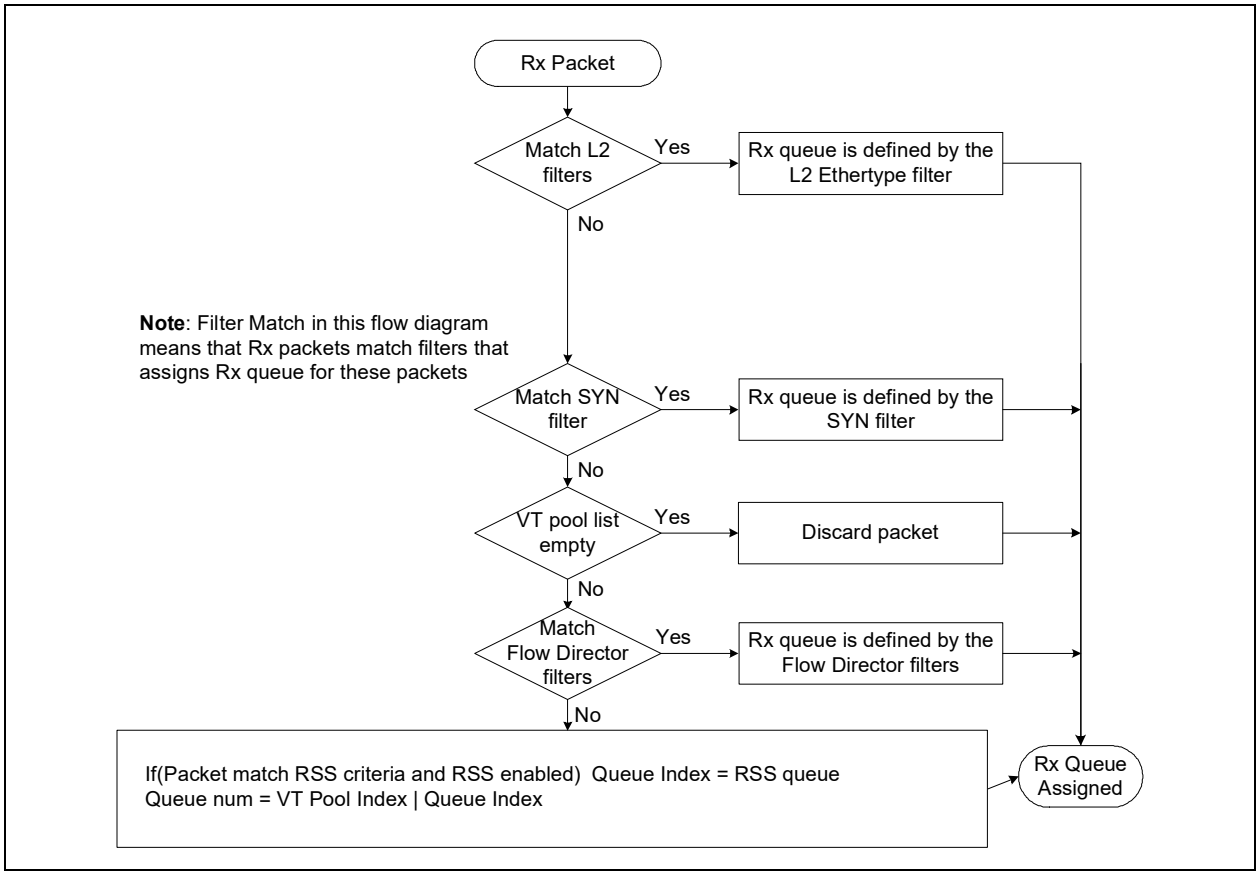
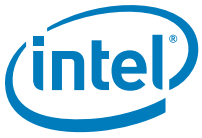


Figure 6.6. Rx Queuing Flow (Virtualization Case)

6.1.3.3 L2 Ethertype Filters

These filters identify packets by their L2 Ethertype, 802.1Q user priority and optionally assign them to a receive queue. The following possible usages have been identified at this time:



- IEEE 802.1X packets — Extensible Authentication Protocol (EAPOL) over LAN
- Time sync packets (such as IEEE 1588) — Identifies Sync or Delay_Req packets
- The L2 type filters should not be set to IP packet type as this might cause unexpected results

The integrated 10 GbE LAN controller 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 6.1.5.2.2 for a description of the *Packet Type* field.

The following flow is used by the Ethertype filters:

1. If the *ETQF.Filter Enable* bit is cleared, the filter is disabled and the following steps are ignored.
2. Receive packet matches any *ETQF* filters if the *EtherType* field in the packet matches the *EType* field of the filter. Note that the following steps are ignored if the packet does not match the *ETQF* filters.

Note: Ethertype Filters should not be configured in a way that may cause a packet to match multiple filters.

3. Packets that match any *ETQF* filters are candidate for the host. If the packet also matches the manageability filters, it is directed to the host as well regardless of the *MNGONLY* register setting.
4. If the *ETQF.1588 Time Stamp* field is set, the packet is identified as an IEEE 1588 packet.
5. If the *ETQS.Queue Enable* bit is cleared, the filter completed its action on the packet. Else, the filter is also used for queuing purposes as described in the sections that follow.
6. If the *ETQF.Pool Enable* field is set, the *Pool* field of the filter determines the target pool for the packet. The packet can still be mirrored to other pools as described in Section 6.6.3. See the sections that follow for more details on the use of the *Pool* field.
7. The *ETQS.RX Queue* field determines the destination queue for the packet. In case of a mirrored packet, only the copy of the packet that is targeted to the pool defined by the *Pool* field in the *ETQF* register is routed according to the *Rx Queue* field.

Setting the *ETQF[n]* registers is described as follows:

- The *Filter Enable* bit enables identification of Rx packets by Ethertype according to this filter. If this bit is cleared, the filter is ignored.
- The *EType* field contains the 16-bit Ethertype compared against all L2 type fields in the Rx packet.
- The *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.

Special considerations for virtualization modes:

- Packets that match an Ethertype filter are diverted from their original pool (as defined by the VLAN and Ethernet MAC address filters) to the pool defined in the *Pool* field in the *ETQF* registers.
- The same applies for multicast packets. A single copy is posted to the pool defined by the filter.
- Mirroring rules
  - A packet sent to a pool by an *ETQF* filter, is still candidate to mirroring using the standard mirroring rules.
  - The Ethertype filter does not take part in the decision on the destination of the mirrored packet.

6.1.3.4 SYN Packet Filters

The integrated 10 GbE LAN controller 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 same filter can be used to forward Geneve packets with the *OAM* bit set (control packets) to a dedicated queue irrespective of the flow director forwarding.

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 SYN_OAM_SELECT bit defines if the filter is to be used for SYN packets or OAM packets (0b = SYN, 1b = OAM)
- 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.

Note: OAM forwarding is not supported for loopback traffic, assuming a VMM does not send OAM packets to itself.

6.1.3.5 Flow Director Filters

The flow director filters identify specific flows or sets of flows and routes them to specific queues. The flow director filters are programmed by FDIRCTRL and all other FDIR registers. The integrated 10 GbE LAN controller shares the Rx packet buffer for the storage of these filters.

The integrated 10 GbE LAN controller supports three different modes of flow director.

- IP filtering (000b)
- MAC, VLAN filtering (001b)
- Cloud: NVGRE or VXLAN filtering (010b)

The following table lists the packets that are candidate to compared to the flow director filters according to the different modes:

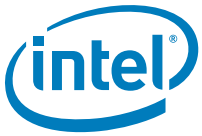


Table 6-5. Candidate for Flow Director per mode

Mode	IP packets TCP/ UDP/SCTP (not tunneled)	Non IP packets	NVGRE packets	VXLAN packets	Other IP packets or Fragmented IP
00 - IP (L4 fields unmasked)	Candidate	Not Candidate	Candidate (inner IP)[1]	Candidate (inner IP)[1]	Not Candidate
00 -IP (L4 fields masked)	Candidate	Not Candidate	Candidate (inner IP)[2]	Candidate (inner IP)[2]	Candidate[2,3]
01 - MAC VLAN	Candidate	Candidate	Candidate	Candidate	Candidate
10 - Cloud	Not Candidate	Not Candidate	Candidate	Candidate	Not Candidate

1. Only if a known L4 header is present (TCP/UDP/SCTP).
2. VXLAN and NVGRE packets without inner IP header (e.g. tunneled ARP) are not candidate in IP mode.
3. IP in IP packets are not candidate in IP mode.

Basic rules for the flow director filters are:

In VT mode, the *Pool* field in FDIRCMD must be valid. If the packet is replicated, only the copy that goes to the pool that matches the *Pool* field is impacted by the filter.The flow director filters cover the following fields:

Table 6-6. Lookup fields for Flow Director per mode

Mode	Lookup Fields
00 - IP	• VLAN header • Source IP and destination IP addresses • Source port and destination port numbers • IPv4 / IPv6[1] and UDP / TCP or SCTP protocol match • Flexible 2-byte tuple anywhere in the first 64 bytes of the packet • Target pool number (relevant only for VT mode)
01 - MAC VLAN	• MAC • VLAN • Flexible 2-byte tuple anywhere in the first 64 bytes of the packet • Target pool number (relevant only for VT mode)
10 - Cloud	• Tunnel type - NVGRE or VXLAN • TNI (NVGRE) / VNI (VXLAN) • Inner MAC • Inner VLAN • Flexible 2-byte tuple anywhere in the first 64 bytes of the packet • Target pool number (relevant only for VT mode)

1. IPv6 extended headers are parsed by the integrated 10 GbE LAN controller, enabling TCP layer header recognition. Still the IPv6 extended header fields are not taken into account for the queue classification by Flow Director filter. This rule do not apply for security headers and fragmentation header. Packets with fragmentation header miss this filter. Packets with security extended headers are parsed only up to these headers and therefore can match only filters that do not require fields from the L4 protocol.

The integrated 10 GbE LAN controller supports two types of filtering modes (static setting by the FDIRCTRL.Perfect-Match bit):

- Perfect match filters — The hardware checks a match between the masked fields of the received packets and the programmed filters. Masked fields should be programmed as zeros in the filter context. The integrated 10 GbE LAN controller supports up to 8 K - 1 perfect match filters.
- Signature filters — The hardware checks a match between a hash-based signature of the masked fields of the received packet. The integrated 10 GbE LAN controller supports up to 32 K - 1 signature filters. This mode can be used only when the filtering mode is IPMODE (*FDIRCTRL.FILTERMODE* = 000b).

- Denote — The *Perfect Match* fields and *Signature* field are denoted as *Flow ID* fields.

The integrated 10 GbE LAN controller supports masking / range for the previously described fields. These masks are defined globally for all filters in the FDIR…M registers.

- The following fields can be masked per bit enabling power of two ranges up to complete enable / disable of the fields: IPv4 addresses and L4 port numbers.
- The following fields can be masked per byte enabling lower granularity ranges up to complete enable / disable of the fields: IPv6 addresses; Inner MAC; Outer MAC; TNI/VNI field.

Note: In perfect match filters the destination IPv6 address can only be compared as a whole (with no range support) to the *IP6AT* filters. A match to any of the *IP6AT* filter is considered as an IPv6 destination match.

- The following fields can be either enabled or disabled completely for the match functionality: VLAN ID tag; outer VLAN priority + DEI bit (the user priority is taken from the outermost tag with UP bits if an inner VLAN exists, otherwise it is ignored); inner or outer VLAN tag; flexible 2-byte tuple and target pool. Target pool can be enabled by software only when VT is enabled as well.

Flow director filters have the following functionality in virtualization mode:

- Flow director filters are programmed by the registers in the PF described in Section 6.1.3.5.13 and Section 6.1.3.5.14.
- Flow director filters can be utilized in virtualization mode to filter on MAC-VLAN by setting *FDIRCTRL.FILTERMODE* = *MACVLANMODE* (001b). *MACVLANMODE* is only used in flow director perfect match mode.

6.1.3.5.1 Flow Director Filters Actions

Flow director filters might have one of the following actions programmed per filter in the *FDIRCMD* register:

- Drop packet or pass to host as defined by the *Drop* bit.
  - Matched packets to a flow director filter is directed to the assigned Rx queue only if the packet does not match the L2 filters for queue assignment nor the SYN filter for queue assignment.
  - Packets that match a filter are directed to the Rx queue defined in the filter context as programmed by the *FDIRCMD.Rx-Queue*. The Rx-Queue field is an absolute receive queue index. In a non-VT setting, it can be programmed to any value. In VT mode, the software should set the Rx-Queue to an index that belongs to the matched pool.
  - Packets that match drop filters are directed to the Rx queue defined per all filters in the *FDIRCTRL.DROP-Queue*. The integrated 10 GbE LAN controller drops these packets if software does not enable the specific Rx queue.

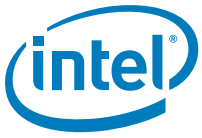
6.1.3.5.2 Flow Director Default Action

A default drop action may be applied by the flow director for packets that are candidate to flow director as defined in Table 6-5. If the *FDIRCTRL.DROP_NO_MATCH* bit is set, any packet candidate for flow director that does not match any of the filters will be sent to the queue defined in the *FDIRCTRL.DROP_QUEUE* field.

6.1.3.5.3 Flow Director Filters Status Reporting

Shared status indications for all packets:

- The integrated 10 GbE LAN controller increments the *FDIRMATCH* counter for packets that match a flow director filter. It also increments the *FDIRMISS* counter for packets that do not match any flow director filter.



- The *Flow Director Filter Match* (*FLM*) bit in the *Extended Status* field of the Rx descriptor is set for packets that match a flow director filter.
- The flow ID parameters are reported in the *Flow Director Filter ID* field in the Rx descriptor if enabled by the *FDIRCTRL.Report-Status*. When the *Report-Status* bit is set, the *RXCSUM.PCSD* bit should be set as well. This field is indicated for all packets that match or do not match the flow director filters.
  - For packets that do not match a flow director filter, if the *FDIRCTRL.REPORT_STATUS_ALWAYS* is set, the *Flow Director Filter ID* field can be used by software for future programming of a matched filter, otherwise, the RSS hash value is reported. Table 6-7 describes the value of the RSS_TYPE field in the receive descriptor in the different cases.

Table 6-7. RSS Type values according to Flow Director Match

REPORT_STATUS (FDIRCTRL[5])	REPORT_STATUS_ALWAYS (FDIRCTRL[7])	Packet matches a Flow Director Entry	RSS Type field value
0	0	X	RSS Type
1	0	0	RSS Type
1	0	1	0xF
0	1	X	illegal configuration
1	1	X	0xF

For packets that match a flow director filter, the *Flow Director Filter ID* field can be used by software to identify the flow of the Rx packet.Too long linked list exception (linked list and too long terms are illustrated in Figure 6.7):

- The maximum recommended linked list length is programmed in the *FDIRCTRL.Max-Length* field
- The length exception is reported in the *FDIRErr* field in the Rx descriptor
- Packets that do not match any flow director filter, reports this exception if the length of the existing linked list is above the maximum recommended length. Software can use it to avoid further programming of additional filters to this linked list before other filters are removed.
- Packets that match a pass filter report this exception if the distance of the matched filter from the beginning of the linked list is higher than the above recommended length.
- Packets that match a drop filter are posted to the Rx queue programmed in the filter context instead of the global *FDIRCTRL.DROP_QUEUE*. The drop exception is reported in addition to the length exception (in the same field in the Rx descriptor).

Collision exception:

- Packets that matches a collided filter report this exception in the *FDIRErr* field in the Rx descriptor.
- Collision events for signature-based filters should be rare. Still it might happen because multiple flows can have the same hash and signature values. Software might leave the setting as is while the collided flows are handled according to the actions of the first programmed flow. Only one flow (out of the collided ones) might remain in the flow director filters. 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.

6.1.3.5.4 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 6.1.3.5.6).

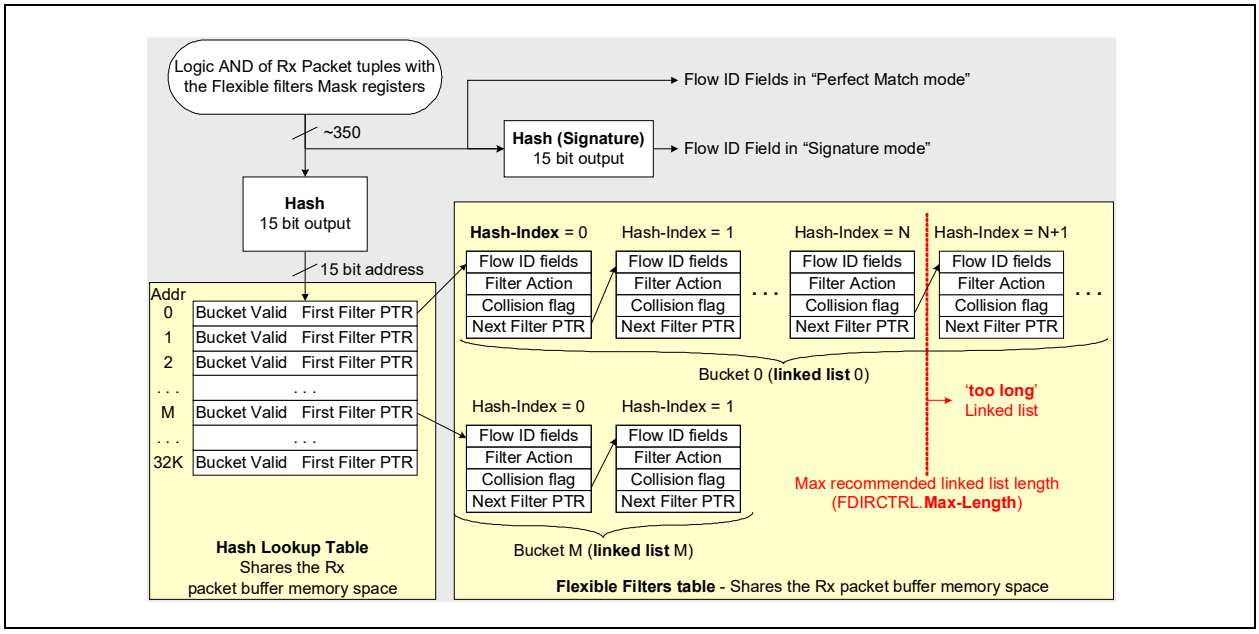


Figure 6.7. Flow Director Filters Block Diagram

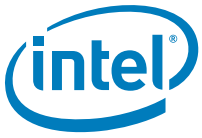
6.1.3.5.5 Rx Packet Buffer Allocation

Flow director filters can consume zero space (when disabled) up to ~256 KB of memory. As shown in Figure 6.7, flow director filters share the same memory with the Rx packet buffer. By setting the *PBALLOC* field in the *FDIRCTRL* register, the software can enable and allocate memory for the flow director filters. The memory allocated to received traffic is the remaining part of the Rx packet buffer.

Table 6.8. Rx Packet Buffer Allocation

PBALLOC (2)	Effective Rx Packet Buffer Size (see following note)	Flow Director Filters Memory	Supported Flow Director Filters			
			Signature		Perfect Match	
			Filters	Bucket Hash	Filters	Bucket Hash
00 Flow Director is disabled	384 KB	0	0	N/A	0	N/A
01	320 KB	64 KB	8 K -1 filters	13 bit	2 K -1 filters	11 bit
10	256 KB	128 KB	16 K -1 filters	14 bit	4 K -1 filters	12 bit
11	128 KB	256 KB	32 K -1 filters	15 bit	8 K -1 filters	13 bit

Note: It is the user's responsibility to ensure that sufficient buffer space is left for reception of traffic. The required buffer space for receive traffic depends on the number of traffic classes, and flow control upper and lower threshold values. If flow director is enabled (*PBALLOC* > 0), software should set the *RXPBSIZE[n]* registers according to the total remaining part of the Rx packet buffer for reception of traffic.



Refer to Section 2.8.2.3.5 for recommended setting of the Rx packet buffer sizes and flow control thresholds.

6.1.3.5.6 Flow Director Filtering Reception Flow

- Rx packet is digested by the filter unit which parse the packet extracting the relevant tuples for the filtering functionality.
- The integrated 10 GbE LAN controller 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 6.1.3.5.16.
- The address in the hash lookup table points to the selected linked list of flow director filters.
- The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller checks the linked list until it reaches the last filter in the linked list or until a matched filter is found.
- Case 1: matched filter is found:
  - Increment the *FDIRMATCH* statistic counter.
  - Process the filter's actions (queue assignment) according to queue assignment priority. Meaning, the actions defined in this filter takes place only if the packet did not match any L2 filter or SYN filter that assigns an Rx queue to the packet.
  - Rx queue assignment according to the filter context takes place if *Queue-EN* is set.
  - If the DROP bit is set in the filter context, the packet is sent to the queue pointed by the *FDIRCTRL.DROP_QUEUE* field.
  - Post the packet to host including the flow director filter match indications as described in Section 6.1.3.5.3.
- Case 2: matched filter is not found:
  - Increment the *FDIRMISS* statistic counter.
  - Post the packet to host including the flow director filter miss indications as described in Section 6.1.3.5.3.

6.1.3.5.7 Add Filter Flow

The software programs the filters parameters in the registers described in Section 6.1.3.5.13 and Section 6.1.3.5.14 while keeping the *FDIRCMD.Filter-Update* bit inactive. As a result, the integrated 10 GbE LAN controller checks the bucket valid indication in the hash lookup table (that matches the *FDIRHASH.Hash*) for the presence of an existing linked list. Following are the two programming flows for the case a link list exists or for the case a new list is required.

- Case 1: Add a filter to existing linked list:
  The integrated 10 GbE LAN controller checks the linked list until it reaches the last filter in the list or until a matched filter is found. The following cases may occur:
  - Matched filter is found (equal flow ID) with the same action parameters — The programming is discarded silently. This is a successful case since the programmed flow is treated as requested.
  - Matched filter is found (equal flow ID) with different action parameters — The integrated 10 GbE LAN controller keeps the old setting of the filter while setting the *Collision* flag in the filter context (see Section 6.1.3.5.3 for software handling of collision during packet reception).
  - Matched filter is found (equal flow ID) with different action parameters and the *Collision* flag is already set — The programming is discarded silently. Software gets the same indications as the previous case.
  - Matched filter is not found (no collision) — The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller 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.

6.1.3.5.8 Update Filter Flow

In some applications, it is useful to update the filter parameters, such as the destination Rx queue. Programing filter parameters is described in Section 6.1.3.5.7.

Setting the *Filter-Update* bit in the *FDIRCMD* register has the following result:

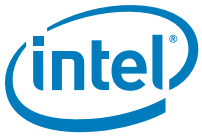
- Case 1: Matched filter does not exist in the filter table — Setting the *Filter-Update* bit has no impact and the command is treated as add filter.
- Case 2: Matched filter already exists in the filter table — Setting the *Filter-Update* bit enables filter parameter's update while keeping the collision indication as is.

When updating an existing filter the software device driver should program the same filter (i.e. the same FDIRHASH.HASH and the same *FDIRHASH.SIGNATURE_SW_INDEX*) while keeping the *FDIRCMD.FILTER_UPDATE* = 1

6.1.3.5.9 Remove Filter Flow

Software programs the filter Hash and Signature / Software-Index in the *FDIRHASH* register. It then should set the *FDIRCMD.CMD* field to *Remove Flow*. Software might use a single 64-bit access to the two registers for atomic operation. As a result, the integrated 10 GbE LAN controller 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.

6.1.3.5.10 Remove all Flow Director Filters

In some cases there is a need to clear the entire flow director table. It might be useful in some applications that might cause the flow director table becoming too occupied. Then, software might clear the entire table enabling its re-programming with new active flows.

Following are steps required to clear the flow director table:

- Poll the *FDIRCMD.CMD* until it is zero indicating any previous pending commands to the flow director table is completed (at worst case the *FDIRCMD.CMD* should be found cleared on the second read cycle). Note that software must not initiate any additional commands (add / remove / query) before this step starts and until this flow completes.
- Clear the *FDIRFREE* register (set *FREE* field to zero).
- Set *FDIRCMD.CLEARHT* to 1b and then clear it back to 0b
- Clear the *FDIRHASH* register to zero
- Re-write *FDIRCTRL* by its previous value while clearing the *INIT-Done* flag.
- Poll the *INIT-Done* flag until it is set to 1b by hardware.
- Clear the following statistic registers: *FDIRUSTAT; FDIRFSTAT; FDIRMATCH; FDIRMISS; FDIRLEN* (note that some of these registers are read clear and some are read write).

6.1.3.5.11 Flow Director Filters Initializing Flow

Following a device reset, the flow director is enabled by programming the FDIRCTRL register, as follows:

- Set *PBALLOC* to non-zero value according to the required buffer allocation to reception and flow director filter (see Section 6.1.3.5.5). 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 integrated 10 GbE LAN controller is in a sleep state).

6.1.3.5.12 Query Filter Flow

Software might query specific filter settings and bucket length using the Query command.

- Program the filter Hash and Signature/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 integrated 10 GbE LAN controller 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[1]
Found Perfect Match Filter	1	1	Filter index within the linked list	Filter's parameters	Filter's parameters

1. The pool parameter is not returned as part of the filter action in signature mode.

6.1.3.5.13 Signature Filter Registers

The signature flow director filter is programmed by setting the *FDIRHASH* and *FDIRCMD* registers. These registers are located in consecutive 8-byte aligned addresses. Software should use a 64-bit register to set these two registers in a single atomic operation. Table 6.9 lists the recommended setting.

Table 6.9. Signature Match Filter Parameters

Filter Bucket Parameters — FDIRHASH	
Hash	15-bit hash function used to define a bucket of filters. This parameter is part of the flow director filter ID that can be reported in the Rx descriptor. It is shared for signature and perfect match filters.
Valid	Should be set to 1b. It is shared for signature and perfect match filters.
Flow ID — FDIRHASH	
Signature	15-bit hash function used as the flow matching field. This parameter is also part of the flow director filter ID that can be reported in the Rx descriptor.
FDIRCMD — Programming Command and Filter action	

6.1.3.5.14 Perfect Match Filter Registers

Perfect match filters are programmed by the following registers: *FDIRSIPv6[n]; FDIRVLAN; FDIRPORT; FDIRIPDA; FDIRIPSA; FDIRHASH; FDIRCMD*. Setting the *FDIRCMD* register, generates the actual programming of the filter. Therefore, write access to this register must be the last cycle after all other registers contain a valid content. Table 6.10 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 6.10. Perfect Match Filter Parameters

Filter Bucket Parameters and Software Index — FDIRHASH	
Hash	See Section 6.1.3.5.13.
Valid	See Section 6.1.3.5.13.
Software-Index	16-bit index provided by software at filter programming used by software to identify the matched flow. This parameter is also part of the flow director filter ID that can be reported in the Rx descriptor.
FDIRCMD — Programming Command and Filter Action Set	

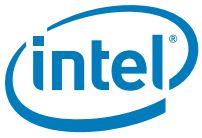


Table 6.10. Perfect Match Filter Parameters

	IP Mode FILTERMODE = 000b	MAC, VLAN mode FILTERMODE = 001b	Cloud modes FILTERMODE = 010b
IPV6DMATCH	Should be set for IPv6 filters only and only if the destination address should be compared to one of the IPAT filters.	Should be cleared	
L4TYPE	Defines if the filter is for TCP, UDP or SCTP flows.	Should be cleared FDIRM.L4P should be set	
IPV6	Defines if the filter is for IPv4 or IPv6 addresses.	Should be cleared FDIRM.L3P should be set FDIRM.DIPV6 should be set	
TUNNEL_FILTER	Defines if the filter is a VXLAN and NVGRE tunneled packet filter. If set, the parameters of the L3 and L4 header relates to the tunneled (inner) header.	Should be cleared.	
POOL	Pool to which the packet is directed (relevant only in virtualization modes).		
Flow ID — Perfect Match Flow ID Parameters are Listed in the Following Registers and Fields			
	IP Mode FILTERMODE = 000b	**MAC, VLAN mode FILTERMODE = 001b**	**Cloud mode FILTERMODE = 010b**
FDIRSIPv6[0…2].IP6SA	Three LS DWord of the source IPv6. Meaningful for IPv6 flows depending on the FDIRIP6M.SIPM setting.	{FDIRSIPv6_0[31:0], FDIRSIPv6_1[15:0]} holds the destination MAC address value FDIRSIPv6_1[31:16], and FDIRSIPv6_2[31:0] are reserved and should be set to 0x0. The FDIRIP6M.SIPM field should be set to 0x0xFC0F	FDIRSIPv6_2[31:0] holds the 32 bits of the TNI/VNI field. {FDIRSIPv6_0[31:0], FDIRSIPv6_1[15:0]} holds the destination MAC address value FDIRSIPv6_1[31:30] holds the cloud mode (00 = NVGRE, 10 = VXLAN, 01 = Geneve, 11 = Reserved). FDIRSIPv6_1[29:16] are reserved and should be set to 0x0. In order to mask one of the fields the matching bits in FDIRIP6M.SIPM should be set: • Bits 15:12 control the masking of the TNI/VNI field. All the 32 bits of the key can be used for filtering. • Bit 11 controls the masking of the cloud mode and should be cleared for normal operation • Bits 3:0 and bit 11 should be set. • Bits 9:4 control the masking of the inner MAC address.
FDIRVLAN.VLAN	VLAN fields are meaningful depending on the FDIRM.VLANID and FDIRM.VLANP setting. The VLAN field should be stored in Little endian format.		Inner VLAN fields are meaningful depending on the FDIRM.VLANID and FDIRM.VLANP setting.The VLAN field should be stored in Little endian format.
FDIRVLAN.FLEX	Flexible 2-byte field at offset FDIRCTRL.Flex-Offset. Meaningful depending on FDIRM.FLEX setting.		

Table 6.10. Perfect Match Filter Parameters

FDIRPORT.Source	L4 source port. Meaningful for TCP, UDP and SCTP packets depending on the FDIRTCPM.SportM, FDIRUDPM.SportM and FDIRSCTPM.SportM setting.	Reserved - Should be zero. FDIRM.L4P should be set.
FDIRPORT.Destination	L4 destination port. Meaningful for TCP, UDP and SCTP packets depending on the FDIRTCPM.DportM, FDIRUDPM.DportM and FDIRSCTPM.DPORTM setting.	Reserved - Should be zero. FDIRM.L4P should be set.
FDIRIPDA.IP4DA	IPv4 destination address. Meaningful depending on the FDIRDIP4M.IPM setting.	Reserved - Should be zero. All bits in FDIRDIP4M.IPM should be set.
FDIRIPSA.IP4SA	IPv4 source address or LS DWord of the source IPv6 address. Meaningful for IPv4 flows depending on the FDIRSIP4M.IPM setting and for IPv6 flows depending on the FDIRIP6M.SIPM setting.	Reserved - Should be zero. All bits in FDIRDSIP4M.IPM should be set. FDIRIP6M.SIPM[3:0] should be set.

6.1.3.5.15 Multiple CPU Cores Considerations

Perfect match filters programming and any query cycles require 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.

6.1.3.5.16 Flow Director Hash Function

The integrated 10 GbE LAN controller supports programmable 16-bit hash functions based on two 32-bit keys, one for the lookup table identifying a bucket of filters and another one for the signature (*FDIRHKEY* and *FDIRSKEY*). The hash function is described in the sections that follows. In some cases, a smaller hash value than 16 bits is required. In such cases, the LS bits of the hash value are used.

```
For (i=0 to 350) {if (Ext_K[i]) then Hash[15: 0] = Hash[15: 0] XOR Ext_S[15+i: i]}
```

While using the following notations:

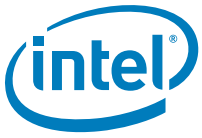
'XOR' - Bitwise XOR of two equal length strings

If (xxx) - Equals ‘true’ if xxx = ‘1’ and equals ‘false’ if xxx = ‘0’

S[335:0] - The input bit string of the flow director tuples: 42 bytes listed in Table 6.11 AND-logic with the filters masks.

```
Ext_S[n] = S[14:0] | S[335:0] | S[335:321] // concatenated
```

K[31:0] - The hash key as defined by the FDIRHKEY or FDIRSKEY registers.



Tmp_K[11*32-1:0] - (Temp Key) equals K[31:0] | K[31:0]... // concatenated Key 11 times

Ext_K[350:0] - (Extended Key) equals T_K[351:1]

The input bit stream for the hash calculation is listed in Table 6.11 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 6.11. Input Bit Stream for Hash Calculation

	Field		
Bytes/Mode	**IP**	**MAC, VLAN**	**Cloud Mode**
Bytes 0…1	VLAN tag (always the outer VLAN)		Inner VLAN Tag
Bytes 2…5	Source IP (16 bytes for IPv6; source IP for IPv4 | 12 bytes of zero's)	Zero	
Bytes 6...11		MAC	Inner MAC
Bytes 12...13		Zero	8 bits of zeros | bit of tunnel_type | 7 bits of zeros
Bytes 14...17		Zero	VNI/TNI
Bytes 18…33	Signature Mode: Destination IP (16 bytes for IPv6; destination IP for IPv4 | 12 bytes of zero's) Exact Mode: Destination IP (7 zero bits |IP6AT match indication | 120 zero bits; destination IP for IPv4 | 12 bytes of zero's)[1]		
34...37	L4 source port number | L4 destination port number		
38...39	Flexible bytes		
40	00b | pool number (as defined by FDIRCMD.Pool)		
41	[7:5] 000b [4] FDIRCMD.TUNNEL_FILTER (should be zero in MAC, VLAN mode). [3] 0b [2] IPv6/IPv4 type (FDIRCMD.IPV6) [1:0] L4 type (FDIRCMD.L4TYPE)		

1. In VXLAN and NVGRE packets, the IP addresses used are of the tunneled header in other packets the outer IP header is used.

6.1.3.6 RSS

RSS is a mechanism to distribute received packets into several descriptor queues. Software can then assign each queue to a different processor, therefore sharing the load of packet processing among several processors.

As described in Section 6.1, the integrated 10 GbE LAN controller uses RSS as one ingredient in its packet assignment policy (the others are the various filters and virtualization). The RSS output is an RSS index. The integrated 10 GbE LAN controller global assignment uses these bits (or only some of the LSBs) as part of the queue number.

Figure 6.8 and Figure 6.9 show the process of computing an RSS output:

1. The receive packet is parsed into the header fields used by the hash operation (such as IP addresses, TCP port, etc.)
2. A hash calculation is performed. The integrated 10 GbE LAN controller supports a single hash function, as defined by Microsoft* (MSFT) RSS. The integrated 10 GbE LAN controller therefore does not indicate to the device driver which hash function is used. The 32-bit result is fed into the *RSS Hash* field in the packet receive descriptor.
3. The integrated 10 GbE LAN controller supports two modes of RSS defined by the *MRQC.Multiple_RSS* bit. When set to 0b a single RSS key and redirection table are supported. In this mode (usually used when virtualization is not enabled) the nine LSBs of the hash result are used as an index into a 512-entry redirection table. Each entry provides up to 6-bit RSS output index (Figure 6.8).

For SRIOV or VMDq enabled mode (set by MRQC.Multiple_RSS set to 1b), The integrated 10 GbE LAN controller supports up to 64 (one per pool) RSS keys and redirection tables (both are controlled and programmed at the VF space). In this mode the 6 LSBs of the hash result are used as an index into a 64-entry redirection table. Each entry provides up to 2-bit RSS output index (Figure 6.9).

When RSS is enabled, the integrated 10 GbE LAN controller provides software with the following information as required by Microsoft RSS and provided for device driver assist:

- A Dword result of the MSFT RSS hash function application to the packet header, to be used by the stack for flow classification, is written into the receive packet descriptor. A 4-bit RSS *Type* field conveys the hash function used for the specific packet.
- Packets to which the RSS function can not be applied (for example non IP packets) will return an *RSS Hash* result of zero, an *RSS Type* of zero and an RSS output index of zero.

6.1.3.6.1 Enabling RSS

- RSS is enabled in the *MRQC* register.
- RSS enabling cannot be done dynamically and must be preceded by a software reset.
- RSS status field in the descriptor write-back is enabled when the *RXCSUM.PCSD* bit is set (fragment checksum is disabled). RSS is therefore mutually exclusive with UDP fragmentation checksum offload.
- Support for RSS is not provided when legacy receive descriptor format is used.

6.1.3.6.2 Disabling RSS

- Disabling RSS on the fly is not allowed, and the integrated 10 GbE LAN controller must be reset after RSS is disabled.
- When RSS is disabled, packets are assigned an RSS output index = zero.

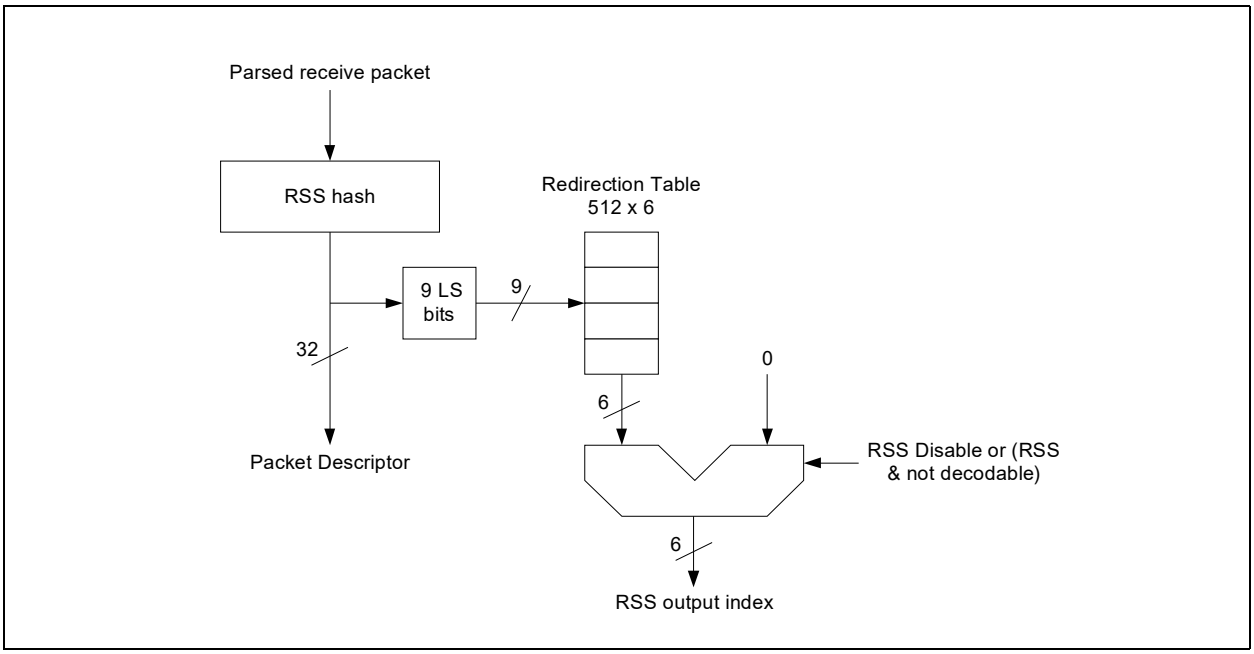


Figure 6.8. RSS Block Diagram (Multiple_RSS = 0)

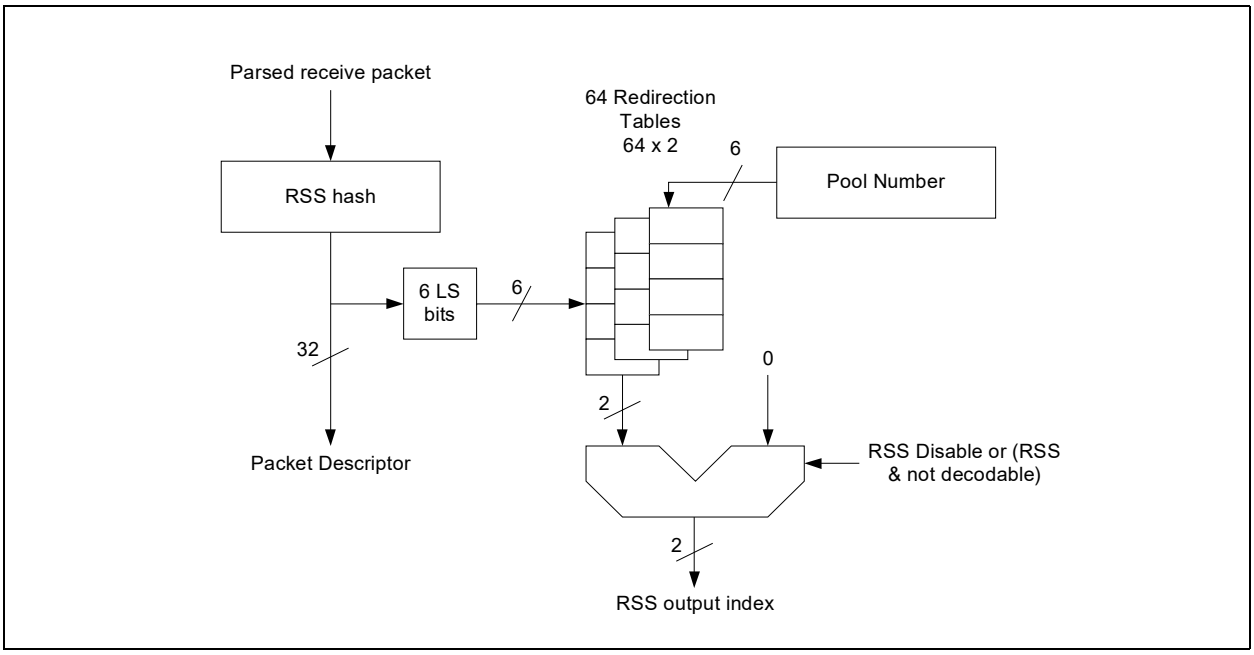


Figure 6.9. RSS Block Diagram (Multiple_RSS = 1)

6.1.3.6.3 RSS Hash Function

This section provides a verification suite used to validate that the hash function is computed according to MSFT nomenclature.

The integrated 10 GbE LAN controller's hash function follows the MSFT definition. A single hash function is defined with several variations for the following cases:

- TcpIPv4 — The integrated 10 GbE LAN controller parses the packet to identify an IPv4 packet containing a TCP segment per the criteria described below. If the packet is not an IPv4 packet containing a TCP segment, RSS is not done for the packet.
- IPv4 — The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller parses the packet to identify an IPv6 packet containing a TCP segment per the criteria described below. If the packet is not an IPv6 packet containing a TCP segment, RSS is not done for the packet.
- IPv6 — The integrated 10 GbE LAN controller parses the packet to identify an IPv6 packet. If the packet is not an IPv6 packet, RSS is not done for the packet.

Tunneled IP to IP packets are considered for the RSS functionality as IP packets. The RSS logic ignores the L4 header while using the outer (first) IP header for the RSS hash.

For NVGRE and VXLAN packets the inner header (IP and TCP/UDP) is used for RSS computations.

The following additional cases are not part of the MSFT RSS specification but are supported RSS modes:

- UdpIPV4 — The integrated 10 GbE LAN controller parses the packet to identify a packet with UDP over IPv4.

- UdpIPV6 — The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller, enabling TCP layer header recognition. Still the IPv6 extended header fields are not taken into account for the queue classification by RSS filter. This rule do not apply for security headers and fragmentation header. Packets with fragmentation header miss this filter. Packets with security extended headers are parsed only up to these headers and therefore can match only filters that do not require fields from the L4 protocol.

Bits[31:16] of the Multiple Receive Queues Command (MRQC), for single RSS and VFMRQC for multiple RSS, registers enable each of the above hash function variations (several might be set at a given time). If several functions are enabled at the same time, priority is defined as follows (skip functions that are not enabled):

- IPv4 packet
  - — Try using the TcpIPv4 function
  - — Try using UdpIPv4 function
  - — Try using the IPv4 function
- IPv6 packet
  - — Try using the TcpIPv6 function.
  - — Try using UdpIPv6 function.
  - — Try using the IPv6 function

The following combinations are currently supported:

- Any combination of IPv4, TcpIPv4, and UdpIPv4.

And/or:

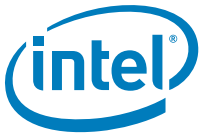
- Any combination of either IPv6, TcpIPv6, and UdpIPv6.

When a packet cannot be parsed by the previous rules, it is assigned an RSS output index = zero. The 32-bit tag (normally a result of the hash function) equals zero.

The 32-bit result of the hash computation is written into the packet descriptor and also provides an index into the redirection table.

The following notation is used to describe the following hash functions:

- Ordering is little endian in both bytes and bits. For example, the IP address 161.142.100.80 translates into 0xa18e6450 in the signature.
- A "^" denotes bit-wise XOR operation of same-width vectors.
- @x-y denotes bytes x through y (including both of them) of the incoming packet, where byte 0 is the first byte of the IP header. In other words, we consider all byte-offsets as offsets into a packet where the framing layer header has been stripped out. Therefore, the source IPv4 address is referred to as @12-15, while the destination v4 address is referred to as @16-19.
- @x-y, @v-w denotes concatenation of bytes x-y, followed by bytes v-w, preserving the order in which they occurred in the packet.



All hash function variations (IPv4 and IPv6) follow the same general structure. Specific details for each variation are described in the following section. The hash uses a random secret key of length 320 bits (40 bytes); the key is stored in the RSS Random Key Register RSSRK for single RSS and VFRSSRK[63:0] for multiple RSS.

The algorithm works by examining each bit of the hash input from left to right. Our nomenclature defines left and right for a byte-array as follows: Given an array K with k bytes, our nomenclature assumes that the array is laid out as follows:

- K[0] K[1] K[2] … K[k-1]

K[0] is the left-most byte, and the MSB of K[0] is the left-most bit. K[k-1] is the right-most byte, and the LSB of K[k-1] is the right-most bit.

ComputeHash(input[], N)

```
For hash-input input[] of length N bytes (8N bits) and a random secret key K of 320 bits

Result = 0;

For each bit b in input[] {

if (b == 1) then Result ^= (left-most 32 bits of K);

shift K left 1 bit position;

}

return Result;
```

6.1.3.6.3.1 Pseudo-code Examples

The following four pseudo-code examples are intended to help clarify exactly how the hash is to be performed in four cases, IPv4 with and without ability to parse the TCP header, and IPv6 with and without a TCP header.

Hash for IPv4 with TCP

Concatenate SourceAddress, DestinationAddress, SourcePort, DestinationPort into one single byte-array, preserving the order in which they occurred in the packet: Input[12] = @12-15, @16-19, @20-21, @22-23.

```
Result = ComputeHash(Input, 12);
```

Hash for IPv4 with UDP

Concatenate SourceAddress, DestinationAddress, SourcePort, DestinationPort into one single byte-array, preserving the order in which they occurred in the packet: Input[12] = @12-15, @16-19, @20-21, @22-23.

```
Result = ComputeHash(Input, 12);
```

Hash for IPv4 without TCP

Concatenate SourceAddress and DestinationAddress into one single byte-array

```
Input[8] = @12-15, @16-19

Result = ComputeHash(Input, 8)
```

Hash for IPv6 with TCP

Similar to previous:

```
Input[36] = @8-23, @24-39, @40-41, @42-43

Result = ComputeHash(Input, 36)
```

Hash for IPv6 with UDP

Similar to previous:

```
Input[36] = @8-23, @24-39, @40-41, @42-43

Result = ComputeHash(Input, 36)
```

Hash for IPv6 without TCP

```
Input[32] = @8-23, @24-39

Result = ComputeHash(Input, 32)
```

6.1.3.6.4 Redirection Tables

The redirection table used for single RSS mode (RETA and ERETA) is a 512-entry structure, indexed by the nine LSBs of the hash function output.

The redirection tables used for multiple RSS mode (VFRETA) are 64-entry structures indexed by the six LSBs of the hash function output. The table to use is defined by the pool to which the packet is sent.

System software might update the redirection tables during run time. Such updates of the table are not synchronized with the arrival time of received packets. Therefore, it is not guaranteed that a table update takes effect on a specific packet boundary.

6.1.3.6.5 RSS Verification Suite

Assume that the random key byte-stream is:

```
0x6d, 0x5a, 0x56, 0xda, 0x25, 0x5b, 0x0e, 0xc2,
0x41, 0x67, 0x25, 0x3d, 0x43, 0xa3, 0x8f, 0xb0,
0xd0, 0xca, 0x2b, 0xcb, 0xae, 0x7b, 0x30, 0xb4,
0x77, 0xcb, 0x2d, 0xa3, 0x80, 0x30, 0xf2, 0x0c,
0x6a, 0x42, 0xb7, 0x3b, 0xbe, 0xac, 0x01, 0xfa
```

Table 6.12. 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.

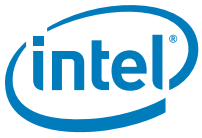


Table 6.13. IPv6

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

6.1.4 Receive Data Storage in System Memory

The integrated 10 GbE LAN controller posts receive packets into data buffers in system memory.

The following controls are provided for the data buffers:

- The SRRCTL[n].BSIZEPACKET field defines the size of the data buffer pointed by each descriptor. The maximum packet size that can be posted to a queue can be limited using the *RXDCTL[n].RLPML* field. This filter enables software to use smaller buffers than the size defined by the *SRRCTL[n].BSIZEPACKET*.

Note: The packet size compared to *RXDCTL[n].RLPML* does not include any parts stripped by the device like CRC VLAN or other tags but include additional TimeStamp appended to the packet.

- The SRRCTL.BSIZEHEADER field defines the size of the header buffer pointed by each descriptor (advanced descriptors only).
- Each queue is provided with a separate SRRCTL register.

Receive memory buffer addresses are word (2 x byte) aligned (both data and headers).

The internal receive buffers are described in Section 6.5.3.1.

6.1.5 Receive Descriptors

6.1.5.1 Legacy Receive Descriptor Format

A receive descriptor is a data structure that contains the receive data buffer address and fields for hardware to store packet information. Upon receipt of a packet for this device, hardware stores the packet data into the indicated buffer and writes the length, status and errors to the receive descriptor. If SRRCTL[n].DESCTYPE = zero, the integrated 10 GbE LAN controller uses the Legacy Rx descriptor as listed in Table 6.14. 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, Time stamp in Packet, tunnel packets checksum or RSC. Packets that match these cases might be dropped from queues that use legacy receive descriptors.

Refer to Table 6.14 and the field descriptions that follow.

Table 6.14. Legacy Receive Descriptor (RDESC) Layout

	63	48 47	40 39	32 31	16 15	0
0	Buffer Address [63:0]					
8	VLAN Tag	Errors	Status	Fragment Checksum	Length	

Buffer Address (64-bit offset 0, 1st line)

Physical address in host memory of the received packet buffer.

Length Field (16-bit offset 0, 2nd line)

The length indicated in this field covers the data written to a receive buffer including CRC bytes (if any). Software must read multiple descriptors to determine the complete length for packets that span multiple receive buffers.

Fragment Checksum (16-bit offset 16, 2nd line)

This field is used to provide the fragment checksum value. This field is equal to the unadjusted 16-bit ones complement of the packet. Checksum calculation starts at the L4 layer (after the IP header) until the end of the packet excluding the CRC bytes. 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 6.1.6.5.

The fragment checksum is always reported in the descriptor with the EOP bit set.

Status Field (8-bit offset 32, 2nd line)

Status information indicates whether the descriptor has been used and whether the referenced buffer is the last one for the packet. Error status information is listed in Table 6.16.

Table 6.15. 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.

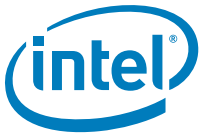
VP (VLAN Packet)

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 6.4.

IPCS (IPv4 Checksum), L4CS (L4 Checksum), UDPCS (UDP Checksum)

These bits are described in the following table.

Note: Switched packets from a local VM that do not use the Tx IP checksum offload by hardware have the IPCS equal to zero; switched packets from a local VM that do not use the Tx L4 checksum offload by hardware have the L4CS and UDPCS equal to zero.



L4CS	UDPCS	IPCS	Functionality
0	0	0	Hardware does not provide checksum offload.
0	0	1	Hardware provides IPv4 checksum offload. Pass/fail indication is provided in the *Error* field – IPE.
1	0	1 / 0	Hardware provides IPv4 checksum offload if IPCS is active along with TCP checksum offload. Pass/fail indication is provided in the *Error* field – IPE and TCPE.
1	1	1 / 0	Hardware provides IPv4 checksum offload if IPCS is active along with UDP checksum offload. Pass/fail indication is provided in the *Error* field – IPE and TCPE.

IPv6 packets do not have the *IPCS* bit set, but might have the *L4CS* bit and *UDPCS* bit set if the integrated 10 GbE LAN controller 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 6.16 and the following text describes the possible errors reported by the hardware.

Table 6.16. Receive Errors (RDESC.ERRORS) Layout

7	6	5	4	3	2	1	0
IPE	TCPE	Reserved	Reserved	Reserved	Reserved	Reserved	RXE

IPE (IPv4 Checksum Error)

The IP checksum error is valid only when the *IPCS* bit in the *Status* field is set (indicating that the hardware validated the IP checksum). This bit is meaningful only on the last descriptor of a packet while the *EOP* bit is set as well. Packets with IP error are posted to host memory regardless of the store bad packet setting (FCTRL.SBP).

TCPE (TCP/UDP Checksum Error)

The TCP/UDP checksum error is valid only when the *L4CS* bit in the *Status* field is set (indicating that the hardware validated the L4 checksum). This bit is meaningful only on the last descriptor of a packet while the *EOP* bit is set as well. Packets with a TCP/UDP error are posted to host memory regardless of the store bad packet setting (FCTRL.SBP).

IPv4/UDP packets that carry a null UDP checksum field are reported with L4CS=0 and TCPE=0 (valid packet with no checksum).

IPv6/UDP packets that carry a null UDP checksum field are reported with L4CS=1 and TPCE=1 (UDP checksum is mandatory over IPv6).

RXE

The RXE error bit is an indication for any MAC error. It is a logic OR function of the following errors:

- CRC or symbol error might be a result of receiving a /V/ symbol on the TBI interface, /FE/ symbol on the GMII/XGMII interface, RX_ER assertion on GMII interface, bad EOP or loss of sync during packet reception.
- Undersize frames shorter than 64 bytes.
- Oversize frames larger than the MFS definition in the MAXFRS register.
- Length error in 802.3 packet format. Length field is not checked in presence of a VLAN or E-tag.

Packets with an RXE error are posted to host memory only when store bad packet bit (FCTRL.SBP) is set.

VLAN Tag Field (16-bit offset 48, 2nd line)

If the RXDCTL.VME is set and the received packet type is 802.1q (as defined by VLNCTRL.VET) then the VLAN header is stripped from the packet data storage. In this case the 16 bits of the VLAN tag, priority tag and DEI from the received packet are posted to the *VLAN Tag* field in the receive descriptor. Otherwise, the *VLAN Tag* field contains 0x0000.

Table 6.17. VLAN Tag Field Layout (for 802.1q Packet)

15 13	12	11 0
PRI	DEI	VLAN

Priority and DEI are part of 802.1Q specifications. The VLAN field is provided in network order.

6.1.5.2 Advanced Receive Descriptors

6.1.5.2.1 Advanced Receive Descriptors — Read Format

Table 6.18 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 6.18. Descriptor Read Format

	63 1	0
0	Packet Buffer Address [63:1]	
8	Header Buffer Address [63:1]	DD

Packet Buffer Address (64)

The physical address in host memory of the packet buffer.

Header Buffer Address (64)

The physical address in host memory of the header buffer with the lowest bit being Descriptor Done (DD). When a packet spans in multiple descriptors, only the header buffer of the first descriptor is used. In subsequent descriptors, only the data buffer is used.

During the programming phase, software must set the *DD* bit to zero (see the description of the *DD* bit in this section). This means that header buffer addresses are always word aligned.

Note: The integrated 10 GbE LAN controller does not support null descriptors meaning packet or header addresses are zero.

6.1.5.2.2 Advanced Receive Descriptors — Write-Back Format

When the integrated 10 GbE LAN controller writes back the descriptors, it uses the format listed in Table 6.19. The advanced descriptor write-back format is used when SRRCTL[n]. DESCTYPE is set to a value other than 000b.

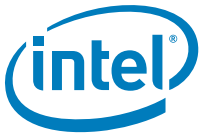


Table 6.19. Descriptor Write-Back Format

	63 48	47 32	31	30 21	20 17	16 4	3 0
0	RSS Hash / Fragment Checksum / Flow Director Filters ID		SPH	HDR_LEN	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 integrated 10 GbE LAN controller must identify the packet type and then choose the appropriate RSS hash function to be used on the packet. The RSS type reports the packet type that was used for the RSS hash function.

RSS Type	Description
0x0	No hash computation done for this packet
0x1	HASH_TCP_IPv4
0x2	HASH_IPv4
0x3	HASH_TCP_IPv6
0x4	Reserved
0x5	HASH_IPv6
0x6	Reserved
0x7	HASH_UDP_IPv4
0x8	HASH_UDP_IPv6
0x9 – 0xE	Reserved
0xF	Packet reports flow director filters status (set if a flow director match, even if a packet is forwarded by another filter).

Packet Type (13-bit at offset 4, 1st line)

The *Packet Type* field reports the packet type identified by the hardware as follows. Note that some of the fields in the receive descriptor are valid for specific packet types.

Bit Index	Bit 11 = 0	Bit 11 = 1 (L2 Packet Matching One of the *ETQF* Filters)
0	IPV4 — IPv4 header present	Ethertype — *ETQF* register index that matches the packet. Special types are defined for 802.1X and 1588.
1	IPV4O — IPv4 with options	
2	IPV6 — IPv6 header present	
3	IPV6E- IPv6 with extensions	Reserved.
4	TCP — TCP header present	

Bit Index	Bit 11 = 0	Bit 11 = 1 (L2 Packet Matching One of the *ETQF* Filters)
5	UDP — UDP header present	Reserved
6	SCTP — SCTP header	
7	If bit 12 = 0 0b = Non-tunneled 1b = Geneve If bit 12 = 1: 0b = NVGRE 1b = VXLAN	
10:8	Reserved	
11	0b = non L2 packet	1b = L2 packet
12	NVGRE or VXLAN Tunnel Packet. If this bit is set, then bit 7 indicates the tunnel type.	Reserved

Notes: UDP, TCP and IPv6 indications are not set in any IPv4 fragmented packet.

In virtualization mode, packets might be received from other local VMs. The integrated 10 GbE LAN controller 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. In order to be identified, the *CC* bit in the transmit descriptor must be set and if it is a tunnel packet, the *TUNNEL.OUTERIPCS* must also be set. If bit 12 (Tunnel Packet) is set, then all the L3/L4 indications (IPV4, IPV4O, IPV6, IPV6E, TCP, UDP, SCTP) reflects the inner header status.

RSC Packet Count- RSCCNT (4-bit offset 17, 1st line)

The *RSCCNT* field is valid only for RSC descriptors while in non-RSC it equals zero. RSCCNT minus one indicates the number of coalesced packets that start in this descriptor. RSCCNT might count up to 14 packets. Once 14 packets are coalesced in a single buffer, RSC is closed enabling accurate coalesced packet count. If the *RSCCNTBP* bit in RDRXCTL is set, coalescing might proceed beyond the 14 packets per buffer while *RSCCNT* stops incrementing beyond 0xF.

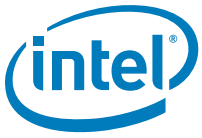
Note: Software can identify RSC descriptors by checking the *RSCCNT* field for non-zero value.

HDR_LEN (10-bit offset 21, 1st line)

The HDR_LEN reflects the size of the packet header in byte units (if the header is decoded by the hardware). This field is meaningful only in the first descriptor of a packet and should be ignored in any subsequent descriptors. Header split is explained in Section 6.1.6 while the packet types for this functionality are enabled by the PSRTYPE[n] registers.

Split Header — SPH (1-bit offset 31, 1st line)

When set to 1b, indicates that the hardware has found the length of the header. If set to 0b, the header buffer may be used only in split always mode. If the received header size is greater or equal to 1024 bytes, the *SPH* bit is not set and header split functionality is not supported. The *SPH* bit is meaningful 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 6.24.



Packet Type	Header Length (Includes All Fields Up to the Field Specified)	Header Split
Un-recognized Ethertype only with / without SNAP and with / without VLAN or packets that match the L2 filters (MTQF).	VLAN header(s) if present Else, EtherType field	No
Pv4 only or fragmented IPv4 with any payload including IPv4-IPv6 tunnelling	IPv4 header	Enabled
Non-fragmented IPv4, TCP / UDP / SCTP	L4 header	Enabled
IPv4-IPv6, only or fragmented IPv4-IPv6 at IPv6 header with any payload	IPv6 header (up to the fragment extension header if exist)	Enabled
IPv4-IPv6,TCP / UDP / SCTP	L4 header	Enabled
VXLAN, NVGREor Geneve	Cloud Header/L2 Header	Enabled[1]

1. If cloud split modes used (PSRTYPE15 or PSRTYPE16 are set). If not, then according to internal header.

RSS Hash or Flow Director Filters ID (32-bit offset 32, 1st line) / Fragment Checksum (16-bit offset 48, 1st line)

This field has multiplexed functionality according to the received packet type (reported on the *Packet Type* field in this descriptor) and device setting.

Fragment Checksum

For fragmented UDP/IP packets, this field holds the UDP fragment checksum (described in Section 6.1.6.5) if both the *RXCSUM.PCSD* bit is cleared (or the RXCSUM.PCSD bit is set, but no RSS was calculated) and *RXCSUM.IPPCSE* bit is set. This field is meaningful only for UDP packets where the *UDPV* bit in the Extended Status word is set. When Fragment Checksum is reported, bits 47:32 are invalid.

The checksum does not include any padding or time stamp added by the device.

- If none of the previous conditions apply, the value is not valid.

Table 6.20. 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 6.1.3.6. Note that the RSS hash is meaningful only for 'RSS Type' in the range 0x1 to 0x8.

Flow Director Filters ID

The flow director filters ID is reported only when the received packet matches a flow directory filter (see Section 6.1.3.5). The flow director filter ID field has a different structure for signature-based filters and perfect match filters as follows:

Filter Type	31	30 29	28 16	15 13	12 0
Hash-based Flow Director Filter ID	Rsv	Bucket Hash		Signature	
Perfect Match Flow Director Filter ID	Rsv		Hash	Rsv	SW-Index

Bucket Hash

A hash value that identifies a flow director bucket. When the Flow Director table is smaller than 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 6.21 lists the extended status word in the last descriptor of a packet (*EOP* bit is set). Table 6.22 lists the extended status word in any descriptor but the last one of a packet (*EOP* bit is cleared).

Table 6.21. Receive Status (RDESC.STATUS) Layout of *Last* Descriptor

19	18	17	16	15	14	13	12	11	10
BMC	LB		TS	TSIP	Rsv				UDPV
VEXT	OUTERIPCS	PIF	IPCS	L4I	UDPCS	VP	FLM	EOP	DD
			FCEOFs	FCSTAT					
9	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**

Table 6.22. Receive Status (RDESC.STATUS) Layout of *Non-Last* Descriptor

19 4	3 : 2	1	0
Next Descriptor Pointer — NEXTP	Rsv	EOP = 0b	DD

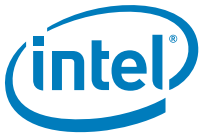
Rsv (14:12) — Reserved at zero.

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 6.1.5.

The VP bit is not set even if a VLAN tag is present if the *PFQDE.HIDE_VLAN* bit is set for the queue.

EOP (1) and DD (0) — *End of Packet* and *Done* bits are listed in the following table:



DD	EOP	Description
0	X	Software setting of the descriptor when it hands it to hardware.
1	0	A completion status indication for a non last descriptor of a packet (or multiple packets in the case of RSC) that spans across multiple descriptors. In a single packet case the *DD* bit indicates that the hardware is done with the descriptor and its buffers. In the case of RSC, the *DD* bit indicates that the hardware is done with the descriptor but might still use its buffers (for the coalesced header) until the last descriptor of the RSC completes. Only the *Length* fields are valid on this descriptor. In the RSC case, the next descriptor pointer and RSCCNT are valid as well.
1	1	A completion status indication of the entire packet (or the multiple packets in the case of RSC) and software might take ownership of its descriptors. All fields in the descriptor are valid (reported by the hardware).

UDPCS(4), L4I (5) / FCSTAT (5:4) — This field has multiplexed functionality. The UDPCS (UDP checksum) is set (together with L4CS bit) when hardware provides UDP checksum offload. The L4I (L4 Integrity) is set when hardware provides any L4 offload as: UDP checksum, TCP checksum or SCTP CRC offload.

IPCS(6) — This bit has multiplexed functionality.

It is IPCS as described in Legacy Rx descriptor (in Section 6.1.5).

OUTERIPCS(8) — Indicates that a checksum was done on the outer IP header of an NVGRE or VXLAN packet.

VEXT (9) — Outer-VLAN is found on a double VLAN packet. This bit is valid only when CTRL_EXT.EXTENDED_VLAN is set. See more details in Section 6.4.5.

UDPV (10) — The *UDP Checksum Valid* bit indicates that a UDP checksum was calculated on the incoming fragmented (non-tunneled) UDP IPv4 packet. It means that the *Fragment Checksum* field in the receive descriptor might contain the UDP checksum as described in Section 6.1.6.5. When this field is cleared in the first fragment that contains the UDP header, it means that the packet does not contain a valid UDP checksum and the checksum field in the Rx descriptor should be ignored.

TSIP (15) - Timestamp in packet. The Timestamp In Packet bit is set to indicate that the received packet arrival time was captured by the hardware and the timestamp was placed in the receive buffer. For more details see Section 6.5 and Section 6.1.6.2.

TS (16) — The *Time Stamp* bit is set when the device recognized a time sync packet. In such a case the hardware captures its arrival time and stores it in the Time Stamp register. For more details see Section 6.5.

SECP (17) — Security processing bit indicates that the hardware identified the security encapsulation and processed it as configured.

> *IPsec processing —* This bit is set only if a matched SA was found. Note that hardware does not process packets with an IPv4 option or IPv6 extension header and the SECP bit is not set. This bit is not set for IPv4 packets shorter than 70 bytes, IPv6 ESP packets shorter than 90 bytes, or IPv6 AH packets shorter than 94 bytes (all excluding CRC). Note that these packet sizes are never expected and set the length error indication in the *SECERR* field.

LB (18) — This bit provides a loopback status indication which means that this packet is sent by a local VM (VM to VM switch indication).

BMC (19) - Packet received from BMC. The BMC bit is set to indicate the packet was sent by the local BMC. Bit is cleared if packet arrives from the network.

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 6.23 and the following text describe the possible errors reported by hardware.

Table 6.23. Receive Errors (RDESC.ERRORS) Layout

11	10	9	8:7	6	5:4	3	2:0
IPE FCEOFe	L4E	RXE		OUTERIPER	Rsv	HBO	FCERR / FDIRERR

FDIRERR (2:0) — This field is relevant when the flow director filters are enabled.

FDIRErr(0) - Length — If the flow director filter matches the *Length* bit, this indicates that the distance of the matched filter from the hash table exceeds the FDIRCTRL.Max-Length. If there is no matched filter, the *Length* bit is set if the flow director linked list of the matched hash value exceeds the FDIRCTRL.Max-Length.

FDIRErr(1) - Drop — The *Drop* bit indicates that a received packet matched a flow director filter with a drop action. In the case of perfect mode filtering, it is expected to find the drop indication only when the linked list in the flow director bucket exceeds the permitted Max-Length. In this case, the packet is not dropped. Instead, it is posted to the Rx queue (indicated in the filter context) for software handling of the Max-Length exception. In the case of hash mode filtering, it is expected that the drop queue is always a valid queue so all packets that match the drop filter are visible to software.

FDIRErr(2) - Coll — A matched flow director filter with a collision indication was found. The collision indicates that software attempted to step over this filter with a different action that was already programmed.

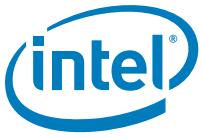
HBO (3) — The *Header Buffer Overflow* bit is set if the packet header (calculated by hardware) is bigger than the header buffer (defined by PSRCTL.BSIZEHEADER). *HBO* reporting might be used by software to allocate bigger buffers for the headers. It is meaningful only if the *SPH* bit in the receive 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*SRRCTL.BSIZEHEADER* bytes of the packet.

Rsv (5:4) — Reserved at zero.

OUTERIPER (6) — Indicates an error was found in the checksum of an outer IP header of a VXLAN or NVGRE packet or that the UDP checksum of the outer UDP header in a VXLAN packet was not zero.

RXE (9) — RXE is described in the legacy descriptor format in Section 6.1.5.



L4E (10) — L4 integrity error is valid only when the *L4I* bit in the *Status* field is set. It is active if L4 processing fails (TCP checksum or UDP checksum or SCTP CRC). Packets with L4 integrity error are posted to host memory regardless of the store bad packet setting (FCTRL.SBP). In case of VXLAN or NVGRE packets, this relates to the internal L4 header.

FCEOFe(11) / IPE(11) — This bit has multiplexed functionality.

Packet
IPE (IPv4 checksum error) is described in Section 6.1.5. In case of VXLAN or NVGRE packets, this relates to the internal IP header.

PKT_LEN (16-bit offset 32, 2nd line)

PKT_LEN holds the number of bytes posted to the packet buffer. The length covers the data written to a receive buffer including posted CRC bytes (if any). Software must read multiple descriptors to determine the complete length for packets that span multiple receive buffers. If SRRCTL.DESCTYPE = 2 (advanced descriptor header splitting) and the buffer is not split because the header is bigger than the allocated header buffer, this field reflects the size of the data written to the data buffer (header + data).

When short packets are padded, the PKT_LEN does not include the padding, so that the software device driver can detect the location of a potential time stamp in the packet.

When a time stamp is added (TSIP bit is set), the PKT_LEN includes the timestamp size (+8).

VLAN Tag (16-bit offset 48, 2nd line)

This field is described in the legacy descriptor format in Section 6.1.5. The VLAN tag is set to 0x0000 even if a VLAN tag is present if the *PFQDE.HIDE_VLAN* bit is set for the queue.

6.1.5.3 Receive Descriptor Fetching

The integrated 10 GbE LAN controller 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.

6.1.5.4 Receive Descriptor Write-Back

The integrated 10 GbE LAN controller 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 6.1.5.3), 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 6.8.5.1 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 integrated 10 GbE LAN controller.

6.1.5.5 Receive Descriptor Queue Structure

Figure 6.10 shows the structure of each of the receive descriptor rings. Note that each ring uses a contiguous memory space.

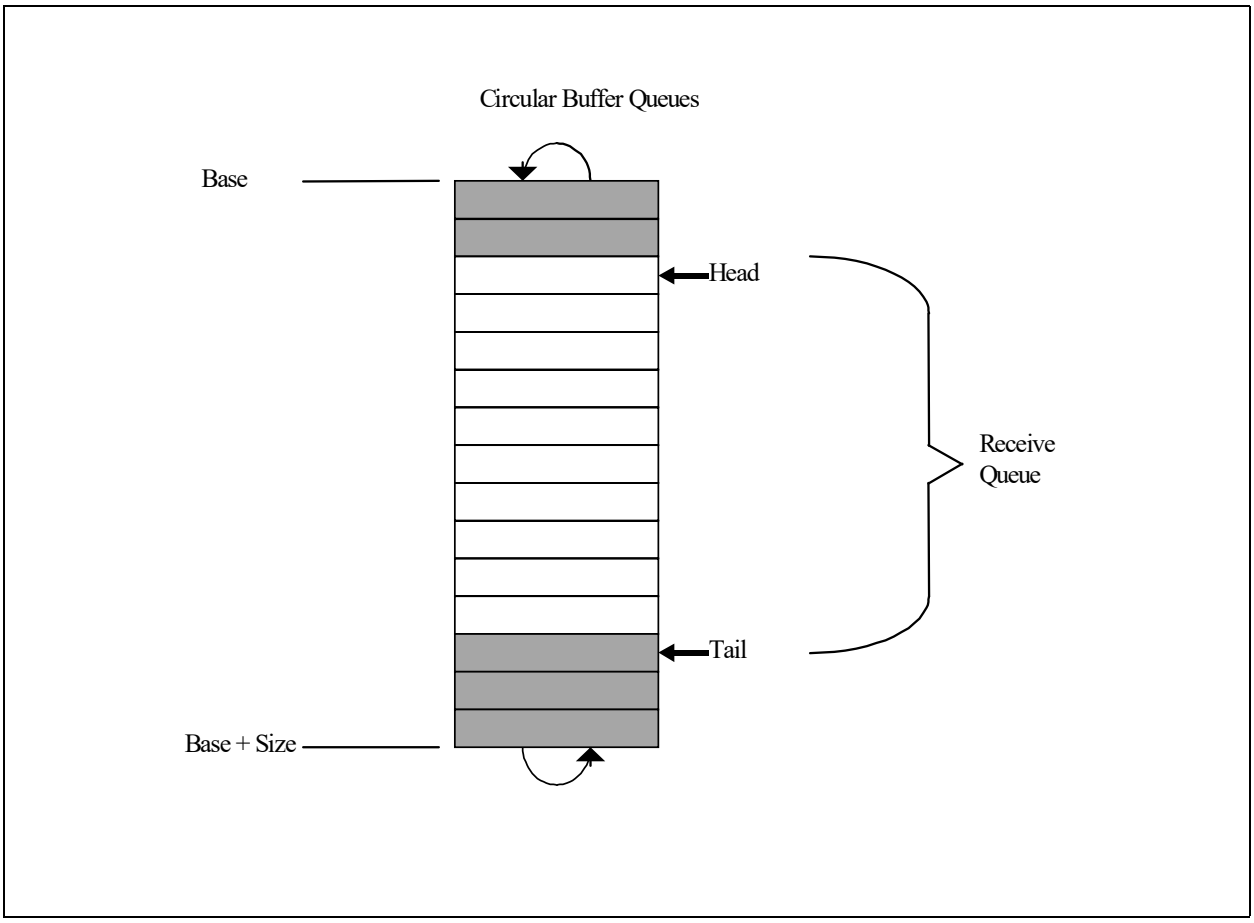


Figure 6.10. Receive Descriptor Ring Structure

Software inserts receive descriptors by advancing the tail pointer(s) to refer to the address of the entry just beyond the last valid descriptor. This is accomplished by writing the descriptor tail register(s) with the offset of the entry beyond the last valid descriptor. The integrated 10 GbE LAN controller adjusts its internal tail pointer(s) accordingly. As packets arrive, they are stored in memory and the internal head pointer(s) is increased by the integrated 10 GbE LAN controller.

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 integrated 10 GbE LAN controller stops storing packets in system memory until software advances the tail pointer(s), making more receive buffers available.

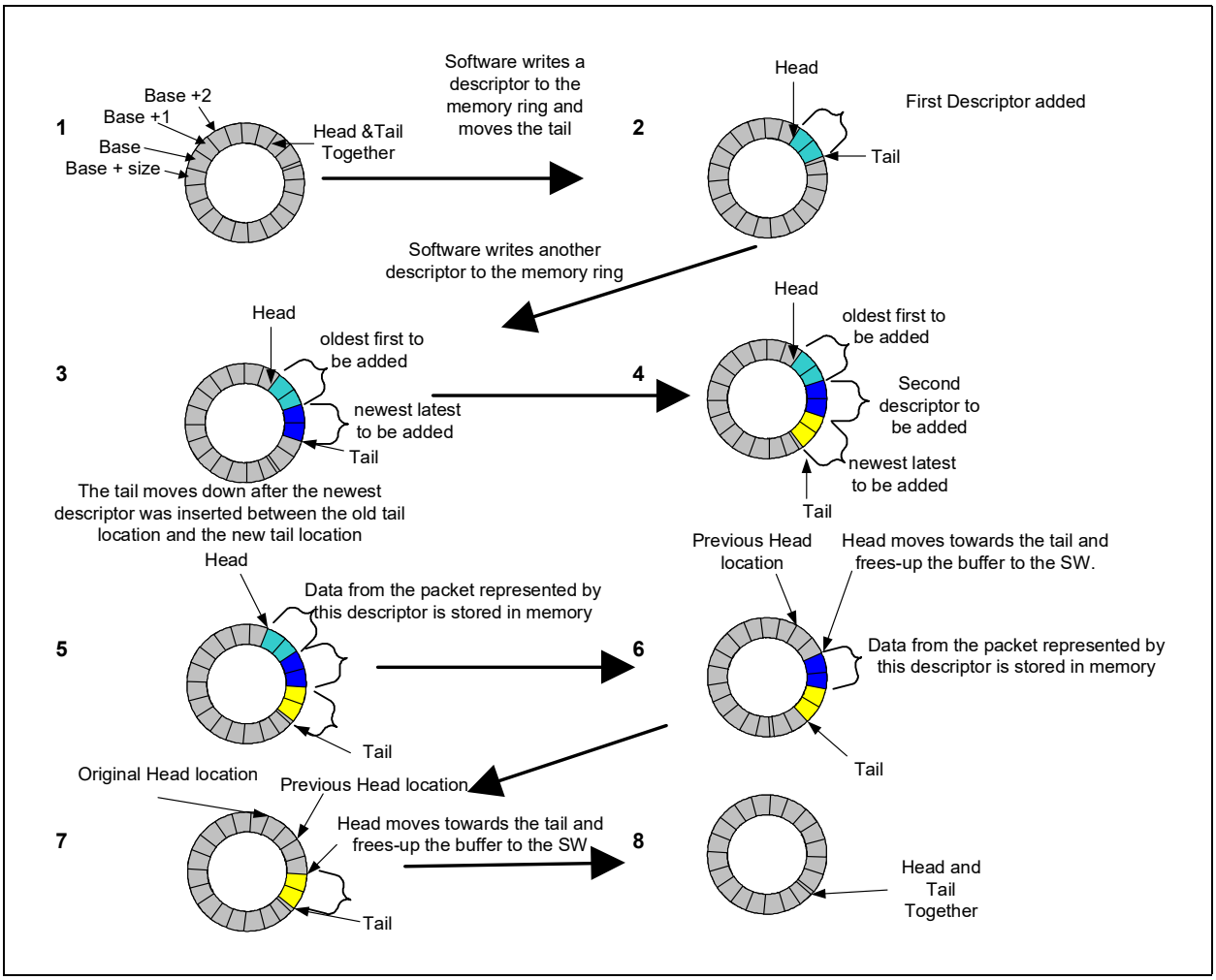


Figure 6.11. Descriptors and Memory Rings

The integrated 10 GbE LAN controller 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 6.11 represent descriptors that have stored incoming packets but have not yet been recognized by software. Software can determine if a receive buffer is valid by reading descriptors in memory rather than by I/O reads. Any descriptor with a *DD* bit set has been used by the hardware, and is ready to be processed by software.

Note: The head pointer points to the next descriptor that is to be written back. At the completion of the descriptor write-back operation, this pointer is increased by the number of descriptors written back. Hardware owns all descriptors between [head... tail]. Any descriptor not in this range is owned by software.

The receive descriptor rings are described by the following registers:

- Receive Descriptor Base Address registers (RDBA) — This register indicates the start of the descriptor ring buffer; this 64-bit address is aligned on a 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 Programmed I/O read of the Receive Descriptor Head register. Checking the descriptor *DD* bit in memory eliminates a potential race condition. All descriptor data is written to the I/O bus prior to incrementing the head register, but a read of the head register could pass the data write in systems performing I/O write buffering. Updates to receive descriptors use the same I/O write path and follow all data writes. Consequently, they are not subject to the race.
- Receive Descriptor Tail registers (RDT) — This register holds a value that is an offset from the base, and identifies the location beyond the last descriptor hardware can process. This is the location where software writes the first new descriptor.

If software statically allocates buffers, and uses a memory read to check for completed descriptors, it simply has to zero the status byte in the descriptor to make it ready for re-use by hardware. This is not a hardware requirement, but is necessary for performing an in-memory scan. This is relevant only to legacy descriptors.

All the registers controlling the descriptor rings behavior should be set before receive is enabled, apart from the tail registers which are used during the regular flow of data.

6.1.5.5.1 Low Receive Descriptors Threshold

As previously described, the size of the receive queues is measured by the number of receive descriptors. During run time, software processes descriptors and upon completion of descriptors, increments the Receive Descriptor Tail registers. At the same time, the hardware may post new received packets incrementing the Receive Descriptor Head registers for each used descriptor.

The number of usable (free) descriptors for the hardware is the distance between the Tail and Head registers. When the tail reaches the head, there are no free descriptors and further packets might be either dropped or block the receive FIFO. In order to avoid this situation, the integrated 10 GbE LAN controller 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.

6.1.6 Receive Offloads

6.1.6.1 Header Splitting

6.1.6.1.1 Purpose

This feature consists of splitting a packet header to a different memory space. This helps the host to fetch headers only for processing: headers are posted through a regular snoop transaction in order to be processed by the host CPU.

The integrated 10 GbE LAN controller'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 integrated 10 GbE LAN controller uses packet splitting when the *SRRCTL[n].DESCTYPE* is greater than one.

6.1.6.1.2 Description

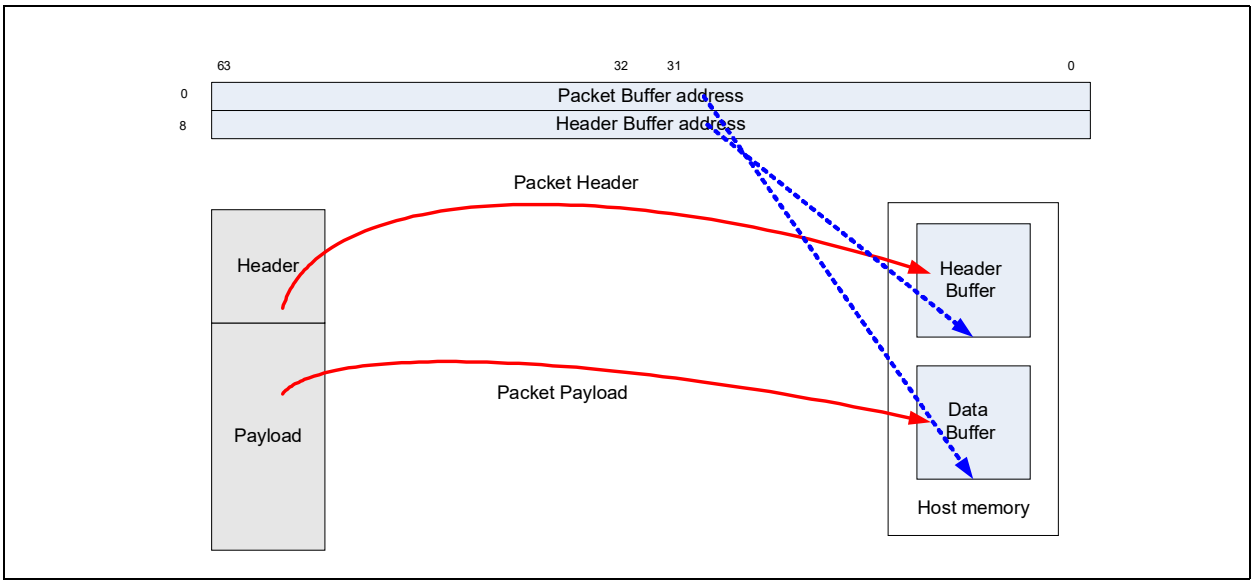


Figure 6.12. Header Splitting Diagram

The physical address of each buffer is written in the *Buffer Addresses* fields:

- The packet buffer address includes the address of the buffer assigned to the packet data.
- The header buffer address includes the address of the buffer that contains the header information. The receive DMA module stores the header portion of the received packets into this buffer.

The sizes of these buffers are statically defined in the SRRCTL[n] registers:

- The *BSIZEPACKET* field defines the size of the buffer for the received packet.
- The *BSIZEHEADER* field defines the size of the buffer for the received header. If header split is enabled, this field must be configured to a non-zero value. The integrated 10 GbE LAN controller 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. In virtualization mode, a separate PSRTYPE register is provided per pool up to the number of pools enabled. In non-virtualization mode, only PSRTYPE[0] is used.

Rules regarding header split:

- Packets that have headers bigger than 1023 bytes are not split.
- The header of a fragmented IPv6 packet is defined until the *fragment* extension header.
- An IP in IP packet (such as any combination of IPv4 and IPv6 tunneling) is not split. Not relevant for NVGRE, Geneve and VXLAN packets.
- Packet header cannot span across buffers, therefore, the size of the header buffer must be larger than any expected header size. In case of header split mode (SRRCTL.DESCTYPE = 010b), a packet with a header larger than the header buffer is not split.

Table 6.24 lists the behavior of the integrated 10 GbE LAN controller in the different modes.

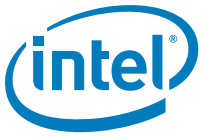
Table 6.24. Behavior in Header Split Modes

DESCTYPE	Condition	SPH	HBO[1]	PKT_LEN[2]	HDR_LEN[2]	Header and Payload DMA[3]
Split	1. Header can't be decoded	0	0	Min (packet length, buffer size)	0x0	Header + Payload (+ TimeStamp) (+ Padding) -> Packet Buffer
	2. Header <= BSIZEHEADER and Payload > 0	1	0	Min (payload length, buffer size)	Header size	Header -> Header Buffer Payload (+ TimeStamp) (+ Padding) -> Packet Buffer
	3. Header <= BSIZEHEADER and Payload = 0 (Header only packet)	1	0	0	Header size	Header (+ TimeStamp) (+ Padding) -> Header Buffer[4]
	4. Header > BSIZEHEADER	1	1	Min (packet length, buffer size)	Header size[5]	Header + Payload (+ TimeStamp) -> Packet Buffer
Split – always use header buffer	1. Header can't be decoded and packet length <= BSIZEHEADER	0	0	0x0	Packet length	Header + Payload (+ TimeStamp) (+ Padding) -> Header Buffer
	2. Header can't be decoded and packet length > BSIZEHEADER	0	0	Min (packet length – BSIZEHEADER, data buffer size)	BSIZEHEADER	Header + Payload (+ TimeStamp) -> Header + Packet Buffers[5]
	3. Header <= BSIZEHEADER	1	0	Min (payload length, data buffer size)	Header Size	Header -> Header Buffer Payload (+ TimeStamp) (+ Padding) -> Packet Buffer
	4. Header > BSIZEHEADER	1	1	Min (packet length – BSIZEHEADER, data buffer size)	Header Size[6]	Header + Payload (+ TimeStamp) -> Header + Packet Buffer

1. HBO is set to 1b if the header size is bigger than BSIZEHEADER and zero otherwise.
2. PKT_LEN and HDR_LEN includes optionally added time stamp if TSIP bit is set and does not include padding added by the device.
3. Partial means up to BSIZEHEADER.
4. Even if there is no payload at all and there is a timestamp in the packet - the timestamp is considered as an 8 bytes payload and is written to the packet buffer.
5. If the packet spans more than one descriptor, only the header buffer of the first descriptor is used.
6. HDR_LEN doesn't reflect the actual data size stored in the header buffer. It reflects the header size determined by the parser.

6.1.6.2 Receive Packet Timestamp in Buffer

The integrated 10 GbE LAN controller supports adding an optional tailored header appended to the end of the packet in the receive buffer. The tailored header includes a 64 bit timestamp composed of the packet reception time measured in the *SYSTIMEL* (Low DW) and *SYSTIMEH* (High DW) registers (See



Section 6.5.3.2 for further information on *SYSTIMEL/H* operation). The timestamp starts right at the end of the packet (after the last byte). It is sent as {*SYSTIMEH, SYSTEML*} - most significant byte first (closest to packet).

PKT_LEN and *HDR_LEN* includes optionally added time stamp.

When the *TSAUXC*.Disable_systime bit is cleared and the *TSYNCRXCTL.TSIP_UP_EN* is set for the UP in the packet (or *TSYNCRXCTL.TSIP_UT_EN* for un-tagged packets), packets received that meet the *TSYNCRXCTL.Type* will be time stamped. A packet that was time stamped is reported as follows:

- Place a 64 bit timestamp, indicating the time a packet was received by the MAC, appended at the end of the received packet within the receive buffer.
- Set the TSIP bit in the RDESC.STATUS field of the last receive descriptor.

Note: When packets are coalesced, the timestamp reflects the reception time of the last coalesced fragment, unless the packet is a pure ACK packet, in which case, the timestamp will be of the first packet.

6.1.6.3 Receive Checksum Offloading

The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller. The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller calculates the TCP or UDP checksum and indicates a pass/fail condition to software via the *TCP/UDP Checksum Error* bit (RDESC.TCPE). For NVGRE or VXLAN packets, the *IPE* and *TCPE* bits relates to the inner IP/TCP header. The outer IPv4 checksum is also checked and a pass/fail indication is indicated to software via the Outer IPv4 Checksum Error bit (*OUTERIPER*).These error bits are valid when the respective status bits indicate the checksum was calculated for the packet (*RDESC.IPCS*, *RDESC.L4CS*, and *RDESC.OUTERIPCS* respectively).

Similarly, if RFCTL.Ipv6_DIS and RFCTL.IP6Xsum_DIS are cleared to zero, the integrated 10 GbE LAN controller calculates the TCP or UDP checksum for IPv6 packets. It then indicates a pass/fail condition in the *TCP/UDP Checksum Error* bit (RDESC.TCPE).

Note: Only Ethernet II frames are supported (no checksum support for SNAP packets).

Table 6.25. Supported Receive Checksum Capabilities

Packet Type	Hardware IP Checksum Calculation	Hardware TCP/UDP Checksum Calculation
IP header's protocol field contains a protocol # other than TCP or UDP.	Yes	No
IPv4 + TCP/UDP packets.	Yes	Yes
IPv6 + TCP/UDP packets.	No (N/A)	Yes
IPv4 packet has IP options (IP header is longer than 20 bytes).	Yes	Yes

Table 6.25. Supported Receive Checksum Capabilities (Continued)

Packet Type	Hardware IP Checksum Calculation	Hardware TCP/UDP Checksum Calculation
IPv6 packet with next header options: • Hop-by-hop options. • Destinations options (without home address). • Destinations options (with home address). • Routing (with segment left 0). • Routing (with segment left > 0). • Fragment.	 No (N/A) No (N/A) No (N/A) No (N/A) No (N/A) No (N/A)	 Yes Yes No Yes No No
Packet has TCP or UDP options.	Yes	Yes
IPv4 tunnels: • IPv4 packet in an IPv4 tunnel. • IPv6 packet in an IPv4 tunnel.	 Yes (outer IPv4 only) Yes (IPv4)	 No No
IPv6 tunnels: • IPv4 packet in an IPv6 tunnel. • IPv6 packet in an IPv6 tunnel.	 No No (N/A)	 No No
Packet is an IPv4 fragment.	Yes	UDP checksum assist
Packet is greater than 1522 bytes.	Yes	Yes
Packet has 802.3ac tag.	Yes	Yes
NVGRE: • Inner Ipv4 packet • Inner Ipv6 packet	 Yes (Inner and outer) Yes (outer)	 Yes Yes
VXLAN: • Inner Ipv4 packet • Inner Ipv6 packet	 Yes (Inner and outer) Yes (outer)	 Yes (inner only)[1] Yes (inner only)

1. The outer UDP header of VXLAN packets do not use a checksum.

6.1.6.4 SCTP Receive Offload

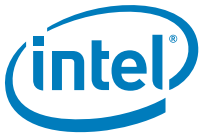
If a receive packet is identified as SCTP, the integrated 10 GbE LAN controller checks the CRC32 checksum of this packet and identifies this packet as SCTP. Software is notified of the CRC check via the L4I and L4E bits in the *Extended Status* field and *Extended Error* field in the Rx descriptor. The 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 6.26. 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			



6.1.6.5 Receive UDP Fragmentation Checksum

The integrated 10 GbE LAN controller might provide a receive fragmented UDP checksum offload for IPv4 non-tunneled packets. The *RXCSUM.PCSD* bit should be cleared and the *RXCSUM.IPPCSE* bit should be set to enable this mode.

The following table lists the outcome descriptor fields for the following incoming packets types.

Incoming Packet Type	Fragment Checksum	UDPV	UDPCS / L4CS
Non-IP packet	0	0	0
IPv6 packet Non fragmented IPv4 packet	0	0	Depends on transport UDP: 1 / 1 TCP: 0 / 1
Fragmented IPv4 with protocol = UDP, first fragment (UDP protocol present)	The unadjusted 1's complement checksum of the IP payload if the checksum in a packet header is different than zero and zero, otherwise.	1 if the UDP header checksum is valid (not 0)	0 / 0
Fragmented IPv4, when not first fragment	The unadjusted 1's complement checksum of the IP payload	1	0 / 0
Fragmented IPv4 with protocol = UDP, first fragment (UDP protocol present) within a VXLAN or NVGRE packet.	The unadjusted 1's complement checksum of the inner IP payload	1 if the UDP header checksum is valid (not 0)	1 / 0
Fragmented IPv4 when not first fragment within a VXLAN or NVGRE packet.	The unadjusted 1's complement checksum of the inner IP payload	0	1 / 0

Note: When the driver computes the 16-bit ones complement sum on the incoming packets of the UDP fragments, it should expect a value of 0xFFFF.

6.1.7 Receive Statistics

6.1.7.1 General rules

- All Statistics registers are cleared on read. In addition, they stick at 0xFF...F when the maximum value is reached.
- For the receive statistics it should be noted that a packet is indicated as received if it passes the device filters and is placed into the packet buffer memory. A packet does not have to be DMA'ed to host memory 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 increased. This is extremely unlikely due to expected delays associated with the system interrupt-collection and ISR delay, but might be an explanation for interrupt statistics values that do not quite make sense. Hardware guarantees that any event noteworthy of inclusion in a statistics count is reflected in the appropriate count within 1 µs; a small time-delay prior to reading the statistics might be required to avoid a potential mismatch between and interrupt and its cause.
- If RSC is enabled, statistics are collected before RSC is applied to the packets.
- All byte (octet) counters composed of 2 registers can be fetched by two consecutive 32-bit accesses while reading the Low 32-bit register first or a single 64-bit access.
- All receive statistic counters count the packets and bytes before coalescing by the RSC logic.

- 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

6.1.7.2 Receive Statistics Hierarchy

The following diagram describes the relations between the packet flow and the different statistic counters.

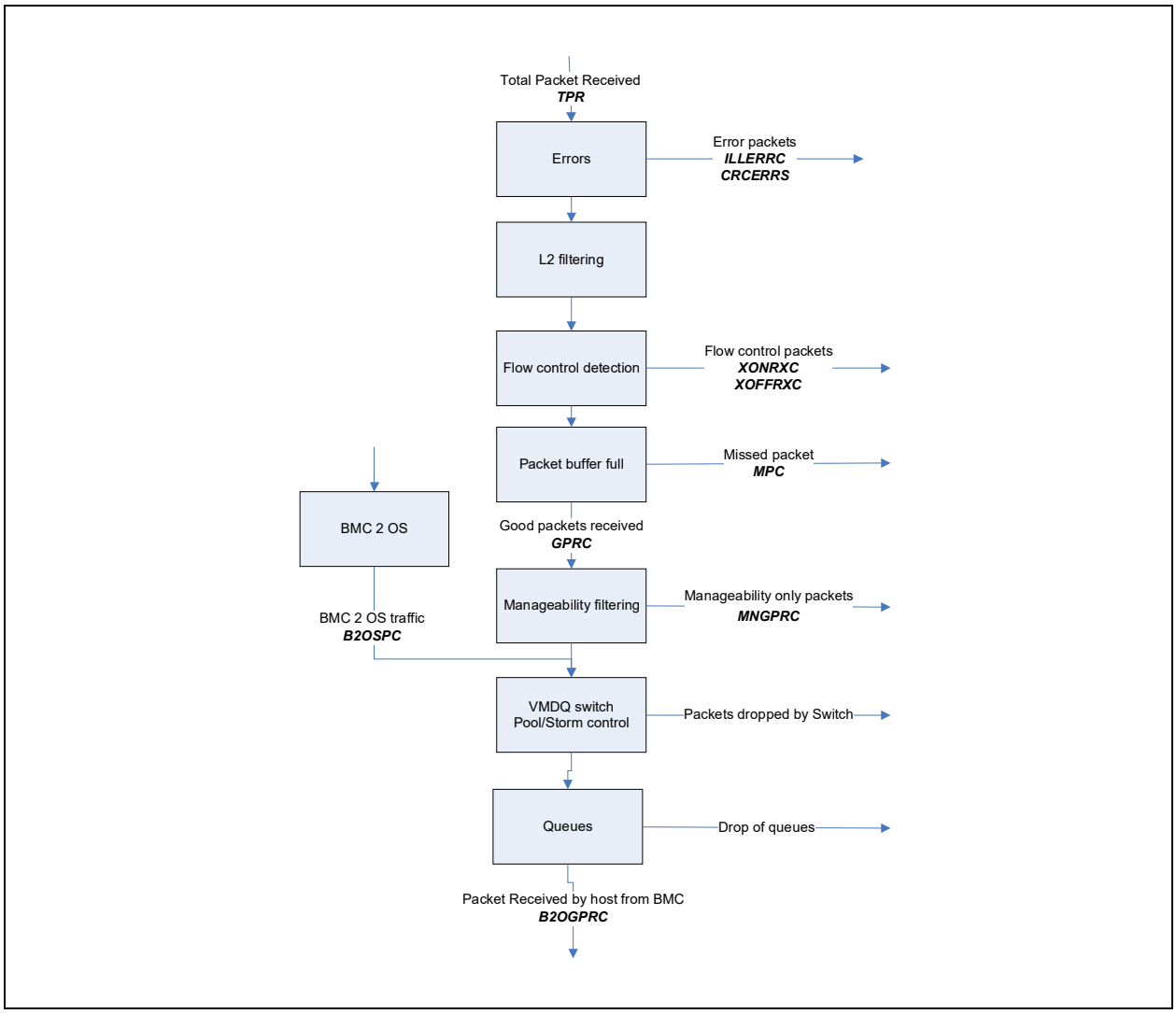


Figure 6.13. Receive Flow Statistics

6.2 Transmit Functionality

6.2.1 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.

6.2.1.1 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 6.14). The following rules must be kept:

- Address alignment of the data buffers can be on any byte boundary.
- Data buffers of any transmitted packet must include at least the 12 bytes of the source and destination Ethernet MAC addresses as well as the 2 bytes of the *Type/Len* field.
- A packet (or multiple packets in transmit segmentation) can span any number of buffers (and their descriptors) up to a limit of 40 minus *WTHRESH* minus 2 (see Section 6.2.3.3 for Tx Ring details and section Section 6.2.3.5.1 for *WTHRESH* details). For best performance it is recommended to minimize the number of buffers as possible.

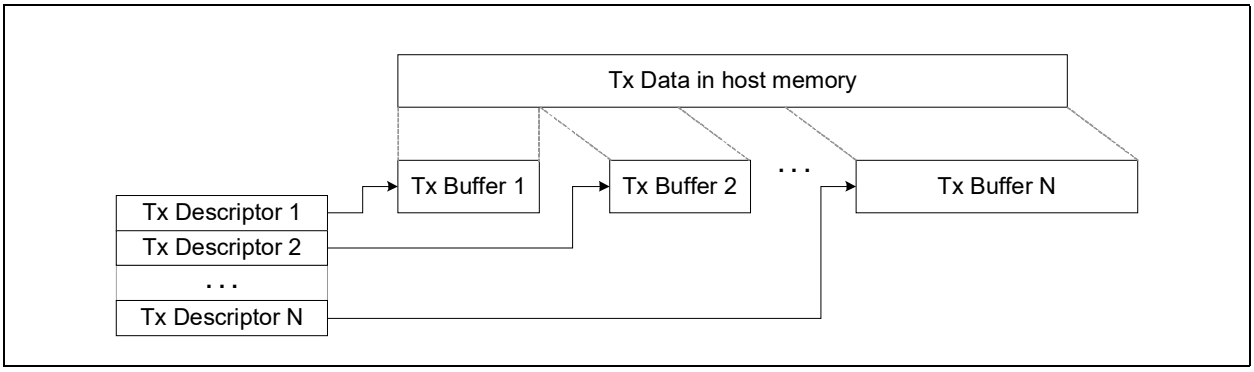


Figure 6.14. Tx Packet in Host Memory

6.2.1.2 Transmit Path in the integrated 10 GbE LAN controller

The transmit path in the integrated 10 GbE LAN controller consists of the following stages:

- Descriptor plane
  - The integrated 10 GbE LAN controller maintains a set of 128 on-die descriptor queues. Each queue is associated with a single descriptor ring in system memory. See Section 6.2.3.3 for more details on the Tx descriptor rings. Each on-die descriptor queue contains up to 40 descriptors in order to achieve the desired performance.

  - A fetch mechanism loads Tx descriptors from the descriptor rings in system memory to the respective descriptor queues in the integrated 10 GbE LAN controller. A descriptor fetch arbiter determines the order in which descriptors are fetched into the various on-die descriptor queues. See Section 6.2.3.4 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 6.6.
  - Once a packet has been fetched into a packet buffer, status is (optionally) written back into system memory. See Section 6.2.3.5 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 6.2.1.2.2.
  - 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 6.6.

6.2.1.2.1 Tx Queues Assignment

The integrated 10 GbE LAN controller 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:

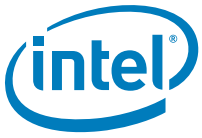
- Virtualization (VT) - In a virtualized environment, DMA resources are shared between more than one software entity (operating system and/or device driver). This is done through allocation of transmit descriptor queues to virtual partitions (VMM, IOVM, VMs, or VFs). Allocation of queues to virtual partitions 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.
- 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 6.27 lists the queuing schemes. Scheme selection is done via the MTQC register.

Table 6.27. Tx Queuing Schemes

VT	Queues Allocation	Packet Buffers Allocation
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	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	32 pools x 4 queues, or 64 pools x 2 queues	A single packet buffer for all traffic.
Yes	16 pools x 8 TCs, or 32 pools x 4 TCs	A separate packet buffer is allocated to each TC (total of four or eight).

Note: Software can use any number of queues per each TC or per each pool within the allocated ranges previously described by disabling any unused queue.



Note: Programming MTQC must be done only during the init phase, while software must also set RTTDCS.ARBDIS before configuring MTQC and then clear RTTDCS.ARBDIS afterwards.

Table 6.28. Queues, Pools and TCs Programming

Device Setting (MTQC)		Device Functionality	
VT_Ena	**NUM_TC_OR_Q**	**Total number of Tx Queues**	**TC & pools**
0	00	64	-
<> 0	00	Non valid configuration	
0	<> 00	Non valid configuration	
0	01	N/A	
0	10	128	TC0 — TC3
0	11	128	TC0 — TC7
1	01	128	64 pools
1	10	128	32 pools
1	11	N/A	
1	01	N/A	
1	10	128	TC0 — TC3 & 32 pools
1	11	128	TC0 — TC7 & 16 pools

Allocating descriptor queues to VFs uses a consistent indexing over the different Tx queuing schemes. The most significant bits of the queue index represent the VF index, and the least significant bits are either the TC index or are used by software to dispatch traffic according to a Transmit Side Scaling (TSS) algorithm — similar to RSS in the Rx path.

The Tx queue numbers associated with the TCs are listed in the following tables: Table 6.29 and Table 6.30.

Table 6.29. Tx Queues Indexing When VT-on

VT Mode	Allocation of Queue Index Bits According to						
	6	**5**	**4**	**3**	**2**	**1**	**0**
64 VFs + TSS	VF or pool (63..0)						TSS
32 VFs + TSS *or* 4 TCs	VF or pool (31..0)					TSS / TC	
16 VFs + 8 TCs	VF (15..0)				TC		

Table 6.30. Tx Queues Indexing When VT-off

TC Mode	TCn	# of Qs	Queues Attached to TCn
4 TCs	TC0	64	0xxxxxx
	TC1	32	10xxxxx
	TC2	16	110xxxx
	TC3	16	111xxxx

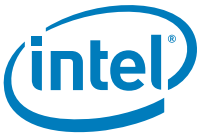


Table 6.30. Tx Queues Indexing When VT-off (Continued)

TC Mode	TCn	# of Qs	Queues Attached to TCn
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.

6.2.1.2.2 Tx Packet Buffers

As previously described, the following modes exist for the integrated 10 GbE LAN controller 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.

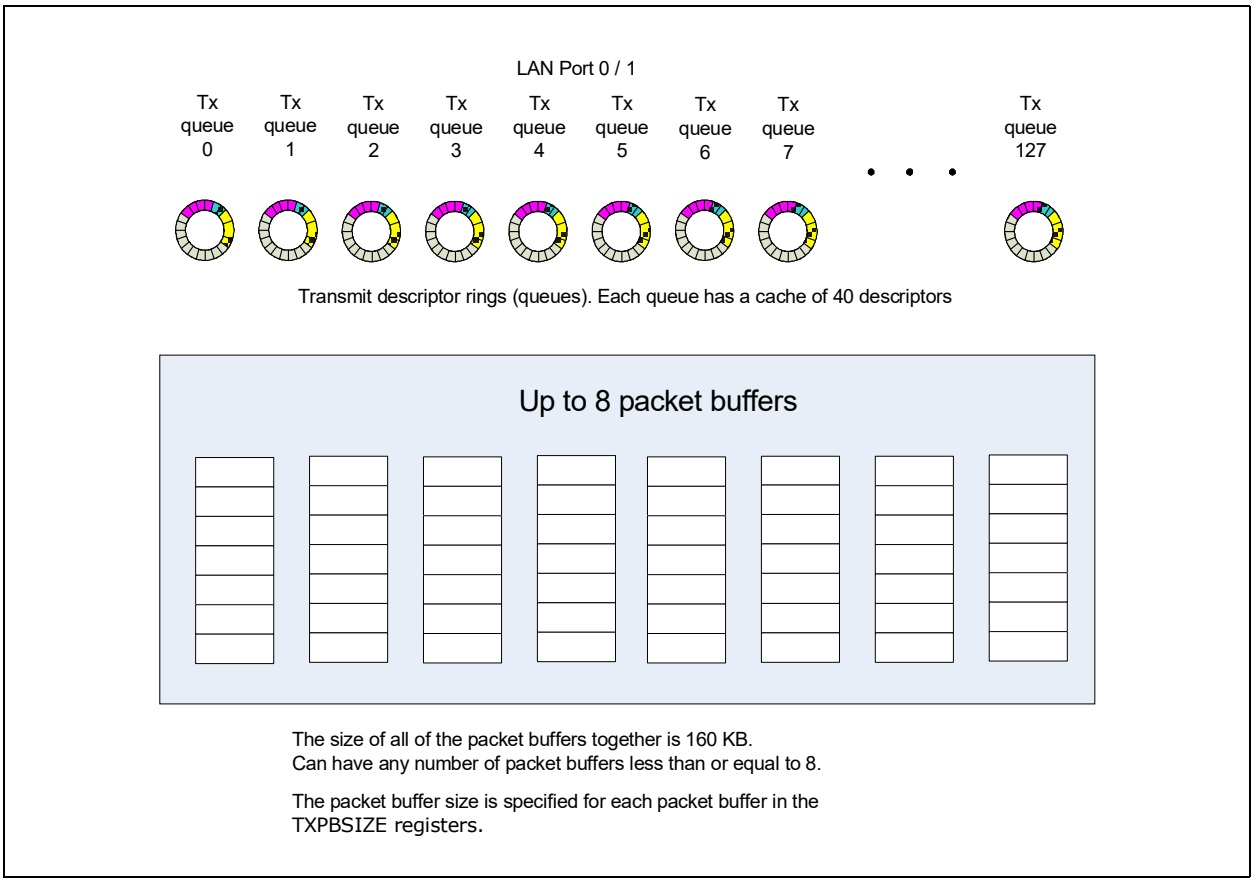


Figure 6.15. Tx Arbitration Schemes

6.2.1.2.3 Tx Arbitration Schemes

There are basically four Tx arbitration schemes, one per each combination of Virtualization (VT) enabled/disabled modes. They are configured via the *MTQC.MTQE* register field.

6.2.1.2.3.1 VT-on

When virtualization is enabled, queues are allocated to the packet buffers in a fixed manner, the same number of queues per each TC.

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 6.5.2.1. 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 6.5.2.1. 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 6.5.2.1.
- **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.

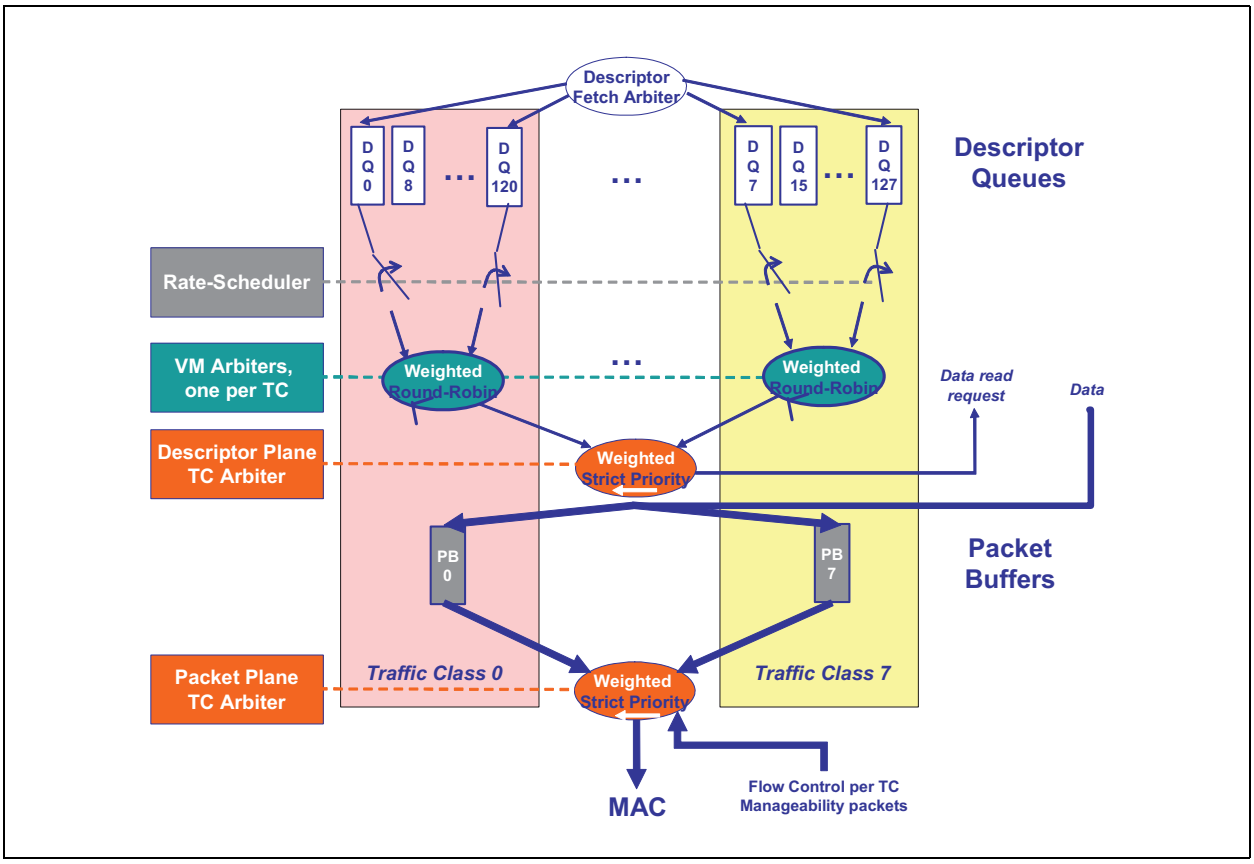


Figure 6.16. Transmit Architecture VT-on — Eight TCs Mode

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.

6.2.1.2.3.2 VT-off

When virtualization disabled, queues are allocated to the packet buffers in a fixed manner according to the number of TCs. In Figure 6.17, eight TCs mode is shown.

- The unique difference with the 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.

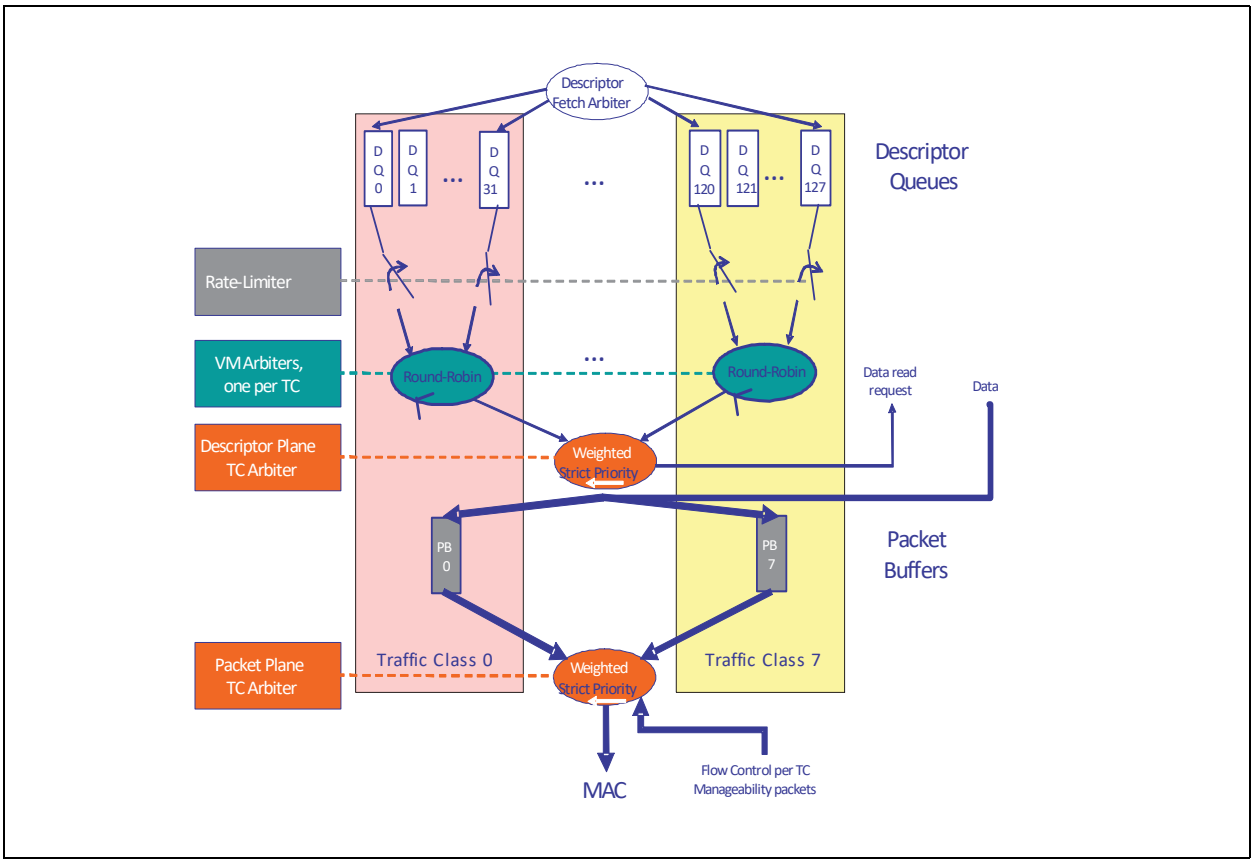


Figure 6.17. Transmit Architecture VT-off — Eight TCs Mode

6.2.1.2.3.3 VT-on

When virtualization is 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 6.5.2.1.

Packet Plane Arbiter:

- **Manageability** packets are inserted with strict priority over data packets.

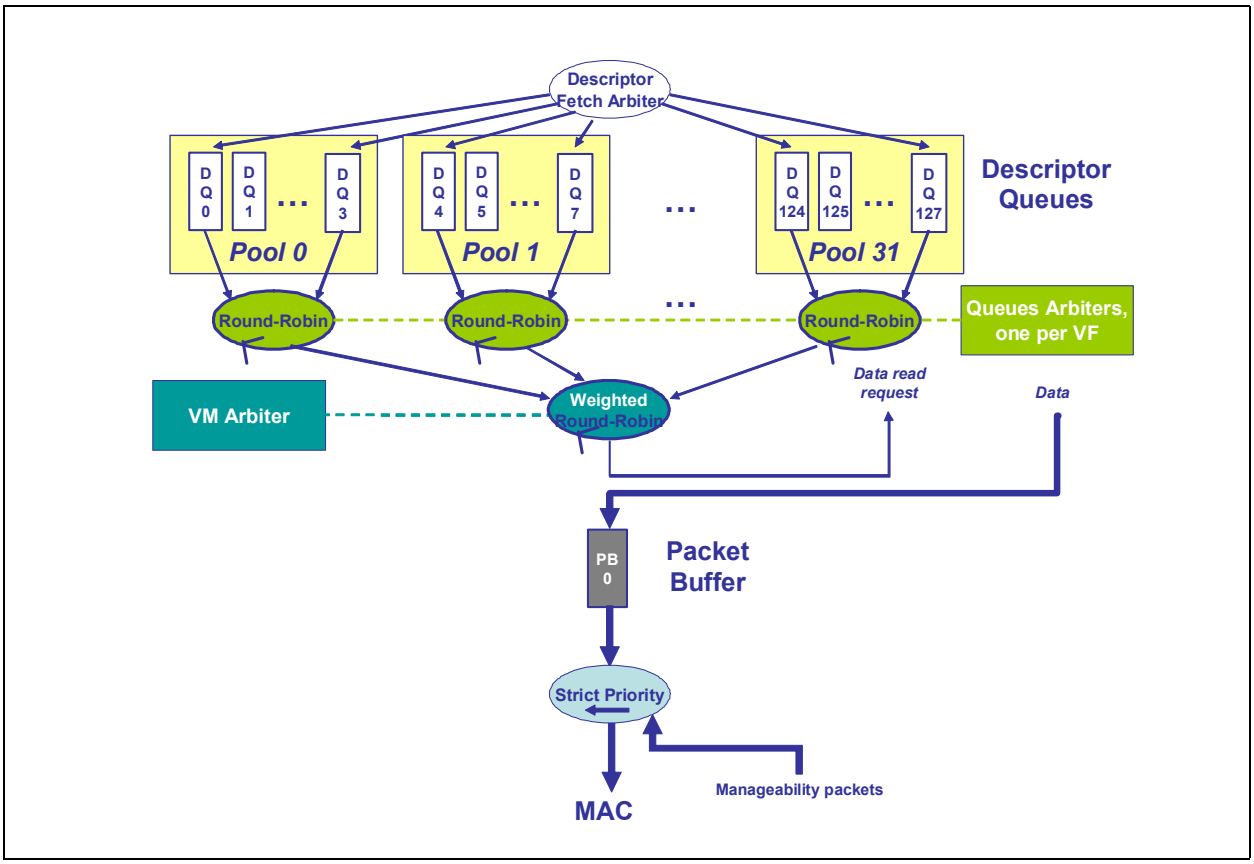


Figure 6.18. Transmit Architecture VT-on — 32 VFs

6.2.1.2.3.4 VT-off

When virtualization is 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.

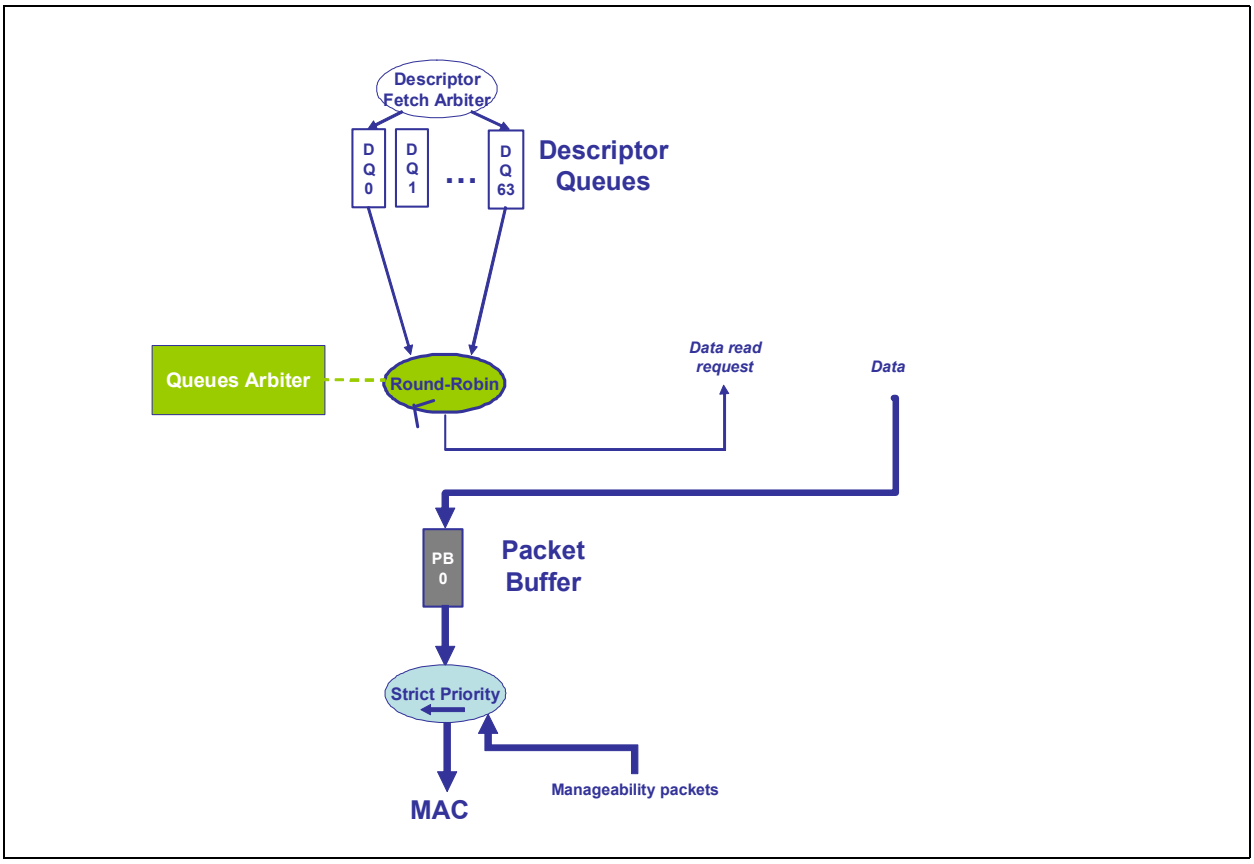


Figure 6.19. Transmit Architecture VT-off

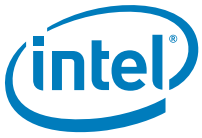
6.2.2 Transmit Contexts

The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller 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.



6.2.3 Transmit Descriptors

6.2.3.1 Introduction

The integrated 10 GbE LAN controller 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).

In addition, the integrated 10 GbE LAN controller 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.

6.2.3.2 Transmit Descriptors Formats

6.2.3.2.1 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 6.31. VLAN Tag

Byte address N + 1 -> first byte on the wire Bit 7 — first on the wire <- Bit 0			Byte address N -> second byte on the wire Bit 7 -> last on the wire — Bit 0
PRI (3 bits)	DEI	VID (4 bits)	VID (8 bits)

6.2.3.2.2 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 6.32. 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 6.32. 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 6.33. Transmit Descriptor Write-Back Format — Legacy Mode

	63 48	47 40	39 36	35 32	31 24	23 16	15 0
0	Reserved				Reserved		
8	VLAN	CSS	Rsvd	STA	CMD	CSO	Length

Buffer Address (64) and Length (16)

The buffer address is a byte-aligned address. Length (*TDESC.LENGTH*) specifies the length in bytes to be fetched from the buffer address provided. The maximum length associated with a single descriptor is 15.5 KB while the total frame size must meet the maximum supported frame size. There is no limitation for the minimum buffer size.

Note: Descriptors with zero length (null descriptors) transfer no data. Null descriptors might appear only between packets and must have their *EOP* bits set.

Checksum Offset and Start — CSO (8) and CSS (8)

A *Checksum Offset* (*TDESC.CSO*) field indicates where, relative to the start of the packet, to insert a TCP checksum if this mode is enabled. A *Checksum Start* (*TDESC.CSS*) field indicates where to begin computing the checksum. Note that *CSO* and *CSS* are meaningful only in the first descriptor of a packet.

Both *CSO* and *CSS* are in units of bytes. These must both be in the range of data provided to the device in the descriptor. This means for short packets that are padded by software, *CSO* and *CSS* must be in the range of the un-padded data length, not the eventual padded length (64 bytes).

CSO must be set to the location of TCP or UDP checksum in the packet. CSS must be set to the beginning of the IP header or the L4 (TCP/UDP) header. Checksum calculation is not done if *CSO* or *CSS* are out of range. This occurs if (*CSS* > length) OR (*CSO* > length - 1).

For the 802.1Q header, the offset values depend on the VLAN insertion enable bit — the *VLE* bit. If they are not set (VLAN tagging included in the packet buffers), the offset values should include the VLAN tagging. If these bits are set (VLAN tagging is taken from the packet descriptor), the offset values should exclude the VLAN tagging.

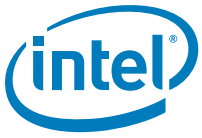
Hardware does not add the 802.1q Ethertype or the VLAN field following the 802.1Q Ethertype to the checksum. So for VLAN packets, software can compute the values to back out only on the encapsulated packet rather than on the added fields.

Note: UDP checksum calculation is not supported by the legacy descriptor because the legacy descriptor does not support the translation of a checksum result of 0x0000 to 0xFFFF needed to differentiate between an UDP packet with a checksum of zero and an UDP packet without checksum.

Because the *CSO* field is eight bits wide, it puts a limit on the location of the checksum to 255 bytes from the beginning of the packet.

Note: *CSO* must be larger than *CSS*.

Software must compute an offsetting entry to back out the bytes of the header that should not be included in the TCP checksum and store it in the position where the hardware computed checksum is to be inserted.



Hardware adds the checksum at the byte offset indicated by the *CSO* field. Checksum calculations are for the entire packet starting at the byte indicated by the *CSS* field. The byte offset is counted from the first byte of the packet fetched from host memory.

Command Byte — CMD (8)

The CMD byte stores the applicable command and has the fields listed in Table 6.34.

Table 6.34. 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 6.4.5 for details about double VLAN.

Table 6.35. VLAN Tag Insertion Decision Table for VLAN Mode Enabled

VLE	Action
0	Send generic Ethernet packet.
1	Send 802.1Q packet; the *Ethernet Type* field comes from the *VET* field of the VLNCTRL register and the VLAN data comes from the *VLAN* field of the TX descriptor.

Note: This table is relevant only if PFVMVIR.VLANA = 00b (use descriptor command) for the queue.

- *DEXT (bit 5)* — Descriptor extension (zero for legacy mode)
- *RSV (bit 4)* — Reserved
- *RS (bit 3)* — Report Status - RS signals hardware to report the DMA completion status indication as well as triggering ITR. Hardware indicates a DMA completion by setting the *DD* bit in the transmit descriptor when *TDWBAL[n].Head_WB_En* = 0b or by Head Write-back if *Head_WB_En* = 1b (see Section 6.2.3.5.2). 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 6.2.3.5.1). 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 6.2.3.5.2.

- *IC (bit 2)* — Insert Checksum - Hardware inserts a checksum at the offset indicated by the *CSO* field if the *Insert Checksum* bit (IC) is set.
- IFCS (bit 1) — Insert FCS - When set, hardware appends the MAC FCS at the end of the packet. When cleared, software should calculate the FCS for proper CRC check. There are several cases in which software must set IFCS as follows:
  — Transmitting a short packet while padding is enabled by the *HLREG0.TXPADEN* bit.
  — Checksum offload is enabled by the *IC* bit in the *TDESC.CMD*.
  — VLAN header insertion enabled by the *VLE* bit in the *TDESC.CMD* or by the *PFVMVIR.VLANA* fields.

— E-tag insertion enabled by the *PFVMVIR.TAGA* fields.

— TSO or TCP/IP checksum offload using a context descriptor.

Note that TSO offload are relevant only to advanced Tx descriptors.

- EOP (bit 0) — End of Packet - A packet can be composed of multiple buffers (each of them indicated by its own descriptor). When EOP is set, it indicates the last descriptor making up the packet.

Note: *VLE*, *IFCS*, and *IC* fields should be set in the first descriptor of a packet. The *RS* bit can be set only on the last descriptor of a packet. The *DEXT* bit must be set to zero for all descriptors. The *EOF* bit is meaningful in all descriptors.

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 6.2.3.2.1 for more details.

Table 6.36. VLAN Field (TDESC.VLAN) Layout

15 13	12	11 0
PRI	DEI	VLAN

6.2.3.2.3 Advanced Transmit Context Descriptor

Table 6.37. Transmit Context Descriptor (TDESC) Layout — (Type = 0010)

	63 56	55 48	47 42	41 32	31 16	15 9	8 0
0	TUNNELLEN	OUTERIPLEN	Reserved		VLAN	MACLEN	IPLEN/HEADLEN

8	MSS	L4LEN	RSV	IDX	Reserved must be 0x3F	DEXT	RSV	DTYP	TUCMD		
	63 48	47 40	39 37	36	35 30	29	28 24	23 20	19 9	8 0	

IPLEN/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.

MACLEN (7)

- This field indicates the length of the MAC header. When an offload is requested, one of the TSE bits (in the advanced transmit data descriptor) or *IXSM* bit or *TXSM* bit are set, MACLEN must be larger than or equal to 14, and less than or equal to 127. This field should include only the part of the L2 header supplied by the driver and not the parts added by hardware. The following table lists the value of MACLEN in the different cases.

	Regular VLAN	Extended Tag	MACLEN
	By hardware or none	None	14
	By hardware or none	VLAN	18
	By hardware or none	E-tag	22
	By software	None	18
	By software	VLAN	22
	By software	E-tag	26

VLAN (16)

This field contains the 802.1Q VLAN tag to be inserted in the packet during transmission. This VLAN tag is inserted when a packet using this context has its DCMD.VLE bit is set. This field should include the entire 16-bit VLAN field including DEI and priority fields as listed in Table 6.36.

Note that the *VLAN* field is provided in network order. See Section 6.2.3.2.1.

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.

- *SOF (bit 2)* — Start of frame delimiter index.
- *ORIS (bit 5)* — Orientation relative to the first frame in an FC sequence.

- PARINC (bit 3) — When this bit is set, hardware relates to the *PARAM* field in the FC header as relative offset. In this case, hardware increments the *PARAM* field in TSO by an MSS value on each transmitted packet of the TSO. Software should set the *PARINC* bit when it sets the *Relative Offset Present* bit in the F_CTL.

OUTERIPLEN (8) - This field holds the value of the outer IP header length.

TUNNELLEN (8) - This field holds the length value of the tunnel headers including the inner L2 header.

TUCMD (11)

- *RSV (bit 10-8)* — Reserved
- *Inner VLAN (bit 7)* — If set, the packet is a tunnel packet with a VLAN in the inner L2 header
- *Reserved (bit 6:4)*
- *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

(bit 0) 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 integrated 10 GbE LAN controller 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.

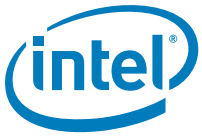
TCP / UDP Segmentation

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] + OUTERIPLEN + TUNNELLEN + IPLEN + L4LEN + MSS

Note: The headers lengths must meet the following: MACLEN + IPLEN + L4LEN <= 512

Address (64)



This field holds the physical address of a data buffer in host memory, which contains a portion of a transmit packet. This field is meaningful in all descriptors.

DTALEN (16)

This field holds the length in bytes of data buffer at the address pointed to by this specific descriptor. This field is meaningful in all descriptors. The maximum length is 16 KB with no limitations on the minimum size. Refer to the comment on descriptors with zero length described in the sections that follow.

TUNNEL Type (2)

00b = VXLAN
10b = NVGRE
10b = Geneve
11b = Reserved

- *OUTERIPCS (bit 1)* — Outer IP checksum enable. Indicates this is a tunnel packet and that the *OUTERIPLEN/TUNNELLEN* parameters in the context descriptor are valid.

MAC (2) — see the following. This field is meaningful on the first descriptor of the packet(s).

- *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 segmentation request. When *TSE* is set in the first descriptor of a TCP 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 6.2.3.2.2.
- *OUTERIPCS (bit 2)* — Outer IP checksum enable. Indicates this is a tunnel packet and that the OUTERIPLEN/TUNNELLEN parameters in the context descriptor are valid.
- *IFCS (bit 1)* — Insert FCS - When this bit is set, the hardware appends the MAC FCS at the end of the packet. When cleared, software should calculate the FCS for proper CRC check. There are several cases in which software must set IFCS as follows:
  - — Transmitting a short packet while padding is enabled by the *HLREG0.TXPADEN* bit.
  - — Checksum offload is enabled by the either *IC, TXSM* or *IXSM* bits in the *TDESC.DCMD.*
  - — VLAN header insertion enabled by the *VLE* bit in the *TDESC.DCMD*.
  - — TCP segmentation offload enabled by the *TSE* bit in the *TDESC.DCMD*.
- *EOP (bit 0)* — End of Packet - A packet might be composed of multiple buffers (each of them is indicated by its own descriptor). When *EOP* is set, it indicates the last descriptor making up the packet. In transmit segmentation (explained later on in this section) the *EOP* flag indicates the last descriptor of the last packet of the segmented transmission.

Note: *TSE*, *VLE* and *IFCS* fields should be set in the first descriptor of the packet(s). The *RS* bit can be set only on the last descriptor of the packet. The *EOP* bit is valid in all descriptors. The *DEXT* bit must be set to 1b for all descriptors.

Descriptors with zero length, transfer no data. If the *RS* bit in the command byte is set, 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 6.2.4.2.2.

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 6.39 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. Tx switching is enabled (including anti spoof checks).

POPTS (6)

- *Rsv (bit 5:2)* — Reserved
- *TXSM (bit 1)* — Insert TCP/UDP Checksum: When set to 1b, the L4 checksum must be inserted. In this case, TUCMD.LP4 indicates whether the checksum is TCP or UDP or SCTP. When DCMD.TSE is set, TXSM must be set to 1b. If this bit is set, the packet should at least contain a TCP header.
- *IXSM (bit 0)* — Insert IP Checksum: This field indicates that IP checksum must be inserted. In IPv6 mode, it must be reset to 0b. If DCMD.TSE and TUCMD.IPV4 are set, IXSM must be set to 1b. If this bit is set, the packet should at least contain an IP header.

This field is relevant only on the first descriptor of the packet(s).

PAYLEN (18)

PAYLEN indicates the size (in byte units) of the data buffer(s) in host memory for transmission.

In a single-send packet, PAYLEN defines the entire packet size fetched from host memory. It does not include the fields that hardware adds such as: optional VLAN tagging and Ethernet CRC or Ethernet padding.

In TSO (regardless if it is transmitted on a single or multiple packets), the PAYLEN defines the protocol payload size fetched from host memory.

- In TCP or UDP segmentation offload, PAYLEN defines the TCP/UDP payload size.

This field is relevant only on the first descriptor of the packet(s). The minimum transmitted packet size excluding VLAN padding and CRC bytes is 17 and the PAYLEN size should meet this limitation. On a single-packet send, the maximum size of the PAYLEN is dictated by the maximum allowed packet size which is 15.5 KB. On TSO, the maximum PAYLEN can be up to 2^{18}-1.

Note: When a packet spreads over multiple descriptors, all of the descriptor fields are valid only on the first descriptor of the packet, except for *RS* and *EOP* bits, which are set on the last descriptor of the packet.

6.2.3.3 Transmit Descriptor Ring

The transmit descriptor ring structure (shown in Figure 6.20) uses a contiguous memory space. A set of four registers (described later in this section) maintain the transmit descriptor ring in the host memory. Hardware maintains internal circular queues of 64 descriptors per queue to hold the descriptors that were fetched from the software ring.

Descriptors handled to hardware should not be manipulated by software until hardware completes its processing. It is indicated by advancing the head pointer beyond these descriptors.

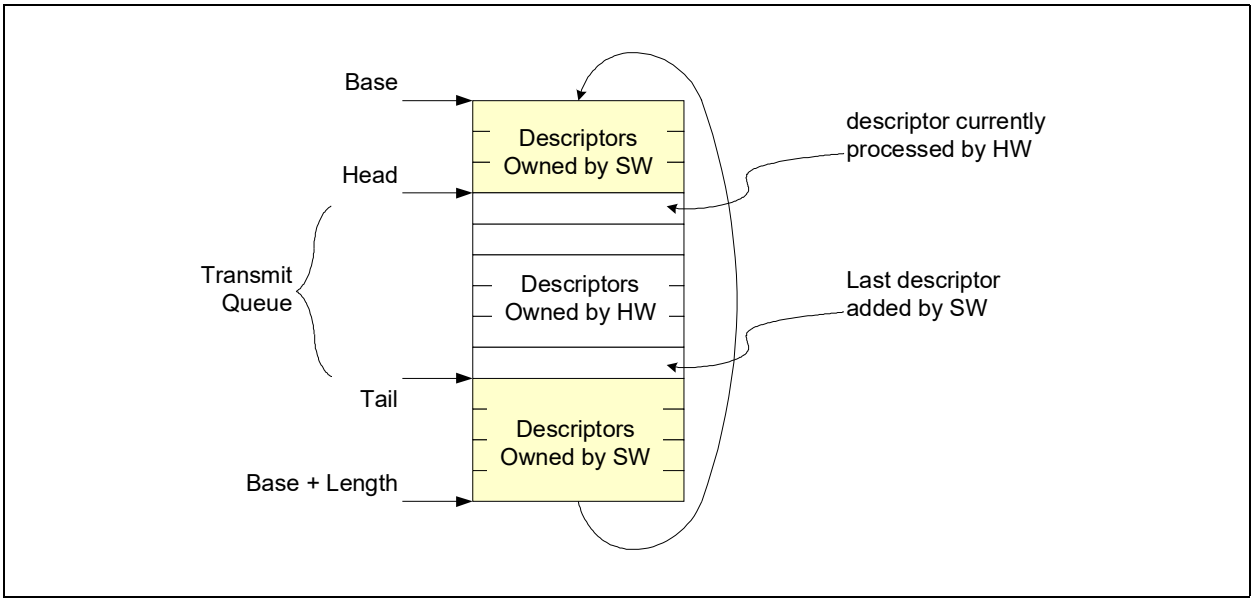


Figure 6.20. Transmit Descriptor Ring Structure

The transmit descriptor ring is defined by the following registers:

- Transmit Descriptor Base Address register (TDBA 0-127) — This register indicates the start address of the descriptor ring buffer in the host memory; this 64-bit address is aligned on a 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.

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 6.2.3.5.2.
- 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.

6.2.3.4 Transmit Descriptor Fetching

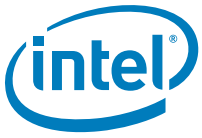
The integrated 10 GbE LAN controller fetches new descriptors as required for packet transmission depending on its on-die descriptor buffer state:

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.

6.2.3.4.1 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 6.2.3.5.1.

- 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.

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.

6.2.3.5 Transmit Write Back

The integrated 10 GbE LAN controller 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

6.2.3.5.1 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.

6.2.3.5.2 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.

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.

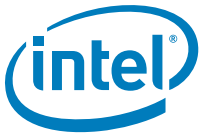
The driver has control of this feature per queue through the TDWBAL and TDWBAH registers.

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.



6.2.4 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 integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller saves significant CPU cycles. The device driver shares the additional tasks to support these options with the integrated 10 GbE LAN controller.

6.2.4.1 Assumptions and Restrictions

The following assumptions apply to the TCP / UDP segmentation implementation in the integrated 10 GbE LAN controller:

- To limit the internal cache dimensions, software is required to spread the header onto a maximum four descriptors, while still allowed to mix header and data in the last header buffer. This limitation stands for up to Layer 4 header included, and for IPv4 or IPv6 independently.
- The maximum size of a single TSO can be as large as defined by the *PAYLEN* field in the Tx data descriptor (such as up to 256 KB).
- The *RS* bit operation is not changed. Interrupts are set after data in the buffers pointed to by individual descriptors is transferred (DMA'ed) to hardware.
- SNAP packets are not supported for segmentation.
- IP in IP tunneled packets are not supported for offloading under TSO operation. VXLAN and NVGRE tunneled packets can be segmented.
- Software must enable the Ethernet CRC offload in the HLREG0.TXCRCEN register since CRC must be inserted by hardware after the checksum has been calculated.
- Software must initialize the appropriate checksum fields in the packet’s header.

6.2.4.2 Transmission Process

The transmission process involves the following:

- The protocol stack receives from an application a block of data that is to be transmitted.
- The protocol stack calculates the number of packets required to transmit this block based on the MTU size of the media and required packet headers.
- The stack interfaces with the device driver and passes the block down with the appropriate header information: Ethernet, IP and TCP / UDP headers.

- The stack interfaces with the device driver and commands the driver to send the individual packet. The device driver sets up the interface to the hardware (via descriptors) for the TCP / UDP segmentation.
- The hardware transfers (DMA's) the packet data and performs the Ethernet packet segmentation and transmission based on offset and payload length parameters in the TCP/IP or UDP/IP context descriptor including:
  — Packet encapsulation
  — Header generation and field updates including IPv4/IPv6 and TCP/UDP checksum generation.
- The driver returns ownership of the block of data to the NOS when the hardware has completed the DMA transfer of the entire data block.

6.2.4.2.1 TCP and UDP Segmentation Data Fetch Control

To perform TCP / UDP segmentation in the integrated 10 GbE LAN controller, 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 integrated 10 GbE LAN controller enables interleaving between different TSO requests at an Ethernet packet level. In other words, the integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller 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.

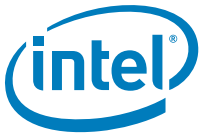
6.2.4.2.2 TCP and UDP Segmentation Write-back Modes

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.

6.2.4.3 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.



6.2.4.4 Packet Format

A TCP / UDP message can be as large as 256 KB and is generally fragmented across multiple pages in host memory. The integrated 10 GbE LAN controller partitions the data packet into standard Ethernet frames prior to transmission. The integrated 10 GbE LAN controller supports calculating the Ethernet, IP, TCP, and UDP headers, including checksum, on a frame-by-frame basis. For tunneled packets (NVGRE, VXLAN) also supports updating the outer IPv4 header.

Table 6.38. TCP/IP and UDP/IP Packet Format Sent by Host

Pseudo Header				Data
Ethernet	Optional tunnel header	IPv4/IPv6	TCP/UDP	DATA (full TCP/UDP message)

Table 6.39. Packets Format Sent by Device

Pseudo Header (updated)	Data (first MSS)	FCS	...	Pseudo Header (updated)	Data (Next MSS)	FCS	...

Frame formats supported by the integrated 10 GbE LAN controller include:

- Ethernet 802.3
- IEEE 802.1Q VLAN (Ethernet 802.3ac)
- Ethernet Type 2
- NVGRE (see frame format in Section A.2.5.1)
- VXLAN (see frame format in Section A.2.5.2)
- IPv4 headers with options
- IPv6 headers with extensions
- TCP with options
- UDP with options

VLAN tag insertion can be handled by hardware.

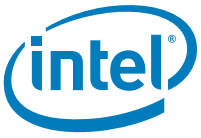
Note: UDP (unlike TCP) is not a reliable protocol and fragmentation is not supported at the UDP level. UDP messages that are larger than the MTU size of the given network medium are normally fragmented at the IP layer. This is different from TCP, where large TCP messages can be fragmented at either the IP or TCP layers depending on the software implementation.

The integrated 10 GbE LAN controller has the ability to segment UDP traffic (in addition to TCP traffic); however, because UDP packets are generally fragmented at the IP layer, the integrated 10 GbE LAN controller's segmentation capability might not be used in practice for UDP.

6.2.4.5 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 6.2.3). The purpose of this descriptor is to provide information to the hardware to be used during the TCP / UDP segmentation offload process.

Setting the *TSE* bit in the DCMD field to one (in the data descriptor) indicates that this descriptor refers to the segmentation context (as opposed to the normal checksum offloading context). This causes the checksum offloading, packet length, header length, and maximum segment size parameters to be loaded from the descriptor into the device.



The TCP / UDP segmentation prototype header is taken from the packet data itself. Software must identity the type of packet that is being sent (IPv4/IPv6, TCP/UDP, other), calculate appropriate checksum off loading values for the desired checksums, and then calculate the length of the header that is pre-appended. The header can be up to 240 bytes in length.

Once the TCP / UDP segmentation context has been set, the next descriptor provides the initial data to transfer. This first descriptor(s) must point to a packet of the type indicated. Furthermore, the data it points to might need to be modified by software as it serves as the prototype header for all packets within the 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 checksums of inner and outer IP headers should be zeroed out unless some adjustment is needed by the driver.

TCP Header

- Sequence number should be set as appropriate for first packet of send (if not already).
- PSH, and FIN flags should be set as appropriate for LAST packet of send.
- TCP checksum should be set to the partial pseudo-header checksum as follows (there is a more detailed discussion of this in Section 6.2.4.6:

Table 6.40. TCP Partial Pseudo-header Checksum for IPv4

IP Source Address		
IP Destination Address		
Zero	Layer 4 Protocol ID	Zero

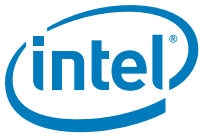
Table 6.41. TCP Partial Pseudo-header Checksum for IPv6

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.



6.2.4.6 Transmit Checksum Offloading with TCP and UDP Segmentation

The integrated 10 GbE LAN controller supports checksum offloading as a component of the TCP / UDP segmentation off-load feature and as stand-alone capability. Section 6.2.5 describes the interface for controlling the checksum off-loading feature. This section describes the feature as it relates to TCP / UDP segmentation.

The integrated 10 GbE LAN controller 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 6.42 for the list of supported transmit checksums per packet type.

6.2.4.7 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 integrated 10 GbE LAN controller.

6.2.4.7.1 TCP/IP/UDP Header for the First Frame

The hardware makes the following changes to the headers of the first packet that is derived from each TCP segmentation context.

Tunnel Header

When tunnel offload is enabled (*Tunnel* bit set in the descriptor) the outer IPv4 header is updated as follow

- IP Total Length = OUTERIPLEN + TUNNELLEN + IPLEN + L4LEN + MSS
- Calculates the Outer IP Checksum

IPv4 Header

- IP Total Length = MSS + L4LEN + IPLEN
- Calculates the IP Checksum

IPv6 Header

- Payload Length = MSS + L4LEN + IPV6_HDR_extension[1]

1. IPV6_HDR_extension is calculated as IPLEN — 40 bytes.

TCP Header

- Sequence Number: The value is the sequence number of the first TCP byte in this frame.
- The flag values of the first frame are set by logic AND function between the flag word in the pseudo header and the DTXTCPFLGL.TCP_flg_first_seg. The default values of the DTXTCPFLGL.TCP_flg_first_seg are set. The flags in a TSO that ends up as a single segment are taken from the in the pseudo header in the Tx data buffers as is.
- Calculates the TCP checksum.

UDP Header

- Calculates the UDP checksum.

6.2.4.7.2 TCP/IP Header for the Subsequent Frames

The hardware makes the following changes to the headers for subsequent packets that are derived as part of a TCP segmentation context:

Number of bytes left for transmission = PAYLEN — (N * MSS). Where N is the number of frames that have been transmitted.

Tunnel Header

When tunnel offload is enabled (*Tunnel* bit set in the descriptor) the outer IPv4 header is updated as follow

- IP Identification: increased from last value (wrap around based on 16-bit width)
- IP Total Length = OUTERIPLEN + TUNNELLEN + IPLEN + L4LEN + MSS
- Calculates the Outer IP Checksum

IPv4 Header

- IP Identification: increased from last value (wrap around)
- IP Total Length = MSS + L4LEN + IPLEN
- Calculate the IP Checksum

IPv6 Header

- Payload Length = MSS + L4LEN + IPV6_HDR_extension[1]

TCP Header

- Sequence Number update: Add previous TCP payload size to the previous sequence number value. This is equivalent to adding the MSS to the previous sequence number.
- The flag values of the subsequent frames are set by logic AND function between the flag word in the pseudo header with the DTXTCPFLGL.TCP_Flg_mid_seg. The default values of the DTXTCPFLGL.TCP_Flg_mid_seg are set.
- Calculate the TCP checksum

UDP Header

- Calculates the UDP checksum.

6.2.4.7.3 TCP/IP Header for the Last Frame

Hardware makes the following changes to the headers for the last frame of a TCP segmentation context:

Last frame payload bytes = PAYLEN — (N * MSS).

Tunnel Header

When tunnel offload is enabled (*Tunnel* bit set in the descriptor) the outer IPv4 header is updated as follow

- IP Identification: increased from last value (wrap around based on 16-bit width)
- IP Total Length = OUTERIPLEN + TUNNELLEN + IPLEN + L4LEN + last frame payload bytes.
- Calculates the Outer IP Checksum

IPv4 Header

- IP Total length = last frame payload bytes + L4LEN + IPLEN
- IP identification: increased from last value (wrap around based on 16-bit width)
- Calculate the IP checksum

IPv6 Header

- Payload length = last frame payload bytes + L4LEN + IPV6_HDR_extension[1]

TCP Header

- Sequence number update: Add previous TCP payload size to the previous sequence number value. This is equivalent to adding the MSS to the previous sequence number.
- The flag values of the last frames are set by logic AND function between the flag word in the pseudo header and the DTXTCPFLGH.TCP_Flg_lst_seg. The default values of the DTXTCPFLGH.TCP_Flg_lst_seg are set. The flags in a TSO that ends up as a single segment are taken from the in the pseudo header in the Tx data buffers as is.
- Calculate the TCP checksum

UDP Header

- Calculates the UDP checksum.

6.2.5 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 integrated 10 GbE LAN controller'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 6.2.3.3.

Enabling checksum offload, software must also enable Ethernet CRC offload by the HLREG0.TXCRCEN since CRC must be inserted by hardware after the checksum has been calculated.

Each checksum operates independently. Insertion of the IP and TCP / UDP checksum for each packet are enabled through the transmit data descriptor POPTS.TXSM and POPTS.IXSM fields, respectively.

6.2.5.1 IP Checksum

Three fields in the transmit context descriptor set the context of the IP checksum offloading feature:

- TUCMD.IPV4
- IPLEN
- MACLEN

TUCMD.IPV4=1 specifies that the packet type for this context is IPv4, and that the IP header checksum should be inserted. TUCMD.IPV4=0 indicates that the packet type is IPv6 (or some other protocol) and that the IP header checksum should not be inserted.

MACLEN specifies the byte offset from the start of the DMA'ed data to the first byte to be included in the checksum, the start of the IP header. The minimal allowed value for this field is 14. Note that the maximum value for this field is 127. This is adequate for typical applications.

Note: The MACLEN+IPLEN value must be less than the total DMA length for a packet. If this is not the case, the results are unpredictable.

IPLEN specifies the IP header length. Maximum allowed value for this field is 511 bytes.

MACLEN+IPLEN specify where the IP checksum should stop. The sum of MACLEN+IPLEN must be smaller equals to the first 638 (127+511) bytes of the packet and obviously must be smaller or equal to the total length of a given packet. If this is not the case, the result is unpredictable.

For IP tunnel packets (IPv4-IPv6), IPLEN must be defined as the length of the two IP headers. Hardware is able to offload the L4 checksum calculation while software should provide the IPv4 checksum.

For NVGRE and VXLAN tunneled packet as defined by the *Tunnel* bit in the descriptor, Outer IPv4 checksum is controlled by OUTERIPCS bit and is determined by MACLEN and OUTERIPLEN similar to the above description. Inner IPv4 checksum is controlled by TUCMD.IPv4 and offset determination should be adjusted by HW such that MACLEN should be replaced by (MACLEN+OUTIPLEN+TUNNELLEN)

The 16-bit IPv4 header checksum is placed at the two bytes starting at MACLEN+10.

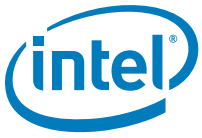
6.2.5.2 TCP and UDP Checksum

Three to five fields in the transmit context descriptor set the context of the TCP / UDP checksum offloading feature:

- MACLEN
- OUTERIPLEN (optional)
- TUNNELLEN (optional)
- IPLEN
- TUCMD.L4T

TUCMD.L4T=01b specifies that the packet type is TCP, and that the 16-bit TCP header checksum should be inserted at byte offset MACLEN+IPLEN+16. TUCMD.L4T=00b indicates that the packet is UDP and that the 16-bit checksum should be inserted starting at byte offset MACLEN+IPLEN+6. If *Tunnel* bit is set, the checksum is inserted at MACLEN+OUTERIPLEN+TUNNELLEN+IPLEN+6

MACLEN+(OUTERIPLEN+TUNNELLEN)+IPLEN specifies the byte offset from the start of the DMA'ed data to the first byte to be included in the checksum, the start of the UDP/TCP header. See MACLEN table in Section 6.2.3.2.3 for its relevant values.



Note: The MACLEN+(OUTERIPLEN+TUNNELLEN)+IPLEN+L4LEN value must be less than the total DMA length for a packet. If this is not the case, the results are unpredictable.

The TCP/UDP checksum always continues to the last byte of the DMA data.

Note: For non-TSO, software still needs to calculate a full checksum for the TCP/UDP pseudo-header. This checksum of the pseudo-header should be placed in the packet data buffer at the appropriate offset for the checksum calculation.

6.2.5.3 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 6.2.3.2.3 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.

6.2.5.4 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 6.42. 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
IPv4 tunnels: • Ipv4 packet in an IPv4 tunnel • Ipv6 packet in an IPv4 tunnel	 No No	 No Yes
IPv6 tunnels: • IPv4 packet in an IPv6 tunnel • IPv6 packet in an IPv6 tunnel	 No No	 No No
Packet is an IPv4 fragment	Yes	No
Packet has 802.3ac tag	Yes	Yes
IPv4 packet has IPSec header without IP options	Yes	Yes
Packet has TCP or UDP options	Yes	Yes
IP header's protocol field contains protocol # other than TCP, UDP, or SCTP	Yes	No
NVGRE: • Inner Ipv4 packet • Inner Ipv6 packet	 Yes (Inner and outer) Yes (outer)	 Yes Yes
VXLAN: • Inner Ipv4 packet • Inner Ipv6 packet	 Yes (Inner and outer) Yes (outer)	 Yes (inner only)[1] Yes (inner only)

1. The outer UDP header of VXLAN packets do not use a checksum.

6.2.6 Transmit Statistics

6.2.6.1 General notes

- All Statistics registers are cleared on read. In addition, they stick at 0xFF...F when the maximum value is reached.
- Due to divergent paths between interrupt-generation and logging of relevant statistics counts, it might be possible to generate an interrupt to the system for a noteworthy event prior to the associated statistics count actually being increased. This is extremely unlikely due to expected delays associated with the system interrupt-collection and ISR delay, but might be an explanation for interrupt statistics values that do not quite make sense. Hardware guarantees that any event noteworthy of inclusion in a statistics count is reflected in the appropriate count within 1 µs; a small time-delay prior to reading the statistics might be required to avoid a potential mismatch between and interrupt and its cause.
- If TSO is enabled, statistics are collected after segmentation.
- All byte (octet) counters composed of 2 registers can be fetched by two consecutive 32-bit accesses while reading the Low 32-bit register first or a single 64-bit access.

6.2.6.2 Transmit Statistics Hierarchy

The following diagram describes the relations between the packet flow and the different statistic counters.

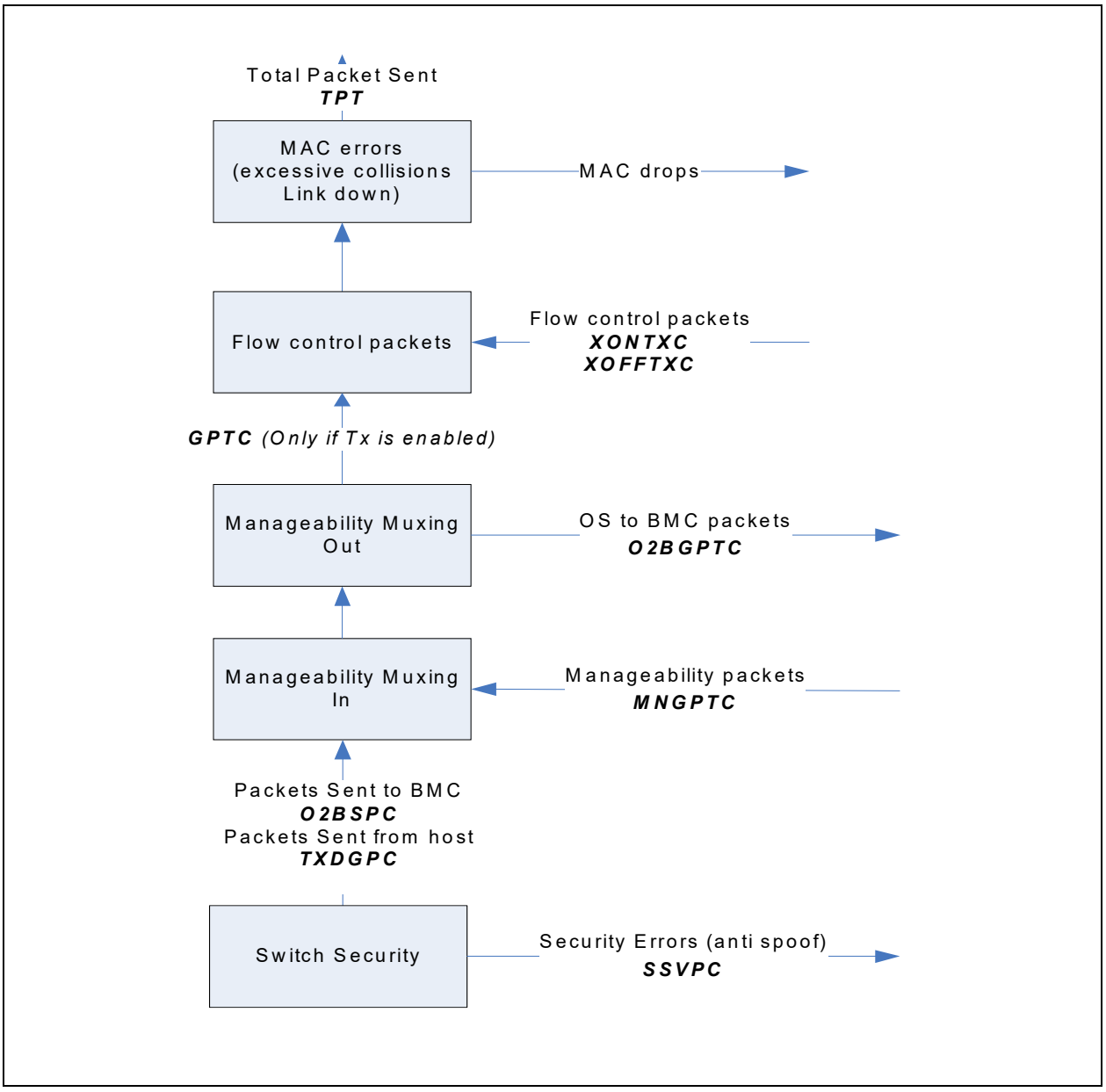
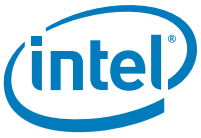


Figure 6.21. Transmit Flow Statistics



6.3 Interrupts

The integrated 10 GbE LAN controller 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.

6.3.1 Interrupt Registers

Physical Function (PF) Registers

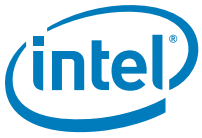
The PF interrupt logic consists of the registers listed in the Table 6.43 followed by their description:

Table 6.43. PF Interrupt Registers

Acronym	Complete Name
EICR	Extended Interrupt Cause register
EICS	Extended Interrupt Cause Set register (enables software to initiate interrupts)
EIMS	Extended Interrupt Mask Set/Read register
EIMC	Extended Interrupt Mask Clear register
EIAC	Extended Interrupt Auto Clear register (following interrupt assertion)
EIAM	Extended Interrupt Auto Mask register (auto set/clear of the EIMS)
EITR	Extended Interrupt Throttling register [throttling]
IVAR	Interrupt Vector Allocation Registers (described in Section 6.3.4)
IVAR_MISC	Miscellaneous Interrupt Vector Allocation Register (described in Section 6.3.4)

These registers are extended to 64 bits by an additional set of two registers. EICR has an additional two registers EICR(1)... EICR(2) and so on for the EICS, EIMS, EIMC, EIAM and EITR registers. The EIAC register is not extended to 64 bits as this extended interrupt causes are always auto cleared. Any reference to EICR... EIAM registers as well as any global interrupt settings in the GPIE register relates to their extended size of 64 bits.

The legacy EICR[15:0] mirror the content of EICR(1)[15:0]. In the same manner the lower 16 bits of EICS, EIMS, EIMC, EIAM mirror the lower 16 bits of EICS(1), EIMS(1), EIMC(1), EIAM(1). For more details on the use of these registers in the various interrupt modes (legacy, MSI, MSI-X) see Section 6.3.4.



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 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 6.3.4.3.2 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.

6.3.1.1 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 as configured in the EITR register
- Mapping the Tx and Rx queues to EICR is done by the IVAR registers as described in Section 6.3.4. 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 6.3.1.4).

6.3.1.2 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 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).

6.3.1.2.1 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 6.3.2.1.1.
- In LLI mode (GPIE.EIMEN = 1b), setting the *EICS* bits impact the RSC completion and interrupt assertion as follow:
  - Interrupt is asserted.
  - Concurrently, hardware triggers RSC completion in all Rx queues associated with the same interrupt.
  - Most likely these RSC(s) are completed to host memory after the interrupt is already asserted. In his case, it is guaranteed that an additional interrupt is asserted when the ITR expires.

6.3.1.3 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.

6.3.1.4 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.

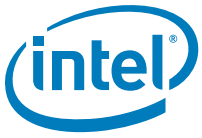
Note: Bits 29:16 should never be set to auto clear since they share the same MSI-X vector.

Writing to the EIAC register changes the setting of the entire register. In IOV mode, some of the bits in this register might affect VF functionality (VF-56...VF-63). It is recommended that software set the register in PF before VF's are enabled. Otherwise, a software semaphore might be required between the VF and the PF to avoid setting corruption.

6.3.1.5 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 29:16 should never be set to auto clear since they share the same MSI-X vector.

Writing to the EIAM register changes the setting of the entire register. In IOV mode, some of the bits in this register might affect VF functionality. It is recommended that software set the register in PF before VF's are enabled. Otherwise, a software semaphore might be required between the VF and the PF to avoid setting corruption.

If any of the *Auto Mask* enable bits is set in the EIAM registers, the GPIE.EIAME bit must be set as well.

6.3.2 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 integrated 10 GbE LAN controller 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 6.3.1.1.

6.3.2.1 Time-based Interrupt Throttling — ITR

Time-based interrupt throttling is useful to limit the maximum interrupt rate regardless of network traffic conditions. The ITR logic is targeted for Rx/Tx interrupts only. It is assumed that the software device driver will not moderate the timer, other and mail box (IOV mode) interrupts. In non-IOV mode, all 64 interrupts can be associated with ITR logic. In IOV mode, the ITR logic is shared between the PF and VFs as shown in Figure 6.22. The ITR mechanism is based on the following parameters:

- **ITR Interval** field in the EITR registers — The minimum inter-interrupt interval is specified in 2.048 µs units (at all speeds). When the ITR Interval equals zero, interrupt throttling is disabled and any event causes an immediate interrupt. The field is composed of nine bits enabling a range of 2.048 µs up to 1046.528 µs. These ITR interval times correspond to interrupt rates in the range of 488 K INT/sec to 955 INT/sec.
  - Due to internal synchronization issues, the ITR interval can be shortened by up to 1 µs at 10 Gb/s or 1 Gb/s link and up to 10 µs at 100 Mb/s link when it is triggered by packet write back or interrupt enablement.
- **ITR Counter** partially exposed in the EITR registers — Down counter that is loaded by the ITR interval each time the associated interrupt is asserted.
  - The counter is decremented by one each 1.024 µs (at 1 Gb/s or 10 Gb/s link) and stops decrementing at zero. At 100 Mb/s link, the speed of the counter is decremented by one each 10.24 µs.
  - If an event happens before the counter is zero, it sets the EICR. The interrupt can be asserted only when the ITR time expires (counter is zero).
  - Else (no events during the entire ITR interval), the EICR register is not set and the interrupt is not asserted on ITR expiration. The next event sets the EICR bit and generates an immediate interrupt. See Section 6.3.2.1.1 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.

6.3.2.1.1 ITR Affect on RSC Functionality

Interrupt assertion is one of the causes for RSC completion (see Section 6.8.6). 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.

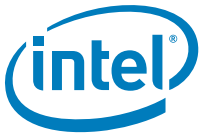
6.3.2.2 Immediate Interrupt

The integrated 10 GbE LAN controller might initiate an immediate interrupt when the receive descriptor ring is almost empty (Rx descriptors below a specific threshold). The threshold is defined by *SRRCTL[n].RDMTS* per Rx queue. This mechanism can protect against memory resources being used up during reception of a long burst of short packets.

6.3.3 TCP Timer Interrupt

6.3.3.1 Introduction

In order to implement TCP timers, software needs to take action periodically (every 10 ms). Today, the driver must rely on software-based timers, whose granularity can change from platform to platform. This software timer generates a software NIC interrupt, which then enables the driver to perform timer functions, avoiding cache thrash and enabling parallelism. The timer interval is system-specific.



It would be more accurate and more efficient for this periodic timer to be implemented in hardware. The driver would program a timeout value (usual value of 10 ms), and each time the timer expires, hardware sets a specific bit in the EICR register. When an interrupt occurs (due to normal interrupt moderation schemes), software reads the EICR register and discovers that it needs to process timer events.

The timeout should be programmable by the driver, and the driver should be able to disable the timer interrupt if it is not needed.

6.3.3.2 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.

6.3.4 Mapping of Interrupt Causes

The following sections describe legacy, MSI and MSI-X interrupt modes.

6.3.4.1 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 6.22. Each entry in the IVAR registers is composed of two fields that identify the associated bit in the EICR[15:0] register. Software might map multiple Tx and Rx queues to the same EICR bit.

INT_Alloc — Defines one of the bits (0...15) in the EICR register that reflects the interrupt status indication.

INT_Alloc_val — Valid bit for the this interrupt cause.

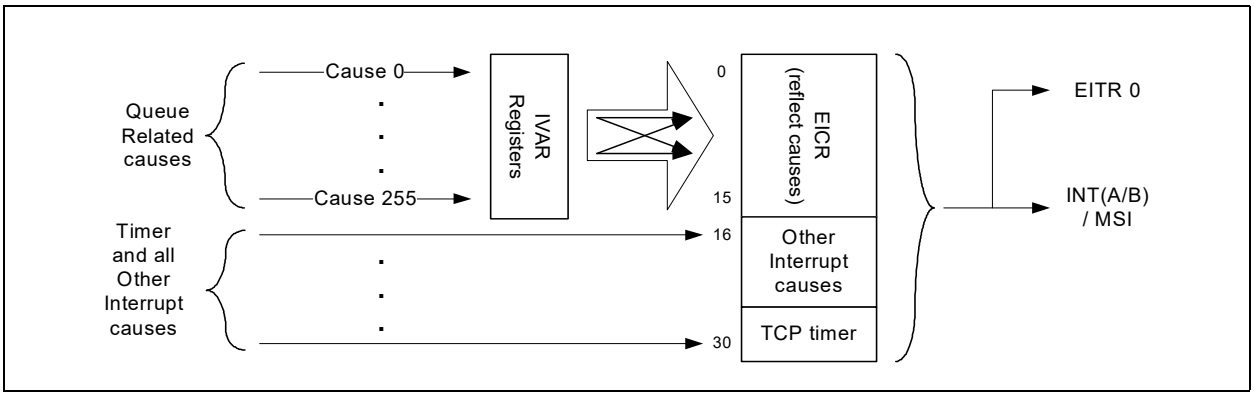


Figure 6.22. Cause Mapping in Legacy and MSI Modes

Mapping between the Tx and Rx queue to the IVAR registers is hard-wired as shown in the Figure 6.23 below:

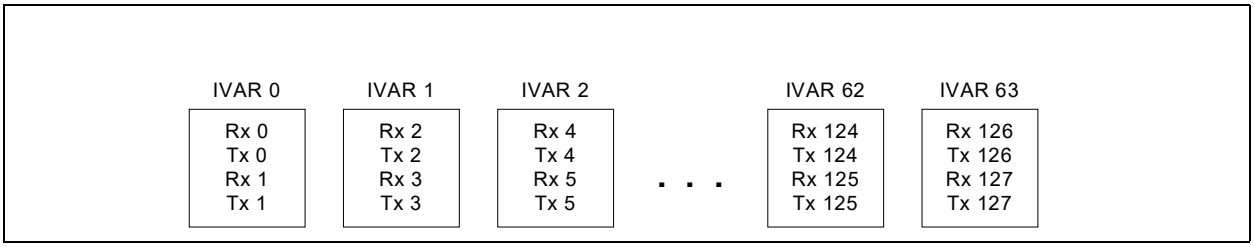


Figure 6.23. Rx and Tx Queue Mapping to IVAR Registers

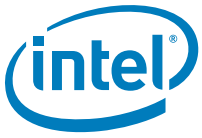
6.3.4.2 MSI-X Mode in Non-IOV Mode

- MSI-X defines a separate optional extension to basic MSI functionality. Hardware indicates the number of requested MSI-X vectors in the table size in the MSI-X capability structure in the configuration space. The number of requested MSI-X vectors is loaded from shared SPI Flash in the *PCI_CNF2.MSI_X_PF_N* field up to maximum of 64 MSI-X vectors. The operating system might allocate any number of MSI-X vectors to the device from a minimum of one up to the requested number of MSI-X vectors.
- Enables interrupts causes allocation to the assigned MSI-X vectors. Interrupt allocation is programmed by the IVAR registers and are described in this section.
- Each vector can use an independent address and data value as programmed directly by the operating system in the MSI-X vector table.
- Each MSI-X vector is associated to an EITR register with the same index (MSI-X 0 to EITR[0], MSI-X 1 to EITR[1],...).

For more information on MSI-X, refer to the PCI Local Bus Specification, Revision 3.0.

MSI-X vectors can be used for several purposes:

1. Dedicated MSI-X vectors per interrupt cause (avoids the need to read the interrupt cause register).
2. Load balancing by MSI-X vectors assignment to different CPUs.
3. Optimized interrupt moderation schemes per MSI-X vector using the EITR registers.



The MSI-X vectors are used for Tx and Rx interrupt causes as well as the other and timer interrupt causes. The remainder of this section describes the mapping of interrupt causes (such as a specific Rx or Tx queue event or any other event) to the interrupts registers and the MSI-X vectors.

The TCP timer and other events are reflected in EICR[30:16] the same as the legacy and MSI mode. It is then mapped to the MSI-X vectors by the IVAR_MISC register as shown in Figure 6.24. The IVAR_MISC register includes two entries for the timer interrupt and an additional entry for all the other causes. The structure of each entry is as follows:

INT_Alloc — Defines the MSI-X vector (0...63) assigned to this interrupt cause.

INT_Alloc_val — Valid bit for the this interrupt cause.

The Tx and Rx queues are associated to the IVAR0...IVAR63 the same as legacy and MSI mode shown in Figure 6.23. The Tx and Rx queues are mapped by the IVAR registers to EICR(1),...EICR(2) registers and MSI-X vectors 0...63 illustrated in Figure 6.24. The IVAR entries have the same structure as the IVAR_MISC register previously shown. Each bit in EICR(1...2) registers is associated to MSI-X vector 0...63 as follows:

- EICR(i).bit_num is associated to MSI-X vector (i x 32 + bit_num).
- The legacy EICR[15:0] mirror the content of EICR(1)[15:0]. In the same manner the lower 16 bits of EICS, EIMS, EIMC, EIAM mirror the lower 16 bits of EICS(1), EIMS(1), EIMC(1), EIAM(1). The use of these registers depends on the number of assigned MSI-X interrupts as follows:
- **16 Tx and Rx Interrupts** — When using up to 16 Tx and Rx interrupts, software might access the Tx and Rx interrupt bits in the legacy EICR, EICS,... registers.
- **More than 16 Tx and Rx Interrupts** — When using more 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 EICS, EIMS... registers when it accesses the higher bits of these registers as follows:
  - EICR, EICS, EIMS and EIMC — When software programs the higher 16 bits of these registers, it should set their lower 16 bits to zero's keeping the EICR(1), EICS(1), EIMS(1) and EIMC(1) unaffected.
  - EIAM — When software programs the higher 16 bits, it should keep the lower 16 bits at their previous setting so the EIAM(1) is unaffected.
  - EIAC — When software programs the higher 16 bits, it should set the lower 16 bits to one's.

Single MSI-X vector — If the operating system allocates only a single MSI-X vector, the driver might use the non-MSI-X mapping method (setting the GPIE.Multiple_MSIX to 0b). In this case, the *INT_Alloc* field in the IVAR registers might define one of the lower 16 bits in the EICR register while using MSI-X vector 0. The IVAR_MISC should be programmed to MSI-X vector 0.

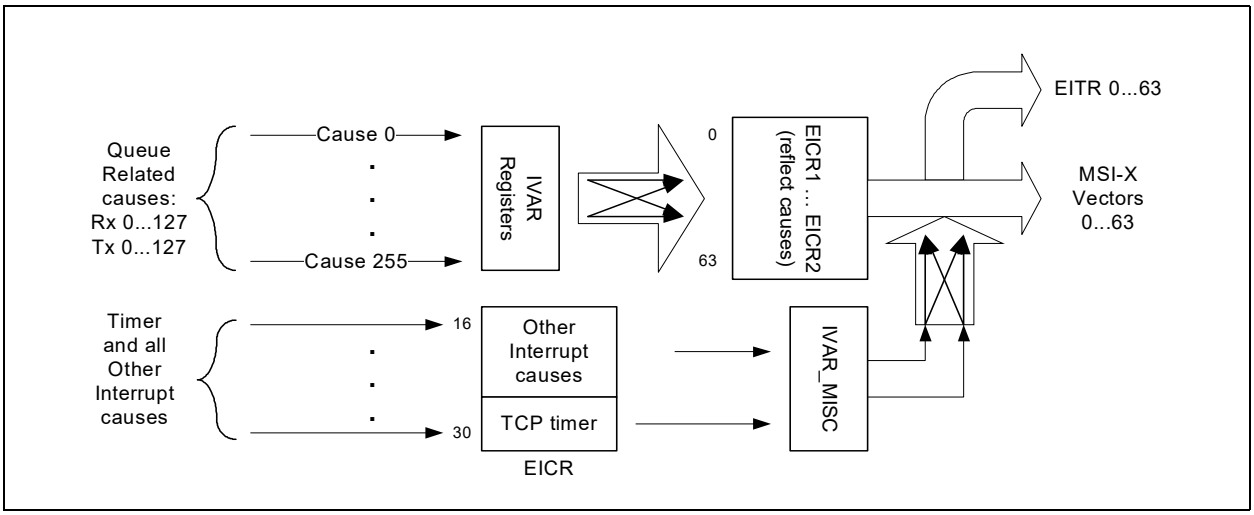


Figure 6.24. Cause Mapping in MSI-X Mode (non-IOV)

6.3.4.3 MSI-X Interrupts In IOV Mode

In IOV mode, interrupts must be implemented by MSI-X vectors. The integrated 10 GbE LAN controller 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 shared SPI Flash in the *PCI_CNF2.MSI_X_VF_N* field. It is reflected in the *Table Size* field in the PCIe MSI-X capability structure of the VF's. In addition, the PF requires its own interrupts. The number of requested MSI-X vectors is loaded from shared SPI Flash in the *PCI_CNF2.MSI_X_PF_N* field up to maximum of 64 MSI-X vectors. It is reflected in the *Table Size* field in the PCIe MSI-X capability structure.

6.3.4.3.1 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 6.22). When there are less than the maximum possible active VF's, some of the Tx and Rx queues can be associated with the PF. These queues can be used for the sake of additional VM's serviced by the hypervisor (the same as VMDq mode) or some Kernel applications handled by the hypervisor. Tx and Rx mapping to the IVAR registers is shown in Figure 6.23 and mapping to the EICR, EICR(1),...EICR(2) registers as well as the MSI-X vectors is shown in Figure 6.24. See Section 6.3.4.3.3 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.

6.3.4.3.2 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 6.25 with the following comments:

- Each VF cannot have more than three MSI-X vectors. It has only three active bits in the VFEICR register while VFEICR.bit_num is associated with MSI-X vector (bit_num).
- The Tx and Rx interrupt can be mapped only to MSI-X 0 and MSI-X 1 (associated with VFEICR.0 and VFEICR.1).
- The mailbox interrupt can be mapped to any of the three MSI-X vectors. However, when all three of them are allocated by the operating system, software should map the mailbox to MSI-X 2 (associated with VFEICR.2). This rule should be kept since only VFEICR.0 and VFEICR.1 have ITR registers (VFEITR-0 and VFEITR-1).
- Association between the Tx and Rx queues and the VFIVAR registers is shown in the Figure 6.25, Figure 6.26 and Figure 6.27 for IOV-64 (64 VF's), IOV-32 and IOV-16. The colored boxes in the figures show the mapping between VF Rx and Tx queues to VFIVAR registers while the dashed boxes show the physical IVAR registers and the associated physical Rx and Tx queues.

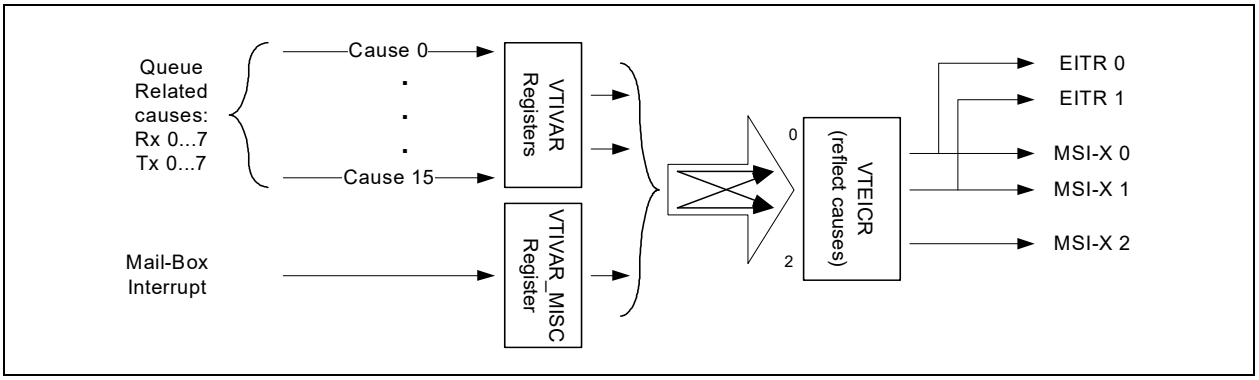


Figure 6.25. VF Interrupt Cause Mapping (MSI-X, IOV)

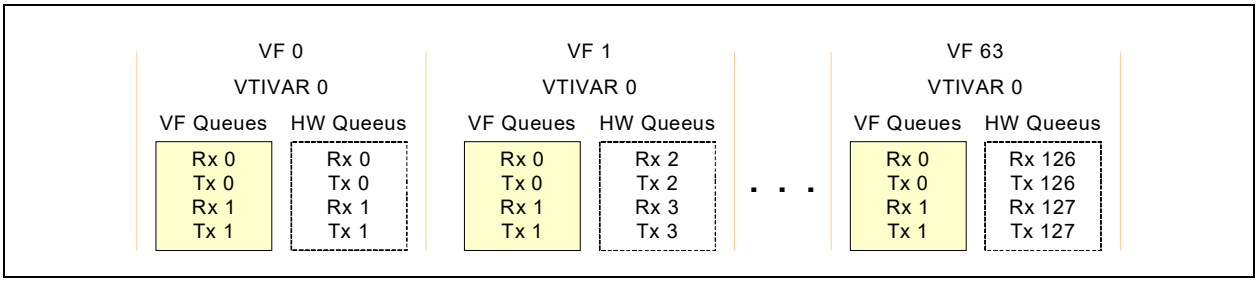


Figure 6.26. VF Mapping of Rx and Tx Queue to VFIVAR in 64 VF's Mode

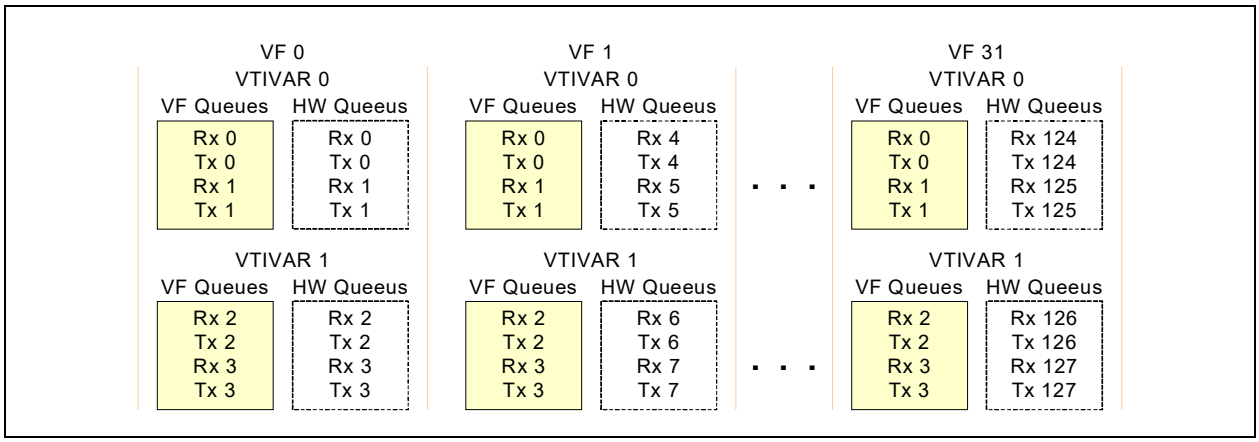


Figure 6.27. VF Mapping of Rx and Tx Queue to VFIVAR in 32 VF's Mode

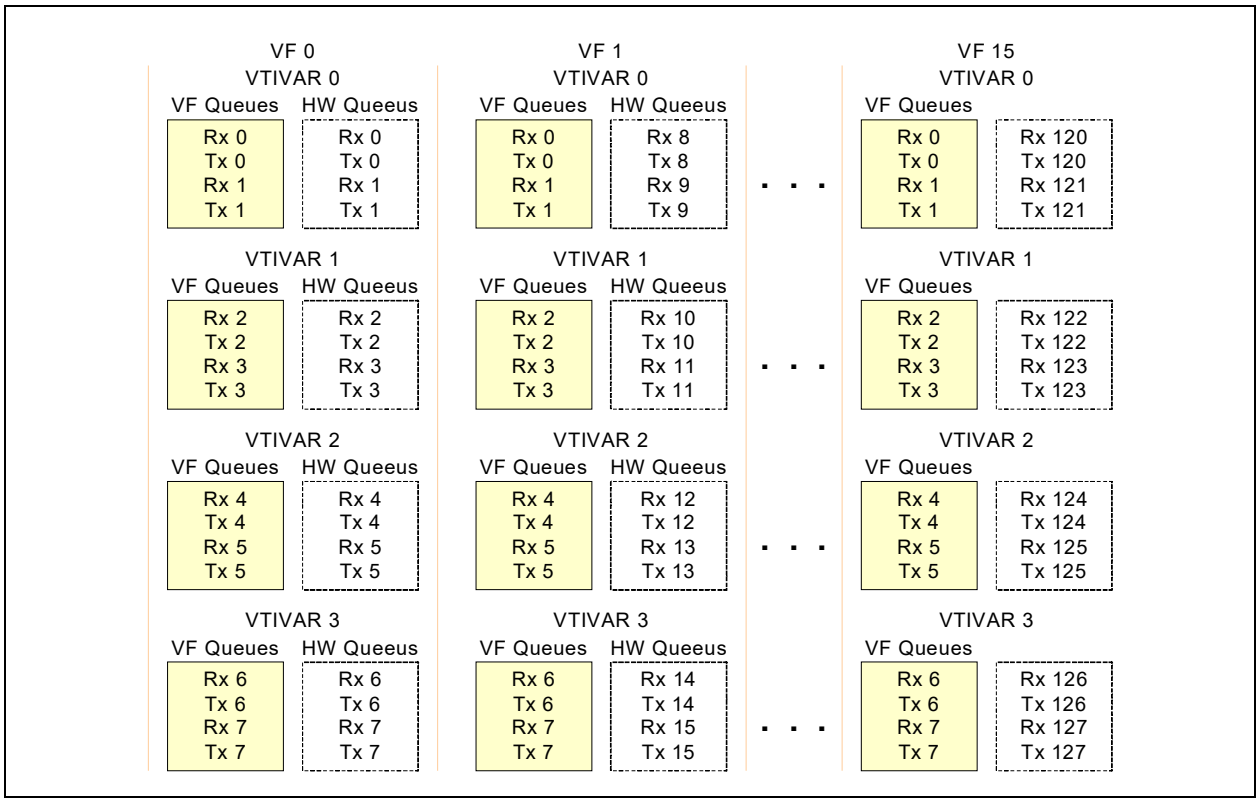


Figure 6.28. VF Mapping of Rx and Tx Queue to VFIVAR in 16 VF's Mode

6.3.4.3.3 MSI-X Vectors Mapping to EITR

EITR registers are aimed for Tx and Rx interrupt throttling. In IOV mode, the Tx and Rx queues might belong to either the PF or to the VF's. EITR(1...63) are multiplexed between the PF and the VF's as configured by the *EITRSEL* register. Figure 6.29 and Table 6.44 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.

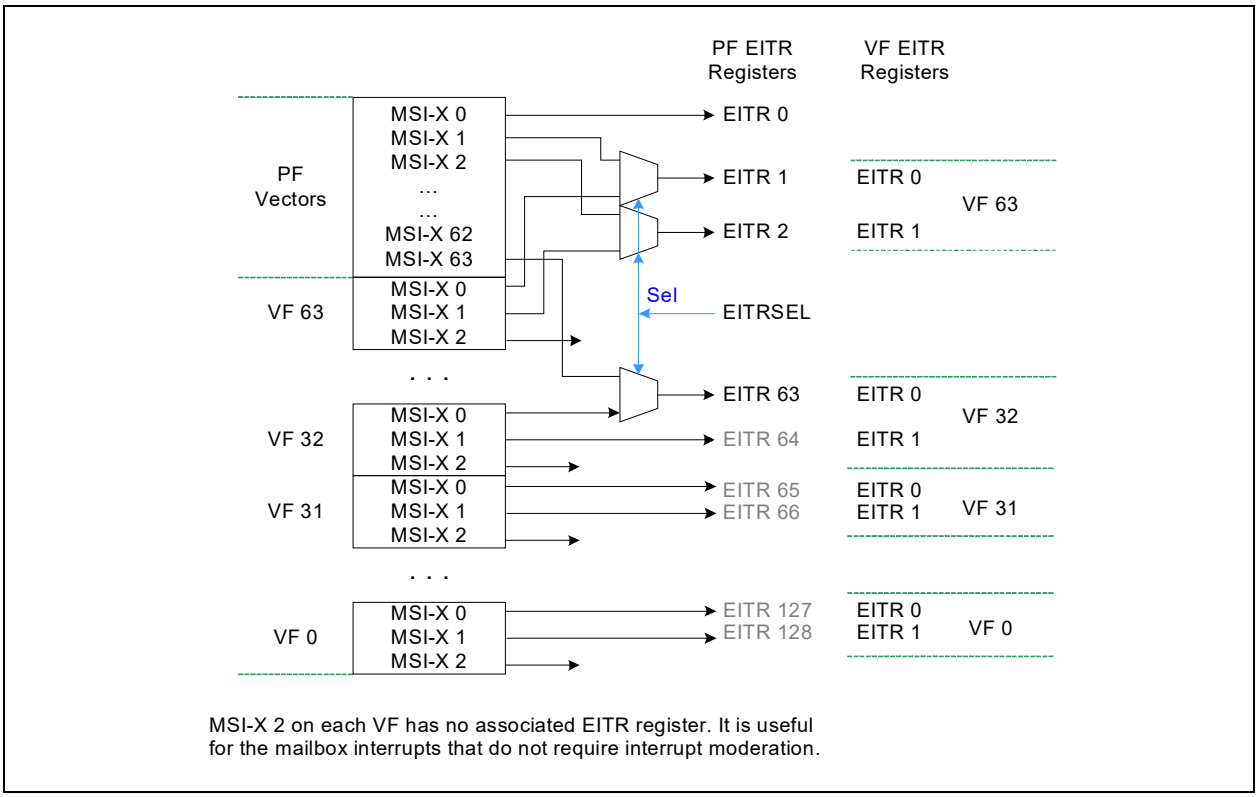


Figure 6.29. PF / VF MSI-X Vectors Mapping to *EITR*

Table 6.44. 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)

Table 6.44. PF / VF MSI-X Vectors Mapping Table to EITR Registers

	. . .	
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)

6.4 802.1q VLAN Support

The integrated 10 GbE LAN controller 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.

6.4.1 802.1q 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.

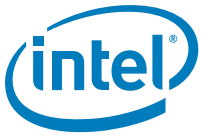
6.4.2 802.1q Tagged Frames

For 802.1q, the *Tag Header* field consists of four octets comprised of the Tag Protocol Identifier (TPID) and Tag Control Information (TCI); each taking two octets. The first 16 bits of the tag header makes up the TPID. It contains the protocol type that identifies the packet as a valid 802.1q tagged packet.

The two octets making up the TCI contain three fields as follows:

- User Priority (UP)
- Drop Eligible Indicator (DEI). Should be set to 0b for transmits. For receives, the device has the capability to filter out packets that have this bit set. See the *DEIEN* and *DEI* bits in the *VLNCTRL*
- VLAN Identifier (VID)

Octet 1		Octet 2
UP	DEI	VID



6.4.3 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 integrated 10 GbE LAN controller has additional capabilities to add and strip tags in hardware. See Section 6.4.3.1 and Section 6.4.3.2.

6.4.3.1 Adding 802.1q Tags on Transmits

The inner VLAN header can be added by software in one of the following methods:

- The header is included in the transmit data buffers.
- Software might instruct the integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller inserts a VLAN tag into the packet that it transmits over the wire. The Tag Protocol Identifier — TPID (VLAN Ether Type) field of the 802.1q tag comes from the DMATXCTL.VT, and the Tag Control Information (TCI) of the 802.1q tag comes from the VLAN field of the legacy transmit descriptor or the *VLAN Tag* field of the advanced data transmit descriptor.
- In IOV mode, the priority tag, DEI and VLAN ID can be taken from the *PFVMVIR* (see details in Section 6.6.4.2)

6.4.3.2 Stripping 802.1q Tags on Receives

Software might instruct the integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller strips the 4-byte VLAN tag from the packet, and stores the TCI in the *VLAN Tag* field of the receive descriptor.

The integrated 10 GbE LAN controller also sets the *VP* bit in the receive descriptor to indicate that the packet had a VLAN tag that was stripped. If the RXDCTL.VME bit is not set, the 802.1q packets can still be received if they pass the receive filter, but the VLAN tag is not stripped and the VP bit is not set.If *PFQDE.HIDE_VLAN* is set, the VLAN tag is stripped as above, but the *VLAN Tag* field and the Status.VP bit in the Rx descriptor are cleared

6.4.4 802.1q 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, *DEIEN* and *DEI*, are also used in conjunction with 802.1q VLAN filtering operations. *DEIEN* enables the comparison of the value of the *DEI* bit in the 802.1q packet to the Receive Control register *DEI* bit as acceptance criteria for the packet.

Note: The *VFE* bit does not effect whether the VLAN tag is stripped. It only effects whether the VLAN packet passes the receive filter.

6.4.5 Double VLAN and Single VLAN Support

The integrated 10 GbE LAN controller supports a mode where all received and sent packets have at least one VLAN tag in addition to the regular tagging that might optionally be added. In this document, when a packet carries two VLAN headers, the first header is referred to as an outer VLAN and the second header as an inner VLAN header (as listed in the table that follows). This mode is used for systems where the near end switch adds the outer VLAN header containing switching information. This mode is enabled by the following configuration:

- This mode is activated by setting the DMATXCTL.GDV and the *Extended VLAN* bit in the CTRL_EXT register.
- The Ethertype of the VLAN tag used for the additional VLAN is defined in the *VET EXT* field in the *EXVET* and *EXVET_T* registers.

6.4.5.1 Cross Functionality with Manageability

The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller is set to double VLAN mode as well as the VLAN Ethertype(s).

Table 6.45. Double VLAN packet format

MAC Address	Outer VLAN	Inner VLAN	L2 Payload	Ethernet CRC

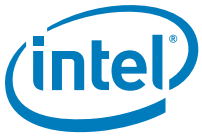
6.4.5.2 Transmit Functionality

6.4.5.2.1 Transmit Functionality on the Outer VLAN Header

- A packet with a single VLAN header is assumed to have only the outer VLAN.
- The outer VLAN header must be added by software as part of the Tx data buffers.
- Hardware does not relate to the outer VLAN header other than the capability of skipping it for parsing inner fields.
- Hardware expects that any transmitted packet (see the disclaimer that follows) has at least the outer VLAN added by software. For any offload that hardware might provide in the transmit data path, hardware assumes that the outer VLAN is present. For those packets that an outer VLAN is not present, any offload that relates to inner fields to the Ethertype might not be provided.

6.4.5.2.2 Transmit Functionality on the Inner VLAN Header

- Inner VLAN insertion is handled as described in Section 6.4.3.1.
- Hardware identifies and skips the VLAN header for parsing inner fields.
- 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 6.6.3.4 and Section 6.6.4.2.



6.4.5.3 Receive Handling of Packets with VLAN Header(s)

A received frame is analyzed for the existence of the outer and inner VLAN headers. The procedure is as follows:

```
Check the Ethertype against the outer VLAN ID. If match then
{
          Check the next Ethertype against the inner VLAN ID. If match
          {
                    This is the double VLAN case. Process both outer and inner VLANs as described
                    below
          }
          Else
          {
                  Only an outer VLAN exists. Process the outer as described below. Assume no inner
                  VLAN
          }
}
Else
{
          Check the Ethertype against the inner VLAN ID. If match
          {
                   This is the case of an inner VLAN only. Handle the frame as an unknown frame –
                   device does
                   not provide any offloads
          }
          Else
          {
                   This is the case of no VLAN. Process the frame (ignore outer and inner VLAN
                   processing)
          }
}
```

6.4.5.3.1 Receive Functionality on the Outer VLAN Header

- Hardware checks the Ethertype of the outer VLAN header against the programmed value in the EXVET register. VLAN header presence is indicated in the Status.VEXT bit in the Rx descriptor.
- The outer VLAN header is posted as is to the receive data buffers.

6.4.5.3.2 Receive Functionality on the Inner VLAN Header

- Hardware checks the Ethertype of the inner VLAN header against the programmed value in the *VLNCTRL.VET. VLAN* header presence is indicated in the Status.VP bit in the Rx descriptor.
- L2 packet filtering is based on the VLAN ID in the inner VLAN header.
- Pool Filtering — Destination pool(s) are defined by the Ethernet MAC address and inner VLAN (if presence) as described in Section 6.6.3.3.
- Inner VLAN tag striping is handled as described in Section 6.4.3.2.

6.4.5.4 Packets With no VLAN Headers in Double VLAN Mode

There are some cases when packets might not carry any VLAN headers, even when extended VLAN is enabled. A few examples for packets that might not carry any VLAN header are: flow control, 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.

6.4.5.4.1 Transmit Functionality

Transmit offload functionality — Software should not enable any offload functions.

6.4.5.4.2 Receive Functionality

Receive offload functionality — pool and queue are selected by the Ethernet MAC address or *ETQF/ETQS* registers. Filtering to host and manageability remains functional.

6.4.5.5 Packets With two VLAN Headers not in Double VLAN Mode

When the *Extended VLAN* bit in the *CTRL_EXT* register and *DMATXCTL.GDV* are not set, hardware expects that Rx and Tx packets might not carry a VLAN header or a single VLAN header. Hardware does not relate to the programming of the *VET EXT* field in the *EXVET* register. Tx and Rx handling of packets with double VLAN headers is unexpected.

6.4.6 E-tag and VLAN

In some systems an additional external tag (E-tag) can be present before the VLAN. This section describes the support for VLANs in presence of external tags.This mode is used for systems where the device adds a tag to identify a subsystem (usually a VM) and the near end switch adds a tag indicating the destination subsystem. External tags may be present on part of the packets and missing in others.

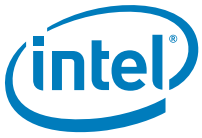
6.4.6.1 Transmit Functionality

6.4.6.1.1 Transmit Functionality on the External Tag

The integrated 10 GbE LAN controller supports insertion of an external tag from a per pool register (*PFVMTIR*) and a VLAN tag from a per pool register (*PFVMVIR*) or from the descriptor as indicated by the *VLE* bit. The following options are supported:

VLAN source/External Tag Source	Hardware Register - PFVMTIR/PFVMVIR	Embedded in Packet
Hardware register - PFVMVIR	Supported	Not supported
Descriptor	Supported	Not supported
Embedded in Packet	Supported	Supported

After the tags are inserted in the packet, the integrated 10 GbE LAN controller uses the VLAN tag and the Ethertype as part of the forwarding decision as described in Section 6.6.3.4. In a packet with at most on external tag and one VLAN, the VLAN and Ethertype will be identified correctly.



6.4.6.2 Receive Handling of Packets With External Tags

The parsing of outer tags is described in Section 6.1.2.1.2. External tags can be extracted from the packet according to the P*FQDE.STRIP_TAG* field. Inner tag extraction is handled as described above (Section 6.4.3.2).

6.4.6.2.1 Packet Priority In Presence Of External Tags

The user priority used to define the traffic class of a packet is always taken from the inner VLAN.

6.4.6.3 Cross Functionality with Manageability

The integrated 10 GbE LAN controller does not provide any stripping or adding VLAN header(s) to manageability packets. Therefore, packets that are directed to/from the manageability controller should include the L2 headers as part of the Rx/Tx data. The manageability controller should know if it is expected to add/receive an external tag as part of the packet.

6.4.6.4 Packet User Priority (802.1P) bits handling

The user priority bits may be used by the device for multiple purposes:

1. Defining the traffic class to which the traffic is associated (Section 6.5.3.1).
2. Defining which packets will get a Timestamp in Buffer (Section 6.1.6.2).
3. Defining if a packet matches and Ethertype filter (Section 6.1.3.3).

For all these purposes, the UP bits are taken from the outermost tag with UP bits (E-tag or VLAN), unless the *VLNCTRL.UP_FIRST_TAG_EN* bit is cleared, in which case, it is always taken from the inner VLAN.

The table below summarize the tag from which the user priority is extracted from the packets

Table 6.46. UP Extraction Rules

Packet type	VLANCTRL.UP_FIRST_TAG_EN = 1	VLANCTRL.UP_FIRST_TAG_EN = 0
Un-tagged Packet	User priority = 0	User priority = 0
Packet with a single VLAN	The user priority field from the VLAN header.	**Single VLAN mode:** The user priority field from the VLAN header. **Double VLAN mode:** User priority = 0
Packet with a VLAN and an outer tag (outer VLAN or E-tag)	The user priority field from the **outer** VLAN header	The user priority field from the **inner** VLAN header.
Packet with E-tag only	The user priority field from the E-tag	User priority = 0

6.5 Time SYNC (IEEE1588 and 802.1AS)

6.5.1 Overview

IEEE 1588 addresses the clock synchronization requirements of measurement and control systems. The protocol supports system-wide synchronization accuracy in the sub-microsecond range with minimal network and local clock computing resources. The protocol is spatially localized and allows simple systems to be installed and operate.

The IEEE802.1AS standard specifies the protocol used to ensure that synchronization requirements are met for time sensitive applications, such as audio and video, across Bridged and Virtual Bridged Local Area Networks consisting of LAN media where the transmission delays are almost fixed and symmetrical; for example, IEEE 802.3 full duplex links. This includes the maintenance of synchronized time during normal operation and following addition, removal, or failure of network components and network re-configuration. It specifies the use of IEEE 1588 specifications where applicable.

Note: The time sync mechanism activation is possible in full-duplex mode only and is not supported at 100 Mb/s link speed.

6.5.2 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 6.30.

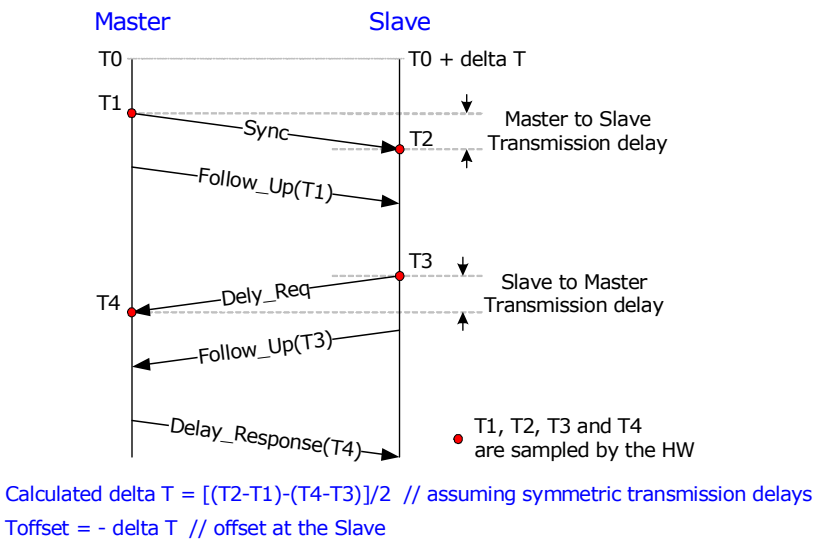


Figure 6.30. Sync Flow and Offset Calculation

Hardware responsibilities are:

1. Identify the packets that require time stamping.
2. Time stamp the packets on both Rx and Tx paths.
3. Store the time stamp value for software.
4. Keep the system time in hardware and give a time adjustment service to software.
5. Maintain auxiliary features related to the system time.

Software responsibilities are:

1. Manageability controller protocol execution, which means defining the node state (master or slave) and selection of the master clock if in slave state.
2. Generate PTP packets, consume PTP packets.
3. Calculate the time offset and adjust the system time using a hardware mechanism for that.
4. Enable configuration and usage of the auxiliary features.

Action	Responsibility	Node Role
Generate a sync packet with time stamp notification in the descriptor.	Software	Master
Time stamp the packet and store the value in registers (T1).	Hardware	Master
Time stamp incoming sync packet, store the value in register	Hardware	Slave
Read the time stamp from register put in a Follow_Up packet and send.	Software	Master
Once received, the Follow_Up store T2 from registers and T1 from Follow_up packet.	Software	Slave
Generate a Delay_Req packet with time stamp notification in the descriptor.	Software	Slave
Time stamp the packet and store the value in registers (T3).	Hardware	Slave
Time stamp incoming Delay_Req packet, store the value in register	Hardware	Master
Read the time stamp from register and send back to slave using a Delay_Response packet.	Software	Master

Action	Responsibility	Node Role
Once received, the Delay_Response packet calculate offset using T1, T2, T3 and T4 values.	Software	Slave

6.5.2.1 Time Sync Indications in Rx and Tx Packet Descriptors

Some indications need to be transferred between software and hardware regarding PTP packets. On the Tx path, software should set the 1588 bit in the Tx packet descriptor (bit 9). On the Rx path, hardware has two indications to transfer to software, one is to indicate that this packet is a PTP packet (whether time stamp is taken or not). This is also for other types of PTP packets needed for management of the protocol and this bit is set only for the L2 type of packets (the PTP packet is identified according to its Ethertype). PTP packets have the *L2 Packet* bit in the packet type field set (bit 11 in the receive descriptor) and the Ethertype matches the filter number set by software in the *ETQF* registers to filter PTP packets. The UDP type of PTP packets 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 *RDESC.STATUS.TS* (bit 16) to indicate to software that time stamp was taken for this packet. Software needs to access the time stamp registers to get the time stamp values.

6.5.3 Hardware Time Sync Elements

All time sync hardware elements are reset to their initial values upon Master reset. Upon change in link speed some of the time sync parameters should be changed accordingly.

6.5.3.1 System Time Structure and Mode of Operation

The *SYSTIME* is a 96 bit register is composed of: *SYSTIMR*, *SYSTIMEL* and *SYSTIMEH* registers: The *SYSTIMR* register holds the sub nsec fraction, the *SYSTIMEL* register holds the nsec fraction and the *SYSTIMEH* register holds the sec fraction of the time (note that the upper two bits of the *SYSTIMEL* register are always zero while the max value of this register is 999,999,999 dec).

- **Initial Setting** — Setting the initial time is done by direct write access to the *SYSTIME* register. Software should set first the *SYSTIMEL* and then set the *SYSTIMEH.* Setting the *SYSTIMR* is meaningless while it represents sub ns units. It is recommended to disable the timer at programming time as follows:
- **Run Time** - During run time the *SYSTIME* timer value in the *SYSTIMEH*, *SYSTIMEL* and *SYSTIMR* registers, is updated periodically each 12.5 nS clock cycle according to the following formula:
  - Define: ***INC_TIME* == 12.5 nsec +/- *TIMINCA.Incvalue* * 2^{-32} nsec**. Add or subtract the TIMINCA.Incvalue is defined by TIMINCA.ISGN (while 0b means Add and 1b means Subtract)
  - Then: ***SYSTIME* = *SYSTIME* + *INC_TIME***
- **Reading the *SYSTIME*** register by software is done by the following sequence:
  - Read the *SYSTIMEL* register
  - Read the *SYSTIMEH* register
- **Dynamic update of SYSTIME** registers is done by using the TIMADJ registers by the following flow:
  - Write the *Tadjust* value and its *Sign* to the *TIMADJ* register (the Sign bit indicates if the Tadjust value should be added or subtracted)
  - Following the write access to the *TIMADJ* register, the hardware repeats the following two steps at each 12.5 nsec clock as long as the Tadjust > 0.
    - ***SYSTIME* = *SYSTIME* + *INC_TIME* +/- 1 nsec**. Add or subtract 1 nsec is defined by *TIMADJ.Sign* (while '0' means Add and '1' means Subtract)

- ***Tadjust = Tadjust* - 1 nsec**

— As shown above, the time adjustment might take multiple clocks. Software might write a new value to the TIMADJ register before the hardware completed the previous adjustment. In such a case, the new value written by software, overrides the above equation. If such a race is not desired, the software could check that the previous adjustment is completed by one of the following methods:

- Wait enough time before accessing the *TIMADJ* register which guarantees that the previous update procedure is completed.
- Poll the matched *TSICR.TADJ* flag which is set by the hardware each time the update procedure is completed.

6.5.3.2 Time Stamping Mechanism

The time stamping logic is located as close as possible to the PHY. Figure 6.31 shows the exact point in time where the time value is captured by the hardware relative to the packet content. This is to reduce delay uncertainties originated from implementation differences. As the time stamp is sampled at a very late phase in the data path, the integrated 10 GbE LAN controller does not insert it to the transferred packet. Instead, the integrated 10 GbE LAN controller supports the two-step operation as follows for Tx and Rx.

Note: Time stamping logic is located at the MAC/PHY interface. There is no time stamping of packets when MAC loopback is activated.

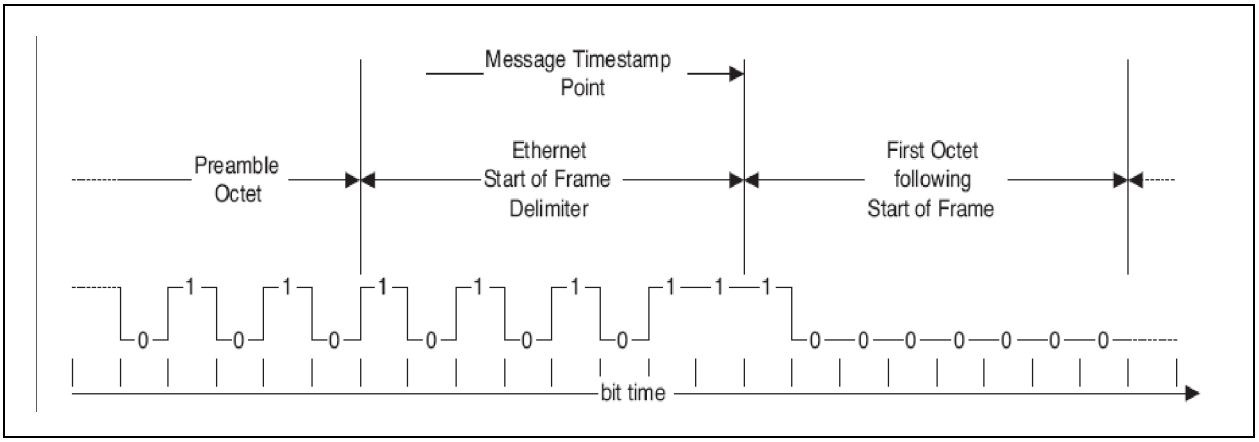


Figure 6.31. Time Stamp Point

6.5.3.2.1 Single Tx Time Stamping

The time stamp logic is activated if enabled by the TSYNCTXCTL.EN bit and the time stamp bit in the packet descriptor is set. In this case, hardware captures the packet's transmission time in the TXSTMPL and TXSTMPH registers. Software is responsible to read the transmission time and append it in the Folow_Up packet as shown in Figure 6.30.

6.5.3.2.2 Single Rx Time Stamping

On the Rx, this logic parses the traversing frame. If it is matching the message type defined in *RXMTRL* register and the *TSYNCRXCTL.TYPE* field, and timestamping is enabled by the *TSYNCRXCTL.EN* bit the reception time stamp is stored in the *RXSTMPL* and *RXSTMPH* registers. In addition, the TS status bits is reported in the Rx descriptor to identify that a time stamp was taken for this packet (stored in the RXSTMPL and RXSTMPH registers).

Note: The time stamp values are locked in the *RXSTMPL* and *RXSTMPH* registers until software accesses them. As long as software does not read these registers, hardware does not capture the time stamp of further Rx packets. 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 posted properly to the software.
Master software must not initiate consecutive sync requests before the previous response is received.

Note: If TSYNCRXCTL.TYPE == 4 (sample time stamp of all packets), then the timestamps are not locked in the *RXSTMPL* and *RXSTMPH* registers and each new packet timestamp will be stored in these registers.

6.5.3.2.3 Multiple Rx Time Stamping

Packets that are identified to be time stamped by hardware are also indicated by the *TS* or *TSIP* flags in the receive descriptors. If the *TS* flag is set, the packet reception time is sampled by the hardware in the *RXSTMPL/H* registers (see Single Rx Time Stamping above). These registers are locked until the software reads its value. If the *TSIP* flag is set, the packet reception time is posted to the packet buffer in host memory. For more information about posting the time stamp in the receive buffer refer to Section 6.1.6.2.

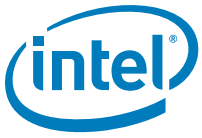
6.5.4 Hardware Time Sync Elements

All time sync hardware is initialized as defined in the registers section upon MAC reset. The time sync logic is enabled if the *TSAUXC.Disable systime* flag is cleared.

The 1588 logic includes multiple registers larger than 32 bits which are indicated as xxxL (Low portion - LS) and xxxH (High portion - MS). When software accesses these registers (either read or write) it should access first the xxxL register (LS) and only then the xxxH register (MS). Accessing the xxxH might impact the hardware functionality which should be triggered only after both portions of the register are valid.

6.5.4.1 Target Time

The two target time registers *TRGTTIML/H0* and *TRGTTIML/H1* enable generating a time triggered event to external hardware using one of the SDP pins according to the setup defined in the *TSSDP* and *TSAUXC* registers. Each target time register is structured the same as the *SYSTIMEL/H* registers. If the value of *SYSTIMEL/H* is equal or larger than the value of the *TRGTTIML/H* registers, a change in level or a pulse is generated on the matched SDP outputs.



6.5.4.1.1 SYSTIM Synchronized Level Change Generation on SDP Pins

To generate a level change on one of the SDP pins when System Time (*SYSTIM*) reaches a pre-defined value, the driver should do the following:

1. Select a specific SDP pin by setting the *TSSDP.TS_SDPx_EN* flag to 1b(while 'x' is 0, 1, 2 or 3).
2. Assign a target time register to the selected SDP by setting the *TSSDP.TS_SDPx_SEL* field to 00b or 01b if level change should occur based on *TRGTTIML/H0 or TRGTTIML/H1,* respectively.
3. Define the selected SDPx pin as output, by setting the appropriate *SDPx_IODIR* bit (while 'x' is 0, 1, 2, or 3) in the *ESDP* register.
4. Define that this SDP is used for TimeSync function by setting the appropriate *SDPx_NATIVE* bit (while 'x' is 0, 1, 2, or 3) in the *ESDP* register. If used with SDP1, clear *ESDP.SDP1_Function* field
5. Program the target time *TRGTTIML/Hx* (while 'x' is 0b or 1b) to the required event time.
6. Program the *TRGTTIML/Hx* to "Level Change" mode by setting the *TSAUXC.PLSG* bit to 0b and *TSAUXC.EN_TTx* bit to 1b (while 'x' is 0b or 1b).
7. To make this a one time operation, the *TSAUXC.DIS_TS_CLEAR* field should be cleared.

When the *SYSTIMEL/H* registers becomes equal or larger than the selected *TRGTTIML/H* registers, the selected SDP changes its output level.

6.5.4.1.2 SYSTIM Synchronized Pulse Generation on SDP Pins

An output pulse can be generated by using one of the target time registers to define the beginning of the pulse and the other target time registers to define the pulse completion time. To generate a pulse on one of the SDP pins when System Time (*SYSTIM*) reaches a pre-defined value, the driver should do the following:

1. Select a specific SDP pin by setting the *TSSDP.TS_SDPx_EN* flag to 1b (while 'x' is 0, 1, 2 or 3).
2. Set *TSSDP.TS_SDPx_SEL* field to 00b to define that the *TRGTTIML/H0* register defines the start of pulse time and *TRGTTIML/H1* register defines the end of pulse time.
3. Define the selected SDPx pin as output, by setting the appropriate *SDPx_IODIR* bit (while 'x' is 0, 1, 2, or 3) in the ESDP register.
4. Define that this SDP is used for TimeSync function by setting the appropriate *SDPx_NATIVE* bit (while 'x' is 0, 1, 2, or 3) in the *ESDP* register. If used with SDP1, clear *ESDP.SDP1_Function* field
5. Program the target time *TRGTTIML/Hx* (while 'x' is 0b or 1b) to the required event time.
6. *TRGTTIML/H0* should be set to a lower value than *TRGTTIML/H1.*
7. Program the *TRGTTIML/H0* defined by the *TSSDP.TS_SDPx_SEL* to "Start of Pulse" mode by setting the *TSAUXC.PLSG0* bit to 1b and *TSAUXC.EN_TT0* bit to 1b (while 'x' defines the SDP used). The*TRGTTIML/H1* register *should be set* to indicate the end of the pulse and *TSAUXC.EN_TT1* bit should be set to 1b.
8. To make this a one time operation, the *TSAUXC.DIS_TS_CLEAR* field should be cleared.
9. When the *SYSTIMEL/H* registers becomes equal or larger than the *TRGTTIML/H0* registers the selected SDP changes its level. Then, when the *SYSTIMEL/H* registers becomes equal or larger than *TRGTTIML/H1* registers (that define the trailing edge of the pulse), the selected SDP changes its level back.

6.5.4.1.3 Synchronized Output Clock on SDP Pins

The integrated 10 GbE LAN controller supports driving a programmable Clock on the SDP pins (up to two output clocks). The output clocks generated are synchronized to the global System time registers (*SYSTIM*). The Target Time registers (*TRGTTIML/H0* or *TRGTTIML/H1*) can be used for the clock output generation. To start an clock output on one of the SDP pins when System Time (*SYSTIM*) reaches a pre-defined value, the driver should do the following:

1. Select a specific SDP pin by setting the *TSSDP.TS_SDPx_EN* flag to 1b (while 'x' is 0, 1, 2 or 3).
2. Select the target time register for a selected SDP, by setting the *TSSDP.TS_SDPx_SEL* field to 10b or 11b if output clock should occur based on *TRGTTIML/H0 or TRGTTIML/H1* respectively.
3. Program the matched *FREQOUT0/1* register to define clock half cycle time.
4. Define the selected SDPx pin as output, by setting the appropriate *SDPx_IODIR* bit (while 'x' is 0, 1, 2, or 3) in the *ESDP* register.
5. Define that this SDP is used for TimeSync function by setting the appropriate *SDPx_NATIVE* bit (while 'x' is 0, 1, 2, or 3) in the *ESDP* register. If used with SDP1, clear *ESDP.SDP1_Function* field
6. If the output clock should start at a specific time, the *TSAUXC.ST0/1* flag should be set to 1b and the matched *TRGTTIML/Hx* should be set to the required start time.
7. Enable the clock operation by setting the relevant *TSAUXC.EN_CLK0/1* bit to 1b.

An interrupt can be generated from the clock output generated by the device by setting the relevant *TSAUXC.EN_TT0/1* bit to 1b and by setting the *TSAUXC.DIS_TS_CLEAR* to allow it to work continuously.The clock out drives initially a logical '0' level on the selected SDP. If the *TSAUXC.ST0/1* flag is cleared, it happens instantly when setting the *TSAUXC.EN_CLK0/1* bit. Otherwise it happens when *SYSTIM* is equal or larger than the *TRGTTIM*. Since then, the hardware repeats endlessly the following two steps:

1. Increment the used *TRGTTIML/Hx* by *FREQOUT*.
2. When *SYSTIM* is equal or larger than the *TRGTTIM*, the SDP reverts its output level.

Note: When clearing *TSAUXC.EN_CLK0/1* while *TSAUXC.EN_TT0/1* was set in order to generate interrupts, clear the matching *TSAUXC.EN_TT0/1* too to avoid one unexpected toggle.

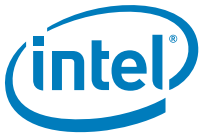
6.5.4.2 Time Stamp Events

Upon a change in the input level of one of the SDP pins that was configured to detect Time stamp events using the TSSDP register or upon a setting of one of the T*SAUXC.SAMP_AUTx* fields, a time stamp of the system time is captured into one of the two auxiliary time stamp registers (*AUXSTMPL/H0* or *AUXSTMPL/H1*). Software enables the timestamp of input event as follow:

1. Define the sampled SDP on AUX time 'x' ('x' = 0b or 1b) by setting the *TSSDP.AUXx_SDP_SEL* field while setting the matched *TSSDP.AUXx_TS_SDP_EN* bit to 1b.
2. Set also the *TSAUXC.EN_TSx* bit ('x' = 0b or 1b) to 1b to enable "timestamping".

Following a transition on the selected SDP, the hardware does the following:

1. The *SYSTIM* registers (low and high) are latched to the selected *AUXSTMP* registers (low and high)
2. The selected *AUTT0* or *AUTT1* flags are set in the *TSICR* and *TSAUXC* registers. If the AUTT interrupt is enabled by the *TSIM* register and the 1588 interrupts are enabled by the Time_Sync flag in the ICR register then an interrupt is asserted as well.After the hardware reports that an event time was latched, the software should read the latched time in the selected *AUXSTMP* registers. Software should read first the Low register and only then the High register. Reading the high register clears the relevant *TSAUC.AUTTx* field and releases the registers to sample a new event.



The software device driver may initiate a sampling of the current time by setting one of the *TSAUXC.SAMP_AUTx* fields. When one of these bits is written the hardware The *SYSTIM* registers (low and high) are latched to the selected *AUXSTMP* registers (low and high) and the selected *AUTT0* or *AUTT1* flags are set in the *TSAUXC* register. Once the device driver reads the selected *AUXSTMP* registers as described above, the hardware clears the relevant *TSAUXC.AUTTx* field and releases the registers to sample a new event.

6.5.5 Time SYNC Interrupts

Time Sync related interrupts can be generated by programming the *TSICR* and *TSIM* registers. The *TSICR* register logs the interrupt cause and the *TSIM* register enables masking specific *TSICR* bits. Occurrence of a Time Sync interrupt sets the *ICR.Time_Sync* interrupt bit.

6.5.6 PTP Packet Structure

The time sync implementation supports both the 1588 V1 and V2 PTP frame formats. The V1 structure can come only as UDP payload over IPv4 while the V2 can come over L2 with its Ethertype or as a UDP payload over IPv4 or IPv6. The 802.1AS uses only the layer 2 V2 format. Note that PTP frame structure over UDP is not supported in the integrated 10 GbE LAN controller for IP tunneling packets.

<table>
<tr><th>Offset in Bytes</th><th>V1 Fields</th><th colspan="2">V2 Fields</th></tr>
<tr><td>Bits</td><td>7 6 5 4 3 2 1 0</td><td colspan="2">7 6 5 4 3 2 1 0</td></tr>
<tr><td>0</td><td rowspan="2">versionPTP</td><td>transportSpecific[1]</td><td>messageType</td></tr>
<tr><td>1</td><td>Reserved</td><td>versionPTP</td></tr>
<tr><td>2</td><td rowspan="2">versionNetwork</td><td colspan="2" rowspan="2">messageLength</td></tr>
<tr><td>3</td></tr>
<tr><td>4</td><td rowspan="16">Subdomain</td><td colspan="2">SubdomainNumber</td></tr>
<tr><td>5</td><td colspan="2">Reserved</td></tr>
<tr><td>6</td><td colspan="2" rowspan="2">flags</td></tr>
<tr><td>7</td></tr>
<tr><td>8</td><td colspan="2" rowspan="8">Correction Field</td></tr>
<tr><td>9</td></tr>
<tr><td>10</td></tr>
<tr><td>11</td></tr>
<tr><td>12</td></tr>
<tr><td>13</td></tr>
<tr><td>14</td></tr>
<tr><td>15</td></tr>
<tr><td>16</td><td colspan="2" rowspan="4">reserved</td></tr>
<tr><td>17</td></tr>
<tr><td>18</td></tr>
<tr><td>19</td></tr>
</table>

Offset in Bytes	V1 Fields	V2 Fields
Bits	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
20	messageType	***Source Port ID***
21	Source communication technology	
22	***Sourceuuid***	
23		
24		
25		
26		
27		
28	sourceportid	
29		
30	***sequenceId***	***sequenceId***
31		
32	***control***	control
33	reserved	logMessagePeriod
34	falgs	n/a
35		

1. Should all be zero.

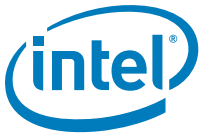
Note: Only the fields with the bold italic format colored red are of interest to hardware.

Ethernet (L2)	VLAN (Optional)	PTP Ethertype	PTP message
Ethernet (L2)	IP (L3)	UDP	PTP message

When a PTP packet is recognized (by Ethertype or UDP port address) on the Rx side the version should be checked if it is V1 then the control field at offset 32 should be compared to message field in the RXMTRL register, 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



MessageId	Message Type	Value (hex)
Unused		4-7
PTP_FOLLOWUP_MESSAGE	General	8
PTP_DELAY_RESP_MESSAGE	General	9
PTP_PATH_DELAY_FOLLOWUP_MESSAGE	General	A
PTP_ANNOUNCE_MESSAGE	General	B
PTP_SIGNALLING_MESSAGE	General	C
PTP_MANAGEMENT_MESSAGE	General	D
Unused		E-F

If V2 mode is configured in TSAUXC register,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: A PTP over UDP packet encapsulated in a VXLAN or NVGRE tunnel will still be identified as a PTP packet.

6.5.6.1 Time Sync Packets identification configuration

The following table summarize the setting needed to identify Time Sync packets.

Table 6-47. Enabling Receive Timestamp

Functionality	Register	Field	Setting Options
Enable receive timestamp in register	TSYNCRXCTL	En	En = 1b (must be set in all the following options).
Sampled V1 Control value	RXMTRL	CTRLT	The CTRLT defines the recognized V1 Control field. This field must be defined if V1 packets recognition is required. 0x0 = PTP_SYNC_MESSAGE
Sampled V2 MessageType value	RXMTRL	MSGT	The MSGT defines the recognized V2 MessageType field. This field must be defined if V2 packets recognition is required. 0x0 = Sync
Enable all packets for timestamp	TSYNCRXCTL	Type	Type equals to 100b enables sampling all packets. Useful only when posting the timestamp to the packet buffer in host memory, enabled per queue by the SRRCTL[n].Timestamp.
Enable L2 1588 packets for timestamp sampling	TSYNCRXCTL	Type	Type equals to 000b or 010b enable V2 packets with MessageType equals to MSGT Type equals to 101b enable all V2 packets with Message Type bit 3 zero (means any event packets)
	ETQF[n]	EType Filter enable IEEE_1588_TIME_STAMP	The EType on one of the enabled ETQF registers (Filter enable is '1') should be set to the 1588 EtherType (equals to 0x88F7) and IEEE_1588_TIME_STAMP field should be set.
Enable 1588 packets over UDP for timestamp sampling	TSYNCRXCTL	Type	Type equals to 001b enables V1 packets with Control field equals to CTRLT parameter Type equals to 010b enables V2 packets with MessageType fields equals to MSGT parameter Type equals to 101b enables all V2 packets with Message Type bit 3 zero (which means any event packets)
	RXMTRL	UDPT	Defines the UDP port to use for V1 detection
Define specific receive queue for the L2 1588 packets	ETQF[n]	Rx Queue Queue Enable	Setting the "Queue Enable" on the same ETQF register as above, the receive queue is defined by the "Rx Queue" field.

6.6 Virtualization

6.6.1 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 integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller provides for these modes.

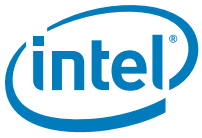
This section assumes a single-root implementation of IOV and no support for multi-root.

6.6.1.1 Direct Assignment Model

The direct assignment support in the integrated 10 GbE LAN controller 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.

6.6.1.1.1 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.

6.6.1.2 System Overview

The following drawings show the various elements involved in the I/O process in a virtualized system. Figure 6.32 shows the flow in software VMDq2 mode and Figure 6.33 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 integrated 10 GbE LAN controller or any other means provided by the VMM vendor.

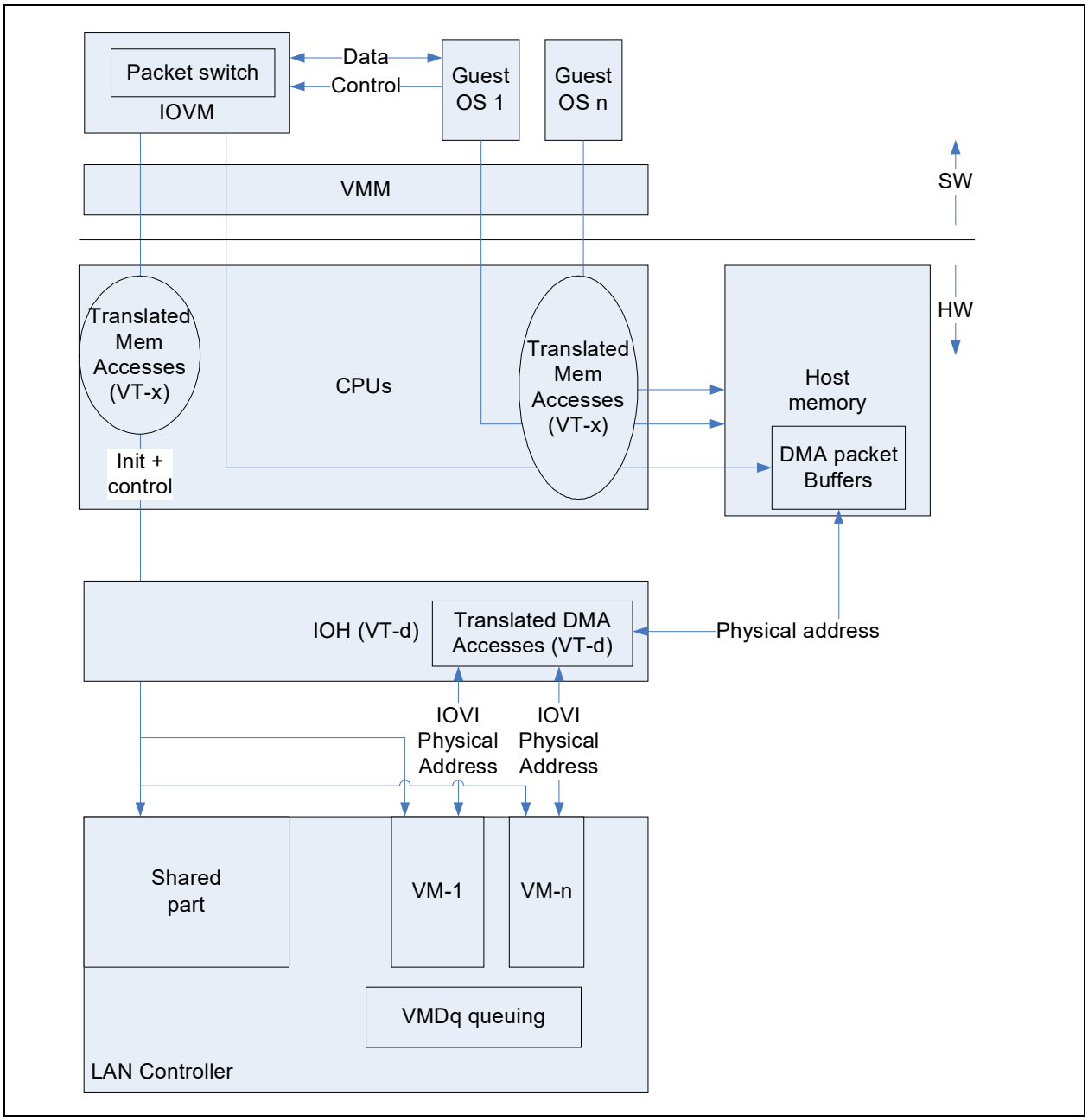


Figure 6.32. System Configuration for VMDq2 Mode

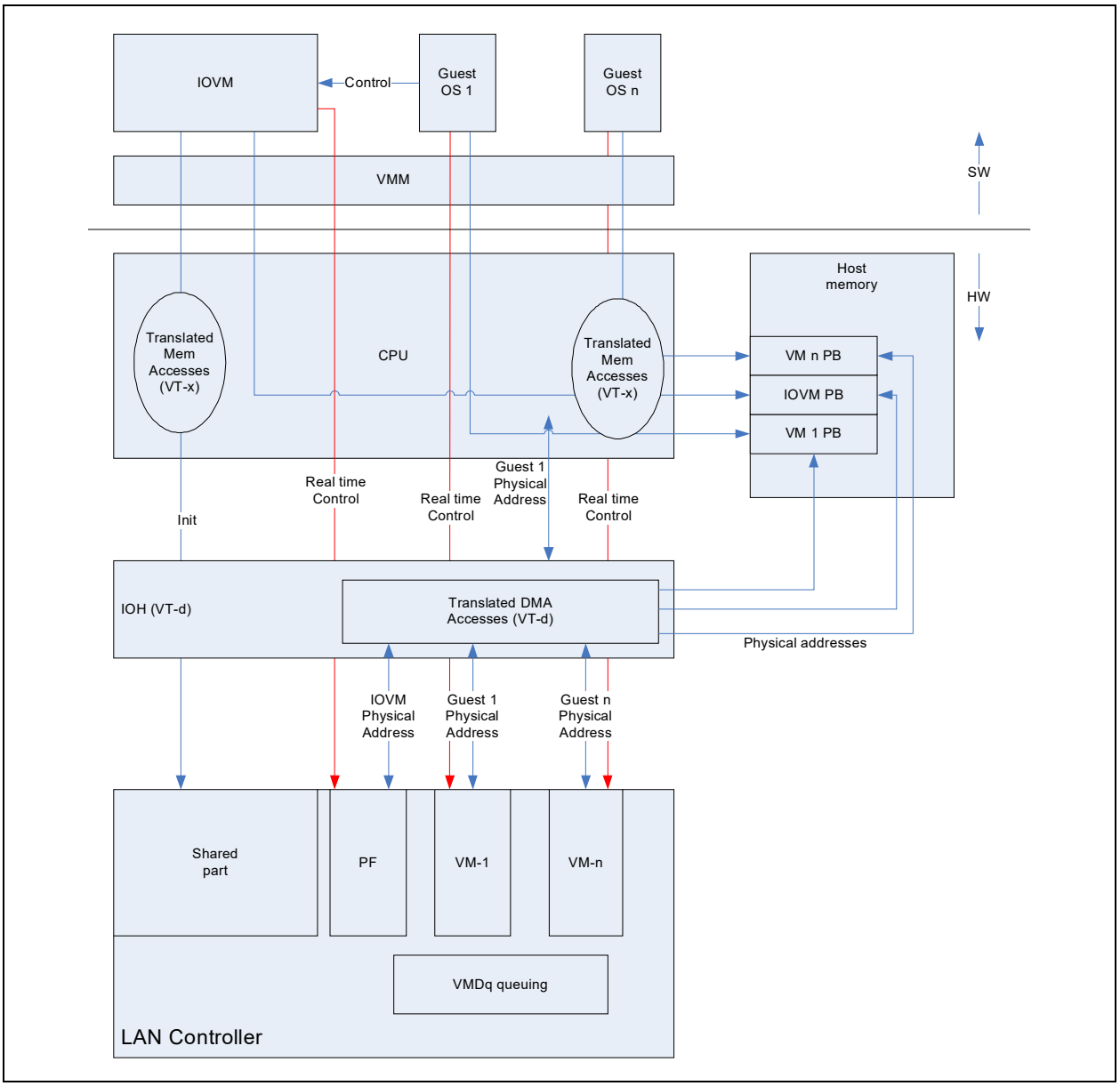


Figure 6.33. System Configuration for IOV Mode

6.6.2 PCI-SIG SR-IOV Support

6.6.2.1 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 integrated 10 GbE LAN controller. 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.

6.6.2.2 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 integrated 10 GbE LAN controller.

6.6.2.2.1 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 6.6.2.4 for details.

6.6.2.2.2 Memory BARs Assignment

The SR-IOV specification defines a fixed stride for all the VF BARs, so that each VF can be allocated part of the memory BARs at a fixed stride from a basic set of BARs. In this method, only two decoders per replicated BAR per PF are required and the BARs reflected to the VF are emulated by the VMM.

The only BARs that are useful for the VFs are BAR0 and BAR3, so only those are replicated. The following table lists the existing BARs and the stride used for the VFs:

Table 6.48. BARs in Integrated 10 GbE LAN Controller (64-bit BARs)

BAR	Type	Usage	Requested Size per VF (=Stride)
0, 1	Mem	CSR space	Maximum (16 KB, page size).
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 6.6.2.7.

The following figure shows the different BARs in an IOV-enabled system:

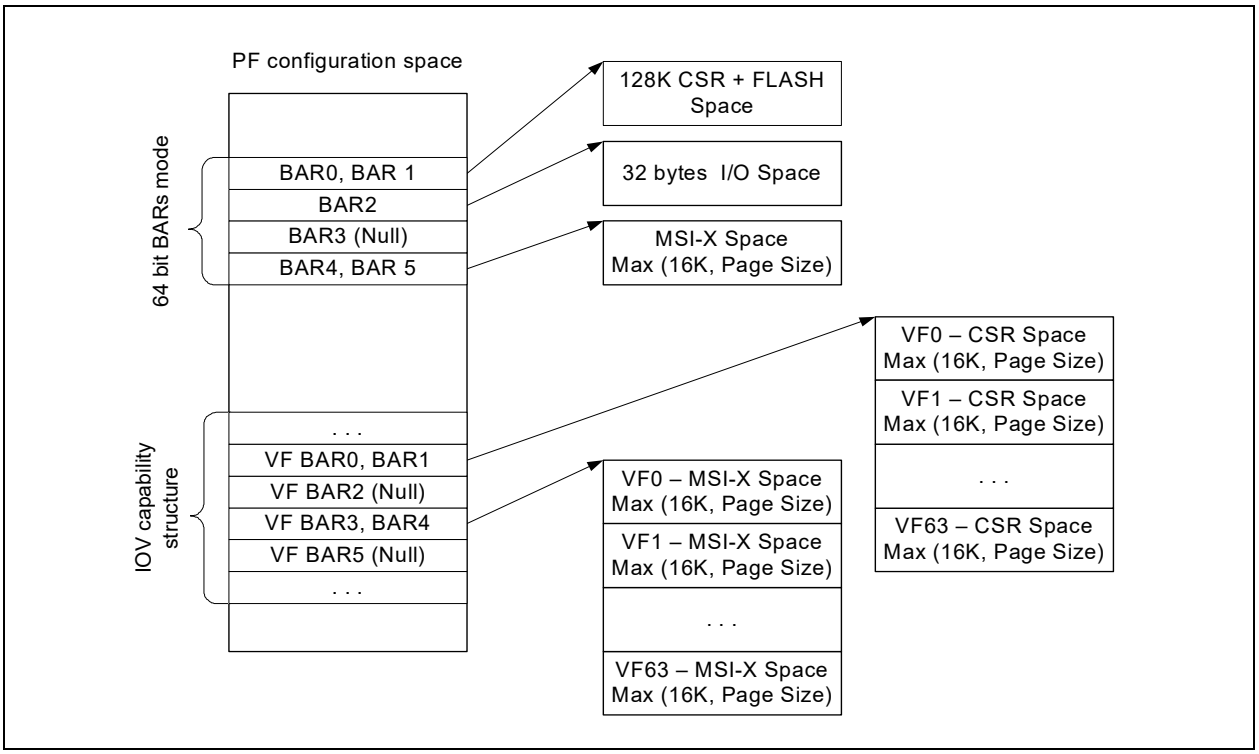


Figure 6.34. BARs in an IOV-enabled System

6.6.2.2.3 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 6.6.2.3 for details.

6.6.2.2.4 MSI and MSI-X Capabilities

Both MSI and MSI-X are implemented in the integrated 10 GbE LAN controller. MSI-X vectors can be assigned per VF. MSI is not supported for the VFs.

6.6.2.2.5 VPD Capability

VPD is implemented only once and is accessible only from the PF.

6.6.2.2.6 Power Management Capability

The integrated 10 GbE LAN controller does not support power management per VF. The power management registers exist for each VF, but only the D0 power state is supported.

6.6.2.2.7 Serial ID

Serial ID capability is not exposed in VFs.

6.6.2.2.8 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 6.6.2.4 for details.

6.6.2.3 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.

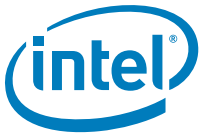
6.6.2.4 Error Reporting

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
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.

6.6.2.5 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. Refer to the following section for details on the Requester ID (RID) allocation to VFs.

6.6.2.6 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.

6.6.2.6.1 Bus Device Function Layout

6.6.2.6.1.1 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 integrated 10 GbE LAN controller is reported to the operating system via the PCIe IOV capability structure.

Table 6.49. 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

6.6.2.6.1.2 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 6.50. 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	

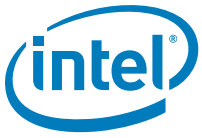


Table 6.50. RID Per VF — Non-ARI Mode

Port	VF#	B,D,F	Binary	Notes
0	1	**B+1**,16,2	B+1,10000,010	
1	1	**B+1**,16,3	B+1,10000,011	
0	2	**B+1**,16,4	B+1,10000,100	
1	2	**B+1**,16,5	B+1,10000,101	
...				
0	63	**B+1**,31,6	B+1,11111,110	
1	63	**B+1**,31,7	B+1,11111,111	Last

Note: When the device ID of a physical function changes (because of LAN disable or LAN function select settings), the VF device IDs changes accordingly.

6.6.2.7 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.

6.6.2.7.1 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.

6.6.2.7.2 Assignment of Queues to VF

See Section 6.2.1.2.1 for allocating Tx queues.

See Section 6.1.3.2 for allocating Rx queues.

The following table lists the Tx and Rx queues to VF allocation.

Table 6.51. Queue to VF Allocation

VF	Queues in 16 VMs Mode	Queues in 32 VMs Mode	Queues in 64 VMs Mode
0	0-7	0-3	0-1
1	8-15	4-7	2-3
...	...	...	...
15	120-127	...	...
...		...	...
31		124-127	...
...			...
63			126-127

6.6.2.7.3 Assignment of MSI-X Vector to VF

See Section 6.3.4.3 for allocating MSI-X vectors in IOV mode.

6.6.2.8 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 integrated 10 GbE LAN controller. 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, 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 VF Device Registers section.

6.6.2.9 SR IOV Control

In order to control the IOV operation, the physical driver is provided with a set of registers. These include:

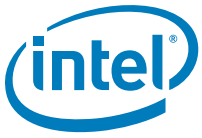
- The mailbox mechanism described in the next section.
- The switch and filtering control registers described in Section 6.6.3.9.
- *PFVFLREC* register indicating that a VFLR reset occurred in one of the VFs (bitmap).

6.6.2.9.1 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 integrated 10 GbE LAN controller provides such a channel that enables direct communication between the two drivers.

The channel consists of a mailbox. Each driver can then receive an indication (either poll or interrupt) when the other side wrote a message.



Assuming a maximum message size of 64 bytes (one cache line), a memory of 64 bytes x 64 VMs = 4 KB is provided per port. The RAM is organized as follows:

Table 6.52. 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 6.53. 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 6.54. 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.

Table 6.54. VF-to-PF Messaging Flow

Step	PF Driver	Hardware	VF #n Driver
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.		
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.

6.6.2.10 DMA

6.6.2.10.1 RID

Each VF is allocated a RID. Each DMA request should use the RID of the VM that requested it. See Section 6.6.2.6 for details.

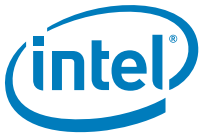
6.6.2.10.2 Sharing of the DMA Resources

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.

6.6.2.11 Timers

6.6.2.11.1 TCP Timer

The TCP timer is available only to the PF. It might indicate an interrupt to the VFs via the mailbox mechanism.



6.6.2.11.2 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.

6.6.2.11.3 Free Running Timer

The free running timer is a PF driver resource the VMs can access. This register is read only to all VFs and is reset only by the PCI reset.

6.6.2.12 Power Management and Wake Up

Power management is a PF resource and is not supported per VF.

6.6.2.13 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 shared SPI Flash. The watermarks of the flow control process and the time-out value are also controllable by the PF only.

Double VLAN is a network setting and as such should be common to all VFs.

6.6.2.13.1 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.

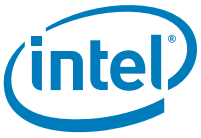
Receiving long packets is enabled separately per Rx queue in the RXDCTL registers. As this impacts the flow control thresholds, the PF should be made aware of the decision of all the VMs. Because of this, the setup of TSO packets is centralized by the PF and each VF might request this setting.

6.6.3 Packet Switching

6.6.3.1 Assumptions

The following assumptions are made:

- The required bandwidth for the VM-to-VM loopback traffic is low. That is, the PCIe bandwidth is not congested by the combination of the VM-to-VM and the regular incoming traffic. This case is handled but not optimized for. Unless specified otherwise, Tx and Rx packets should not be dropped or lost due to congestion caused by loopback traffic.
- If the buffer allocated for the VM-to-VM loopback traffic is full, it is acceptable to back pressure the transmit traffic of the same TC. This means that the outgoing traffic might be blocked if the loopback traffic is congested.
- The decision on local traffic is done only according to the Ethernet DA address and the VLAN tag. There is no filtering according to other parameters (IP, L4, etc.). The switch has no learning capabilities. In case of double VLAN mode, the inner VLAN is used for the switching functionality.
- The forwarding decisions are based on the receive filtering programming.
- No packet switching between TCs.
- Coexistence with TimeSync: time stamp is not sampled for any VM-to-VM loopback traffic.



- Coexistence with Double VLAN: When double VLAN is enabled by DMATXCTL.GDV and it is expected to transmit untagged packets by software, transmit to receive packet switching should not be enabled.

6.6.3.2 Pool Selection

Pool selection is described in the following sections. A packet might be forwarded to a single pool or replicated to multiple pools. Multicast and broadcast packets are cases of replication, as is mirroring.

The following capabilities determine the destination pools of each packet:

- 128 Ethernet MAC address filters (RAH/RAL registers) for both unicast and multicast filtering. These are shared with L2 filtering. For example, the same Ethernet MAC addresses are used to determine if a packet is received by the switch and to determine the forwarding destination. In "E-tag pool select" mode RAL registers bits [13:0] are used for E-tag filtering {GRP, E-CID_base} and pool selection.
- 64 shared VLAN filters (PFVLVF and PFVLVFB registers) — each VM can be made a member of each VLAN.
- Hash filtering of unicast and multicast addresses (if the direct filters previously mentioned are not sufficient)
- Forwarding of broadcast packets to multiple pools
- Forwarding by Ethertype
- Mirroring by pool, VLAN, or link

Receive pool/queue allocation refer to Section 6.1.3.2.

6.6.3.3 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 6.1.3.

As far as switching is concerned, it doesn't matter whether the integrated 10 GbE LAN controller's virtual environment operates in IOV mode or in VMDq2 mode.

When operating in replication mode, broadcast and multicast packets can be forwarded to more than one pool, and is replicated to more than one Rx queue. Replication is enabled by the Rpl_En bit in the PFVTCTL register.

Note: For the following algorithms packets with no E-tag in *PFVTCTL.POOLING_MODE* = 01b (E-tag mode) or all packets in *PFVTCTL.POOLING_MODE* = 00b (MAC mode) are defined as packets with no external tags.

6.6.3.3.1 Replication Mode Enabled

When replication mode is enabled, each broadcast/multicast packet can go to more than one pool. Finding the pool list of any packet is provided in the following steps:

1. **Exact unicast or multicast match** — If there is a match in one of the exact filters (*RAL/RAH*), for unicast or multicast packets, use the *MAC Pool Select Array* (*MPSAR[n]*) bits as a candidate for the pool list. Note that MPSAR[n] must not enable more than one pool for unicast RAL/RAH filters. The compared field differs according to the filtering mode (*PFVTCTL.POOLING_MODE*):
   a. If *POOLING_MODE* = 00b (MAC mode) or for packets with no external tags, the Destination MAC address is compared to {*RAH[15:0], RAL[31:0]*} for all the registers for which RAH.ADTYPE = 0 (MAC)
   b. If *POOLING_MODE* = 01b (E-tag mode) and the packet has an E-tag, the E-tag E-PID (as extracted from offset 47:34, starting from the Ethertype, in the tag identified by the *ETAG_ETYPE* register) is compared to RAL[13:0] for all the registers for which *RAH.ADTYPE* = 1 (E-tag)
2. **PFVFRE** — If any bit in the *PFVFRE* register is cleared, clear the respective bit in the pool list.
3. **Broadcast** — If the packet is a broadcast packet with no external tag, add pools for which their PFVML2FLT.BAM bit (*Broadcast Accept Mode*) is set.
4. **Unicast hash** — If the packet is a unicast packet with no external tag, and the prior steps yielded no pools, check it against the Unicast Hash Table (*PFUTA*). If there is a match, add pools for which their PFVML2FLT.ROPE bit (*Accept Unicast Hash*) is set.
5. **Multicast hash** — If the packet is a multicast packet with no external tag and the prior steps yielded no pools, check it against the Multicast Hash Table (*MTA*). If there is a match, add pools for which their *PFVML2FLT.ROMPE* bit (*Receive Multicast Packet Enable*) is set.
6. **Multicast promiscuous** — If the packet is a multicast packet with no external tag, take the candidate list from prior steps and add pools for which their *PFVML2FLT.MPE* bit (*Multicast Promiscuous Enable*) is set.
7. **Unicast Promiscuous** - If the packet is a unicast packet with no external tag, take the candidate list from prior steps and add pools for which their *PFVML2FLT.UPE* bit (Unicast Promiscuous Enable) is set.
8. **VLAN groups** — This step is relevant only when VLAN filtering is enabled by the *VLNCTRL.VFE* bit.
   a. Tagged packets: enable only pools in the packet's VLAN group as defined by the VLAN filters — *PFVLVF[n]* and their pool list — *PFVLVFB[n]* or pools for which the *PFVML2FLT.VPE* (VLAN Promiscuous Enable) is set.
   b. Untagged packets: enable only pools with their P*FVML2FLT.AUPE* bit set.
   c. If there is no match, the pool list should be empty.

Note: In a VLAN network, un-tagged packets are not expected. Such packets received by the switch should be dropped, unless their destination is a virtual port set to receive these packets. The setting is done through the *PFVML2FLT.AUPE* bit. It is assumed that VMs for which this bit is set are members of a default VLAN and thus only MAC queuing is done on these packets.

9. **Default Pool** — If the pool list is empty at this stage and the *PFVTCTL.Dis_Def_Pool* bit is cleared, 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 pooling action is requested and the *Pool Enable* bit in the *ETQF* is set, the pool list is set to the pool pointed to by the filter.
11. **Filter Local Packets (source address pruning)** - The pruning operation depends on the filtering mode (*PFVTCTL.POOLING_MODE*):
    a. If *POOLING_MODE* = 00b (MAC mode) or for packets with no external tag, the Source MAC address is compared to {RAH[15:0], RAL[31:0]} for all the registers for which RAH.ADTYPE = 0 (MAC).

b. If *POOLING_MODE* = 01b (E-tag mode) and the packet has an E-tag, {00,E-tag ingress-E-CID_base} (as extracted from offset 31:20, starting from the Ethertype, in the tag identified by the *ETAG_ETYPE* register) is compared to RAL[13:0] for all the registers for which *RAH.ADTYPE* = 1 (E-tag).

If there is a match to one of the RAL/RAH(n) clear all the pools that are set in both the matching MPSAR[n] and the PFFLP from the pool list. This will prune multicast packets from getting back to the sender in VEPA mode.

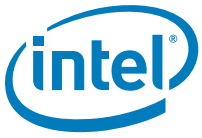
12. **PFVFRE** — If any bit in the *PFVFRE* register is cleared, clear the respective bit in the pool list. The PFVFRE register blocks reception by a VF while the PF configures its registers.
13. **Mirroring** — Each of the four mirroring rules adds its destination pool (PFMRCTL.MP) to the pool list if the following applies:
    a. **Pool mirroring** — *PFMRCTL.VPME* is set and one of the bits in the pool list matches one of the bits in the PFMRVM register.
    b. **VLAN port mirroring** — *PFMRCTL.VLME* is set and the index of the VLAN of the packet in the PFVLVF table matches one of the bits in the *PFMRVLAN* register.
    c. **Uplink port mirroring** — *PFMRCTL.UPME* is set, the pool list is not empty.
14. **PFVFRE** — If any bit in the *PFVFRE* register is cleared, clear the respective bit in the pool list. The PFVFRE register blocks reception by a VF while the PF configures its registers. Note that this stage appears twice in order to handle mirroring cases.

6.6.3.3.2 Replication Mode Disabled

When replication mode is disabled, software should take care of multicast and broadcast packets and check which of the VMs should get them. In this mode, the pool list of any packet always contains one pool only according to the following steps:

1. **Exact unicast or multicast match** — The compared field differs according to the filtering mode (*PFVTCTL.POOLING_MODE*):
    a. If *POOLING_MODE* = 00b (MAC mode) or for packets with no external tags, the Destination MAC address is compared to {RAH[15:0], RAL[31:0]} for all the registers for which *RAH.ADTYPE* = 0 (MAC)
    b. If *POOLING_MODE* = 01b (E-tag mode) and the packet has an E-tag, the E-tag E-PID (as extracted from offset 47:34, starting from the Ethertype, in the tag identified by the *ETAG_ETYPE* register) is compared to RAL[13:0] for all the registers for which *RAH.ADTYPE* = 1 (E-tag or VN-tag)
    c. If *POOLING_MODE* = 10b (VN-tag mode) and the packet has a VN-tag, the VN-tag {P, dst_vif/vif_list_id} (as extracted from offset 31:17, starting from the Ethertype, in the tag identified by the *VNTAGTBL* register) is compared to RAL[14:0] for all the registers for which *RAH.ADTYPE* = 1 (E-tag or VN-tag).

    If the compared field matches one of the exact filters (*RAL/RAH*), use the *MAC Pool Select Array* (MPSAR[n]) bits as a candidate for the pool list. Note that MPSAR[n] must not enable more than one pool for unicast RAL/RAH filters.
2. **PFVFRE** — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list.
3. **Unicast hash** — If the packet is a unicast packet with no external tag, and the prior steps yielded no pools, check it against the Unicast Hash Table (*PFUTA*). If there is a match, add the pool for which their PFVML2FLT.ROPE (*Accept Unicast Hash*) bit is set. Refer to the software limitations described after step 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 with no external tag 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.
- Software should not filter out local packets by setting PFFLP bits

6.6.3.4 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 integrated 10 GbE LAN controller.

Tx switching rules are very similar to Rx switching in a virtualized environment, with the following exceptions:

- If a target pool is not found, the default pool is used only for broadcast and multicast packets.
- A unicast packet that matches an exact filter is not sent to the LAN.
- Broadcast and multicast packets are always sent to the external LAN.
- A packet might not be sent back to the originating pool (even if the destination address is equal to the source address) unless loopback is enabled for that pool by the PFVMTXSW[n] register.

The detailed flow for pool selection as well as the rules that apply to loopback traffic is as follows:

- Loopback is disabled when the network link is disconnected. It is expected (but not required) that system software (including VMs) does not post packets for transmission when the link is disconnected. Loopback is disabled when the RXEN (*Receive Enable*) bit is cleared.
- Loopback packets are identified by the *LB* bit in the receive descriptor.

Notes: When Tx switching is enabled, the host shall avoid sending packets longer than 9.5KB.

Tx Switching should be enabled only if *PFVTCTL.POOLING_MODE* = 00b (MAC mode).

6.6.3.4.1 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. **Ether-type filters** - if one of the enabled (*ETQF.TX_ANTISPOOF*) Transmit Ether-type filters (*ETQF.ETYPE*) is matched by the packet and the appropriate Pool loopback/anti spoof bit in the *PFVFSPOOF* is set (*PFVFSPOOF.ETHERTYPELB/PFVFSPOOF.ETHERTYPEAS*), the pool list is set to the pool pointed to by the *ETQS* matching register (loopback) or the packet is dropped (anti spoof). All the subsequent steps after step 3. are skipped

Note: Loopback on *ETQF* match is possible even if the general loopback enable bit (*PFDTXGSWC.LBE*) is clear.

3. **PFVFRE** — If any bit in the *PFVFRE* register is cleared, clear the respective bit in the pool list.
4. **Broadcast** — If the packet is a broadcast packet, add pools for which their *PFVML2FLT.BAM* bit (*Broadcast Accept Mode*) is set.
5. **Unicast hash** — If the packet is a unicast packet, and the prior steps yielded no pools, check it against the Unicast Hash Table (*PFUTA*). If there is a match, add pools for which their PFVML2FLT.ROPE bit (*Accept Unicast Hash*) is set.
6. **Multicast hash** — If the packet is a multicast packet and the prior steps yielded no pools, check it against the Multicast Hash Table (*MTA*). If there is a match, add pools for which their PFVML2FLT.ROMPE bit (*Receive Multicast Packet Enable*) is set.
7. **Unicast Promiscuous** - If the packet is a unicast packet, take the candidate list from prior steps and add pools for which their *PFVML2FLT.UPE* bit (Unicast Promiscuous Enable) is set.
8. **Multicast Promiscuous** — If the packet is a multicast packet, take the candidate list from prior steps and add pools for which their *PFVML2FLT.MPE* bit (*Multicast Promiscuous Enable*) is set.
9. **Filter source pool** — The pool from which the packet was sent is removed from the pool list unless the PFVMTXSW.LLE bit is set.
10. **VLAN groups** — This step is relevant only when VLAN filtering is enabled by the *VLNCTRL.VFE* bit.
    a. Tagged packets: enable only pools in the packet's VLAN group as defined by the VLAN filters — *PFVLVF[n]* and their pool list — *PFVLVFB[n]* or pools for which the *PFVML2FLT.VPE* (VLAN Promiscuous Enable) is set.
    b. Untagged packets: enable only pools with their *PFVML2FLT.AUPE* bit set.
    c. If there is no match, the pool list should be empty.
11. **Forwarding to the network** — Packets are forwarded to the network in the following cases:
    a. All broadcast and multicast packets.
    b. Unicast packets that do not match any exact filter. A match of an exact filter that also points to a pool disabled via *PFVFRE* is not considered a match.
12. PFVFRE — If any bit in the *PFVFRE* register is cleared, clear the respective bit in the pool list (pre mirroring step).

13. **Mirroring** — Each of the following three mirroring rules adds its destination pool (*PFMRCTL.MP*) to the pool list if the following applies:
    a. **Pool mirroring** — *PFMRCTL.VPME* is set and one of the bits in the pool list matches one of the bits in the PFMRVM register.
    b. **VLAN port mirroring** — *PFMRCTL.VLME* is set and the index of the VLAN of the packet in the PFVLVF table matches one of the bits in the *PFMRVLAN* register.
    c. **Downlink port mirroring** — *PFMRCTL.DPME* is set and the packet is sent to the network.
14. PFVFRE — If any bit in the *PFVFRE* register is cleared, clear the respective bit in the pool list (post mirroring step).

6.6.3.4.2 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. **Ether-type filters** - if one of the enabled (*ETQF.TX_ANTISPOOF*) Transmit Ether-type filters (*ETQF.ETYPE*) is matched by the packet and the appropriate Poll loopback/anti spoof bit in the PFVFSPOOF is set (*PFVFSPOOF.ETHERTYPELB/PFVFSPOOF.ETHERTYPEAS*), the pool list is set to the pool pointed to by the *ETQS* matching register (loopback) or the packet is dropped (anti spoof). All the subsequent steps after step 3. are skipped

Note: Loopback on Ethertype match is possible even if the general loopback enable bit (*PFDTXGSWC.LBE*) is clear.

3. **PFVFRE** — If any bit in the *PFVFRE* register is cleared, clear the respective bit in the pool list.
4. **Unicast hash** — If the packet is a unicast packet, and the prior steps yielded no pools, check it against the Unicast Hash Table (*PFUTA*). If there is a match, add the pool for which their PFVML2FLT.ROPE bit (*Accept Unicast Hash*) is set. Refer to the software limitations that follow.
5. **VLAN groups** — This step is relevant only when VLAN filtering is enabled by the *VLNCTRL.VFE* bit.
    a. Tagged packets: enable only pools in the packet's VLAN group as defined by the VLAN filters — PFVLVF[n] and their pool list — *PFVLVFB[n]* or pools for which the *PFVML2FLT.VPE* (VLAN Promiscuous Enable) is set.
    b. Untagged packets: enable only pools with their *PFVML2FLT.AUPE* bit set.
    c. If there is no match, the pool list should be empty.
6. **Multicast or Broadcast** — If the packet is a multicast or broadcast packet and was not forwarded in step 1 and 2, set the default pool bit in the pool list (from *PFVTCTL.DEF_PL*).
7. **Filter source pool —** The pool from which the packet was sent is removed from the pool list unless the *PFVMTXSW.LLE* bit is set.
8. **Forwarding to the Network** — Packets are forwarded to the network in the following cases:
    a. All broadcast and multicast packets.
    b. Unicast packets that do not match any exact filter. A match of an exact filter that also points to a pool disabled via *PFVFRE* is not considered a match.
9. PFVFRE — If any bit in the *PFVFRE* register is cleared, clear the respective bit in the pool list.

The following software limitations apply when replication is disabled:

1. It is software's responsibility not to set more than one bit in the bitmaps of the exact filters. Note that multiple bits can be set in an *RAH* register as long as it is guaranteed that the packet is sent to only one queue by other means (such as VLAN).

2. Software must not set per-VM promiscuous bits (multicast or broadcast).
3. Software must not set the *ROPE* bit in more than one *PFVML2FLT* register.
4. Software should not activate mirroring.

6.6.3.5 Mirroring Support

The integrated 10 GbE LAN controller 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 — reflects all the packets received to a pool from the network.
2. Uplink port mirroring — reflects all the traffic received from the network.
3. Downlink port mirroring — reflects all the traffic transmitted to the network.
4. VLAN mirroring — reflects all the traffic received from the network in a set of given VLANs (either from the network or from local VMs).

Note: Reflects 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: In order to get all traffic irrespective of the filtering rules, the mirroring pool should be set to promiscuous mode and promiscuous VLAN mode.

Mirroring modes are controlled by a set of rule control registers:

- *PFMRCTL* — controls the rules to be applied and the destination port.
- *PFMRVLAN* — controls the VLAN ports as listed in the *PFVLVF* table taking part in the VLAN mirror rule.
- *PFMRVM* — controls the pools taking part in the pool mirror rule.

6.6.3.6 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.

6.6.3.6.1 Local Traffic Offload

The following capabilities are not supported on the loopback path:

The RSS and VLAN strip offload capabilities are only supported if the *CC* bit in the transmit descriptor of the packet is set and if it is a tunnel packet, the *TUNNEL.OUTERIPCS* is also set. The reason is that when these bits are not set, software does not provide the necessary offload offsets with the Tx packet.

- Receive Side Coalescing (RSC) is not supported.

6.6.3.6.2 Rx Traffic Offload

- CRC offload is a global policy. CRC strip is enabled or disabled for all received packets.

6.6.3.7 Rate Control Features

6.6.3.7.1 Congestion Control

Tx packets going through the local switch are stored in the Rx packet buffer, similar to packets received from the network. Tx to Rx switching always avoids packet drop as if flow control is enabled. Therefore, the software must set the *FCRTH[n].RTH* fields regardless if flow control is activated on the integrated 10 GbE LAN controller.

The integrated 10 GbE LAN controller 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.

6.6.3.7.2 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 integrated 10 GbE LAN controller. 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.

6.6.3.7.3 Receive Priority

As the switch might decide to loopback packets from the transmit path to the receive path, in case the receive path is full, the transmit path might be blocked (including the traffic to the LAN). The integrated 10 GbE LAN controller guarantees that packets are not dropped because of that and that one traffic class flow is not affected by congestion in another traffic class. The PF driver might decide to program the integrated 10 GbE LAN controller to drop packets from receive queues without available descriptors.

In order to keep the congestion effect locality, receive traffic from the LAN have higher priority than loop back traffic. This way large loopback traffic does not impact the network.

6.6.3.8 Small Packets Padding

In virtualized systems, the driver receiving the packet in the VM might not be aware of all the hardware offloads applied to the packet. Thus, in case of stripping actions by the hardware (VLAN strip), it might receive packets which are smaller than a legal packet. The integrated 10 GbE LAN controller provides an option to pad small packets in such cases so that all packets have a legal size. This option can be enabled only if the CRC is stripped. In these cases, all small packets are padded to 60 bytes (legal packet - 4 bytes CRC). The padding is done with zero data. This function is enabled via the *RDRXCTL.PSP* bit.

Packet padding is done after tag extractions and do not take into account if tags are exposed in the Rx descriptor status.

6.6.3.9 Switch Control

The PF driver has some control of the switch logic. The following registers are available to the PF for this purpose:

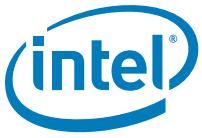
PFVTCTL: - PF Virtual Control register — contains the following fields:

- 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.
- 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.
- 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 6.6.3.5.
- PFVFTE — Enables/disables transmission of packets to the link to a specific VF. Used during initialization of the VF.
- 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.

In addition, the rate-control mechanism is programmed.

6.6.4 Security Features

The integrated 10 GbE LAN controller allows some security checks on the inbound and outbound traffic of the switch.



6.6.4.1 Inbound Security

Each incoming packet (either from the LAN or from a local VM) is filtered according to the VLAN tag so that packets from one VLAN cannot be received by pools that are not members of that VLAN.

6.6.4.2 Outbound Security

6.6.4.2.1 MAC Anti-spoofing

Each pool is associated with one or more Ethernet MAC addresses on the receive path. The association is determined through the MPSAR registers. The MAC anti-spoofing capability insures that a VM always uses a source Ethernet MAC address on the transmit path that is part of the set of Ethernet MAC addresses defined on the Rx path. A packet with a non-matching SA is dropped, preventing spoofing of the Ethernet MAC address. This feature is enabled in the PFVFSPOOF.MACAS field, and can be enabled per Tx pool.

Note: Anti-spoofing is not available for VMs that hide behind other VMs whose Ethernet MAC addresses are not part of the RAH/RAL Ethernet MAC Address registers. In this case, anti-spoofing should be done by software switching, handling these VMs.

6.6.4.2.2 VLAN Anti-spoofing

Each pool is associated with one or more VLAN tags on the receive path. The association is determined through the PFVLVF and PFVLVFB registers. The VLAN anti-spoofing capability insures that a VM always uses a VLAN tag on the transmit path that is part of the set of VLAN tags defined on the Rx path. A packet with a non-matching VLAN tag is dropped, preventing spoofing of the VLAN tag. This feature is enabled in the PFVFSPOOF.VLANAS field, and can be enabled per Tx pool.

Notes: If VLAN anti-spoofing is enabled, 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.

Un-tagged packets are not checked by VLAN anti-spoofing.

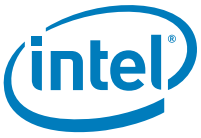
6.6.4.2.3 VLAN Tag Validation

In PCI-SIG IOV scenarios the driver might be malicious, and thus might fake a VLAN tag. The integrated 10 GbE LAN controller provides the ability to force a specific VLAN value for a VM. The possible behaviors are controlled by the PFVMVIR[n] registers as follows:

- Use descriptor value — to be used in case of a trusted VM that can decide which VLAN to send. This option should also be used in case one VM is member of multiple VLANs.
- Always insert default VLAN — this mode should be used for non-trusted or non-VLAN aware VMs. In this case, any VLAN insertion command from the VM is ignored. If a packet is received with a VLAN, the packet should be dropped.
- Never insert VLAN — This mode should be used in a non-VLAN network. In this case, any VLAN insertion command from the VM is ignored. If a packet is received with a VLAN, the packet should be dropped.

Notes: The VLAN insertion settings should be done before any of the queues of the VM are enabled.

When double VLAN is enabled by DMATXCTL.GDV and it is expected to transmit untagged packets by software, VLAN validation should not be enabled.



6.6.4.2.4 E-tag Insertion

In addition to the VLAN insertion described above, the device can add an E-tag to the packets sent by a VF. The PFVMVIR[n].TAGA bit defines if a tag should be added. If this bit is set, the added tag is as follow:

- TAGA = 00b (MAC mode) or 11b (Reserved): Insert no tag.
- TAGA = 01b (E-tag mode): An E-tag is added as follow: {TAG_ETYPE.ETAG_ETHERTYPE[15:0], PFVMTIR.PORT_TAG_ID[31:0], 0x0000}.

Notes: The E-tag insertion settings should be done before any of the queues of the VM are enabled.

6.6.4.2.5 Ether-type Anti-spoofing/Pruning

Each pool can be set to drop packets associated with one or more (up to 8) Ether-types on the transmit path. The Ether-types are defined using the *ETQF* registers. The Ether-type pruning (anti spoof) capability ensures that a VM doesn't send out LLDP/ECP control frames to external switches. The Ether-type loopback capability enables the loopback of link layer packets to a pre defined pool destination.

Ether-type pruning (anti spoof) feature is enabled in the *PFVFSPOOF.ETHERTYPEAS* field, and can be enabled per transmit pool. Ether-type loopback feature is enabled in the *PFVFSPOOF.ETHERTYPELB* field, and can be enabled per transmit pool.

6.6.4.3 Malicious Driver Detection

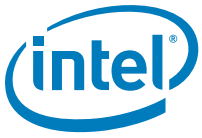
The hardware can be programmed to take some action as a result of some misbehavior of a VM. These actions might hint to the fact that some VM is malicious and the VMM should remedy the situation. In order to inform the VMM of this fact, The Mailbox bit (*EICR.MAILBOX* bit) may be set to indicate the occurrence of such behavior. The *LMVM_RX* and *LMVM_TX* register contains information on which queue (*LMVM_TX.Last_Q* or *LMVM_RX.Last_Q*) and port (*LMVM_TX.Mal_PF* and *LMVM_RX.Mal_PF*) the malicious behavior was detected. The *LVMMC_TX* and *LVMMC_RX* registers contain information on the type of error detected and are clear by read.

Malicious driver behavior detection is enabled by setting the *DMATXCTL.MDP_EN* and *RDRXCTL.MDP_EN* bits to 1. This capability should be enabled before queues are exposed to non trusted virtual machines.

On detection of a malicious driver event the integrated 10 GbE LAN controller stops activity of the offending queue, and sets the bit matching the offending queue in *WQBR_RX* and *WQBR_TX* registers. If *DMATXCTL.MBINTEN or RDRXCTL.MBINTEN* are set, the integrated 10 GbE LAN controller generates an interrupt for transmit or receive errors respectively by asserting the *EICR.MAILBOX* bit. Cause of Malicious driver activation is reported in the *LVMMC_TX* and *LVMMC_RX* registers. To re-activate offending queue, driver should release it by setting the matching bit in *WQBR_RX* or *WQBR_TX* register.

The *WQBR_RX* and *WQBR_TX* registers keeps track of all the queues on which a malicious activity was detected and can be used by the driver to track multiple events.

After the queue is disabled, the initialization flow defined for receive queues should be applied.



6.6.4.3.1 Queue Context Validation

The integrated 10 GbE LAN controller checks that the queue context submitted when a queue is enabled is valid. It also prevents change to static configuration while the queue is enabled. These checks are done both for receive and transmit queues. The table below describes the checks done on the queue context:

Check type	Description	Bit in LVMMC_TX/RX	Action
Receive queue context validity	Check that: • RDLEN >0 • SRRCTL.BSIZEPACKET > 0 • SRRCTL.BSIZEHEADER > 0 if DESCTYP requires header split • SRRCTL.BSIZEHEADER <= 1024. • SRRCTL.DESCTYPE is valid	INVALID_RXQ_CONTEXT	Prevent Enable of queue
Receive queue context change on the fly	Attempt to write RDBAL, RDBAH, RDLEN, or SRRCTL while queue is enabled		Disable queue.
Transmit queue context validity	Check that: • TDLEN >0	INVALID_TXQ_CONTEXT	Prevent Enable of queue
Transmit queue context change on the fly	Attempt to write TDBAL, TDBAH, TDLEN, TDWBAL, or TDWBAH while queue is enabled		Disable queue.

Note: Queue zero is enabled by default. Thus, if malicious driver protection is enabled, in order to change Queue zero configuration, the queue should be disabled.

6.6.4.3.2 Transmit Descriptor Validity Checks

The table below describes the checks are done by the integrated 10 GbE LAN controller to define if a transmit packet descriptor is valid. All the checks are done on the descriptors. The checks on the packet header are described in the previous sections.

Table 6.55. Malicious Driver - Tx Descriptor Checks

Check type	Description	Bit in LVMMC_TX	Action
Mac Header size	Checks that the MAC header size in the context descriptor is at least 14 (or 18 in case of offloaded packet and double VLAN).	MAC_HEADER	Drop Packet and stop offending queue
IPV4 Header	If a checksum or TSO offload is required, checks that the IPv4 header size in the context descriptor is at least 20.	IPV4_HEADER	Drop Packet and stop offending queue
IPV6 Header	If a TSO offload is required, checks that the IPv6 header size in the context descriptor is at least 40.	IPV6_HEADER	Drop Packet and stop offending queue
Wrong MAC_IP	Check that the MAC+ IP (+ OUTERIP + TUNNEL) header size is not bigger than the packet size.	WRONG_MAC_IP	Drop Packet and stop offending queue
TCP header size	If a TCP TSO offload is required, checks that the TCP header size in the context descriptor is at least 20.	TCP_LSO	Drop Packet and stop offending queue
UDP header size	If a UDP TSO offload is required, checks that the TCP header size in the context descriptor is at least 8.	UDP_LSO	Drop Packet and stop offending queue
SCTP data size	If a SCTP checksum offload is required, checks that the SCTP L4 packet size (including header and data) is at least 12.	STCP_CS	Drop Packet and stop offending queue
Packet too big	In case of a single send, check that the packet is not larger than 15.5KBytes.	SIZE	Drop Packet and stop offending queue
Packet too small	Check that the total length of the packet transmitted, not including FCS is at least 13 bytes.	N/A	Silently drop packet.

Table 6.55. Malicious Driver - Tx Descriptor Checks

Check type	Description	Bit in LVMMC_TX	Action
Illegal offload request.	Check that TSO is no requested for SCTP or that a checksum offload is not requested for IPv6 packets.	OFF_ILL	Drop Packet and stop offending queue
Illegal IPSec offload request by VF	Check that a VF had not requested IPSec offload	IPSEC_OFFLOAD	Drop Packet and stop offending queue
SCTP alignment	If an SCTP CRC offload is requested, check that the data size is 4 byte aligned.	SCTP_ALIGNED	Drop Packet and stop offending queue
Zero MSS	Check that the MSS size is larger than zero.	ZERO_MSS	Drop Packet and stop offending queue
Context in middle of packet	Check that a context descriptor is not sent in the middle of a packet.	CONTEXT_IN_PACKET	Drop Packet and stop offending queue
Number of large send header buffers	Check that the Large send header is contained in at most 4 buffers.	LSO_MORE_THAN_4	Drop Packet and stop offending queue
Buffers size and length match - Single Send	For single send, check that the total of all buffers size and the packet length match.	OOS_SSO	Drop Packet and stop offending queue
Buffers size and length match - Large Send	For LSO, check that the total of all buffers size and the packet length match.	OOS_LSO	Drop Packet and stop offending queue
LSO wrong header size	For LSO, check that the size of the header fetched actually match the expected header size.	N/A	Silently drop packet. Counted in SSVPC
UDP data size	If a UDP checksum offload is required, checks that the UDP L4 packet size in the context descriptor is at least 8.	SSO_UDP	Drop Packet and stop offending queue
TCP data size	If a TCP checksum offload is required, checks that the TCP L4 packet size in the context descriptor is at least 20.	SSO_TCP	Drop Packet and stop offending queue
Descriptor Type	Check that only descriptor types 2 (context) or 3 (advanced data descriptor) are used.	DESC_TYPE	Drop Packet and stop offending queue
CC bit not set when needed	CC (Check Context) bit in descriptor is not set in virtualization mode.		Drop Packet and stop offending queue
Null packet check	Check that a Null packet has the EOP bit set.	WRONG_NULL	Drop Packet and stop offending queue
Packet without EOP	Check that a only entire packets are provided by the driver	NO_EOP	Drop Packet and stop offending queue
Burst of contexts	Check that less than *4* Contiguous context descriptor are sent by the driver.	CONTEXT_BURST	Drop Packet and stop offending queue
Legacy Descriptor in SR-IOV	Legacy descriptor used when SR-IOV is enabled.	LEGACY_DESC_IOV	Drop Packet and stop offending queue

6.6.4.3.3 Reactive Malicious Behavior Detection

The table below describes the checks are done by the integrated 10 GbE LAN controller to detect a malicious behavior, even if the packet seems valid.

Table 6.56. Reactive Malicious Checks

Check type	Description	Bit in LVMMC_TX/RX	Action
Malicious VF memory access	A PCIe DMA access initiated by a VF ended with Unsupported Request (UR) or Completer Abort (CA). This check is done for both Tx and Rx queues.	INV_MACC	Drop Packet and stop offending queue
DMA access outside of active memory [1]	A Queue attempt to read memory outside of the active memory range of the system.	INV_MACC	Drop PCIe read transaction. Stop queue that requested the transaction
	A Queue attempt to write memory outside of the active memory range of the system.	N/A	Silently drop PCIe write transaction.

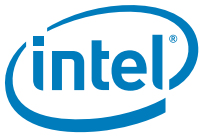


Table 6.56. Reactive Malicious Checks

Check type	Description	Bit in LVMMC_TX/RX	Action
VLAN not expected in packet	A packet where VLAN should be added by device is in Tx descriptor. See Section 6.6.4.2.3	VLAN_IERR	Drop Packet
Anti spoof checks	See Section 6.6.4.2.1 and Section 6.6.4.2.2 and Section 6.6.4.2.5 for details	MAC_VLAN_SPOOF	Drop Packets.

1. A PCIe error interrupt can be asserted when such a transaction is dropped. See Section 6.1.5 for details.

6.6.5 Virtualization of Hardware

This section describes additional features used in both IOV and VMDq2 modes.

6.6.5.1 Per-pool Statistics

Part of the statistics are by definition shared and cannot be allocated to a specific VM. For example, CRC error count cannot be allocated to a specific VM, as the destination of such a packet is not known if the CRC is wrong.

All the non-specific statistics are handled by the PF driver in the same way it is done in non-virtualized systems. A VM might request a statistic from the PF driver but might not access it directly.

The conceptual model used to gather statistics in a virtualization context is that each queue pool is considered as a virtual link and the Ethernet link is considered as the uplink of the switch. Thus, any packet sent by a pool is counted in the Tx statistics, even if it was forwarded to another pool internally or was dropped by the MAC for some reason. In the same way, a replicated packet is counted in each of the pools receiving it.

The following statistics are provided per pool:

- Good packet received count
- Good packet transmitted count
- Good octets received count
- Good octets transmitted count
- Multicast packets received count

Note: All the per VF statistics are read only and wrap around after reaching their maximum value.

6.7 Tunneling Support

The integrated 10 GbE LAN controller supports the VXLAN and NVGRE tunneling packet formats. As part of this support the following features are supported:

- LSO and transmit checksum offloads for tunneled packets as described in Section 6.2.4 and Section 6.2.5.
- RSS forwarding based on inner L3/L4 header as described in Section 6.1.3.6
- Flow director forwarding based on Tenant ID, inner MAC and inner VLAN as described in Section 6.1.3.5
- New indications in the receive descriptor with information on the tunnel headers and on the outer header checksum as described in Section 6.1.5.2
- New packet split modes based on the tunnel header or outer L2 header as described in Section A.2.5

6.8 Receive Side Coalescing (RSC)

The integrated 10 GbE LAN controller can merge multiple received frames from the same TCP/IP connection (referred to as flow in this section) into a single structure. The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller digests received packets and categorizes them by their TCP/IP connections (flows). For each flow, hardware coalesces the packets as shown in Figure 6.35 and Figure 6.36 (the colored parameters are explained in the RSC context table and receive descriptor sections). The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller opens and closes the RSC contexts autonomously with no need for any software intervention. Software needs only to enable RSC in the selected receive queues.

Note: When RSC is enabled, advanced receive descriptors should be used and CRC strip should be enabled.

Figure 6.35 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.

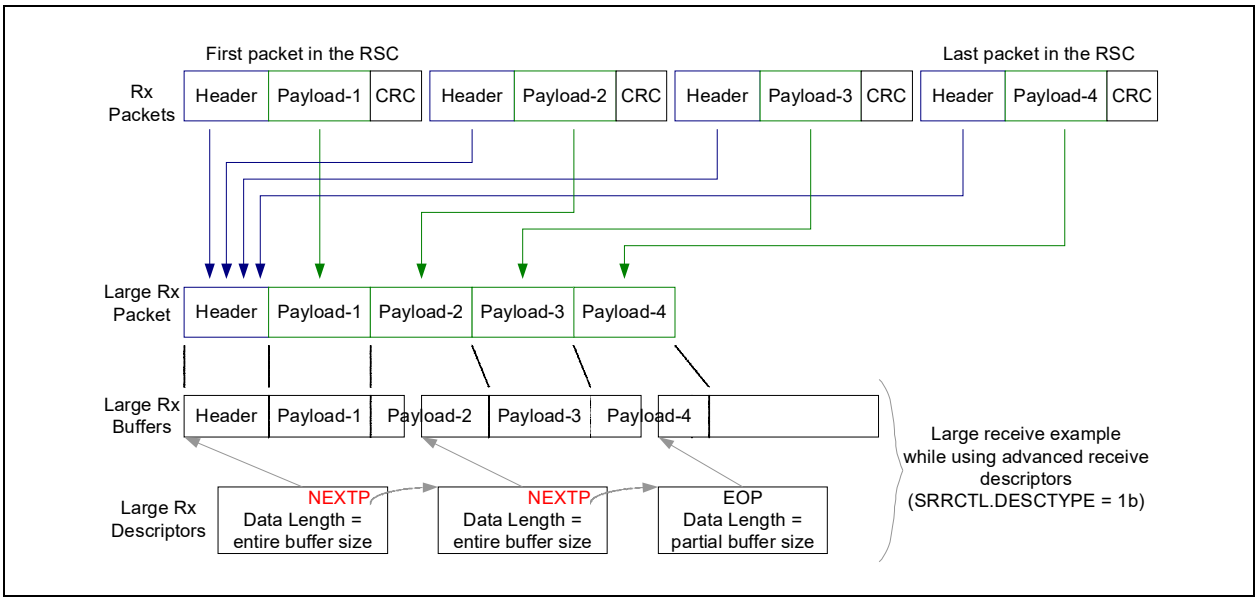


Figure 6.35. RSC Functionality (No Header Split)

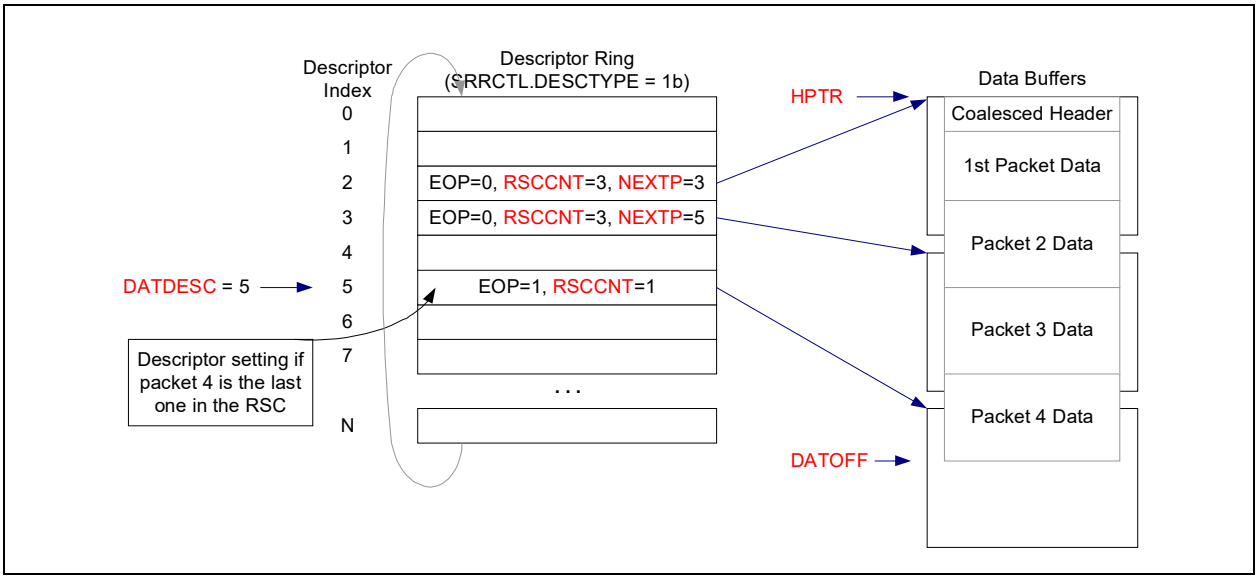


Figure 6.36. RSC Functionality (No Header Split)

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).

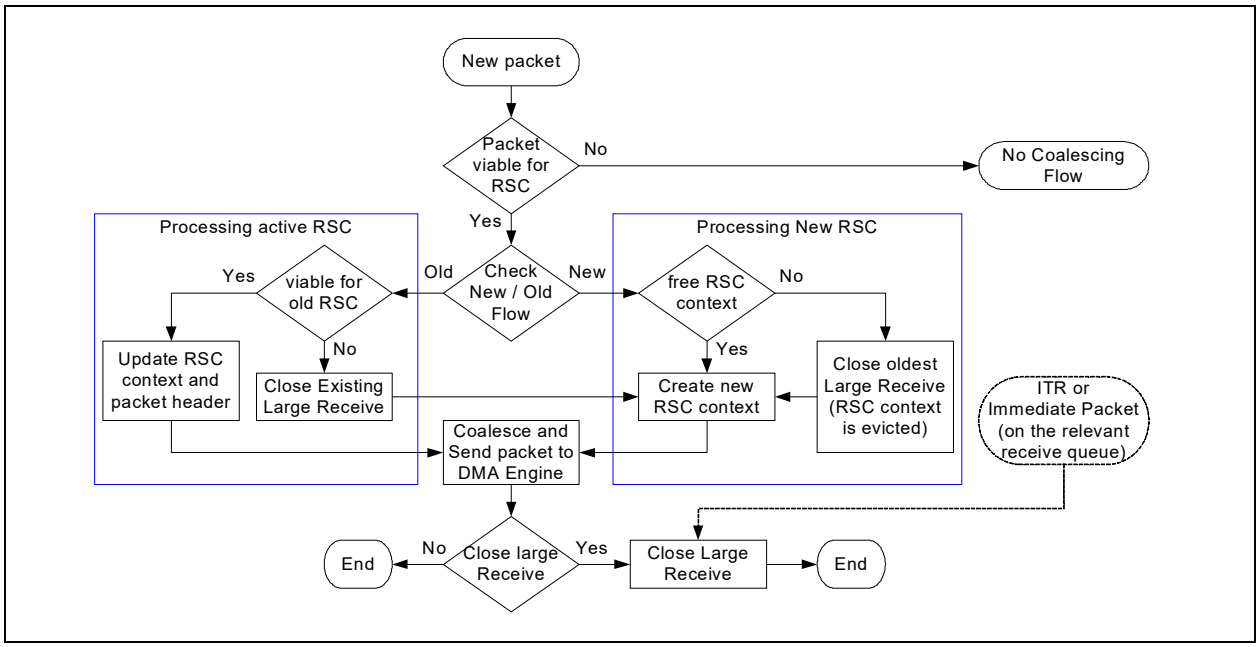
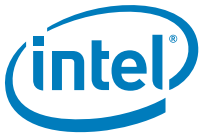


Figure 6.37. RSC Event Flow

6.8.1 Packet Candidacy for RSC

Incoming packets can be good candidates for RSC offload when the following conditions are met. If any of the these conditions are not met, and assuming the queue is configured as required, the received packet is processed in the legacy (non-coalescing) scheme.

- RSC is enabled in the destination receive queue by the *RSCCTL.RSCEN*. In this case, software must set the *SRRCTL.DESCTYPE* field in the relevant queues to advanced descriptor modes.
- RSC is further enabled by the *RSCINT.RSCEN* for the receive queues associated to the interrupts defined by the *RSCINT* registers.
- The *SRRCTL[n].BSIZEHEADER* (header buffer size) must be larger than the packet header (even if header split is not enabled). A minimum size of 128 bytes for the header buffer addresses this requirement.
- The received packet has no MAC errors and no TCP/IP checksum errors. MAC errors are: CRC error 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.
- The packet type is TCP/IPv4 (non-SNAP) with optional VLAN header(s). RSC is not supported for IPv6 packets.
- The packet is not an NVGRE or VXLAN packet.
- IP header does not carry any option headers.
- Enable or disable NFS coalescing.
- The TCP segment is not fragmented.
- The following TCP flags are inactive: FIN, SYN, RST, URG, ECE, CWR, NS and the other three reserved TCP flags (see TCP Flags mapping in Table 6.57).



- The ECT and CE bits in the TOS field in the IP header are not equal to 11b (see the flags in Table 6.58).
- 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 6.8.4.

The supported packet format is as follows:

Size	Packet Fields
6 bytes	Destination Ethernet MAC address.
6 bytes	Source Ethernet MAC address.
[4/6/8 bytes]	Optional External Tag (VLAN, E-tag, VN-tag).
[4 bytes]	Optional VLAN.
2 bytes	Ethernet type field equals 0x0800 (MS byte first on the wire).
20 bytes	IPv4 header with no options.
20 bytes	Basic TCP header (no options — refer to the rows that follow).
[10 bytes]	Optional TCP time stamp header: 1 byte Kind 0x08 1 byte Length 0x0A 4 bytes TS value variable 4 bytes TS echo reply variable
[1 byte]	Optional TCP no operation header: 1 byte Kind 0x01
[1 byte]	Optional TCP end of option header list: 1 byte Kind 0x00
Variable length	TCP payload (RSC candidate must have payload size greater than zero).

Table 6.57. 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 6.58. IP TOS Field — Bit Map

7	6	5	4	3	2	1	0
TOS (DS)						ECT	CE

Table 6.59. 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)

6.8.2 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 6.60). 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 6.8.3.
- If the packet's flow matches an active RSC context then the packet might be appended to the existing RSC as described in Chapter 6.8.4.

Table 6.60. 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

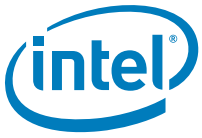


Table 6.60. RSC Context

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.
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 6.8.5.2).
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.

6.8.3 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.

6.8.3.1 RSC Context Setting

The integrated 10 GbE LAN controller 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.

6.8.4 Processing Active RSC

Received packets that belong to an active RSC can be added to the large receive if all the following conditions are met:

- The L2 header size equals the size of previous packets in the RSC as recorded in the internal IPOFF parameter in the RSC context table.
- The packet header length as reported in the HDR_LEN field is assumed to be the same as the first packet in the RSC (not checked by hardware).
- The ACK flag in the TCP header is equal to the RSCACK bit in the RSC context (The value of the ACK flag should be constant in all the coalesced packets).
- The packet type remains the same as indicated by the RSCACKTYPE bit in the RSC context. Packet type can be either pure ACK packet (with no TCP payload) or other.
- For non-RSCACKTYPE (packet with TCP payload): The sequence number in the TCP header matches the expected value in the RSC context (RSCSEQ).

- For RSCACKTYPE: The Acknowledgment number in the TCP header is greater than the RSCSEQ number in the RSC context. Note that the integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller closes the old large receive it is guaranteed that there is at least one free context.

Note: See Section 6.1.6.2 for impact of Time Stamp in packet on RSC decisions.

If the received packet meets all the above conditions, the integrated 10 GbE LAN controller appends this packet payload to the active large receive and updates the context and header as follows. The packet is then DMA'ed to the RSC buffers (as described in Section 6.8.5).

- Update the TCP PSH: The PSH bit in the Large Receive context gets the value of the PSH bit in the TCP header of the received packet.
- Update the expected sequence number for non-RSCACKTYPE: The RSCSEQ in the large receive context is increased by the value of the TCP payload size of the received packet.
- Update the expected sequence number for RSCACKTYPE: The RSCSEQ in the large receive context is updated to the value of the ACK sequence number field in the received packet.
- Update the total length: The RSCIPLEN in the large receive context is increased by the value of the TCP payload size of the received packet. The value of the *Total Length* field in the IP header in the received packet gets the updated RSCIPLEN. Note that in RSCACKTYPE packets the received payload size is zero.
- 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 increased by one on each packet that starts on the same descriptor as long as it does not exceed a value of 0xF. When the RSCCNT is set to 0xF (14 packets) the RSC completes.
  - Decrement by one the Remaining Descriptors (RSCDESC) for each new descriptor.
  - Update the receive descriptor index (DATDESC) for each new descriptor.
  - Update the offset within the data buffer (DATOFF) at the end of the DMA to its valid value for the next packet.

- All other fields are kept as defined by the first packet in the large receive.

6.8.5 Packet DMA and Descriptor Write Back

The Figure 6.38 shows a top view of the RSC buffers using advanced receive descriptors and header split descriptors.

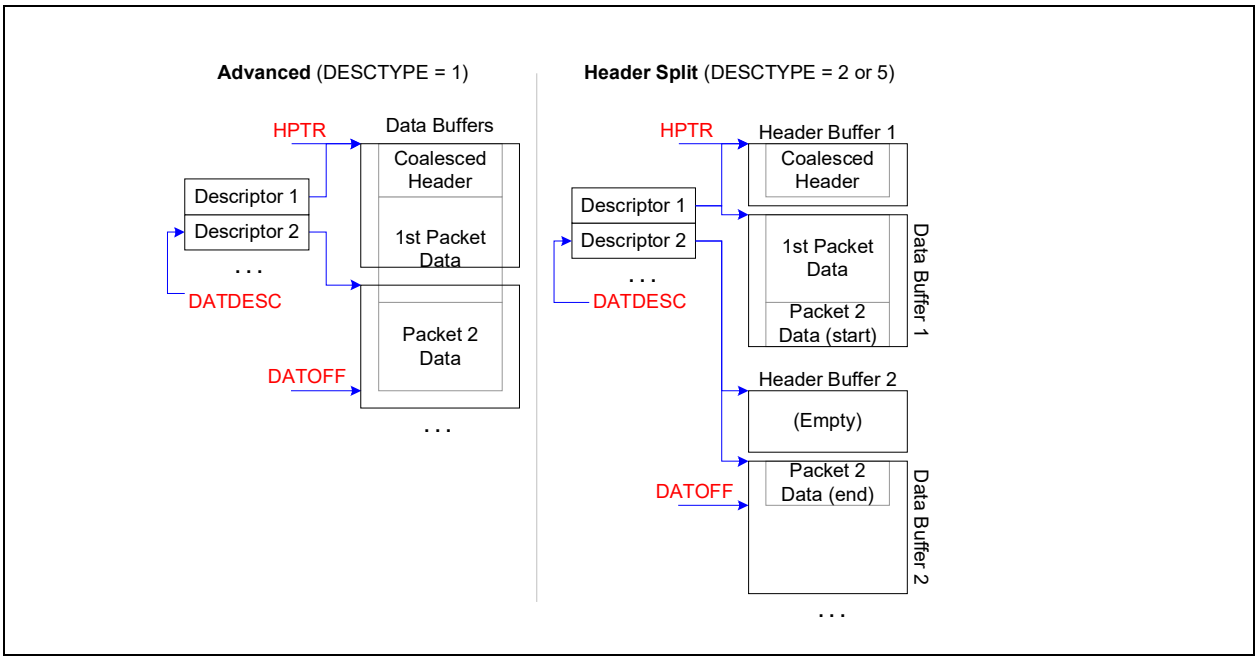


Figure 6.38. RSC — Header and Data Buffers

6.8.5.1 RSC Descriptor Indication (Write Back)

After receiving each packet, the integrated 10 GbE LAN controller 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 6.8.6 summarizes all the causes for RSC completion. Any other descriptor in the middle of the RSC is indicated (write back) when the hardware requires the next descriptor so it can report the NEXTP field explained in the table that follows.

Fields on the Last Descriptors of Large Receive	Fields on All Descriptors Except for the Last One
EOP: End of packet, and all other fields that are reported together with the EOP.	**NEXTP**: Points to the next descriptor of the same large receive.
DD: indicates that this descriptor is completed by the hardware and can be processed by the software. The descriptor and the header of the packet may still be updated by the hardware.	
RSCCNT: indicates the number of coalesced packets in this descriptor.	

6.8.5.2 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 6.8.5.3 and Section 6.8.5.4.

6.8.5.3 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 6.38 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.

6.8.5.4 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 6.8.5.1 details the indicated descriptor fields).
- Decrement the RSCDESC parameter by one and update the DATDESC for each new opened descriptor.

DMA completion.

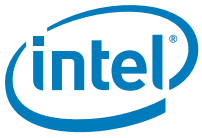
- Following DMA completion, set the DATOFF to the byte offset of the next packet.
- If the PSH TCP flag is active in the coalesced packet then the large receive is completed.

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 integrated 10 GbE LAN controller completes the current large receive while the new packet starts a new large receive.



6.8.6 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 integrated 10 GbE LAN controller 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 6.8.4). 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 6.3.2.1.1.
- EITR expiration while interrupt is disabled — The ITR counter continues to count even when its interrupt is disabled. Every time the timer expires it triggers RSC completion on the associated Rx queues.
- Low number of available descriptors — Whenever crossing the number of free Rx descriptors, the receive descriptor minimum threshold size defined in the SRRCTL[n] registers.
- Interrupt assertion by setting the EICS register has the same impact on packet reception as described in Section 6.3.1.2.1.

Note: In some cases packets that do not meet coalescing conditions might have active RSC of the same flow. As an example: received packets with ECE or CWR TCP flags. Such packets bypass completely the RSC logic (posted as single packets), and do not cause a completion of the active RSC. The active RSC would eventually be closed by either reception of a legitimate packet that is processed by the RSC logic but would not have the expected TCP sequence number. Or, an interrupt event closes all RSC's in its Rx queue. When software processes the packets, it gets them in order even though the RSC completes after the previous packet(s) that bypassed the RSC logic.

Any interrupt closes all RSC's on the associated receive queues. Therefore, when ITR is not enabled any receive packet causes an immediate interrupt and receive coalescing should not be enabled on the associated Rx queues.

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.

6.9 Reliability

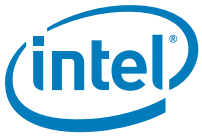
6.9.1 Memory Integrity Protection

All the integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller halts the traffic and sets the ECC error interrupt. Software is then required to initiate a complete initialization cycle to resume normal operation.

6.9.2 PCIe Error Handling

For PCIe error events and error reporting see Receive Descriptors - Section 6.1.5.



7.0 Programming Interface

7.1 General

The integrated 10 GbE LAN controller's address space is mapped into four regions with PCI Base Address Registers listed in Table 7.1.

Table 7.1. Integrated 10 GbE LAN Controller Address Regions

Addressable Content	Mapping Style	Region Size
Internal registers memories and shared SPI Flash (Memory BAR)	Direct memory mapped	128 KB + shared SPI Flash Size[1]
FLASH (optional)[2]	Direct memory-mapped	2 MB - 4 MB
Expansion ROM (optional)	Direct memory-mapped	64 KB - 512 KB
Internal registers and memories (optional)[3]	I/O window mapped	32 bytes
MSI-X (optional)	Direct memory mapped	16 KB[4]

1. The Flash size is defined by the BARCTRL register.
2. The Flash space in the Memory CSR and Expansion ROM Base Address are mapped to different Flash memory regions. Accesses to the Memory BAR at offset 256 KB are mapped to the Flash device at offset 0x0, while accesses to the Expansion ROM at offset 0x0 are mapped to the Flash region pointed by the PXE Driver Module Pointer (read from shared SPI Flash word address 0x05). See Section 5.1. The Expansion ROM region has a size limited to 512 KB.
3. The internal registers and memories can be accessed though I/O space as explained in the sections that follow.
4. See Section 7.1.1.3 for the MSI-X BAR offset in 32-bit and 64-bit BAR options.

7.1.1 Memory-Mapped Access

7.1.1.1 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.

7.1.1.2 Memory-Mapped Accesses to Flash

The external Flash can be accessed using direct memory-mapped offsets from the CSR base address register (BAR0/BAR1). The Flash is only accessible if enabled through the *PCI_LBARCTRL.FLASH_EXPOSE* bit in the PCIe General Configuration Shared SPI Flash Module. 256 KB should be added to the physical address within the external Flash device to get the offset from the BAR.

7.1.1.3 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.

7.1.1.4 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 shared SPI Flash Initialization Control Word, and if the Expansion ROM Base Address register contains a valid (non-zero) base memory address.

7.1.2 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:

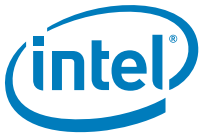
Offset	Abbreviation	Name	RW	Size
0x00	IOADDR	Internal Register, Internal Memory, or Flash Location Address. 0x00000-0x1FFFF – Internal registers/memories 0x20000-0x7FFFF – Undefined	RW	4 bytes
0x04	IODATA	Data field for reads or writes to the internal register, internal memory, or Flash location as identified by the current value in IOADDR. All 32 bits of this register are read/write-able.	RW	4 bytes
0x08-0x1F	Reserved	Reserved.	O	4 bytes

7.1.2.1 IOADDR (I/O Offset 0x00; RW)

The IOADDR register must always be written as a Dword access. Writes that are less than 32 bits are ignored. Reads of any size returns a Dword of data; however, the chipset or CPU might only return a subset of that Dword.

For software programmers, the IN and OUT instructions must be used to cause I/O cycles to be used on the PCIe bus. Because writes must be to a 32-bit quantity, the source register of the OUT instruction must be EAX (the only 32-bit register supported by the OUT command). For reads, the IN instruction can have any size target register, but it is recommended that the 32-bit EAX register be used.

Bits 31 through 20 are ignored by the integrated 10 GbE LAN controller and should be set to zero by software at hardware reset (global reset) or PCI reset, this register value resets to 0x00000000. Once written, the value is retained until the next write or reset.



7.1.2.2 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 integrated 10 GbE LAN controller 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.

7.1.2.3 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.

7.1.3 Configuration Access to Internal Registers and Memories

To support legacy pre-boot 16-bit operating environments without requiring I/O address space, the integrated 10 GbE LAN controller 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 7-2.

Table 7-2. 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:

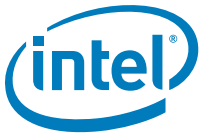
1. CSR address is written to the IOADDR register where:
   a. Bit 31 (*IOADDR.Configuration IO Access Enable*) of the IOADDR register should be set to 1b.
   b. Bits 30:0 of IOADDR should hold the actual address of the internal register or memory being read.
2. CSR value is read from the IODATA register.
   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 *PCIE_CNF.IO_SUP* shared SPI Flash bit must be cleared when the *PCIE_CAPSUP.CSR_CONF_EN* shared SPI Flash bit is set, and vice versa.

7.1.4 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.
ROS	Read Only Status. Writes to ROS fields has no effect. The value of the field can change due to Hardware events.
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/RSV	Reserved. field can return any value on read access and must be set to its initialized value on write access unless specified differently in the field description.



7.1.5 VF Registers Allocated Per Queue

Depending on configuration, each pool has 2, 4, or 8 queues allocated to it. Note that in IOV mode, any queues not allocated to a VF are allocated to the PF. The registers assigned to a queue are accessible both in its VF address space and in the PF address space. This section describes the address mapping of registers that belong to queues.

Section 6.6.2.7.2 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 (such as [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).

7.1.6 Non-queue VF Registers

Registers that do not correspond to a specific queue are allocated addresses in the VF space according to these rules:

- Registers that are read-only by the VF (like 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: Note, however, that 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.

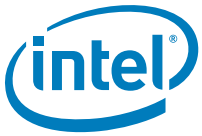
7.1.7 Access to MAC Registers

When working in SGMII 100 Mb/s or 10 Mb/s speeds, the core and MAC layer have separate clock frequencies therefore there is a delay between the time a posted write is issued by software until the time the actual hardware operation completes.

In order to prevent coherency problems, software is required to check and insure that the register is ready to be accessed.

This limitation is not applicable to all registers. The relevant registers that are subject to these limitations each have a BUSY signal in the MAC SGMII Busy register.

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



7.2 Device Registers - PF

7.2.1 BAR0 Registers Summary

Table 7.3. BAR0 Registers Summary

Offset / Alias Offset	Abbreviation	Name	Block	Page
General Control Registers				
0x00000000	CTRL	Device Control Register	Target	312
0x00000008	STATUS	Device Status Register	Target	313
0x00000018	CTRL_EXT	Extended Device Control Register	Target	313
0x00000020	ESDP	Extended SDP Control	Target	314
0x00000028	PHY_GPIO	PHY GPIO Register	Target	315
0x00000030	MAC_GPIO	MAC GPIO Register	Target	315
0x00000100	PHYINT_STATUS0	PHY Interrupt Status Register 0	Target	315
0x00000104	PHYINT_STATUS1	PHY Interrupt Status Register 1	Target	316
0x00000108	PHYINT_STATUS2	PHY Interrupt Status Register 2	Target	317
0x00000200	LEDCTL	LED Control	Target	318
0x00005078	EXVET	Extended VLAN Ether Type - Receive	RX_Filter	319
0x00008224	EXVET_T	Extended VLAN Ether Type - Transmit	DMA_TX	319
0x00015F58	I2CCMD	SFP I^2C Command	MNG	320
0x00015F5C	I2CPARAMS	SFP I^2C Parameters	MNG	320
0x00015F64	GRC	General Receive Control	MNG	321
0x00015FEC	FACTPS	Function Active and Power State to Manageability	MNG	321
0x00015FF0	DEV_FUNC_EN	Device and Functions Enable Control	MNG	322
NVM Registers				
0x00015F48	FLSWCTL	Software Flash Burst Control Register	MNG	322
0x00015F4C	FLSWDATA	Software Flash Burst Data Register	MNG	323
0x00015F50	FLSWCNT	Software Flash Burst Access Counter	MNG	323
0x00015F54	FLUPDATE	Flash Firmware Code Update	MNG	323
0x00015F68	FLA	Flash Access Register	MNG	323
0x00015FF8	EEC	EEPROM Mode Control Register	MNG	324
Flow Control Registers				
0x00003200 + 0x4*n, n=0...3	FCTTVN[n]	Flow Control Transmit Timer Value n	DBU_RX	325
0x00003220 + 0x4*n, n=0...7	FCRTL[n]	Flow Control Receive Threshold Low	DBU_RX	325
0x00003260 + 0x4*n, n=0...7	FCRTH[n]	Flow Control Receive Threshold High	DBU_RX	326
0x000032A0	FCRTV	Flow Control Refresh Threshold Value	DBU_RX	326
0x00003D00	FCCFG	Flow Control Configuration	DBU_RX	326
0x00004294	MFLCN	MAC Flow Control Register	MAC	327
0x0000CE00	TFCS	Transmit Flow Control Status	DBU_TX	327

Table 7.3. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Page
PCIe Registers				
0x00011028	PCI_STATUS1	PCIe Function Status 1	PCIe_GLUE	327
0x00011040	PCI_FWCTRL	PCIe Firmware Control	PCIe_GLUE	328
0x00011044	PCI_CSRTO	PCIe CSR Access Timeout	PCIe_GLUE	328
0x00011050	GCR_EXT	PCIe Control Extended Register	PCIe_GLUE	328
0x00011054	PCI_FLASHTO	PCI Flash Access Timeout	PCIe_GLUE	328
0x00011070	FUNC_RID	Function Requester ID Information Register	PCIe_GLUE	328
0x00011098	PCI_REVID	PCIe Revision ID	PCIe_GLUE	329
0x00011140	PCI_PCIERR	PCIe Errors Reported	PCIe	329
0x00011520	PCI_ICAUSE	PCIe Interrupt Cause	PCIe	329
0x00011528	PCI_IENA	PCIe Interrupts Enable	PCIe	329
0x00011530	PCI_VMINDEX	PCIe VM Pending Index	PCIe	329
0x00011538	PCI_VMPEND	PCIe VM Pending Status	PCIe	329
0x00011540	PCI_DREVID	PCIe Default Revision ID	PCIe	329
0x00011544	PCI_BYTCTH	PCIe Byte Counter High	PCIe	330
0x00011548	PCI_BYTCTL	PCIe Byte Counter Low	PCIe	330
0x00011720	PCI_LATCT	PCIe Latency Counter	PCIe	330
0x00011734	PCI_LCBDATA	PCIe LCB Data Port	PCIe	330
0x00011740	PCI_PKTCT	PCIe Packet Counter	PCIe	330
0x00011788	PCI_LCBADD	PCIe LCB Address Port	PCIe	330
0x00011800	PCI_GSCL_1	PCIe Statistic Control Register #1	PCIe	331
0x00011804	PCI_GSCL_2	PCIe Statistic Control Registers #2	PCIe	332
0x00011810 + 0x4*n, n=0...3	PCI_GSCL_5_8[n]	PCIe Statistic Control Register #5...#8	PCIe	332
0x00011820 + 0x4*n, n=0...3	PCI_GSCN_0_3[n]	PCIe Statistic Counter Registers #0...#3	PCIe	333
PCIe Configuration Space Setting Registers				
0x00011000	PCI_CNF	PCIe PF Configuration	PCIe_GLUE	333
0x00011008	PCI_PFDEVID	PCIe PF Device ID	PCIe_GLUE	333
0x00011010	PCI_VFDEVID	PCIe VF Device ID	PCIe_GLUE	334
0x00011038	PCI_CLASS	PCIe Storage Class	PCIe_GLUE	334
0x0001103C	PCI_VENDORID	PCIe Vendor ID	PCIe_GLUE	334
0x00011048	PCI_LBARCTRL	PCI BAR Control	PCIe_GLUE	334
0x00011058	PCI_SUBSYSID	PCIe Subsystem ID	PCIe_GLUE	335
0x00011060	PCI_PWRDATA	PCIe Power Data Register	PCIe_GLUE	335
0x00011078	PCI_SERH	PCIe Serial Number MAC Address High	PCIe_GLUE	335
0x00011080	PCI_CAPCTRL	PCIe Capabilities Control	PCIe_GLUE	335
0x00011088	PCI_CAPSUP	PCIe Capabilities Support	PCIe_GLUE	335
0x00011090	PCI_LINKCAP	PCIe Link Capabilities	PCIe_GLUE	336
0x000110A0	PCI_PMSUP	PCIe PM Support	PCIe_GLUE	337
0x000110B8	PCI_GLBL_CNF	PCIe Global Config	PCIe_GLUE	337

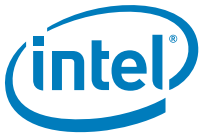


Table 7.3. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Page
0x000110E8	PCI_UPADD	PCIe Upper Address	PCIe_GLUE	337
0x000110F0	PCI_SERL	PCIe Serial Number MAC Address Low	PCIe_GLUE	338
0x000110F8	PCI_CNF2	PCIe Global Config 2	PCIe_GLUE	338
Interrupt Registers				
0x00000800	EICR	Extended Interrupt Cause Register	Interrupt	338
0x00000808	EICS	Extended Interrupt Cause Set Register	Interrupt	339
0x00000810	EIAC	Extended Interrupt Auto Clear Register	Interrupt	340
0x00000820 + 0x4*n, n=0...23 and 0x00012300 + 0x4*(n-24), n=24...128	EITR[n]	Extended Interrupt Throttle Registers	Interrupt	340
0x00000880	EIMS	Extended Interrupt Mask Set/Read Register	Interrupt	341
0x00000888	EIMC	Extended Interrupt Mask Clear Register	Interrupt	341
0x00000890	EIAM	Extended Interrupt Auto Mask Enable register	Interrupt	341
0x00000894	EITRSEL	MSIX to EITR Select	Interrupt	341
0x00000898	GPIE	General Purpose Interrupt Enable	Interrupt	341
0x00000900 + 0x4*n, n=0...63	IVAR[n]	Interrupt Vector Allocation Registers	Interrupt	342
0x00000A00	IVAR_MISC	Miscellaneous Interrupt Vector Allocation	Interrupt	343
0x00000A90	EICS1	Extended Interrupt Cause Set Registers 1	Interrupt	343
0x00000A94	EICS2	Extended Interrupt Cause Set Registers 2	Interrupt	343
0x00000AA0	EIMS1	Extended Interrupt Mask Set/Read Registers	Interrupt	343
0x00000AA4	EIMS2	Extended Interrupt Mask Set/Read Registers	Interrupt	343
0x00000AB0	EIMC1	Extended Interrupt Mask Clear Registers 1	Interrupt	344
0x00000AB4	EIMC2	Extended Interrupt Mask Clear Registers 2	Interrupt	344
0x00000AD0	EIAM1	Extended Interrupt Auto Mask Enable registers 1	Interrupt	344
0x00000AD4	EIAM2	Extended Interrupt Auto Mask Enable registers 2	Interrupt	344
0x00012000 + 0x4*n, n=0...128	RSCINT[n]	RSC Enable Interrupt	Interrupt	344
MSI-X Table Registers				
0x000110C0 + 0x4*n, n=0...1	PBACL[n]	MSI-X PBA Clear	PCIe	345
0x000110C8 + 0x4*n, n=0...5	VFPBACL[n]	VF MSI-X PBA Clear	PCIe	345
Receive Registers				
0x00005000	RXCSUM	Receive Checksum Control	RX_Filter	345
0x00005008	RFCTL	Receive Filter Control Register	RX_Filter	346
0x0000507C	VXLANCTRL	VXLAN control	RX_Filter	346
0x00005080	FCTRL	Filter Control Register	RX_Filter	346
0x00005084	ETAG_ETYPE	E-tag Ethertype Register	RX_Filter	347
0x00005088	VLNCTRL	VLAN Control Register	RX_Filter	347

Table 7.3. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Page
0x00005090	MCSTCTRL	Multicast Control Register	RX_Filter	347
0x00005128 + 0x4*n, n=0...7	ETQF[n]	EType Queue Filter	RX_Filter	348
0x00005200 + 0x4*n, n=0...127	MTA[n]	Multicast Table Array	RX_Filter	348
0x00005400 + 0x8*n, n=0...15	RAL_ALIAS[n]	Receive Address Low	RX_Filter	348
0x00005404 + 0x8*n, n=0...15	RAH_ALIAS[n]	Receive Address High	RX_Filter	348
0x00005818	MRQC_ALIAS	Multiple Receive Queues Command Register	DBU_RX	349
0x0000A000 + 0x4*n, n=0...127	VFTA[n]	VLAN Filter Table Array	RX_Filter	349
0x0000A200 + 0x8*n, n=0...127	RAL[n]	Receive Address Low	RX_Filter	349
0x0000A204 + 0x8*n, n=0...127	RAH[n]	Receive Address High	RX_Filter	349
0x0000A600 + 0x4*n, n=0...255	MPSAR[n]	MAC Pool Select Array	RX_Filter	350
0x0000EA00 + 0x4*n, n=0...63	PSRTYPE[n]	Packet Split Receive Type Register	DBU_RX	350
0x0000EB00 + 0x4*n, n=0...31	RETA[n]	Redirection Table	DBU_RX	351
0x0000EB80 + 0x4*n, n=0...9	RSSRK[n]	RSS Random Key Register	DBU_RX	351
0x0000EC00 + 0x4*n, n=0...7	ETQS[n]	EType Queue Select	DBU_RX	352
0x0000EC30	SYNQF	SYN Packet Queue Filter	DBU_RX	352
0x0000EC70	RQTC	RSS Queues per Traffic Class Register	DBU_RX	352
0x0000EC80	MRQC	Multiple Receive Queues Command Register	DBU_RX	353
0x0000EE80 + 0x4*n, n=0...95	ERETA[n]	Extended Redirection Table	DBU_RX	354
0x00018000 + 0x4*n + 0x40*m, n=0...15, m=0...63	VFRSSRK[n,m]	Per Pool RSS Random Key Register	DBU_RX	354
0x00019000 + 0x4*n + 0x40*m, n=0...15, m=0...63	VFRETA[n,m]	Per Pool Redirection Table	DBU_RX	355
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	355
0x00001004 + 0x40*n, n=0...63 and 0x0000D004 + 0x40*(n-64), n=64...127	RDBAH[n]	Receive Descriptor Base Address High	DMA_RX	355
0x00001008 + 0x40*n, n=0...63 and 0x0000D008 + 0x40*(n-64), n=64...127	RDLEN[n]	Receive Descriptor Length	DMA_RX	356

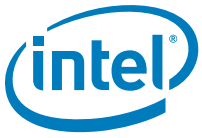


Table 7.3. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Page
0x00001010 + 0x40*n, n=0...63 and 0x0000D010 + 0x40*(n-64), n=64...127	RDH[n]	Receive Descriptor Head	DMA_RX	356
0x00001014 + 0x40*n, n=0...63 and 0x0000D014 + 0x40*(n-64), n=64...127	SRRCTL[n]	Split Receive Control Registers	DMA_RX	356
0x00001018 + 0x40*n, n=0...63 and 0x0000D018 + 0x40*(n-64), n=64...127	RDT[n]	Receive Descriptor Tail	DMA_RX	357
0x00001028 + 0x40*n, n=0...63 and 0x0000D028 + 0x40*(n-64), n=64...127	RXDCTL[n]	Receive Descriptor Control	DMA_RX	357
0x0000102C + 0x40*n, n=0...63 and 0x0000D02C + 0x40*(n-64), n=64...127	RSCCTL[n]	RSC Control	DMA_RX	358
0x00002100 + 0x4*n, n=0...15	SRRCTL_ALIAS[n]	Split Receive Control Registers	DMA_RX	358
0x00002F00	RDRXCTL	Receive DMA Control Register	DMA_RX	358
0x00003000	RXCTRL	Receive Control Register	DBU_RX	359
0x00003C00 + 0x4*n, n=0...7	RXPBSIZE[n]	Receive Packet Buffer Size	DBU_RX	359
Transmit Registers				
0x00004950 + 0x4*n, n=0...7	TXPBTHRESH[n]	Tx Packet Buffer Threshold	DMA_TX	359
0x00004A80	DMATXCTL	DMA Tx Control	DMA_TX	359
0x00004A88	DTXTCPFLGL	DMA Tx TCP Flags Control Low	DMA_TX	360
0x00004A8C	DTXTCPFLGH	DMA Tx TCP Flags Control High	DMA_TX	360
0x00006000 + 0x40*n, n=0...127	TDBAL[n]	Transmit Descriptor Base Address Low	DMA_TX	360
0x00006004 + 0x40*n, n=0...127	TDBAH[n]	Transmit Descriptor Base Address High	DMA_TX	361
0x00006008 + 0x40*n, n=0...127	TDLEN[n]	Transmit Descriptor Length	DMA_TX	361
0x00006010 + 0x40*n, n=0...127	TDH[n]	Transmit Descriptor Head	DMA_TX	362
0x00006018 + 0x40*n, n=0...127	TDT[n]	Transmit Descriptor Tail	DMA_TX	362
0x00006028 + 0x40*n, n=0...127	TXDCTL[n]	Transmit Descriptor Control	DMA_TX	362
0x00006038 + 0x40*n, n=0...127	TDWBAL[n]	Tx Descriptor Completion Write Back Address Low	DMA_TX	363
0x0000603C + 0x40*n, n=0...127	TDWBAH[n]	Tx Descriptor Completion Write Back Address High	DMA_TX	363
0x00008100	DTXMXSZRQ	DMA Tx TCP Max Allow Size Requests	DMA_TX	364
0x00008120	MTQC	Multiple Transmit Queues Command Register	DMA_TX	364
0x0000CC00 + 0x4*n, n=0...7	TXPBSIZE[n]	Transmit Packet Buffer Size	DBU_TX	364

Table 7.3. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Page
0x0000CD10	MNGTXMAP	Manageability Transmit TC Mapping	DBU_TX	364
0x00017100	TAG_ETYPE	Tags Ethertypes	DMA_TX	365
Timers Registers				
0x0000004C	TCPTIMER	TCP Timer	Target	365
Flow Director Registers				
0x0000EE00	FDIRCTRL	Flow Director Filters Control Register	DBU_RX	366
0x0000EE0C + 0x4*n, n=0...2	FDIRSIPV6[n]	Flow Director Filters Source IPv6	DBU_RX	367
0x0000EE18	FDIRIPSA	Flow Director Filters IP SA	DBU_RX	367
0x0000EE1C	FDIRIPDA	Flow Director Filters IP DA	DBU_RX	367
0x0000EE20	FDIRPORT	Flow Director Filters Port	DBU_RX	367
0x0000EE24	FDIRVLAN	Flow Director Filters VLAN and FLEX bytes	DBU_RX	368
0x0000EE28	FDIRHASH	Flow Director Filters Hash Signature	DBU_RX	368
0x0000EE2C	FDIRCMD	Flow Director Filters Command Register	DBU_RX	368
0x0000EE38	FDIRFREE	Flow Director Filters Free	DBU_RX	369
0x0000EE3C	FDIRDIP4M	Flow Director Filters DIPv4 Mask	DBU_RX	369
0x0000EE40	FDIRSIP4M	Flow Director Filters Source IPv4 Mask	DBU_RX	370
0x0000EE44	FDIRTCPM	Flow Director Filters TCP Mask	DBU_RX	370
0x0000EE48	FDIRUDPM	Flow Director Filters UDP Mask	DBU_RX	370
0x0000EE4C	FDIRLEN	Flow Director Filters Length	DBU_RX	370
0x0000EE50	FDIRUSTAT	Flow Director Filters Usage Statistics	DBU_RX	371
0x0000EE54	FDIRFSTAT	Flow Director Filters Failed Usage Statistics	DBU_RX	371
0x0000EE58	FDIRMATCH	Flow Director Filters Match Statistics	DBU_RX	371
0x0000EE5C	FDIRMISS	Flow Director Filters Miss Match Statistics	DBU_RX	371
0x0000EE68	FDIRHKEY	Flow Director Filters Lookup Table HASH Key	DBU_RX	371
0x0000EE6C	FDIRSKEY	Flow Director Filters Signature HASH Key	DBU_RX	371
0x0000EE70	FDIRM	Flow Director Filters Other Mask	DBU_RX	372
0x0000EE74	FDIRIP6M	Flow Director Filters IPv6 Mask	DBU_RX	372
0x0000EE78	FDIRSCTPM	Flow Director Filters SCTP Mask	DBU_RX	373
MAC Registers				
0x00004240	HLREG0	Highlander Control 0 Register	MAC	373
0x00004244	HLREG1	Highlander Status 1 Register	MAC	374
0x00004248	PAP	Pause and Pace Register	MAC	375
0x0000425C	MSCA	MDI Single Command and Address ***Note:*** Values in this register should not be changed.	MAC	
0x00004260	MSRWD	MDI Single Read and Write Data ***Note:*** Values in this register should not be changed.	MAC	
0x00004268	MAXFRS	Max Frame Size	MAC	375
0x000042A4	LINKS	Link Status Register	MAC	375
0x000042D0	MMNGC	MAC Manageability Control Register	MAC	376

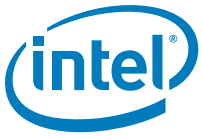


Table 7.3. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Page
0x00004330	MACC	MAC Control register	MAC	376
0x00011144	PHY_INDIRECT_CTRL	PHY Indirect Access Control	PCIe_GLUE	378
0x00011148	PHY_INDIRECT_DATA	PHY Indirect Access Data	PCIe_GLUE	378
Statistic Registers				
0x00001030 + 0x40*n, n=0...15	QPRC[n]	Queue Packets Received Count	DMA_RX	379
0x00001034 + 0x40*n, n=0...15	QBRC_L[n]	Queue Bytes Received Count Low	DMA_RX	379
0x00001038 + 0x40*n, n=0...15	QBRC_H[n]	Queue Bytes Received Count High	DMA_RX	380
0x00001430 + 0x40*n, n=0...15	QPRDC[n]	Queue Packets Received Drop Count	DMA_RX	380
0x00002300 + 0x4*n, n=0...31	RQSMR[n]	Receive Queue Statistic Mapping Registers	DMA_RX	380
0x00002F40	RXDSTATCTRL	Rx DMA Statistic Counter Control	DMA_RX	381
0x00002F50	RXDGPC	DMA Good Rx Packet Counter	DMA_RX	381
0x00002F54	RXDGBCL	DMA Good Rx Byte Counter Low	DMA_RX	381
0x00002F58	RXDGBCH	DMA Good Rx Byte Counter High	DMA_RX	381
0x00002F5C	RXDDPC	DMA Duplicated Good Rx Packet Counter	DMA_RX	381
0x00002F60	RXDDBCL	DMA Duplicated Good Rx Byte Counter Low	DMA_RX	381
0x00002F64	RXDDBCH	DMA Duplicated Good Rx Byte Counter High	DMA_RX	382
0x00002F68	RXLPBKPC	DMA Good Rx LPBK Packet Counter	DMA_RX	382
0x00002F6C	RXLPBKBCL	DMA Good Rx LPBK Byte Counter Low	DMA_RX	382
0x00002F70	RXLPBKBCH	DMA Good Rx LPBK Byte Counter High	DMA_RX	382
0x00002F74	RXDLPBKPC	DMA Duplicated Good Rx LPBK Packet Counter	DMA_RX	382
0x00002F78	RXDLPBKBCL	DMA Duplicated Good Rx LPBK Byte Counter Low	DMA_RX	382
0x00002F7C	RXDLPBKBCH	DMA Duplicated Good Rx LPBK Byte Counter High	DMA_RX	382
0x00002F90	B2OGPRC	BMC2OS Packets Received by Host	DMA_RX	383
0x00004000	CRCERRS	CRC Error Count	STAT	383
0x00004004	ILLERRC	Illegal Byte Error Count	STAT	383
0x00004008	ERRBC	Error Byte Count	STAT	383
0x00004010	MSPDC	MAC short Packet Discard Count	STAT	383
0x00004018	MBSDC	Bad SFD Count	STAT	383
0x00004034	MLFC	MAC Local Fault Count	STAT	383
0x00004038	MRFC	MAC Remote Fault Count	STAT	384
0x0000403C	LINK_DN_CNT	LINK Down Counter	STAT	384
0x00004040	RLEC	Receive Length Error Count	STAT	384
0x00004044	ERR_PKT_VLAN_MIS_CNT	Error Packets VLAN Mismatch Counter	STAT	384
0x00004048	ERR_PKT_ADD_MIS_CNT	Error Packet Address Mismatch Counter	STAT	384
0x0000405C	PRC64	Packets Received [64 Bytes] Count	STAT	384

Table 7.3. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Page
0x00004060	PRC127	Packets Received [65-127 Bytes] Count	STAT	385
0x00004064	PRC255	Packets Received [128-255 Bytes] Count	STAT	385
0x00004068	PRC511	Packets Received [256-511 Bytes] Count	STAT	385
0x0000406C	PRC1023	Packets Received [512-1023 Bytes] Count	STAT	385
0x00004070	PRC1522	Packets Received [1024 to Max Bytes] Count	STAT	385
0x00004074	GPRC	Good Packets Received Count	STAT	385
0x00004078	BPRC	Broadcast Packets Received Count	STAT	386
0x0000407C	MPRC	Multicast Packets Received Count	STAT	386
0x00004080	GPTC	Good Packets Transmitted Count	STAT	386
0x00004088	GORCL	Good Octets Received Count Low	STAT	386
0x0000408C	GORCH	Good Octets Received Count High	STAT	386
0x00004090	GOTCL	Good Octets Transmitted Count Low	STAT	386
0x00004094	GOTCH	Good Octets Transmitted Count High	STAT	386
0x000040A4	RUC	Receive Undersize Count	STAT	387
0x000040A8	RFC	Receive Fragment Count	STAT	387
0x000040AC	ROC	Receive Oversize Count	STAT	387
0x000040B0	RJC	Receive Jabber Count	STAT	387
0x000040B4	MNGPRC	Management Packets Received Count	STAT	387
0x000040B8	MNGPDC	Management Packets Dropped Count	STAT	388
0x000040C0	TORL	Total Octets Received Low	STAT	388
0x000040C4	TORH	Total Octets Received High	STAT	388
0x000040D0	TPR	Total Packets Received	STAT	388
0x000040D4	TPT	Total Packets Transmitted	STAT	388
0x000040D8	PTC64	Packets Transmitted (64 Bytes) Count	STAT	389
0x000040DC	PTC127	Packets Transmitted [65-127 Bytes] Count	STAT	389
0x000040E0	PTC255	Packets Transmitted [128-255 Bytes] Count	STAT	389
0x000040E4	PTC511	Packets Transmitted [256-511 Bytes] Count	STAT	389
0x000040E8	PTC1023	Packets Transmitted [512-1023 Bytes] Count	STAT	389
0x000040EC	PTC1522	Packets Transmitted [Greater than 1024 Bytes] Count	STAT	389
0x000040F0	MPTC	Multicast Packets Transmitted Count	STAT	390
0x000040F4	BPTC	Broadcast Packets Transmitted Count	STAT	390
0x00004120	XEC	XSUM Error Count	STAT	390
0x00004140 + 0x4*n, n=0...7	PXONRXCNT[n]	Priority XON Received Count	STAT	390
0x00004160 + 0x4*n, n=0...7	PXOFFRXCNT[n]	Priority XOFF Received Count	STAT	390
0x00004180	BUPRC	Total Unicast Packets Received (BMC copy)	STAT	391
0x00004184	BMPRC	BMC Total Multicast Packets Received	STAT	391
0x00004188	BBPRC	Total Broadcast Packets Received (BMC copy)	STAT	391
0x0000418C	BUPTC	Total Unicast Packets Transmitted (BMC copy)	STAT	391
0x00004190	BMPTC	BMC Total Multicast Packets Transmitted	STAT	391

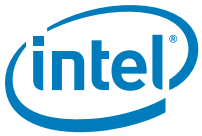


Table 7.3. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Page
0x00004194	BBPTC	Total Broadcast Packets Transmitted (BMC copy)	STAT	392
0x00004198	BCRCERRS	BMC FCS Receive Errors	STAT	392
0x0000419C	BXONRXC	BMC Pause XON Frames Received	STAT	392
0x000041A4	LXONRXCNT	Link XON Received Count	STAT	392
0x000041A8	LXOFFRXCNT	Link XOFF Received Count	STAT	393
0x000041B0	RXNFGPC	Good Rx Non-Filtered Packet Counter	STAT	393
0x000041B4	RXNFGBCL	Good Rx Non-Filter Byte Counter Low	STAT	393
0x000041B8	RXNFGBCH	Good Rx Non-Filter Byte Counter High	STAT	393
0x000041C0	B2OSPC	BMC2OS Packets Sent by BMC	STAT	393
0x000041C4	O2BGPTC	OS2BMC Packets Received by BMC	STAT	393
0x000041E0	BXOFFRXC	BMC Pause XOFF Frames Received	STAT	394
0x000041E4	BXONTXC	BMC Pause XON Frames Transmitted	STAT	394
0x000041E8	BXOFFTXC	BMC Pause XOFF Frames Transmitted	STAT	394
0x000041F0	B2OSDPC	Sideband Receive Dropped Packet Count	STAT	394
0x00005118	FCCRC	Fiber Channel CRC Error Count	RX_Filter	395
0x00006030 + 0x40*n, n=0...15	QPTC_ALIAS[n]	Queue Packets Transmitted Count	DMA_TX	395
0x00008600 + 0x4*n, n=0...31	TQSM[n]	Transmit Queue Statistic Mapping Registers	DMA_TX	395
0x00008680 + 0x4*n, n=0...15	QPTC[n]	Queue Packets Transmitted Count	DMA_TX	395
0x00008700 + 0x8*n, n=0...15	QBTC_L[n]	Queue Bytes Transmitted Count Low	DMA_TX	396
0x00008704 + 0x8*n, n=0...15	QBTC_H[n]	Queue Bytes Transmitted Count High	DMA_TX	396
0x00008780	SSVPC	Switch Security Violation Packet Count	DMA_TX	396
0x000087A0	TXDGPC	DMA Good Tx Packet Counter	DMA_TX	396
0x000087A4	TXDGBCL	DMA Good Tx Byte Counter Low	DMA_TX	396
0x000087A8	TXDGBCH	DMA Good Tx Byte Counter High	DMA_TX	396
0x000087B0	O2BSPC	OS2BMC Packets Transmitted by Host	DMA_TX	397
Wake-Up and Proxy Control Registers				
0x00005800	WUC	Wake Up Control Register	RX_Filter	397
0x00005808	WUFC	Wake Up Filter Control Register	RX_Filter	397
0x00005810	WUS	Wake Up Status Register	RX_Filter	398
0x00005838	IPAV	IP Address Valid	RX_Filter	399
0x00005840 + 0x8*n, n=0...3	IP4AT[n]	IPv4 Address Table	RX_Filter	399
0x00005880 + 0x4*n, n=0...3	IP6AT[n]	IPv6 Address Table	RX_Filter	400
0x00005990 + 0x4*n, n=0...11	IP6AT_EXT[n]	IPv6 Address Table Extended	RX_Filter	400
0x00005A00 + 0x4*n, n=0...31	WUPM[n]	Wake Up Packet Memory (128 Bytes)	RX_Filter	400

Table 7.3. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Page
0x00005F60	PROXYS	Proxying Status Register	RX_Filter	400
0x00005F64	PROXYFC	Proxying Filter Control Register	RX_Filter	401
0x00009000 + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x00009600 + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3	FHFT_FILTER_DW_EVEN[n,m]	Filter DW Even	RX_Filter	402
0x00009004 + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x00009604 + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3	FHFT_FILTER_DW_ODD[n,m]	Filter DW Odd	RX_Filter	402
0x00009008 + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x00009608 + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3	FHFT_FILTER_MASK[n,m]	Filter Mask	RX_Filter	402
0x0000900C + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x0000960C + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3	FHFT_FILTER_LENGTH[n,m]	Filter Length	RX_Filter	402
Management Filters Registers				
0x00005010 + 0x4*n, n=0...7	MAVTV[n]	Management VLAN TAG Value	RX_Filter	402
0x00005030 + 0x4*n, n=0...7	MFUTP[n]	Management Flex UDP/TCP Ports	RX_Filter	403
0x00005050 + 0x4*n, n=0...3	BMCIP[n]	BMC IP address Register	RX_Filter	403
0x00005060	BMCIPVAL	BMC IP Valid Register	RX_Filter	403
0x00005160 + 0x4*n, n=0...7	MDEF_EXT[n]	Manageability Decision Filters Ext	RX_Filter	403
0x00005190 + 0x4*n, n=0...3	METF[n]	Management Ethernet Type Filters	RX_Filter	404
0x00005820	MANC	Management Control Register	RX_Filter	405
0x00005864	MNGONLY	Manageability Only Traffic	RX_Filter	406
0x00005890 + 0x4*n, n=0...7	MDEF[n]	Manageability Decision Filters	RX_Filter	406
0x000058B0 + 0x4*n + 0x10*m, n=0...3, m=0...3	MIPAF[n,m]	Manageability IP Address Filter	RX_Filter	407
0x00005910 + 0x8*n, n=0...3	MMAL[n]	Manageability Ethernet MAC Address Low	RX_Filter	407
0x00005914 + 0x8*n, n=0...3	MMAH[n]	Manageability Ethernet MAC Address High	RX_Filter	408
0x00009400 + 0x10*n, n=0...15	FTFT_FILTER_EVEN[n]	FTFT Filter DW Even Words	RX_Filter	408
0x00009404 + 0x10*n, n=0...15	FTFT_FILTER_ODD[n]	FTFT Filter DW Odd Words	RX_Filter	408
0x00009408 + 0x10*n, n=0...15	FTFT_FILTER0_MASK[n]	FTFT Filter Mask	RX_Filter	408

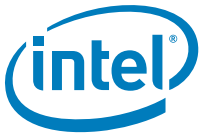


Table 7.3. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Page
0x0000940C + 0x10*n, n=0...15	FTFT_FILTER0_LENGTH[n]	FTFT Filter Length	RX_Filter	409
Manageability (ARC subsystem) HOST Interface				
0x00015800 + 0x4*n, n=0...447	ARCRAM[n]	Host ARC Data RAM	MNG	409
0x00015F00	HICR	HOST Interface Control Register	MNG	409
0x00015F40	FWRESETCNT	Firmware Resets Count	MNG	409
0x00015F70	SWSM	Software Semaphore Register	MNG	410
0x00015F74	FWSM	Firmware Semaphore Register	MNG	410
0x00015F78	SW_FW_SYNC	Software-Firmware Synchronization	MNG	411
0x00015F7C	SW_FW_SYNC_MIRR	Software-Firmware Synchronization	MNG	412
Time Sync (IEEE 1588) Registers				
0x0000003C	TSSDP	Time Sync SDP Configuration Register	Target	412
0x00005120	RXMTRL	Rx Message Type Register Low	RX_Filter	413
0x00005188	TSYNCRXCTL	Rx Time Sync Control register	RX_Filter	414
0x000051A4	RXSTMPH	Rx timestamp High	RX_Filter	414
0x000051E8	RXSTMPL	Rx timestamp Low	RX_Filter	414
0x00008C00	TSYNCTXCTL	Tx Time Sync Control Register	SEC_TX	414
0x00008C04	TXSTMPL	Tx Timestamp Value Low	SEC_TX	415
0x00008C08	TXSTMPH	Tx Timestamp Value High	SEC_TX	415
0x00008C0C	SYSTIMEL	System Time Register Low	SEC_TX	415
0x00008C10	SYSTIMEH	System Time Register High	SEC_TX	415
0x00008C14	TIMEINCA	Increment Attributes Register	SEC_TX	415
0x00008C18	TIMADJ	Time Adjustment Offset Register	SEC_TX	415
0x00008C20	TSAUXC	TimeSync Auxiliary Control Register	SEC_TX	415
0x00008C24	TRGTTIMEL0	Target Time Register 0 Low	SEC_TX	417
0x00008C28	TRGTTIMEH0	Target Time Register 0 High	SEC_TX	417
0x00008C2C	TRGTTIMEL1	Target Time Register 1 Low	SEC_TX	417
0x00008C30	TRGTTIMEH1	Target Time Register 1 High	SEC_TX	417
0x00008C34	FREQOUT0	Frequency Out 0 Control Register	SEC_TX	417
0x00008C38	FREQOUT1	Frequency Out 1 Control Register	SEC_TX	417
0x00008C3C	AUXSTMPL0	Auxiliary Time Stamp 0 Register Low	SEC_TX	417
0x00008C40	AUXSTMPH0	Auxiliary Time Stamp 0 Register High	SEC_TX	418
0x00008C44	AUXSTMPL1	Auxiliary Time Stamp 1 Register Low	SEC_TX	418
0x00008C48	AUXSTMPH1	Auxiliary Time Stamp 1 Register High	SEC_TX	418
0x00008C58	SYSTIMR	System Time Register residue	SEC_TX	418
0x00008C60	TSICR	Time Sync Interrupt Cause Register	SEC_TX	418
0x00008C68	TSIM	Time Sync Interrupt Mask Register	SEC_TX	419
Virtualization PF Registers				
0x00000700 + 0x4*n, n=0...1	PFVFLREC[n]	PF VFLR Events Clear	Target	419

Table 7.3. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Page
0x00000710 + 0x4*n, n=0...3	PFMBICR[n]	PF Mailbox Interrupt Causes Register	Target	419
0x00000720 + 0x4*n, n=0...1	PFMBIMR[n]	PF Mailbox Interrupt Mask Register	Target	420
0x00002F04	PFQDE	PF Queue Drop Enable Register	DMA_RX	420
0x00002FA4	LMVM_RX	Last Malicious VM - RX	DMA_RX	420
0x00002FA8	LVMMC_RX	Last VM Misbehavior Cause - RX	DMA_RX	421
0x00002FB0 + 0x4*n, n=0...3	WQBR_RX[n]	Wrong Queue Behavior Register - Rx	DMA_RX	421
0x00004B00 + 0x4*n, n=0...63	PFMAILBOX[n]	PF Mailbox	Target	421
0x000050B0	PFFLPL	Filter Local Packets Low	RX_Filter	421
0x000050B4	PFFLPH	Filter Local Packets High	RX_Filter	422
0x00005180 + 0x4*n, n=0...1	PFVMTXSW[n]	PF VM Tx Switch Loopback Enable	RX_Filter	422
0x000051B0	PFVTCTL	PF Virtual Control Register	RX_Filter	422
0x000051E0 + 0x4*n, n=0...1	PFVFRE[n]	PF VF Receive Enable	RX_Filter	423
0x00008000 + 0x4*n, n=0...63	PFVMVIR[n]	PF VM VLAN Insert Register	DMA_TX	423
0x00008108	LVMMC_TX	Last VM Misbehavior Cause - TX	DMA_TX	423
0x00008110 + 0x4*n, n=0...1	PFVFTE[n]	PF VF Transmit Enable	DMA_TX	424
0x00008124	LMVM_TX	Last Malicious VM - TX	DMA_TX	424
0x00008130 + 0x4*n, n=0...3	WQBR_TX[n]	Wrong Queue Behavior Register - Tx	DMA_TX	425
0x00008200 + 0x4*n, n=0...7	PFVFSPOOF[n]	PFVF Anti Spoof Control	DMA_TX	425
0x00008220	PFDTXGSWC	PF DMA Tx General Switch Control	DMA_TX	425
0x00008790	PFVMECM0	PF VM 0:31 Error Count Mask	DMA_TX	425
0x00008794	PFVMECM1	PF VM 32:63 Error Count Mask	DMA_TX	425
0x0000F000 + 0x4*n, n=0...63	PFVML2FLT[n]	PF VM L2Control Register	RX_Filter	426
0x0000F100 + 0x4*n, n=0...63	PFVLVF[n]	PF VM VLAN Pool Filter	RX_Filter	426
0x0000F200 + 0x4*n, n=0...127	PFVLVFB[n]	PF VM VLAN Pool Filter Bitmap	RX_Filter	426
0x0000F400 + 0x4*n, n=0...127	PFUTA[n]	PF Unicast Table Array	RX_Filter	427
0x0000F600 + 0x4*n, n=0...3	PFMRCTL[n]	PF Mirror Rule Control	RX_Filter	427
0x0000F610 + 0x4*n, n=0...7	PFMRVLAN[n]	PF Mirror Rule VLAN	RX_Filter	427
0x0000F630 + 0x4*n, n=0...7	PFMRVM[n]	PF Mirror Rule Pool	RX_Filter	427
0x00017000 + 0x4*n, n=0...63	PFVMTIR[n]	PF VM Tag Insert Register	DMA_TX	428
Power Management				

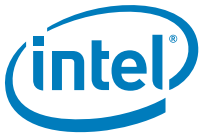


Table 7.3. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Page
0x00002400	DMACR	DMA Coalescing Control Register	DMA_RX	428
0x00002404	DMCTLX	DMA Coalescing Time to Lx Request	DMA_RX	429
0x00003300 + 0x4*n, n=0...7	DMCTH[n]	DMA Coalescing Threshold	DBU_RX	429
0x000041F4	TLPIC	EEE TX LPI Count	STAT	429
0x000041F8	RLPIC	EEE RX LPI Count	STAT	429
0x00004380	EEE_SU	Energy Efficient Ethernet (EEE) Setup Register	MAC	430
0x00004398	EEE_STAT	Energy Efficient Ethernet (EEE) STATUS	MAC	430
0x000043A0	EEER	Energy Efficient Ethernet (EEE) Register	MAC	431
0x00011708	LTRC	Latency Tolerance Reporting (LTR) Control	PCIe	432
0x00015F20	DMCMNGTH	DMA Coalescing Management Threshold	MNG	433
VF Registers Mapping in the PF space				
0x00000300 + 0x4*n, n=0...63	VFCTRL[n]	VF Control Register	Target	433
0x00000B00 + 0x4*n, n=0...63	VFEICR[n]	VF Extended Interrupt Cause	Interrupt	433
0x00000C00 + 0x4*n, n=0...63	VFEICS[n]	VF Extended Interrupt Cause Set	Interrupt	433
0x00000D00 + 0x4*n, n=0...63	VFEIMS[n]	VF Extended Interrupt Mask Set/Read	Interrupt	433
0x00000E00 + 0x4*n, n=0...63	VFEIMC[n]	VF Extended Interrupt Mask Clear	Interrupt	433
0x0000101C + 0x40*n, n=0...63	VFGPRC[n]	VF Good Packets Received Count	DMA_RX	433
0x00001020 + 0x40*n, n=0...63	VFGORC_LSB[n]	VF Good Octets Received Count Low	DMA_RX	433
0x00003400 + 0x4*n, n=0...63	VFMRQC[n]	VF Multiple Receive Queues Command Register	DBU_RX	434
0x00004C00 + 0x4*n, n=0...63	VFMAILBOX[n]	VF Mailbox	Target	434
0x00004D00 + 0x4*n, n=0...63	VFEIAM[n]	VF Extended Interrupt Auto Mask Enable	Interrupt	434
0x00004E00 + 0x4*n, n=0...63	VFIVAR_MISC[n]	VF Interrupt Vector Allocation Registers Misc	Interrupt	434
0x00008300 + 0x4*n, n=0...63	VFGPTC[n]	VF Good Packets Transmitted Count	STAT	434
0x00008400 + 0x8*n, n=0...63	VFGOTC_LSB[n]	VF Good Octets Transmitted Count LSB	STAT	434
0x00008404 + 0x8*n, n=0...63	VFGOTC_MSB[n]	VF Good Octets Transmitted Count MSB	STAT	434
0x0000D01C + 0x40*n, n=0...63	VFMPRC[n]	VF Multicast Packets Received Count	DMA_RX	434
0x0000D020 + 0x40*n, n=0...63	VFGORC_MSB[n]	VF Good Octets Received Count High	DMA_RX	435
0x00012500 + 0x4*n, n=0...63	VFIVAR[n]	VF Interrupt Vector Allocation Registers	Interrupt	435

Table 7.3. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Page
0x00013000 + 0x4*n + 0x40*m, n=0...15, m=0...63	PFMBMEM[n,m]	PF Mailbox Memory	Target	435
MNG_IOSF_SB				
0x00010020	MNGSB_MSGCTL	MNG SB-IOSF Request Control-Status	MNG	435
0x00010024	MNGSB_RSPCTL	MNG SB-IOSF Response Control-Status	MNG	436
0x00010030	MNGSB_DADD	MNG SB-IOSF DMA Address	MNG	436
0x00010034	MNGSB_DCNT	MNG SB-IOSF Auto-Inc Counter	MNG	436
0x000100F4	MNGSB_WHDR0	MNG SB-IOSF Request Hdr0	MNG	436
0x000100F8	MNGSB_WHDR1	MNG SB-IOSF Request Hdr1	MNG	437
0x000100FC	MNGSB_WHDR2	MNG SB-IOSF Request Hdr2	MNG	437
0x00010100	MNGSB_WDATA	MNG SB-IOSF Write Data	MNG	437
0x000102FC	MNGSB_RHDR0	MNG SB-IOSF Response Hdr0	MNG	437
0x00010300	MNGSB_RDATA	MNG SB-IOSF Read Data	MNG	438

7.2.2 Detailed Register Description - PF BAR0

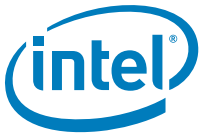
7.2.2.1 General Control Registers

7.2.2.1.1 Device Control Register - CTRL (0x00000000) Target

CTRL is also mapped to address 0x00004 to maintain compatibility with predecessors.

Note: LRST and RST can be used to globally reset the entire 10GBase-T Controller. This register is provided primarily as a software mechanism to recover from an indeterminate or suspected hung hardware state. Most registers (receive, transmit, interrupt, statistics, etc.) and state machines are set to their power-on reset values, approximating the state following a power-on or PCI reset. However, PCIe configuration registers are not reset, thereby leaving the device mapped into system memory space and accessible by a software device driver. To ensure that a global device reset has fully completed and that the device responds to subsequent accesses, programmers must wait approximately 1 ms after setting before attempting to check if the bit has cleared or to access (read or write) any other device register.

Field	Bit(s)	Init.	Type	Description
RESERVED	1:0	0x0	RSV	Reserved.
PCIE_MASTER_DISABLE	2	0b	RW	When set, the device blocks new master requests, including manageability requests, by using this function. Once no master requests are pending by using this function, the PCIe Master Enable Status bit is cleared. ***Note:*** After any change to this bit, the host must read that the bit has been modified as expected before reading STATUS.*PCIE_MASTER_ENABLE_STATUS* bit.
LRST	3	0b	RW	Link Reset. This bit performs a reset of the MAC, PHY, and the entire Controller (software reset), resulting in a state nearly approximating the state following a power-up reset or internal P-IOSF reset, except for the system PCI configuration. Normally 0b. Writing 1b initiates the reset. This bit is self-clearing. Also referred to as MAC reset.



Field	Bit(s)	Init.	Type	Description
RESERVED	25:4	0x0	RSV	Reserved.
RST	26	0b	RW	Device Reset. This bit performs a complete reset of the controller, resulting in a state nearly approximating the state following a power-up reset or internal P-IOSF reset, except for the system PCI configuration. Normally 0b, writing 1b initiates the reset. This bit is self-clearing. Also referred to as a software reset or global reset.
RESERVED	31:27	0x0	RSV	Reserved.

7.2.2.1.2 Device Status Register - STATUS (0x00000008) Target

Field	Bit(s)	Init.	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.
RESERVED	6:4	0x0	RW	Reserved.
LINKUP	7	0b	RW	Linkup Status Indication. This bit is useful for IOV mode. The PF software driver sets it according to the LINKS register and PHY state. It is reflected in the VFSTATUS register indicating link up to the VF drivers.
RESERVED	9:8	0x0	RSV	Reserved.
NUM_VFS	17:10	0x0	RO	Num VFs. This field reflects the value of the Num VFs in the IOV capability structure (note that bit 17 is always 0b).
IOV_ACTIVE	18	0b	RO	IOV Active. This bit reflects the value of the VF Enable (VFE) bit in the IOV Control/Status register.
PCIE_MASTER_ENABLE_STATUS	19	1b	RO	This is a status bit of the appropriate CTRL.*PCIE_MASTER_DISABLE* bit. 0 = Associated LAN function does not issue any master request and all previously issued requests are complete. 1 = Associated LAN function can issue master requests.
RESERVED	31:20	0x0	RSV	Reserved. Reads as 0b.

7.2.2.1.3 Extended Device Control Register - CTRL_EXT (0x00000018) Target

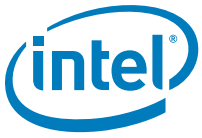
Field	Bit(s)	Init.	Type	Description
RESERVED	13:0	0x00	RSV	Reserved.
PFRSTD	14	0b	SC	PF Reset Done. When set, the *RSTI* bit in all the VFMAILOBOX registers are cleared, and the *RSTD* bit in all the VFMAILOBOX registers are set.
RESERVED	16:15	0x0	RSV	Reserved.
RO_DIS	17	0b	RW	Relaxed Ordering Disable. When this bit is cleared and the Enable Relaxed Ordering bit in the Device Control register is set, the device requests relaxed ordering transactions per queues as configured in the TPH_RXCTRL[n] and TPH_TXCTRL[n] registers. When set to 1b, the device does not request any relaxed ordering transactions.
RESERVED	25:18	0x0	RSV	Reserved.

Field	Bit(s)	Init.	Type	Description
EXTENDED_VLAN	26	0b	RW	Extended VLAN. When set, all incoming Rx packets are expected to have at least one VLAN with the Ether type as defined in the EXVET register. This bit should only be reset by a PCIe reset and should only be changed while Tx and Rx processes are stopped. Should be set to the same value as DMATXCTL.*GDV*.
RESERVED	27	0b	RSV	Reserved.
DRV_LOAD	28	0b	RW	Software Device Driver loaded and the corresponding network interface is enabled. This bit should be set by the software device driver after it is loaded, and cleared when it unloads or at PCIe reset. The Manageability Controller (MC) loads this bit as an indication that the software device driver successfully loaded to it.
RESERVED	31:29	0x0	RSV	Reserved.

7.2.2.1.4 Extended SDP Control - ESDP (0x00000020) Target

This register is initialized only at global reset, preserving the SDP states across software and PCIe resets. Some specific I/O pins are initialized in other resets in native mode as expected for the specific behavior, and described explicitly as follows:

Field	Bit(s)	Init.	Type	Description
SDP0_DATA	0	0b	RW	SDP0 Data Value. Used to read (write) a value of the software-controlled I/O pin SDP0. If SDP0 is configured as an output (*SDP0_IODIR* = 1b), this bit controls the value driven on the pin. If SDP0 is configured as an input, all reads return the current value of the pin. See Notes 1 and 2.
SDP1_DATA	1	0b	RW	SDP1 Data Value. Used to read (write) a value of the software-controlled I/O pin SDP1. If SDP1 is configured as an output (*SDP1_IODIR* = 1b), this bit controls the value driven on the pin. If SDP1 is configured as an input, all reads return the current value of the pin. See Note 1.
RESERVED	7:2	0x0	RSV	Reserved.
SDP0_IODIR	8	0b	RW	SDP0 Pin Directionality. Controls whether software-controlled pin SDP0 is configured as an input or an output. 0 = Input 1 = Output See Notes 1 and 2.
SDP1_IODIR	9	0b	RW	SDP1 Pin Directionality. Controls whether software-controlled pin SDP1 is configured as an input or an output. 0b = Input 1b = Output See Note 1.
RESERVED	15:10	0x0	RSV	Reserved.
SDP0_NATIVE	16	0b	RW	SDP0 Operating Mode. 0 = Generic software controlled I/O by SPD0_DATA and SDP0_IODIR. 1 = Native mode operation (connected to hardware function) is IEEE 1588 functionality. See Note 2.



Field	Bit(s)	Init.	Type	Description
SDP1_NATIVE	17	0b	RW	SDP1 Operating Mode. 0 = Generic software controlled I/O by SPD1_DATA and SDP1_IODIR. 1 = Native mode operation (connected to hardware function) according to the *SDP1_FUNCTION* bit. See Note 1
RESERVED	24:18	0x0	RSV	Reserved.
SDP1_FUNCTION	25	0b	RW	SDP1 Native Mode Functionality (*SDP1_NATIVE* = 1b). 0 = 1588 functionality. SDP1_IODIR should be configured as an output. 1 = Reserved *SDP1_IODIR* should be configured as an output.
RESERVED	31:26	0x0	RSV	Reserved.

Notes:

1. Initial values are read from the shared SPI Flash.
2. It is assumed that bit 15 of NC-SI Configuration 1 word in the shared SPI Flash is cleared. Otherwise, the SDP0 pin of function 0 is used as input pins that encode the NC-SI Package ID of the controller.
3. It is assumed that bit *SDP_FUNC_OFF_EN* of shared SPI Flash Control word 2 is cleared. Otherwise, SDP1 pins are strapped during PE_RST_N to determine that both PCIe functions are disabled

7.2.2.1.5 PHY GPIO Register - PHY_GPIO (0x00000028) Target

Field	Bit(s)	Init.	Type	Description
PHY_GPIO	3:0	0x0	RO	PHY-to-MAC GPIO. Used to read the four internal PHY to MAC general purpose signals.
RESERVED	31:4	0x0	RSV	Reserved.

7.2.2.1.6 MAC GPIO Register - MAC_GPIO (0x00000030) Target

Field	Bit(s)	Init.	Type	Description
MAC_GPIO	3:0	0x0	RW	MAC-to-PHY GPIO. Used to set the four internal MAC-to-PHY general purpose signals.
RESERVED	31:4	0x0	RSV	Reserved.

7.2.2.1.7 PHY Interrupt Status Register 0 - PHYINT_STATUS0 (0x00000100) Target

Register contents is valid once the corresponding EEMNGCTL.*CFG_DONE0/1* bit is asserted by firmware. Bits are set by firmware to notify the host of the PHY interrupts that were triggered. This register is reset by hardware only at power-up events. The host is responsible to clear the PHY interrupts once it completes the PHY interrupt handling routine.

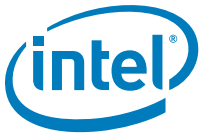
Field	Bit(s)	Init.	Type	Description
PMA_RECEIVE_LINK_STATUS	0	0b	RW	Reflects the inverse state of PHY register bit 1.1.2 before it was set by a firmware read.
PMA_TRANSMIT_FAULT	1	0b	RW	Reflects the state of PHY register bit 1.8.B before it was cleared by a firmware read.
PMA_RECEIVE_FAULT	2	0b	RW	Reflects the state of PHY register bit 1.8.A before it was cleared by a firmware read.
PMA_RESERVED	7:3	0x0	RW	Reserved. Read as written.
PCS_RECEIVE_LINK_STATUS	8	0b	RW	Reflects the inverse state of PHY register bit 3.1.2 before it was set by a firmware read.
PCS_TRANSMIT_FAULT	9	0b	RW	Reflects the state of PHY register bit 3.8.B before it was cleared by a firmware read.

Field	Bit(s)	Init.	Type	Description
PCS_RECEIVE_FAULT	10	0b	RW	Reflects the state of PHY register bit 3.8.A before it was cleared by a firmware read.
PCS_10GBASE_T_BLOCK_LOCK_LATCHED	11	0b	RW	Reflects the inverse state of PHY register bit 3.21.F before it was set by a firmware read.
PCS_10GBASE_T_HIGH_BER_LATCHED	12	0b	RW	Reflects the state of PHY register bit 3.21.E before it was cleared by a firmware read.
PCS_CRC_ERROR	13	0b	RW	Reflects the state of PHY register bit 3.EC00.F before it was cleared by firmware read.
PCS_LDPC_DECODE_FAILURE	14	0b	RW	Reflects the state of PHY register bit 3.EC00.E before it was cleared by a firmware read.
PCS_INVALID_65B_BLOCK	15	0b	RW	Reflects the state of PHY register bit 3.EC00.8 before it was cleared by a firmware read.
PCS_CHANGE_IN_AUXILIARY_BIT	16	0b	RW	Reflects the state of PHY register bit 3.EC00.0 before it was cleared by a firmware read.
PCS_RESERVED	23:17	0x0	RW	Reserved. Read as written.
PHY_XS_RESERVED	31:24	0x0	RW	Reserved. Read as written.

7.2.2.1.8 PHY Interrupt Status Register 1 - PHYINT_STATUS1 (0x00000104) Target

Register contents is valid once the corresponding EEMNGCTL.*CFG_DONE0/1* bit is asserted by firmware. Bits are set by firmware to notify the host of the PHY interrupts that were triggered. This register is reset by hardware only at power-up events. The host is responsible to clear the PHY interrupts once it completes the PHY interrupt handling routine.

Field	Bit(s)	Init.	Type	Description
AUTO_NEGOTIATION_EXTENDED_NEXT_PAGE_RECEIVED	0	0b	RW	Reflects the state of PHY register bit 7.1.6 before it was cleared by a firmware read.
AUTO_NEGOTIATION_REMOTE_FAULT	1	0b	RW	Reflects the state of PHY register bit 7.1.4 before it was cleared by a firmware read.
AUTO_NEGOTIATION_LINK_STATUS	2	0b	RW	Reflects the inverse state of PHY register bit 7.1.2 before it was cleared by a firmware read.
AUTO_NEGOTIATION_MASTER_SLAVE_CONFIGURATION_FAULT	3	0b	RW	Reflects the state of PHY register bit 7.21.F before it was cleared by a firmware read.
AUTO_NEGOTIATION_COMPLETED_FOR_NON_SUPPORTED_RATE	4	0b	RW	Reflects the state of PHY register bit 7.CC00.3 before it was cleared by a firmware read.
AUTO_NEGOTIATION_COMPLETED_FOR_SUPPORTED_RATE	5	0b	RW	Reflects the state of PHY register bit 7.CC00.2 before it was cleared by a firmware read.
AUTO_NEGOTIATION_AUTOMATIC_DOWNSHIFT	6	0b	RW	Reflects the state of PHY register bit 7.CC00.1 before it was cleared by a firmware read.
AUTO_NEGOTIATION_CONNECTION_STATE_CHANGE	7	0b	RW	Reflects the state of PHY register bit 7.CC00.0 before it was cleared by a firmware read.
AUTO_NEGOTIATION_100BASE_TX_DEVICE_DETECT	8	0b	RW	Reflects the state of PHY register bit 7.EC01.F before it was cleared by a firmware read.
AUTO_NEGOTIATION_ENERGY_ON_LINE_DETECT	9	0b	RW	Reflects the state of PHY register bit 7.EC01.E before it was cleared by a firmware read.



Field	Bit(s)	Init.	Type	Description
AUTO_NEGOTIATION_NEXT_PAGE_3RD_RECEIVED	10	0b	RW	Reflects the state of PHY register bit 7.EC01.3 before it was cleared by a firmware read.
AUTO_NEGOTIATION_NEXT_PAGE_2ND_RECEIVED	11	0b	RW	Reflects the state of PHY register bit 7.EC01.2 before it was cleared by a firmware read.
AUTO_NEGOTIATION_NEXT_PAGE_1ST_RECEIVED	12	0b	RW	Reflects the state of PHY register bit 7.EC01.1 before it was cleared by a firmware read.
AUTO_NEGOTIATION_BASE_PAGE_RECEIVED	13	0b	RW	Reflects the state of PHY register bit 7.EC01.0 before it was cleared by a firmware read.
AUTO_NEGOTIATION_10BASE_T_DEVICE_DETECT	14	0b	RW	Reflects the inverse state of PHY register bit 7.EC02.2 before it was cleared by a firmware read.
AUTO_NEGOTIATION_PROTOCOL_ERROR	15	0b	RW	Reflects the state of PHY register bit 7.EC01.D before it was cleared by a firmware read.
FLP_IDLE_ERROR	16	0b	RW	Reflects the state of PHY register bit 7.EC01.C before it was cleared by a firmware read.
AUTO_NEGOTIATION_GBE_PHY_RESERVED	27:17	0x0	RW	Reserved. Read as written.
PCIE_SERDES_LOSS_OF_SIGNAL_3_0	31:28	0x0	RW	Reflects the state of PHY register bit 4.CC02.F:C before it was cleared by a firmware read.

7.2.2.1.9 PHY Interrupt Status Register 2 - PHYINT_STATUS2 (0x00000108) Target

Register contents is valid once the corresponding EEMNGCTL.*CFG_DONE0/1* bit is asserted by firmware. Bits are set by firmware to notify the host of the PHY interrupts that were triggered. This register is reset by hardware only at power-up events. The host is responsible to clear the PHY interrupts once it completes the PHY interrupt handling routine.

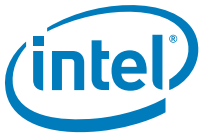
Field	Bit(s)	Init.	Type	Description
GLOBAL_MAC_RESET	0	0b	RW	Reflects the inverse state of PHY register bit 1E.1200.0 before it was cleared by a firmware read.
GLOBAL_MAC_LOW_POWER_LINK_UP_MODE	1	0b	RW	Reflects the state of PHY register bit 1E.1204.4 before it was cleared by a firmware read.
GLOBAL_MAC_PHY_DISABLE_MODE	2	0b	RW	Reflects the inverse state of PHY register bit 1E.1204.1 before it was cleared by a firmware read.
GLOBAL_MAC_LOW_POWER_MODE	3	0b	RW	Reflects the inverse state of PHY register bit 1E.1204.0 before it was cleared by a firmware read.
GLOBAL_MAC_SPI_GRANT	4	0b	RW	Reflects the state of PHY register bit 1E.1206.2 before it was cleared by a firmware read.
GLOBAL_MAC_SPI_CONTROL	5	0b	RW	Reflects the inverse state of PHY register bit 1E.1206.1 before it was cleared by a firmware read.
GLOBAL_HIGH_TEMPERATURE_FAILURE	6	0b	RW	Reflects the state of PHY register bit 1E.CC00.E before it was cleared by a firmware read.
GLOBAL_LOW_TEMPERATURE_FAILURE	7	0b	RW	Reflects the state of PHY register bit 1E.CC00.D before it was cleared by a firmware read.
GLOBAL_HIGH_TEMPERATURE_WARNING	8	0b	RW	Reflects the state of PHY register bit 1E.CC00.C before it was cleared by a firmware read.
GLOBAL_LOW_TEMPERATURE_WARNING	9	0b	RW	Reflects the state of PHY register bit 1E.CC00.B before it was cleared by a firmware read.

Field	Bit(s)	Init.	Type	Description
GLOBAL_RESET_COMPLETED	10	0b	RW	Reflects the state of PHY register bit 1E.CC00.6 before it was cleared by a firmware read.
GLOBAL_DEVICE_FAULT	11	0b	RW	Reflects the state of PHY register bit 1E.CC00.4 before it was cleared by a firmware read.
GLOBAL_PAIR_A_CHANGE_OF_STATUS	12	0b	RW	Reflects the state of PHY register bit 1E.CC00.3 before it was cleared by a firmware read.
GLOBAL_PAIR_B_CHANGE_OF_STATUS	13	0b	RW	Reflects the state of PHY register bit 1E.CC00.2 before it was cleared by a firmware read.
GLOBAL_PAIR_C_CHANGE_OF_STATUS	14	0b	RW	Reflects the state of PHY register bit 1E.CC00.1 before it was cleared by a firmware read.
GLOBAL_PAIR_D_CHANGE_OF_STATUS	15	0b	RW	Reflects the state of PHY register bit 1E.CC00.0 before it was cleared by a firmware read.
GLOBAL_MEDIUM_BER	16	0b	RW	Reflects the state of PHY register bit 1E.CC01.E before it was cleared by a firmware read.
GLOBAL_RESERVED	31:17	0x0	RW	Reserved. Read as written.

7.2.2.1.10 LED Control - LEDCTL (0x00000200) Target

Note: All init bits in this register are read from the shared SPI Flash. Refer to the LED Configuration section in the shared SPI Flash chapter

Field	Bit(s)	Init.	Type	Description
LED0_MODE	3:0	0x0	RW	LED0 Mode. This field specifies the control source for the LED0 output. An initial value of 0000b selects the LINK_UP indication.
RESERVED	4	0b	RSV	Reserved.
GLOBAL_BLINK_MODE	5	0b	RW	Global Blink Mode. This field specifies the blink mode of all LEDs. 0 = Blink at 200 ms on and 200 ms off. 1 = Blink at 83 ms on and 83 ms off.
LED0_IVRT	6	0b	RW	LED0 Invert. This field specifies the polarity/inversion of the LED0 source prior to output or blink control. By default, the output drives the cathode of the LED0. Therefore, when the LED output is 0b, the LED0 is on. 0 = LED0 output is active low. 1 = LED0 output is active high.
LED0_BLINK	7	0b	RW	LED0 Blink. This field specifies whether to apply blink logic to the (inverted) LED0 control source prior to the LED0 output. 0 = Do not blink LED0 output. 1 = Blink LED0 output.
LED1_MODE	11:8	0x1	RW	LED1 Mode. This field specifies the control source for the LED1 output. An initial value of 0001b selects the 10 Gb/s link indication.
RESERVED	13:12	0x0	RSV	Reserved.
LED1_IVRT	14	0b	RW	LED1 Invert. This field specifies the polarity/inversion of the LED1 source prior to output or blink control. By default, the output drives the cathode of the LED. Therefore, when the LED1 output is 0b, the LED1 is on. 0 = LED1 output is active low. 1 = LED output is active high.



Field	Bit(s)	Init.	Type	Description
LED1_BLINK	15	1b	RW	LED1 Blink. This field specifies whether to apply blink logic to the (inverted) LED1 control source prior to the LED output. 0 = Do not blink LED output. 1 = Blink LED output.
LED2_MODE	19:16	0x4	RW	LED2 Mode. This field specifies the control source for the LED2 output. An initial value of 0100b selects LINK/ACTIVITY indication.
RESERVED	21:20	0x0	RSV	Reserved.
LED2_IVRT	22	0b	RW	LED2 Invert. This field specifies the polarity/inversion of the LED2 source prior to output or blink control. By default, the output drives the cathode of the LED2. Therefore, when the LED2 output is 0b, the LED2 is on. 0 = LED2 output is active low. 1 = LED2 output is active high.
LED2_BLINK	23	0b	RW	LED2 Blink. This field specifies whether to apply blink logic to the (inverted) LED2 control source prior to the LED2 output. 0 = Do not blink LED2 output. 1 = Blink LED2 output.
LED3_MODE	27:24	0x5	RW	LED3 Mode. This field specifies the control source for the LED3 output. An initial value of 0101b selects the 1 Gb/s link indication.
RESERVED	29:28	0x0	RSV	Reserved.
LED3_IVRT	30	0b	RW	LED3 Invert. This field specifies the polarity/inversion of the LED3 source prior to output or blink control. By default, the output drives the cathode of the LED3. Therefore, when the LED3 output is 0b, the LED3 is on. 0 = LED3 output is active low. 1 = LED3 output is active high.
LED3_BLINK	31	0b	RW	LED3 Blink. This field specifies whether to apply blink logic to the (inverted) LED3 control source prior to the LED3 output. 0 = Do not blink LED3 output. 1 = Blink LED3 output.

7.2.2.1.11 Extended VLAN Ether Type - Receive - EXVET (0x00005078) RX_Filter

Field	Bit(s)	Init.	Type	Description
RESERVED	15:0	0x00	RSV	Reserved.
VET_EXT	31:16	0x8100	RW	Outer VLAN EtherType. The VLAN Tag Protocol Identifier (TPID). ***Note:*** This field appears in little endian (MS byte first on the wire).

7.2.2.1.12 Extended VLAN Ether Type - Transmit - EXVET_T (0x00008224) DMA_TX

Field	Bit(s)	Init.	Type	Description
RESERVED	15:0	0x00	RSV	Reserved.
VET_EXT	31:16	0x8100	RW	Outer VLAN EtherType. The VLAN Tag Protocol Identifier (TPID). ***Note:*** This field appears in little endian (MS byte first on the wire).

7.2.2.1.13 SFP I²C Command - I2CCMD (0x00015F58) MNG

This register is used by software to read or write to the configuration registers in an SFP module when the *SDP23_FUNCTION* bit and the *SDP[23]_NATIVE* bit are set in ESDP register.

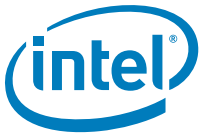
Field	Bit(s)	Init.	Type	Description
DATA	15:0	X	RW	Data in a write command: Software places the data bits and then the MAC shifts them out to the I²C bus. Data in a read command: The MAC reads these bits serially from the I²C bus and then software reads them from this location. This field is interpreted as two consecutive bytes unless the I2CPARAMS.*I2C_DATA_ORDER* bit is set, in which case the data is considered a single 16-bit word. When I2CPARAMS.*ACCESS_WIDTH* = 0, bits 15:8 are not used.
REGADD	23:16	0x0	RW	I²C Register Address. For example, register 0, 1, 2... 255.
RESERVED	26:24	0x0	RSV	Reserved.
OP	27	0b	RW	Op Code. 0 = I²C write. 1 = I²C read.
RESET	28	0b	RW	Reset Sequence. If set, sends a reset sequence before the actual read or write. This bit is self clearing. A reset sequence is defined as nine consecutive stop conditions.
R	29	1b	RW	Ready Bit. Set to 1b by the device at the end of the I²C transaction. Indicates the read or write completed. Reset by a software write of a command.
RESERVED	30	0b	RSV	Reserved
E	31	0b	RW	Error. This bit set is to 1b by hardware when it fails to complete an I²C read. Reset by a software write of a command. ***Note:*** This bit is valid only when the Ready (*R*) bit is set.

7.2.2.1.14 SFP I²C Parameters - I2CPARAMS (0x00015F5C) MNG

This register is used to set the parameters for I²C access and to allow bit-banging access to the I²C interface.

Note: This register is reset only on global reset.

Field	Bit(s)	Init.	Type	Description
WRITE_TIME	4:0	0x6	RW	Write Time. Defines the delay between a write access and the next access. The value is in milliseconds. ***Note:*** A value of zero is not valid.
READ_TIME	7:5	0x2	RW	Read Time. Defines the delay between a read access and the next access. The value is in microseconds. ***Note:*** A value of zero is not valid
I2CBB_EN	8	0b	RW	I²C Bit-bang Enable. 0 = The I2C_CLK and I2C_DATA lines are controlled by hardware once activated via the I2CCMD register. 1 = The I2C_CLK and I2C_DATA lines are controlled via the *CLK*, *DATA*, and *DATA_OE_N* fields of this register.
CLK_OUT	9	0b	RW	I2C Clock. While in bit-bang mode, controls the value driven on the I2C_CLK pad of this port.



Field	Bit(s)	Init.	Type	Description
DATA_OUT	10	0b	RW	I2C_DATA. While in bit-bang mode and when the *DATA_OE_N* field is zero, controls the value driven on the I2C_DATA pad of this port.
DATA_OE_N	11	0b	RW	I2C_DATA_OE_N. While in bit-bang mode, controls the direction of the I2C_DATA pad of this port. 0 = Pad is output. 1 = Pad is input.
DATA_IN	12	0b	RO	I^2C Data In. Reflects the value of the I2C_DATA pad. While in bit-bang mode when the *DATA_OE_N* field is zero, this field reflects the value set in the *DATA_OUT* field.
CLK_OE_N	13	0b	RW	I^2C Clock Output Enable. While in bit-bang mode, controls the direction of the I2C_CLK pad of this port. 0 = Pad is output. 1 = Pad is input.
CLK_IN	14	0b	RO	I^2C Clock In Value. Reflects the value of the I2C_CLK pad. While in bit-bang mode when the *CLK_OE_N* field is zero, this field reflects the value set in the *CLK_OUT* field.
CLK_STRETCH_DIS	15	0b	RW	Clock Stretch Disable. 0 = Enable slave clock stretching support in an I^2C access. 1 = Disable clock stretching support in an I^2C access.
ACCESS_WIDTH	16	0b	RW	I^2C Access Width: 0 = Byte access 1 = Word access
RESERVED	23:17	0x0	RSV	Reserved. Write 0x0, ignore on read.
PHYADD	30:24	0x0	RW	Device Address Bits 7:1. The actual address used is b{PHYADD[6:0], 0}.
I2C_DATA_ORDER	31	0b	RW	I^2C Data Order. 0 = I2CCMD.DATA field read in byte order. 1 = I2CCMD.DATA field read in word order.

7.2.2.1.15 General Receive Control - GRC (0x00015F64) MNG

LAN Wake Enables.

Field	Bit(s)	Init.	Type	Description
RESERVED	0	0b	RSV	Reserved.
APME	1	0b	RW	APM Wake Enable Port 1 (copied from shared SPI Flash3[1] on CW load) APM Wake Enable Port 0 (copied from shared SPI Flash3[0] on CW load)
RESERVED	31:2	0x0	RSV	Reserved.

7.2.2.1.16 Function Active and Power State to Manageability - FACTPS (0x00015FEC) MNG

Activity / Power States.

Note: In regular mode (port swap disabled - *LAN_FUNCTION_SEL* = 0), the power state indication of port 0 is mapped to the *FUNC0_POWER_STATE* field and the power state indication of port 1 is mapped the *FUNC1_POWER_STATE* field. And vice versa when port swap mode is enabled (*LAN_FUNCTION_SEL* = 1).

Field	Bit(s)	Init.	Type	Description
RESERVED	1:0	0x0	RSV	Reserved.
LAN0_VALID	2	0b	RO	Shadow of Enable Pin.

Field	Bit(s)	Init.	Type	Description
FUNC0_AUX_EN	3	0b	RO	Enable bit shadow from the configuration space.
RESERVED	7:4	0x0	RSV	Reserved.
LAN1_VALID	8	0b	RO	Shadow of Enable Pin.
FUNC1_AUX_EN	9	0b	RO	Enable bit shadow from the configuration space
RESERVED	29:10	0x0	RSV	Reserved.
LAN_FUNCTION_SEL	30	0b	RO	Value of ControlWord2[15].
RESERVED	31	0b	RSV	Reserved.

7.2.2.1.17 Device and Functions Enable Control - DEV_FUNC_EN (0x00015FF0) MNG

This register is common to the two ports and reflects the LAN disable and device disable enable bits in the shared SPI Flash.

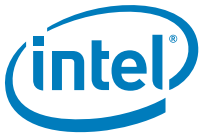
Field	Bit(s)	Init.	Type	Description
LAN_PCI_DISABLE	0	0b	RO	Value of ControlWord2[8].
LAN_DISABLE_SELECT	1	0b	RO	Value of ControlWord2[9].
DEV_OFF_EN	2	0b	RO	Value of ControlWord2[10].
RESERVED	3	0b	RSV	Reserved.
IGNORE_PHY_FW_VALID	4	0b	RO	Value of ControlWord2[12].
RESERVED	31:5	0x0	RSV	Reserved

7.2.2.2 Shared SPI Flash Registers

7.2.2.2.1 Software Flash Burst Control Register - FLSWCTL (0x00015F48) MNG

Flash access control. When Software writes this register, the Firmware gets an interrupt.

Field	Bit(s)	Init.	Type	Description
ADDR	23:0	0x0	RW	Address. This field is written by software along with *CMD* to indicate the Flash address to do operation (read/write/erase etc.). Refer to *CMD* description below.
CMD	27:24	0x0	RW	Command. Indicates which command should be executed. Valid only when the *CMDV* bit is set 0000b = Read 0001b = Write 0010b = Sector Erase All other values are reserved.
CMDV	28	0b	RO	Command Valid. Set by hardware when a successful write of this register is received. When set, indicates that software issued a new command and the command is in progress. Cleared by Firmware at the end of the command. If the software tries to set this bit while a command is in progress (*DONE* is not set), the hardware will clear the bit.
COMMAND_FAIL	29	0b	RO	When set, means the command was not executed due to security issues, for example attempt to write to a read only area or request of an unknown command. This bit is valid only if *DONE* bit is set.



Field	Bit(s)	Init.	Type	Description
DONE	30	1b	RO	Partial Command Done. This bit clears after register is written by software and is set back again when the Flash transaction is done. When writing a burst transaction the bit is cleared by hardware every time software writes FLSWDATA. When doing a read burst transaction the bit is cleared by hardware every time software reads FLSWDATA. Firmware writes sets this bit after the data was written/read from the flash.
GLDONE	31	1b	RO	Global Done. This bit is cleared by hardware after register is written by software and is set back again by Firmware when all Flash read/write transactions completes (FLSWCNT.*NVCNT* bytes where read/written).

7.2.2.2.2 Software Flash Burst Data Register - FLSWDATA (0x00015F4C) MNG

This register holds the data read/written to the flash. Writing or reading of this register will send an interrupt to the Firmware and will clear the FLSWCTL.*DONE* bit.

Field	Bit(s)	Init.	Type	Description
NVDATA	31:0	0x0	RW	Shared SPI Flash Data. Data written or read to/from the shared SPI Flash.

7.2.2.2.3 Software Flash Burst Access Counter - FLSWCNT (0x00015F50) MNG

Field	Bit(s)	Init.	Type	Description
NVCNT	12:0	0x0	RW	Shared SPI Flash Burst Counter. This counter holds the size in bytes of the Flash burst read or write.
RESERVED	31:13	0x0	RSV	Reserved.

7.2.2.2.4 Flash Firmware Code Update - FLUPDATE (0x00015F54) MNG

Field	Bit(s)	Init.	Type	Description
MODULEID	15:0	0x0	RW	The module in the flash to be updated from the free area when the FLUPDATE.*UPDATE* bit is set. Should be compared to the LSB of the *MODULEID* in the CSS Authentication header.
RESERVED	28:16	0x0	RSV	Reserved.
AUT_DONE	29	0b	ROS	Authentication Cycle Done. Set to 1b by firmware when done. This bit is self-cleared once the update request is set to 1b.
AUT_FAIL	30	0b	ROS	Authentication Failed. Set to 1b by firmware when authentication failed.
UPDATE	31	0b	SC	Request authentication of the new secure section written. If the authentication succeeds, firmware resets itself to load its new code. This bit is self-cleared, always read as 0b.

7.2.2.2.5 Flash Access Register - FLA (0x00015F68) MNG

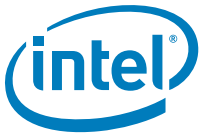
Note: When shared SPI Flash is locked, this register is RO to Software.

Field	Bit(s)	Init.	Type	Description
RESERVED	5:0	0x0	RSV	Reserved.

Field	Bit(s)	Init.	Type	Description
LOCKED	6	0b	RO	RO Mode is ON and Security Disable Strap pin is OFF.
NVM_SEC_DIS	7	0b	RO	Reflects the value of the NVM_SEC_DIS strap.
RESERVED	31:8	0x0	RSV	Reserved. Reads as 0b.

7.2.2.2.6 EEPROM Mode Control Register - EEC (0x00015FF8) MNG

Field	Bit(s)	Init.	Type	Description
RESERVED	7:0	0x0	RSV	Reserved.
EE_PRES	8	0b	RO	Shared SPI Flash Present (set by FW). When this bit is set, indicates that an shared SPI Flash is present and has the correct signature field.
AUTO_RD	9	0b	RO	Shared SPI Flash Auto-Read Done. When set to 1b, this bit indicates that the auto-read by hardware from the shared SPI Flash caused by the latest reset for this port is done. This bit is also set when the shared SPI Flash is not present or when its signature field is not valid.
MNG_READY	10	0b	RO	Management Auto-load Done. When set, indicates the sections needed for manageability are valid.
EE_SIZE	14:11	0x7	RO	Shared SPI Flash Size via EEPROM Mode. This field defines the size of the shared SPI Flash that is accessible via EEPROM mode. This is equal to the size of the internal shadow RAM, fixed to 16 KB (0x7).
RESERVED	16:15	0x0	RSV	Reserved.
PCI__GENERAL__DONE	17	0b	RO	PCIe General Done. When set to 1b, indicates that the PCIe general section read from the shared SPI Flash is done. This bit is cleared when auto-read starts. This bit is also set when the shared SPI Flash is not present or when its signature field is not valid.
PCI_FUNC_DONE	18	0b	RO	PCIe Function Done. When set to 1b, indicates that the PCIe function section read from the shared SPI Flash is done. This bit is cleared when auto-read starts. This bit is also set when the shared SPI Flash 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	When set to 1b, indicates that the core section read from the shared SPI Flash is done. This bit is cleared when auto-read starts. This bit is also set when the shared SPI Flash 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	When set to 1b, indicates that the core CSR section read from the shared SPI Flash is done. This bit is cleared when auto-read starts. This bit is also set when the shared SPI Flash is not present or when its signature field is not valid. ***Note:*** This bit returns the relevant done indication for the function that reads the register.



Field	Bit(s)	Init.	Type	Description
MAC_DONE	21	0b	RO	When set to 1b, indicates that the MAC section read from the shared SPI Flash is done. This bit is cleared when auto-read starts. This bit is also set when the shared SPI Flash is not present or when its signature field is not valid. ***Note:*** This bit returns the relevant done indication for the function that reads the register.
HIP_DONE	22	0b	RO	When set, indicates that the HIP section read from the shared SPI Flash is done. This bit is cleared when auto-read starts. This bit is also set when the shared SPI Flash is not present or when its signature field is not valid.
FLUPD	23	0b	RW	Set by FW. Cleared by FW or when FLUPD is set.
RESERVED	24	0b	RSV	Reserved.
SEC1VAL	25	0b	RO	Sector 1 Valid (set by FW). 0 = Indicates that the content of the first section (from byte address 0x0000 to 0x3FFF) is valid. Meaningful only when the EE_PRES bit is read as 1b. 1 = Indicates that the content of the second shadow RAM section (from byte address 0x4000 to 0x7FFF) of the Flash device is valid.
FLUDONE	26	0b	WO	(Set) Send Interrupt to FW (reads as 0).
RESERVED	31:27	0x0	RSV	Reserved.

7.2.2.3 Flow Control Registers

7.2.2.3.1 Flow Control Transmit Timer Value n - FCTTVN[n] (0x00003200 + 0x4*n, n=0...3) DBU_RX

Each 32-bit register (n=0... 3) refers to two timer values (register 0 refers to timer 0 and 1; register 1 refers to timer 2 and 3, etc.).

Note: The 16-bit value in the *TTV* field is inserted into a transmitted frame (either XOFF frames or any pause frame value in any software transmitted packets). It counts in units of slot time (usually 64 bytes). The device uses a fixed slot time value of 64-byte times.

Field	Bit(s)	Init.	Type	Description
TTV_2N	15:0	0x0	RW	Transmit Timer Value 2n. Timer value included in XOFF frames as Timer (2n). The same value must be set to User Priorities (UPs) attached to the same TC, as defined in the RTTUP2TC register. For legacy 802.3X flow control packets, TTV0 is the only timer that is used.
TTV_2N_1	31:16	0x0	RW	Transmit Timer Value 2n+1. Timer value included in XOFF frames as Timer 2n+1. The same value must be set to UPs attached to the same TC, as defined in the RTTUP2TC register.

7.2.2.3.2 Flow Control Receive Threshold Low - FCRTL[n] (0x00003220 + 0x4*n, n=0...7) DBU_RX

Each 32-bit register (n=0... 7) refers to a different receive packet buffer.

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.

Field	Bit(s)	Init.	Type	Description
RESERVED	4:0	0x0	RSV	Reserved.
RTL	18:5	0x0	RW	Receive Threshold Low n. Receive packet buffer n FIFO low water mark for flow control transmission (32 bytes granularity).
RESERVED	30:19	0x0	RSV	Reserved.
XONE	31	0b	RW	XON Enable n. Per the receive packet buffer XON enable. 0 = Disabled 1 = Enabled

7.2.2.3.3 Flow Control Receive Threshold High - FCRTH[n] (0x00003260 + 0x4*n, n=0...7) DBU_RX

Each 32-bit register (n=0... 7) refers to a different receive packet buffer.

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.

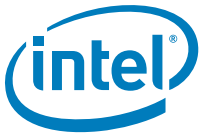
Field	Bit(s)	Init.	Type	Description
RESERVED	4:0	0x0	RSV	Reserved.
RTH	18:5	0x0	RW	Receive Threshold High n. Receive packet buffer n FIFO high water mark for flow control transmission (32 bytes granularity).
RESERVED	30:19	0x0	RSV	Reserved.
FCEN	31	0b	RW	Transmit Flow Control Enable for packet buffer n. Should be set only for traffic classes to which UPs are mapped (in RTRUP2TC register).

7.2.2.3.4 Flow Control Refresh Threshold Value - FCRTV (0x000032A0) DBU_RX

Field	Bit(s)	Init.	Type	Description
FC_REFRESH_TH	15:0	0x0	RW	Flow Control Refresh Threshold. This value is used to calculate the actual refresh period for sending the next pause frame if conditions for a pause state are still valid (buffer fullness above low threshold value). The formula for the refresh period for priority group N is: `FCTTV[N/2].TTV[Nmod2] - FCRTV.FC_REFRESH_TH` ***Note:*** The *FC_REFRESH_TH* must be smaller than TTV of the TC and larger than the maximum packet size in the TC + FC packet size + link latency and Tx latency and Rx latency in 64-byte units.
RESERVED	31:16	0x0	RSV	Reserved.

7.2.2.3.5 Flow Control Configuration - FCCFG (0x00003D00) DBU_RX

Field	Bit(s)	Init.	Type	Description
RESERVED	2:0	0x0	RSV	Reserved.



Field	Bit(s)	Init.	Type	Description
TFCE	4:3	0x0	RW	Transmit Flow Control Enable. These bits indicate that the device transmits flow control packets (XON/XOFF frames) based on receive fullness. If auto negotiation is enabled, this bit should be set by software to the negotiated flow control value. 00b = Transmit flow control disabled. 01b = Link flow control enabled. 10b = Reserved. 11b = Reserved.
RESERVED	31:5	0x0	RSV	Reserved.

7.2.2.3.6 MAC Flow Control Register - MFLCN (0x00004294) MAC

Field	Bit(s)	Init.	Type	Description
PMCF	0	0b	RW	Pass MAC Control Frames. Filter out unrecognized pause (Flow Control OpCode does not match) and other control frames. 0 = Filter Unrecognized Pause Frames. 1 = 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* is set to 1b. If *RFCE* is set to 0b, this bit has no effect on incoming PAUSE frames.
RESERVED	2	0b	RW	Reserved.
RFCE	3	0b	RW	Receive Link Flow Control Enable. Indicates that the integrated 10 GbE LAN controller 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.
RESERVED	11:4	0x0	RW	Reserved.
RESERVED	31:12	0x0	RSV	Reserved.

7.2.2.3.7 Transmit Flow Control Status - TFCS (0x0000CE00) DBU_TX

Field	Bit(s)	Init.	Type	Description
TC_XON	7:0	0xFF	RO	TC is in FC XON state.
RESERVED	31:8	0x0	RSV	Reserved.

7.2.2.4 PCIe Registers

This section contains the registers used to control the PCIe core behavior.

7.2.2.4.1 PCIe Function Status 1 - PCI_STATUS1 (0x00011028) PCIe_GLUE

Field	Bit(s)	Init.	Type	Description
FUNC_VALID	0	0b	RO	Function Valid. 0 = Function is disabled 1 = Function is enabled ***Note:*** This bit is valid to firmware even when the function is disabled
RESERVED	31:1	0x0	RSV	Reserved.

7.2.2.4.2 PCIe Firmware Control - PCI_FWCTRL (0x00011040) PCIe_GLUE

Field	Bit(s)	Init.	Type	Description
TCO_ISOLATE	0	0b	RO	Set by Firmware to block all host write access if instructed to do so by BMC.
RESERVED	31:1	0x0	RW	Reserved.

7.2.2.4.3 PCIe CSR Access Timeout - PCI_CSRTO (0x00011044) PCIe_GLUE

Field	Bit(s)	Init.	Type	Description
CSR_TO	15:0	0x000F	RO	Defines the timeout value in micro second for CSR accesses.
MSIX_TO	31:16	0x0028	RO	Defines the timeout value in micro second for MSI-x table accesses.

7.2.2.4.4 PCIe Control Extended Register - GCR_EXT (0x00011050) PCIe_GLUE

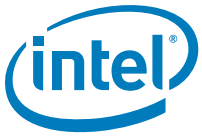
Field	Bit(s)	Init.	Type	Description
VT_MODE	1:0	0x0	RW	VT Mode of operation. Defines the allocation of physical registers to the VFs. Software must set this field the same as GPIE. 00b = noVT — Reserved. 01b = VT16 — Resources are allocated to 16 VFs. 10b = VT32 — Resources are allocated to 32 VFs 11b = VT64 — Resources are allocated to 64 VFs.
RESERVED	3:2	0x0	RSV	Reserved.
APBACD	4	0b	RW	Auto PBA Clear Disable. 0 = Any active PBA entry is cleared on the falling edge of the appropriate interrupt request to the PCIe block. 1 = Software can clear the PBA only by direct write to clear access to the PBA bit. The appropriate interrupt request is cleared when software sets the associated interrupt mask bit in the EIMS (re-enabling the interrupt) or by direct write to clear access to the PBA.
RESERVED	31:5	0x00	RSV	Reserved.

7.2.2.4.5 PCI Flash Access Timeout - PCI_FLASHTO (0x00011054) PCIe_GLUE

Field	Bit(s)	Init.	Type	Description
PCI_FLASHTO	31:0	0x00000FA0	RO	Defines the timeout value in micro second for flash accesses.

7.2.2.4.6 Function Requester ID Information Register - FUNC_RID (0x00011070) PCIe_GLUE

Field	Bit(s)	Init.	Type	Description
FUNCTION_NUMBER	2:0	0x0	RO	Function Number. Function number assigned to the function based on BIOS/OS Enumeration.
DEVICE_NUMBER	7:3	0x0	RO	Device Number. Device number assigned to the function based on BIOS/OS Enumeration.
BUS_NUMBER	15:8	0x0	RO	Bus Number. Bus Number assigned to the function based on BIOS/OS Enumeration.
RESERVED	31:16	0x0	RSV	Reserved.



7.2.2.4.7 PCIe Revision ID - PCI_REVID (0x00011098) PCIe_GLUE

Field	Bit(s)	Init.	Type	Description
NVM_REVID	7:0	0x00	RW	Value of Rev ID loaded from shared SPI Flash.
RESERVED	31:8	0x00	RSV	Reserved.

7.2.2.4.8 PCIe Errors Reported - PCI_PCIERR (0x00011140) PCIe

This register indicates which PCIe errors are reported to device software.

Field	Bit(s)	Init.	Type	Description
PCIE_ERR_REP	31:0	0x0	RO	Each bit corresponds to a particular error event as described in section "Proprietary Error Reporting".

7.2.2.4.9 PCIe Interrupt Cause - PCI_ICAUSE (0x00011520) PCIe

Field	Bit(s)	Init.	Type	Description
PCIE_ERR_CAUSE	31:0	0x0	RW1C	Each bit corresponds to a particular error event as described in section "Proprietary Error Reporting".

7.2.2.4.10 PCIe Interrupts Enable - PCI_IENA (0x00011528) PCIe

Field	Bit(s)	Init.	Type	Description
PCIE_ERR_EN	31:0	0x0	RW	Each bit corresponds to a particular error event as described in section "Proprietary Error Reporting".

7.2.2.4.11 PCIe VM Pending Index - PCI_VMINDEX (0x00011530) PCIe

Field	Bit(s)	Init.	Type	Description
VMINDEX	8:0	0x0	RW	VM Index. Software sets the *VMINDEX* that its transaction pending flag should be reflected in the PFPCI_VMPEND register. The VM index is an absolute index in the range of 0 through 63. It can be set by the software only to VMs that the PF owns and only to VMs which are not assigned to a VF.
RSVD	31:9	0x0	RSV	Reserved.

7.2.2.4.12 PCIe VM Pending Status - PCI_VMPEND (0x00011538) PCIe

Field	Bit(s)	Init.	Type	Description
PENDING	0	0b	RO	PCIe Transaction Pending Status. The reported VM is controlled by the *VMINDEX* field in the PFPCI_VMINDEX register. This flag is set to 1b as long as there is at least one PCIe transaction, pending for its completion.
RSVD	31:1	0x0	RSV	Reserved.

7.2.2.4.13 PCIe Default Revision ID - PCI_DREVID (0x00011540) PCIe

Field	Bit(s)	Init.	Type	Description
DEFAULT_REVID	7:0	0x00	RO	Mirroring of Default Rev ID prior to shared SPI Flash load.
RESERVED	31:8	0x00	RSV	Reserved.

7.2.2.4.14 PCIe Byte Counter High - PCI_BYTCTH (0x00011544) PCIe

A byte counter used by the PCIe performance counters.

Field	Bit(s)	Init.	Type	Description
PCI_COUNT_BW_BCT	31:0	0x00	RO	This register contains the high double-word of a 64-bit counter that counts PCIe payload bytes. This register gets stuck at its maximum value of 0xFF...F.

7.2.2.4.15 PCIe Byte Counter Low - PCI_BYTCTL (0x00011548) PCIe

A byte counter used by the PCIe performance counters.

Field	Bit(s)	Init.	Type	Description
PCI_COUNT_BW_BCT	31:0	0x00	RO	This register contains the low double-word of a 64-bit counter that counts PCIe payload bytes. This register gets stuck at its maximum value of 0xFF...F.

7.2.2.4.16 PCIe Latency Counter - PCI_LATCT (0x00011720) PCIe

A latency counter used by the PCIe performance counters.

Field	Bit(s)	Init.	Type	Description
PCI_COUNT_LAT_CT	31:0	0x00	RO	A 32-bit counter that counts time in multiples of 100 ns. This register gets stuck at its maximum value of 0xFF...F.

7.2.2.4.17 PCIe LCB Data Port - PCI_LCBDATA (0x00011734) PCIe

Field	Bit(s)	Init.	Type	Description
LCB_DATA	31:0	0x0	RW	A 32-bit data register. Part of a port. Used to load shared SPI Flash configuration into the PCIe LCB unit. Operates together with the PCI_LCBADDR register.

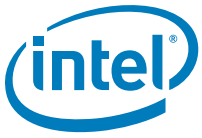
7.2.2.4.18 PCIe Packet Counter - PCI_PKTCT (0x00011740) PCIe

A packet counter used by the PCIe performance counters.

Field	Bit(s)	Init.	Type	Description
PCI_COUNT_BW_PCT	31:0	0x0	RO	A 32-bit counter that counts PCIe packets. This register gets stuck at its maximum value of 0xFF...F.

7.2.2.4.19 PCIe LCB Address Port - PCI_LCBADD (0x00011788) PCIe

Field	Bit(s)	Init.	Type	Description
ADDRESS	17:0	0x0	RW	An 18-bit address register. Part of a port. Used to load shared SPI Flash configuration into the PCIe LCB unit. Operates together with the PCI_LCBDATA register
RESERVED	19:18	0x0	RSV	Reserved.



Field	Bit(s)	Init.	Type	Description
BLOCK_ID	27:20	0x0	RW	An ID of a sub-unit in the PCIe unit. Supported values are: 00000000b = NPQ: RLAN 00000001b = NPQ: TLAN 00000010b = NPQ: RX_PE 00000011b = NPQ: TX_PE 00000100b = NPQ: PMAT 00000101b = NPQ: MNG 00000110b = NPQ: TDPU 00000111b = PQ: RLAN 00001000b = PQ: TLAN 00001001b = PQ: RX_PE 00001010b = PQ: TX_PE 00001011b = PQ: PMAT 00001100b = PQ: MNG 00001101b = PQ: RDPU 00001110b = VDM 00001111b = Don't care 00010000b = HIU: CFG/Slave/MSG/MSI-X 01111110b = LCB's internal config space registers 01111111b = LCB's internal memory space registers All other values are reserved.
RESERVED	31:28	0x0	RSV	Reserved.

7.2.2.4.20 PCIe Statistic Control Register #1 - PCI_GSCL_1 (0x00011800) PCIe

This register controls the operation of the PCIe performance counters.

Field	Bit(s)	Init.	Type	Description
GIO_COUNT_EN_0	0	0b	RW	Enables PCIe* statistic counter number 0.
GIO_COUNT_EN_1	1	0b	RW	Enables PCIe* statistic counter number 1.
GIO_COUNT_EN_2	2	0b	RW	Enables PCIe* statistic counter number 2.
GIO_COUNT_EN_3	3	0b	RW	Enables PCIe* statistic counter number 3.
LBC_ENABLE_0	4	0b	RW	Leaky Bucket Counter Enable 0. 0 = Leaky Bucket mode is disabled and the counter is incremented by one for each event. 1 = Statistics counter 0 operates in Leaky Bucket mode.
LBC_ENABLE_1	5	0b	RW	Leaky Bucket Counter Enable 1. 0 = Leaky Bucket mode is disabled and the counter is incremented by one for each event. 1 = Statistics counter 1 operates in Leaky Bucket mode.
LBC_ENABLE_2	6	0b	RW	Leaky Bucket Counter Enable 2. 0 = Leaky Bucket mode is disabled and the counter is incremented by one for each event. 1 = Statistics counter 2 operates in Leaky Bucket mode.
LBC_ENABLE_3	7	0b	RW	Leaky Bucket Counter Enable 3. 0 = Leaky Bucket mode is disabled and the counter is incremented by one for each event. 1 = Statistics counter 3 operates in Leaky Bucket mode.
PCI_COUNT_LAT_EN	8	0b	RW	PCIe Count Latency Enable. Enables the latency counter. 0 = Disable the latency counter. 1 = Enable the latency counter.

Field	Bit(s)	Init.	Type	Description
PCI_COUNT_LAT_EV	13:9	0x00	RW	PCIe Count Latency Event. Selects the event to be measured.
PCI_COUNT_BW_EN	14	0b	RW	PCIe Count Bandwidth Enable. Enables the bandwidth counter. 0 = Disable the bandwidth counter 1 = Enable the bandwidth counter
PCI_COUNT_BW_EV	19:15	0x00	RW	PCIe Count Bandwidth Event. Selects the event to be measured.
RESERVED	27:20	0x0	RSV	Reserved
GIO_64_BIT_EN	28	0b	RW	Enables two 64-bit counters instead of four 32-bit counters.
GIO_COUNT_RESET	29	0b	RWS	Reset indication of PCIe* statistic counters. Write of zero has no effect.
GIO_COUNT_STOP	30	0b	RWS	Stop indication of PCIe* statistic counters. Write of zero has no effect.
GIO_COUNT_START	31	0b	RWS	Start indication of PCIe* statistic counters. Write of zero has no effect.

7.2.2.4.21 PCIe Statistic Control Registers #2 - PCI_GSCL_2 (0x00011804) PCIe

This register defines the events counted by the performance counters.

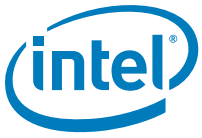
Field	Bit(s)	Init.	Type	Description
GIO_EVENT_NUM_0	7:0	0x0	RW	Event number that counter 0 counts (GSCN_0).
GIO_EVENT_NUM_1	15:8	0x0	RW	Event number that counter 1 counts (GSCN_1).
GIO_EVENT_NUM_2	23:16	0x0	RW	Event number that counter 2 counts (GSCN_2).
GIO_EVENT_NUM_3	31:24	0x0	RW	Event number that counter 3 counts (GSCN_3).

7.2.2.4.22 PCIe Statistic Control Register #5...#8 - PCI_GSCL_5_8[n] (0x00011810 + 0x4*n, n=0...3) PCIe

These registers control the operation of the leaky bucket counter n.

- GSCL_5 corresponds to n=0.
- GSCL_6 corresponds to n=1.
- GSCL_7 corresponds to n=2.
- GSCL_8 corresponds to n=3.

Field	Bit(s)	Init.	Type	Description
LBC_THRESHOLD_N	15:0	0x00	RW	Threshold for the Leaky Bucket Counter n.
LBC_TIMER_N	31:16	0x00	RW	Time period between decrementing the value in Leaky Bucket Counter n. The time period is defined in μs units.



7.2.2.4.23 PCIe Statistic Counter Registers #0...#3 - PCI_GSCN_0_3[n] (0x00011820 + 0x4*n, n=0...3) PCIe

These registers contain the performance counters 0-3.

- GSCL_0 corresponds to n=0.
- GSCL_1 corresponds to n=1.
- GSCL_2 corresponds to n=2.
- GSCL_3 corresponds to n=3.

Field	Bit(s)	Init.	Type	Description
EVENT_COUNTER	31:0	0x0	RO	Event counter as defined in PCI_GSCL_2.*GIO_EVENT_NUM_X* fields. These registers are stuck at their maximum value of 0xFF...F.

7.2.2.5 PCIe Configuration Space Setting Registers

This section contains registers used to set the default setting of the PCIe configuration space. These registers are reset and loaded from the shared SPI Flash at PCIe reset.

7.2.2.5.1 PCIe PF Configuration - PCI_CNF (0x00011000) PCIe_GLUE

Contains the per-PF configuration loaded from the shared SPI Flash.

Field	Bit(s)	Init.	Type	Description
RESERVED	2:0	0x0	RW	Reserved.
EXROM_DIS	3	1b	RW	Expansion ROM Disable. 0 = The Expansion ROM BAR in the PCI configuration space is enabled. 1 = The Expansion ROM BAR in the PCI configuration space is disabled.
IO_BAR	4	0b	RW	I/O BAR Support. 0 = I/O BAR is not supported. 1 = I/O BAR is supported.
INT_PIN	6:5	0x0	RW	Controls the value advertised in the Interrupt Pin field of the PCI configuration header for this function. 00b = INTA# 01b = INTB# 10b = INTC# 11b = INTD# The value advertised in the PCI configuration header is the value loaded from shared SPI Flash + 1. The default value for port 1 is 0x1.
RESERVED	31:7	0x0	RSV	Reserved.

7.2.2.5.2 PCIe PF Device ID - PCI_PFDEVID (0x00011008) PCIe_GLUE

Contains the per-PF Device ID.

Field	Bit(s)	Init.	Type	Description
PF_DEV_ID_LAN	15:0	0x1562	RW	Contains the Device ID for this PF when the function has an Ethernet device Class Code.
RESERVED	31:16	0x1562	RW	Reserved.

7.2.2.5.3 PCIe VF Device ID - PCI_VFDEVID (0x00011010) PCIe_GLUE

Contains the per-PF Device IDs for its VFs.

Field	Bit(s)	Init.	Type	Description
VF_DEV_ID_LAN	15:0	0x1565	RW	Contains the device ID for this PF's VFs when the function has an Ethernet device class code.
RESERVED	31:16	0x1565	RW	Reserved.

7.2.2.5.4 PCIe Storage Class - PCI_CLASS (0x00011038) PCIe_GLUE

Contains the per-PF configuration loaded from the shared SPI Flash.

Field	Bit(s)	Init.	Type	Description
STORAGE_CLASS	0	0b	RW	Storage Class. 0 = The class code of this port is set to 0x020000 (LAN). 1 = The class code of this port is set to 0x010000 (SCSI).
RESERVED	31:1	0x0	RSV	Reserved.

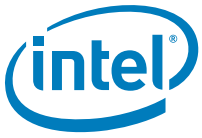
7.2.2.5.5 PCIe Vendor ID - PCI_VENDORID (0x0001103C) PCIe_GLUE

The Vendor ID exposed in config space. A value of 0xFFFF is not loaded to config space.
Shared for all PFs.

Field	Bit(s)	Init.	Type	Description
VENDOR_ID	15:0	0x8086	RW	Contains the Vendor ID exposed in offset 0x0 in the config space of all functions. A value of 0xFFFF is ignored.
RESERVED	31:16	0x0	RSV	Reserved.

7.2.2.5.6 PCI BAR Control - PCI_LBARCTRL (0x00011048) PCIe_GLUE

Field	Bit(s)	Init.	Type	Description
PREFBAR	0	1b	RW	Prefetchable bit indication in the memory BARs (should be set when 64-bit BARs are used) 0 = BARs are marked as non prefetchable. 1 = BARs are marked as prefetchable.
BAR32	1	0b	RW	BAR 32-bit Enable. 0 = 64-bit BAR addressing mode is selected. 1 = 32-bit BARs are enabled.
CSRSIZE	2	0b	RW	The *CSRSIXE* and *FL_BAR_SIZE* fields define the usable FLASH size and CSR mapping window size.
FLASH_EXPOSE	3	1b	RW	When set, the Flash memory is accessible through the memory BAR.
RESERVED	5:4	0x0	RSV	Reserved.
FL_BAR_SIZE	8:6	0x4	RW	This field indicates the size of the external Flash as: `64KB x (2 ** FL_BAR_SIZE)` as used to define exposure in the BAR. The following values are supported: 100b = 1 MB 101b = 2 MB 110b = 4 MB 111b = 8 MB All other values are reserved.
RESERVED	10:9	0x0	RW	Reserved.



Field	Bit(s)	Init.	Type	Description
EXROM_BAR_SIZE	13:11	0x3	RW	This field indicates the size of the expansion ROM BAR as `64KB x (2 ** EXROM_BAR_SIZE)` Values are: 000b = Corresponds to a 64 KB size 111b = Corresponds to a 8 MB size Default value is 512 KB.
RESERVED	31:14	0x0	RSV	Reserved.

7.2.2.5.7 PCIe Subsystem ID - PCI_SUBSYSID (0x00011058) PCIe_GLUE

Field	Bit(s)	Init.	Type	Description
SUB_VEN_ID	15:0	0x8086	RW	Loaded to the PCI configuration Subsystem Vendor ID Register.
SUB_ID	31:16	0x0	RW	Loaded to the PCI configuration Subsystem ID Register.

7.2.2.5.8 PCIe Power Data Register - PCI_PWRDATA (0x00011060) PCIe_GLUE

Field	Bit(s)	Init.	Type	Description
D0_POWER	7:0	0x0	RW	The value in this field is reflected in the PCI Power Management Data register of the LAN functions for D0 power consumption and dissipation (Data_Select = 0 or 4).
COMM_POWER	15:8	0x0	RW	The value in this field is reflected in the PCI Power Management Data register of function 0 when the Data_Select field is set to 8 (common function).
D3_POWER	23:16	0x0	RW	The value in this field is reflected in the PCI Power Management Data register of the LAN functions for D3 power consumption and dissipation (Data_Select = 3 or 7).
RESERVED	31:24	0x0	RSV	Reserved.

7.2.2.5.9 PCIe Serial Number MAC Address High - PCI_SERH (0x00011078) PCIe_GLUE

Field	Bit(s)	Init.	Type	Description
SER_NUM_H	15:0	0x0	RW	The high Word of the Ethernet MAC address used to generate the PCIe serial number.
RESERVED	31:16	0x0	RSV	Reserved.

7.2.2.5.10 PCIe Capabilities Control - PCI_CAPCTRL (0x00011080) PCIe_GLUE

Determines PCIe capabilities supported by the device and that software is allowed to enable or disable.

Field	Bit(s)	Init.	Type	Description
VPD_EN	0	0b	RW	VPD Enable. 0 = The PCIe VPD Capability is not present and is not exposed. 1 = The PCIe VPD Capability is present and exposed.
RESERVED	31:1	0x0	RSV	Reserved.

7.2.2.5.11 PCIe Capabilities Support - PCI_CAPSUP (0x00011088) PCIe_GLUE

Determines PCIe capabilities supported by the device.

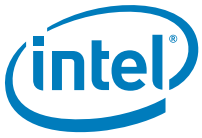
Field	Bit(s)	Init.	Type	Description
PCIE_VER	0	1b	RW	Determines the PCIe capability version. 0 = Capability version: 0x1. 1 = Capability version: 0x2.

Field	Bit(s)	Init.	Type	Description
RESERVED	1	0b	RSV	Reserved.
LTR_EN	2	1b	RW	A value of 1b indicates support for the PCIe Latency Tolerance Reporting (LTR) Capability.
ARI_EN	4	1b	RW	A value of 1b indicates support for the PCIe ARI Capability.
IOV_EN	5	1b	RW	A value of 1b indicates support for the PCIe SR-IOV Capability.
ACS_EN	6	1b	RW	A value of 1b indicates support for the PCIe ACS Capability.
SEC_EN	7	0b	RW	A value of 1b indicates support for the Secondary PCI Express Extended Capability.
RESERVED	14:8	0x0	RSV	Reserved.
ECRC_MCTP_GEN	15	0b	RW	ECRC Generation for MCTP. 0 = Do not add ECRC to MCTP packets even if ECRC is enabled. 1 = Add ECRC to MCTP packets if ECRC is enabled via the ECRC Generation Enable field in PCIe Advanced Error Capabilities and Control Register.
ECRC_GEN_EN	16	1b	RW	Loaded into the ECRC Generation Capable bit of the PCIe Configuration registers.
ECRC_CHK_EN	17	1b	RW	Loaded into the ECRC Check Capable bit of the PCIe Configuration registers.
IDO_EN	18	1b	RW	Enables ID-based ordering (IDO).
MSI_MASK	19	1b	RW	MSI per-vector masking setting. This bit is loaded to the masking bit (bit 8) in the Message Control of the MSI Configuration Capability structure.
CSR_CONF_EN	20	1b	RW	Enables access to CSRs via the PCI Configuration Space. See Section "Configuration Access to Internal Registers and Memories".
RESERVED	29:21	0x0	RSV	Reserved.
LOAD_SUBSYS_ID	30	0b	RW	Load Subsystem IDs. When set to 1b, indicates that the device loads its PCIe subsystem ID and sub-system vendor ID from shared SPI Flash.
LOAD_DEV_ID	31	0b	RW	Load Device ID. When set to 1b, indicates that the device loads its PCI device ID's from shared SPI Flash.

7.2.2.5.12 PCIe Link Capabilities - PCI_LINKCAP (0x00011090) PCIe_GLUE

Determines PCIe Link capabilities supported by the device.

Field	Bit(s)	Init.	Type	Description
RESERVED	5:0	0x00	RSV	Reserved.
MAX_PAYLOAD	8:6	0x2	RW	Max Payload Size Supported. Loaded to the PCIe Device Capabilities Register. Supported values are 000b to 100b (128 bytes to 2 KB). Otherwise, keep HW default.
MAX_LINK_WIDTH	12:9	0x07	RW	Max Link Width. Loaded to the PCIe Link Capabilities Register. Values: 0001b = Limit max link width to x1. 0011b = Limit max link width to x4. 0100b = Limit max link width to x8. 0111b = Do not limit max link width. Negotiate to the max width supported by the link. All other values are reserved.
RESERVED	31:13	0x0	RSV	Reserved.



7.2.2.5.13 PCIe PM Support - PCI_PMSUP (0x000110A0) PCIe_GLUE

This register contains parameters that define PCIe power management support.

Field	Bit(s)	Init.	Type	Description
ASPM_SUP	1:0	0x3	RW	Active State Link PM Support is loaded to the PCIe Link Capabilities register.
L0S_EXIT_LAT	4:2	0x5	RW	L0s Exit Latency. Loaded to the L0s Exit Latency field in the PCIe Link Capabilities register.
L1_EXIT_LAT	7:5	0x4	RW	L1 Exit Latency. Loaded to L1 Exit Latency field in the PCIe Link Capabilities register.
L0S_ACC_LAT	10:8	0x3	RW	Loaded to the Endpoint L0s Acceptable Latency field in the PCIe Device Capabilities register.
L1_ACC_LAT	13:11	0x6	RW	Loaded to the Endpoint L1 Acceptable Latency field in the PCIe Device Capabilities register.
SLOT_CLK	14	1b	RW	Slot Clock Configuration. Loaded to the PCIe Link Status register.
OBFF_SUP	16:15	0x0	RW	Loaded to the OBFF Supported field in the PCIe Device Capabilities 2 register. Must be set to 0b in the shared SPI Flash (OBFF is not supported).
RESERVED	31:17	0x0	RSV	Reserved.

7.2.2.5.14 PCIe Global Config - PCI_GLBL_CNF (0x000110B8) PCIe_GLUE

Field	Bit(s)	Init.	Type	Description
RESERVED	1:0	0x0	RSV	Reserved.
WAKE_PIN_EN	2	0b	RW	When set to 1b, enables the use of PE_WAKE_N pin for PME event in all power states.
PCIE_CLKGATE_DIS	3	1b	RW	When set to 0b, enables dynamic clock gating of the PCIE clocks (LCB, HIU & CSR).
PCIE_CLKGATE_L1_ONLY	4	1b	RW	When set to 1b, and if *PCIE_CLKGATE_DIS* is 0b, the clock gating of the PCIE clocks will be only when the PCIE is in L1 state.
PCIE_PCLK_GATE_EN	5	0b	RW	When set to 1b, and if *PCIE_CLKGATE_DIS* is 0b, the PCIE PCLK is also dynamically gated.
RESERVED	7:6	0x0	RSV	Reserved.
PCIE_CLKGATE_TIMER	15:8	0x10	RW	Clock Gating Idle Timer. This field defines the number of clocks (CSR clock) of idle-detect before gating the PCIE clocks. ***Note:*** It is not recommended to set a value smaller than 0x10.
RESERVED	31:16	0x0	RSV	Reserved.

7.2.2.5.15 PCIe Upper Address - PCI_UPADD (0x000110E8) PCIe_GLUE

This register is used to block PCIe master accesses above some address.

Field	Bit(s)	Init.	Type	Description
RESERVED	0	0b	RSV	Reserved.
ADDRESS	31:1	0x0	RW	Address. Bits [31:1] correspond to bits [63:33] in the PCIe address space, respectively.

7.2.2.5.16 PCIe Serial Number MAC Address Low - PCI_SERL (0x000110F0) PCIe_GLUE

Field	Bit(s)	Init.	Type	Description
SER_NUM_L	31:0	0x0	RW	The low DW of the Ethernet MAC address used to generate the PCIe serial number.

7.2.2.5.17 PCIe Global Config 2 - PCI_CNF2 (0x000110F8) PCIe_GLUE

This register contains global status fields of the PCIe configuration.

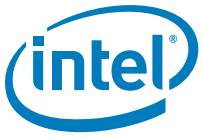
Field	Bit(s)	Init.	Type	Description
RESERVED	0	0b	RSV	Reserved.
CACHELINE_SIZE	1	0b	RW	Cache Line Size. Determines the system cache line size. 0 = 64 bytes 1 = 128 bytes This field is loaded from shared SPI Flash.
MSI_X_PF_N	12:2	0x3F	RW	MSI_X PF Table Size. System software reads this field to determine the MSI-X Table Size N, which is encoded as N-1. This field is loaded from the shared SPI Flash *MSI_X_N_PF* field and reflects the same field in the PCIe MSI-X configuration (Message Control Register)
MSI_X_VF_N	23:13	0x02	RW	MSI_X VF Table Size. System software reads this field to determine the MSI-X Table Size N for VFs, which is encoded as N-1. This field is loaded from the shared SPI Flash *MSI_X_N_VF* field and reflects the same field in the PCIe MSI-X configuration (VF MSI-X Control Register)
NUM_VFS	31:24	0x40	RW	The number of VFs requested by the function. Valid values are 0-64. If value is zero, then SR-IOV capability is hidden.

7.2.2.6 Interrupt Registers

7.2.2.6.1 Extended Interrupt Cause Register - EICR (0x00000800) Interrupt

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.	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*. 3. Filter programming failed due to no space in the flow director table. Note that this case should not happen if the driver handles the FDIRCTRL.*FULL_THRESH* event.
RX_MISS	17	0b	RW1C	Missed packet interrupt is activated for each received packet that overflows the Rx packet buffer (overrun). ***Note:*** The packet is dropped and also increments the associated RXMPC[n] counter.
PCI_EXCEPTION	18	0b	RW1C	The PCI exception is activated by one of the events described in Proprietary Error Reporting section. The specific PCI event is error reported in the PCI_ICAUSE register:



Field	Bit(s)	Init.	Type	Description
MAILBOX	19	0b	RW1C	VF to PF mailbox interrupt. Caused by a VF write access to the PF mailbox or by a malicious event detection
LSC	20	0b	RW1C	Link Status Change. This bit is set each time the link status changes (either from up to down, or from down to up).
LINKSEC	21	0b	RW1C	Indicates that the Tx LinkSec packet counter reached the threshold requiring key exchange.
MNG	22	0b	RW1C	Manageability Event Detected. Indicates that a manageability event happened. When the device is in power-down mode, the BMC might generate a PME for the same events that would cause an interrupt when the device is at the D0 state. This interrupt bit is also used to alert the host when the BMC IP address was changed or when EEC.*MAC_DONE* bit is set by firmware.
TS	23	0b	RW1C	Thermal Sensor Event. Indicates that a thermal sensor trip point was crossed.
TIMESYNC	24	0b	RW1C	TimeSync Interrupt. Indicates that a Time Sync event has occurred. Check TSIM register for precise cause.
GPI_SDP0	25	0b	RW1C	General Purpose Interrupt on SDP0. If GPI interrupt detection is enabled on this pin (via GPIE), this interrupt cause is set when a transition to high is sampled on SDP0.
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 shared SPI Flash. • 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	30	0b	RW1C	TCP Timer Expired. This bit is set when the timer expires.
OTHER_CAUSE	31	0b	ROS	Other Cause Interrupt. Activated when any bit (29:16) in the Extended Interrupt Cause Register (EICR) is set and its relevant mask bit in the EIMS is enabled.

7.2.2.6.2 Extended Interrupt Cause Set Register - EICS (0x00000808) Interrupt

Field	Bit(s)	Init.	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.

7.2.2.6.3 Extended Interrupt Auto Clear Register - EIAC (0x00000810) Interrupt

Note: Bits 29:16 should never be set to auto clear since they share the same MSI-X vector.

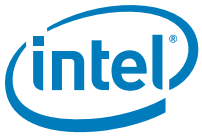
Field	Bit(s)	Init.	Type	Description
RTXQ_AUTO_CLEAR	15:0	0x0	RW	0b = The corresponding bits in the EICR are not auto cleared. 1b = Each bit enables auto clear of the corresponding *RTXQ* bits in the EICR following interrupt assertion.
RESERVED	29:16	0x0	RSV	Reserved.
TCP_TIMER_AUTO_CLEAR	30	0b	RW	0b = Auto clear is not enabled. 1b = Enables auto clear of the TCP timer interrupt cause in the EICR following interrupt assertion.
RESERVED	31	0b	RSV	Reserved.

7.2.2.6.4 Extended Interrupt Throttle Registers - EITR[n] (0x00000820 + 0x4*n, n=0...23 and 0x00012300 + 0x4*(n-24), n=24...128) Interrupt

Mapping of the EITR registers to the MSI-X vectors is described in MSI-X Vectors Mapping to EITR Section (7.3.4.3.3).

Note: Additional address(es): 0x012300 + 4*(n-24), n=24...128.

Field	Bit(s)	Init.	Type	Description
RESERVED	2:0	0x0	RSV	Reserved.
ITR_INTERVAL	11:3	0x0	RW	Minimum inter-interrupt interval specified in 2.048 µs units at 1 GbE and 10 GbE link. At 100 Mb/s link the interval is specified in 20.48 µs units. At 0x0 interrupt throttling is disabled while any event causes an immediate interrupt.
RESERVED	13:12	0x0	RSV	Reserved.
HIGH_PRIORITY	14	0b	RW	High Priority Vector. Setting this bit will cause assertion of this vector to break DMA coalescing.
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.
ITR_COUNTER	27:21	0x0	RW	This field represents the 7 MS bits (out of 9 bits) of the ITR counter. It is a down counter that is loaded with ITR_INTERVAL value each time the associated interrupt is asserted. When the ITR counter reaches zero it stops counting and triggers an interrupt. On a write cycle, software must set this field to 0 if *CNT_WDIS* in this register is cleared (write enable to the ITR counter).
RESERVED	30:28	0x0	RSV	Reserved.
CNT_WDIS	31	0b	RW	Write disable to the LLI credit and ITR counter. 0 = Software must set the LLI credit and ITR counter to zero, which enables an immediate interrupt on packet reception. 1 = The LLI credit and ITR counter are not overwritten by the write access. This bit is write only and always read as zero.



7.2.2.6.5 Extended Interrupt Mask Set/Read Register - EIMS (0x00000880) Interrupt

Field	Bit(s)	Init.	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.

7.2.2.6.6 Extended Interrupt Mask Clear Register - EIMC (0x00000888) Interrupt

Field	Bit(s)	Init.	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.

7.2.2.6.7 Extended Interrupt Auto Mask Enable register - EIAM (0x00000890) Interrupt

Field	Bit(s)	Init.	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:*** If any of the Auto Mask enable bits is set, the GPIE.EIAME bit must be set as well.
RESERVED	31	0b	RSV	Reserved.

7.2.2.6.8 MSIX to EITR Select - EITRSEL (0x00000894) Interrupt

Field	Bit(s)	Init.	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 Section 7.3.4.3.3). 0 = Selects the PF. 1 = Selects the VF.

7.2.2.6.9 General Purpose Interrupt Enable - GPIE (0x00000898) Interrupt

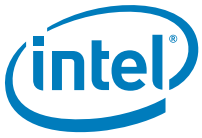
Field	Bit(s)	Init.	Type	Description
RESERVED	0	0b	RSV	Reserved.
SDP0_GPIEN	1	0b	RW	General Purpose Interrupt Detection Enable for SDP0. If software-controllable IO pin SDP0 is configured as an input, this bit (when 1b) enables use for GPI interrupt detection, resulting in setting EICR[25] bit.
SDP1_GPIEN	2	0b	RW	General Purpose Interrupt Detection Enable for SDP1. If software-controllable IO pin SDP1 is configured as an input, this bit (when 1b) enables use for GPI interrupt detection, resulting in setting EICR[26] bit.
SDP2_GPIEN	3	0b	RW	General Purpose Interrupt Detection Enable for SDP2. If software-controllable IO pin SDP2 is configured as an input, this bit (when 1b) enables use for GPI interrupt detection, resulting in setting EICR[27] bit.
MULTIPLE_MSIX	4	0b	RW	MSI-X Mode. Selects between MSI-X interrupts and other interrupt modes. 0 = Legacy and MSI mode (non-MSI-X mode) 1 = MSI-X mode
OCD	5	0b	RW	Other Clear Disable. 0 = The whole EICR is cleared on read. 1 = Indicates that only bits 16...29 of the EICR are cleared on read.

Field	Bit(s)	Init.	Type	Description
EIMEN	6	0b	RW	EICS Immediate Interrupt Enable. 0 = The EICS interrupt waits for EITR expiration. 1 = Setting bit in the EICS causes a Low Latency Interrupt.
RESERVED	10:7	0x00	RSV	Reserved.
RSC_DELAY	13:11	0x0	RW	Delay from RSC completion triggered by ITR and interrupt assertion. In 10 GbE or 1 GbE mode, the delay equals: ("RSC Delay" + 1) x 4 µs = 4, 8, 12... 32 µs In 100 Mb/s mode, the delay equals: ("RSC Delay" + 1) x 40 µs = 40, 80, 120... 320 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 01b = 16 VF mode 10b = 32 VF mode 11b = 64 VF mode
RESERVED	29:16	0x0	RSV	Reserved.
EIAME	30	0b	RW	Extended Interrupt Auto Mask Enable. When set the EIMS register can be auto-cleared (depending on EIAM setting) upon interrupt assertion. In any case, the EIMS register can be auto-cleared (depending on EIAM setting) following a write-to-clear (or read) to the EICR register. Software might set the *EIAME* only in MSI-X mode.
PBA_SUPPORT	31	0b	RW	PBA Support. When set, setting one of the extended interrupts masks via EIMS causes the PBA bit of the associated MSI-X vector to be cleared. Otherwise, the device behaves in a way supporting legacy INT-x interrupts. ***Note:*** Should be cleared when working in INT-x or MSI mode and set in MSI-X mode.

7.2.2.6.10 Interrupt Vector Allocation Registers - IVAR[n] (0x00000900 + 0x4*n, n=0...63) Interrupt

These registers map interrupt causes into EICR entries (legacy/MSI modes) or into MSI-X vectors (MSI-X modes). See Section 7.3.4 for mapping and use of these registers.

Field	Bit(s)	Init.	Type	Description
INT_ALLOC_0	5:0	X	RW	The interrupt allocation for Rx queue ('2xN' for IVAR register 'N').
RESERVED	6	0b	RSV	Reserved.
INT_ALLOC_VAL_0	7	0b	RW	Interrupt Allocation Valid indication for INT_Alloc[0].
INT_ALLOC_1	13:8	X	RW	The interrupt allocation for Tx queue ('2xN' for IVAR register 'N').
RESERVED	14	0b	RSV	Reserved.
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 ('2xN'+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 ('2xN'+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].



7.2.2.6.11 Miscellaneous Interrupt Vector Allocation - IVAR_MISC (0x00000A00) Interrupt

These register maps interrupt causes into MSI-X vectors (MSI-X modes). See Section 7.3.4 for mapping and use of these registers.

Note: The INT_ALLOC_VAL[1] bit default value is one — to enable legacy driver functionality.

Field	Bit(s)	Init.	Type	Description
INT_ALLOC_0	6:0	X	RW	Defines the MSI-X vector assigned to the TCP timer interrupt cause. The value must be in the 0-63 range.
INT_ALLOC_VAL_0	7	0b	RW	Valid bit for INT_Alloc[0].
INT_ALLOC_1	14:8	X	RW	Defines the MSI-X vector assigned to the “Other” interrupt cause. The value must be in the 0-63 range.
INT_ALLOC_VAL_1	15	1b	RW	Valid bit for INT_Alloc[1].
RESERVED	31:16	0x0	RSV	Reserved.

7.2.2.6.12 Extended Interrupt Cause Set Registers 1 - EICS1 (0x00000A90) Interrupt

Field	Bit(s)	Init.	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.

7.2.2.6.13 Extended Interrupt Cause Set Registers 2 - EICS2 (0x00000A94) Interrupt

Field	Bit(s)	Init.	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.

7.2.2.6.14 Extended Interrupt Mask Set/Read Registers - EIMS1 (0x00000AA0) Interrupt

Field	Bit(s)	Init.	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.

7.2.2.6.15 Extended Interrupt Mask Set/Read Registers - EIMS2 (0x00000AA4) Interrupt

Field	Bit(s)	Init.	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.

7.2.2.6.16 Extended Interrupt Mask Clear Registers 1 - EIMC1 (0x00000AB0) Interrupt

Field	Bit(s)	Init.	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.

7.2.2.6.17 Extended Interrupt Mask Clear Registers 2 - EIMC2 (0x00000AB4) Interrupt

Field	Bit(s)	Init.	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.

7.2.2.6.18 Extended Interrupt Auto Mask Enable Registers 1 - EIAM1 (0x00000AD0) Interrupt

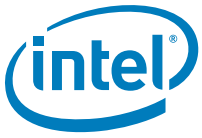
Field	Bit(s)	Init.	Type	Description
AUTO_MASK	31:0	0x0	RW	At 1b each bit enables auto set and clear of its corresponding bits in the EIMS[n]. Bits 15:0 of EIAM1 are mirrored in EIAM bits 15:0. ***Note:*** If any of the Auto Mask enable bits is set, the GPIE.*EIAME* bit must be set as well.

7.2.2.6.19 Extended Interrupt Auto Mask Enable Registers 2 - EIAM2 (0x00000AD4) Interrupt

Field	Bit(s)	Init.	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:*** If any of the Auto Mask enable bits is set, the GPIE.*EIAME* bit must be set as well.

7.2.2.6.20 RSC Enable Interrupt - RSCINT[n] (0x00012000 + 0x4*n, n=0...128) Interrupt

Field	Bit(s)	Init.	Type	Description
RSCEN	0	1b	RW	RSC Enable. This bit enables RSC on the receive queues associated with interrupt vector n.
RESERVED	31:1	0x0	RSV	Reserved.



7.2.2.7 MSI-X Table Registers

The MSI-X capability is described in Section 9.3.8. The MSI-X table is described in Section 9.3.8.2 and the Pending Bit Array (PBA) is described in Section 9.3.8.3. These registers are located in the MSI-X BAR.

7.2.2.7.1 MSI-X PBA Clear - PBACL[n] (0x000110C0 + 0x4*n, n=0...1) PCIe

PBACL[0] is also mapped to address 0x11068 to maintain compatibility with previous products.

Field	Bit(s)	Init.	Type	Description
PENBITCLR	31:0	0x0	RW	MSI-X Pending Bits Clear. Writing 1b to any bit clears it's content. Writing 0b has no effect. Reading this register returns the MSIPBA.*PENBIT* value.

7.2.2.7.2 VF MSI-X PBA Clear - VFPBACL[n] (0x000110C8 + 0x4*n, n=0...5) PCIe

Note: These registers reflects the VFPBACL bits of the VFs. The PBA is a vector of 192 bits. The vector starts at bit 31 of register p=5 and ends at bit 0 of register p=0 (?reverse? ordering). Each VF has 3 bits in this vector while, PENBIT[2...0] of VF=vi are mapped to bits vi * 3... vi * 3 + 2. Explicitly, PENBIT[2] of VF0 is at bit 31 of register p=5 and so on.

Field	Bit(s)	Init.	Type	Description
PENBITCLR	31:0	0x0	RW	MSI-X Pending Bits Clear. Writing 1b to any bit clears it's content. Writing 0b has no effect. Reading this register returns the MSIPBA.*PENBIT* value.

7.2.2.8 Receive Registers

7.2.2.8.1 Receive Checksum Control - RXCSUM (0x00005000) RX_Filter

The Receive Checksum Control register controls the receive checksum off loading features of the device.

Note: This register should only be initialized (written) when the receiver is not enabled (for example, only write this register when RXCTRL.*RXEN* = 0).

Field	Bit(s)	Init.	Type	Description
RESERVED	9:0	0x00	RSV	Reserved.
ICMPV6XSUM	10	1b	RW	ICMPv6 Checksum Enable. 0 = Disable ICMPv6 checksum calculation. 1 = Enable ICMPv6 checksum calculation. ***Note:*** ICMPv6 checksum offload is supported only for packets sent to firmware for Proxying.
RESERVED	11	0b	RSV	Reserved,
IPPCSE	12	0b	RW	IP Payload Checksum Enable,
PCSD	13	0b	RW	RSS/Fragment Checksum Status Selection. The Fragment Checksum and IP Identification fields are mutually exclusive with the RSS hash. Only one of the two options is reported in the Rx descriptor. 0 = The extended descriptor write-back contains the fragment checksum, 1 = The extended descriptor write-back has the RSS field.
RESERVED	31:14	0x0	RSV	Reserved.

7.2.2.8.2 Receive Filter Control Register - RFCTL (0x00005008) RX_Filter

Field	Bit(s)	Init.	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.
RESERVED	31:10	0x0	RSV	Reserved. Must always be 0x0.

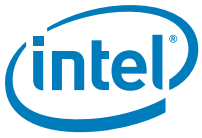
7.2.2.8.3 VXLAN Control - VXLANCTRL (0x0000507C) RX_Filter

Field	Bit(s)	Init.	Type	Description
UDPPORT	15:0	0x0	RW	Defines the UDP port used to identify VXLAN traffic.
RESERVED	31:16	0x0	RSV	Reserved.

7.2.2.8.4 Filter Control Register - FCTRL (0x00005080) RX_Filter

Note: Before receive filters are updated/modified the RXCTRL.RXEN bit should be set to zero. After the proper filters have been set the RXCTRL.RXEN bit can be set to 1 to re-enable the receiver.

Field	Bit(s)	Init.	Type	Description
RESERVED	0	0b	RSV	Reserved.
SBP	1	0b	RW	Store Bad Packets. 0 = Do not store 1 = Store. ***Note:*** CRC errors before the SFD are ignored. Any packet must have a valid SFD (RX_DV with no RX_ER in the XGMII/GMII i/f) to be recognized by the device (even bad packets).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 store-bad- packet bit is set.
RESERVED	6:2	0x0	RSV	Reserved.
TPE	7	0b	RW	Tag Promiscuous Enable. When set, any packet with active tag is accepted. The active tag to accept is defined by the PFVTCTL.*POOLING_MODE* field: PFVTCTL.*POOLING_MODE* = 01b - Accept all packets with E-tag PFVTCTL.*POOLING_MODE* = 10b - Reserved Other - Ignore this bit
MPE	8	0b	RW	Multicast Promiscuous Enable. 0 = Disabled 1 = Enabled When set, all received multicast and broadcast packets pass L2 filtering and can be directed to the MNG or Host by a one of the decision filters.



Field	Bit(s)	Init.	Type	Description
UPE	9	0b	RW	Unicast Promiscuous Enable. 0 = Disabled 1 = Enabled
BAM	10	0b	RW	Broadcast Accept Mode. 0 = Ignore broadcast packets to Host 1 = Accept broadcast packets to Host
RESERVED	31:11	0x0	RSV	Reserved.

7.2.2.8.5 E-tag Ethertype Register - ETAG_ETYPE (0x00005084) RX_Filter

Field	Bit(s)	Init.	Type	Description
ETAG_ETYPE	15:0	0x893F	RW	Etag Ethertype Value.
RESERVED	30:16	0x0	RSV	Reserved.
VALID	31	0b	RW	Valid Bit. 0 = Entry is disabled. 1 = Entry is enabled.

7.2.2.8.6 VLAN Control Register - VLNCTRL (0x00005088) RX_Filter

Field	Bit(s)	Init.	Type	Description
VET	15:0	0x8100	RW	VLAN Ether Type (the VLAN Tag Protocol Identifier - TPID). This register contains the type field hardware that's matched against to recognize an 802.1Q (VLAN) Ethernet packet. For proper operation, software must not change the default setting of this field (802.3ac standard defines it as 0x8100). This field must be set to the same value as the VT field in the DMATXCTL register. ***Note:*** This field appears in little endian order (the upper byte is first on the wire (VLNCTRL.VET[15:8]).
RESERVED	26:16	0x0	RSV	Reserved.
UP_FIRST_TAG_EN	27	1b	RW	If set, the UP used for traffic class determination is taken from the first tag with UP. Otherwise, it is taken from the inner VLAN.
DEI	28	0b	RW	Drop Eligible Indicator Bit Value. If DEIEN is set to 1b, then.1q packets with DEI equal to this field are accepted. Otherwise, the.1q packet is discarded.
DEIEN	29	0b	RW	Drop Eligible Indicator Enable. 0 = Disabled (DEI bit not compared to decide packet acceptance). 1 = Enabled (DEI from packet must match the DEI field to accept a.1q packet).
VFE	30	0b	RW	VLAN Filter Enable. 0 = Disabled (VFTA filter table does not decide packet acceptance). 1 = Enabled (VFTA filter table decides packet acceptance for.1q packets).
RESERVED	31	0b	RSV	Reserved.

7.2.2.8.7 Multicast Control Register - MCSTCTRL (0x00005090) RX_Filter

Field	Bit(s)	Init.	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]

Field	Bit(s)	Init.	Type	Description
MFE	2	0b	RW	Multicast Filter Enable. 0 = The multicast table array filter (MTA[n]) is disabled. 1 = The multicast table array (MTA[n]) is enabled.
RESERVED	31:3	0x0	RSV	Reserved.

7.2.2.8.8 EType Queue Filter - ETQF[n] (0x00005128 + 0x4*n, n=0...7) RX_Filter

Field	Bit(s)	Init.	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).
RESERVED	19:16	0x0	RSV	Reserved.
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.
RESERVED	27	0b	RW	Reserved.
CNM	28	0b	RW	When set, a packet with this EType is identified as a CNM packet.
TX_ANTISPOOF	29	0b	RSV	If set, this Ethertype is a candidate for anti-spoof action on transmitted packets. The actual action applied is set according to the PFVFSPOOF.*ETHERTYPEAS* or PFVFSPOOF.*ETHERTYPELB*.
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.

7.2.2.8.9 Multicast Table Array - MTA[n] (0x00005200 + 0x4*n, n=0...127) RX_Filter

This table should be initialized by software before transmit and receive are enabled.

Field	Bit(s)	Init.	Type	Description
BIT_VECTOR	31:0	X	RW	Word wide bit vector specifying 32 bits in the multicast address filter table. This device provides Multicast filtering for 4096 Multicast Addresses by providing single bit entry per Multicast Address. Those 4096 address locations organized in Multicast Table Array 128 registers of 32 bits each one. Only 12 bits out of 48-bit Destination Address are considered as Multicast Address. Those 12 bits can be selected by *MO* field of MCSTCTRL register. The 7 MS bits of the Ethernet MAC Address (out of the 12 bits) selects the register index, while the 5 LS bits (out of the 12 bits) selects the bit within a register.

7.2.2.8.10 Receive Address Low - RAL_ALIAS[n] (0x00005400 + 0x8*n, n=0...15; RW) RX_Filter

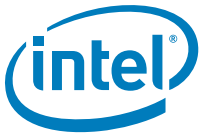
These registers are aliases to the first 16 RAL addresses.

Fields definitions are the same as defined in Section 7.2.2.8.14.

7.2.2.8.11 Receive Address High - RAH_ALIAS[n] (0x00005404 + 0x8*n, n=0...15; RW) RX_Filter

These registers are aliases to the first 16 RAH addresses.

Fields definitions are the same as defined in Section 7.2.2.8.15.



7.2.2.8.12 Multiple Receive Queues Command Register - MRQC_ALIAS (0x00005818; RW) DBU_RX

This is an alias to the MRQC register.

Fields definitions are the same as defined in Section 7.2.2.8.23.

7.2.2.8.13 VLAN Filter Table Array - VFTA[n] (0x0000A000 + 0x4*n, n=0...127) RX_Filter

This table should be initialized by software before transmit and receive are enabled.

Field	Bit(s)	Init.	Type	Description
VLAN_FLT	31:0	X	RW	VLAN Filter. Each bit 'i' in register 'n' affects packets with VLAN VID equal to 32*n+i. 128 VLAN Filter registers compose a table of 4096 bits that cover all possible VIDs. Each bit when set, allows packets with the associated VID to pass. Each bit when cleared, blocks packets with this VID.

7.2.2.8.14 Receive Address Low - RAL[n] (0x0000A200 + 0x8*n, n=0...127) RX_Filter

These registers contain the lower bits of the 48 bit Ethernet MAC Address. All 32 bits are valid. If the shared SPI Flash is present the first register (RAL0) is loaded from the shared SPI Flash.

Note: The first 16 MAC addresses are also mapped to CSR addresses 0x5400 - 0x5478 for backward compatibility.

Field	Bit(s)	Init.	Type	Description
RAL	31:0	X	RW	Receive Address Low. The lower 32 bits of the 48-bit Ethernet MAC Address. ***Note:*** Field is defined in Big Endian (LS byte of RAL is first on the wire).

7.2.2.8.15 Receive Address High - RAH[n] (0x0000A204 + 0x8*n, n=0...127) RX_Filter

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 field is cleared by a master reset.

Note: The first receive address register (RAR0) is also used for exact match pause frame checking (DA matches the first register). RAR0 should always be used to store the individual Ethernet MAC address of the adapter. After reset, if the shared SPI Flash is present, the first register (Receive Address Register 0) is loaded from the IA field in the shared SPI Flash, its Address Select field is 00b, and its Address Valid field is 1b. If no shared SPI Flash is present, the Address Valid field is 0b. The Address Valid field for all of the other registers are 0b. The first 16 MAC addresses are also mapped to CSR addresses 0x5404 - 0x547C for backward compatibility.

Field	Bit(s)	Init.	Type	Description
RAH	15:0	X	RW	Receive Address High. The upper 16 bits of the 48-bit Ethernet MAC Address. ***Note:*** Field is defined in Big Endian (MS byte of RAH is Last on the wire).
RESERVED	29:16	0x0	RSV	Reserved.
ADTYPE	30	0b	RW	Address Type: 0 = MAC 1 = E-tag Fixed to zero in RAH[0].

Field	Bit(s)	Init.	Type	Description
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 shared SPI Flash is present, the Receive Address[0] is loaded from the shared SPI Flash and its Address Valid field is set to 1b after a software or PCI reset or shared SPI Flash read.

7.2.2.8.16 MAC Pool Select Array - MPSAR[n] (0x0000A600 + 0x4*n, n=0...255) RX_Filter

Software should initialize these registers before transmit and receive are enabled.

Field	Bit(s)	Init.	Type	Description
POOL_ENA	31:0	X	RW	Pool Enable Bit Array. Each couple of registers '2*n' and '2*n+1' are associated with Ethernet MAC Address Filter 'n' as defined by RAL[n] and RAH[n]. Each bit when set, enables packet reception with the associated Pools as described 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'.

7.2.2.8.17 Packet Split Receive Type Register - PSRTYPE[n] (0x0000EA00 + 0x4*n, n=0...63) DBU_RX

Registers 0...15 are mapped also to 0x05480 to maintain compatibility with the 82598.

Notes: This register must be initialized by software.

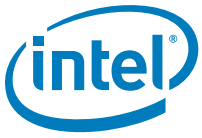
Packets are split according to the lowest-indexed entry that applies to the packet and that is enabled. For example, if bits 4 and 8 are set, then an IPv4 packet that is not TCP is split after the IPv4 header. Exception to this rule is for tunnel packets. If *PSR_TYPE15* (split on tunnel) or *PSR_TYPE16* (Split on outer L2 header) is set, they take precedence on the other filters. At most one of those should be set.

This bit mask table enables or disables each type of header to be split. A value of 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.	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_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.



Field	Bit(s)	Init.	Type	Description
PSR_TYPE13	13	0b	RW	Reserved.
PSR_TYPE14	14	0b	RW	Reserved.
PSR_TYPE15	15	0b	RW	Split on tunnel header
PSR_TYPE16	16	0b	RW	Split on outer L2 header
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 1001b, 1010b, 1011b, 1110b or 1111b.

7.2.2.8.18 Redirection Table - RETA[n] (0x0000EB00 + 0x4*n, n=0...31) DBU_RX

RETA is also mapped to addresses 0x05C00-0x05C7C to maintain compatibility with the 82598. The redirection table has 128-entries in 32 registers.

Note: The contents of the redirection table are not defined following reset of the Memory Configuration Registers. System software must initialize the table prior to enabling multiple receive queues. It might also update the redirection table during run time. Such updates of the table are not synchronized with the arrival time of received packets. Therefore, it is not guaranteed that a table update takes effect on a specific packet boundary.

Field	Bit(s)	Init.	Type	Description
ENTRY0	5:0	X	RW	Entry0 defines the RSS output index for hash value '4 4*n+0', where 'n' is the register index equal to 0...31.
RESERVED	7:6	0x0	RSV	Reserved.
ENTRY1	13:8	X	RW	Entry1 defines the RSS output index for hash value '4*n+1', where 'n' is the register index equal to 0...31.
RESERVED	15:14	0x0	RSV	Reserved.
ENTRY2	21:16	X	RW	Entry2 defines the RSS output index for hash value '4*n+2', where 'n' is the register index equal to 0...31.
RESERVED	23:22	0x0	RSV	Reserved.
ENTRY3	29:24	X	RW	Entry3 defines the RSS output index for hash value '4*n+3', where 'n' is the register index equal to 0...31.
RESERVED	31:30	0x0	RSV	Reserved.

7.2.2.8.19 RSS Random Key Register - RSSRK[n] (0x0000EB80 + 0x4*n, n=0...9) DBU_RX

RSSRK is also mapped to addresses 0x05C80-0x05CA4 to maintain compatibility with the 82598. The RSS Random Key is 40 bytes wide (see RSS hash in Section 7.1.2.8.1).

Note: Additional address(es): 0x05C80 + 4*n, n=0...9

Field	Bit(s)	Init.	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'.

Field	Bit(s)	Init.	Type	Description
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'.

7.2.2.8.20 EType Queue Select - ETQS[n] (0x0000EC00 + 0x4*n, n=0...7) DBU_RX

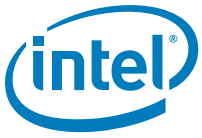
Field	Bit(s)	Init.	Type	Description
RESERVED	15:0	0x00	RSV	Reserved.
RX_QUEUE	22:16	0x00	RW	Identifies the Rx queue associated with this EType.
RESERVED	30:23	0x00	RSV	Reserved.
QUEUE_ENABLE	31	0b	RW	When set, enables queueing of Rx packets by the EType defined in the matching ETQF register to the queue indicated in this register.

7.2.2.8.21 SYN Packet Queue Filter - SYNQF (0x0000EC30) DBU_RX

Field	Bit(s)	Init.	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	31:8	0x0	RSV	Reserved.

7.2.2.8.22 RSS queues per Traffic class register - RQTC (0x0000EC70) DBU_RX

Field	Bit(s)	Init.	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.*MRQE* = 0100b or 0101b. A value of 7 in this field is not valid and is considered as if zero is written.
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.*MRQE* = 0100b or 0101b. A value of 7 in this field is not valid and is considered as if zero is written.
RESERVED	7	0b	RSV	Reserved.
RQTC2	10:8	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.*MRQE* = 0100b or 0101b. A value of 7 in this field is not valid and is considered as if zero is written.
RESERVED	11	0b	RSV	Reserved.
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.*MRQE* = 0100b or 0101b. A value of 7 in this field is not valid and is will be considered as if zero is written.
RESERVED	15	0b	RSV	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.*MRQE* = 0100b. A value of 7 in this field is not valid and is considered as if zero is written.



Field	Bit(s)	Init.	Type	Description
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.*MRQE* = 0100b. A value of 7 in this field is not valid and is considered as if zero is written.
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.*MRQE* = 0100b. A value of 7 in this field is not valid and is considered as if zero is written.
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.*MRQE* = 0100b. A value of 7 in this field is not valid and is considered as if zero is written.
RESERVED	31	0b	RSV	Reserved.

7.2.2.8.23 Multiple Receive Queues Command Register - MRQC (0x0000EC80) DBU_RX

MRQC is also mapped to address 0x05818 to maintain compatibility with the 82598.

Field	Bit(s)	Init.	Type	Description
MRQE	3:0	0x0	RW	Multiple Receive Queues Enable Defines allocation of the Rx queues per RSS and virtualization. 0000b = RSS disabled. 0001b = RSS only - Single set of RSS 64 queues. 0010b = Reserved. 0011b = Reserved. 0100b = Reserved. 0101b = Reserved. 0110b = Reserved. 0111b = Reserved. 1000b = Virtualization only - 64 pools, no RSS, each pool allocated 2 queues. 1001b = Virtualization and RSS - 62 pools, each allocated 2 RSS queues - 1 pool allocated 4 queues. 1010b = Virtualization and RSS - 32 pools, each allocated 4 RSS queues. 1011b = Virtualization and RSS - 64 pools, each allocated 2 RSS queues. 1100b = Reserved. 1101b = Reserved. 1110b = Reserved. 1111b = Virtualization and RSS - 60 pools, each allocated 2 RSS queues - 2 pools, each allocated 4 queues.
RESERVED	12:4	0x0	RSV	Reserved.
MULTIPLE_RSS	13	0b	RW	Multiple RSS Enable. 0 = The device uses a single RSS Key (Legacy) 1 = The device uses up to 64 RSS Keys (one per pool). Each RSS has a redirection table with 64 entry of 2 bits each. This bit is relevant only if MRQC.*MRQE* = 1001b, 1010b, 1011b, 1110b, or 1111b
RSC_DIS_LP	14	0b	RW	RSC Disable LLI & Push Coalescing. If set, RSC does not coalesce packets with LLI or PSH.
RESERVED	15	0b	RW	Reserved.

Field	Bit(s)	Init.	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:*** On Tunnel packets IPv4-IPv6 only, the IPv4 header might be used for the RSS filtering.

7.2.2.8.24 Extended Redirection Table - ERETA[n] (0x0000EE80 + 0x4*n, n=0...95) DBU_RX

The extended redirection table adds 384-entries to extend RETA in 96 registers.

Note: The contents of the extended redirection table are not defined following reset of the Memory Configuration Registers. System software must initialize the table prior to enabling multiple receive queues. It might also update the redirection table during run time. Such updates of the table are not synchronized with the arrival time of received packets. Therefore, it is not guaranteed that a table update takes effect on a specific packet boundary.

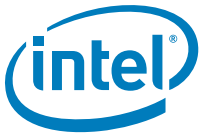
Field	Bit(s)	Init.	Type	Description
ENTRY0	5:0	X	RW	Entry0 defines the RSS output index for hash value '4*[n+32]', where 'n' is the register index equal to 0...95.
RESERVED	7:6	0x0	RSV	Reserved.
ENTRY1	13:8	X	RW	Entry1 defines the RSS output index for hash value '4*[n+32]+1', where 'n' is the register index equal to 0...95.
RESERVED	15:14	0x0	RSV	Reserved.
ENTRY2	21:16	X	RW	Entry2 defines the RSS output index for hash value '4*[n+32]+2, where 'n' is the register index equal to 0...95.
RESERVED	23:22	0x0	RSV	Reserved.
ENTRY3	29:24	X	RW	Entry3 defines the RSS output index for hash value '4*[n+32]+3', where 'n' is the register index equal to 0...95.
RESERVED	31:30	0x0	RSV	Reserved.

7.2.2.8.25 Per Pool RSS Random Key Register - VFRSSRK[n,m] (0x00018000 + 0x4*n + 0x40*m, n=0...15, m=0...63) DBU_RX

The RSS Random Key is 40 bytes wide.

Note: Only the 10 first registers in each VF array are implemented (0x00 - 0x24).

Field	Bit(s)	Init.	Type	Description
K0	7:0	0x0	RW	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'.



Field	Bit(s)	Init.	Type	Description
K3	31:24	0x0	RW	RSS Key Byte '4*n+3' of the RSS random key, for each register 'n'.

7.2.2.8.26 Per Pool Redirection Table - VFRETA[n,m] (0x00019000 + 0x4*n + 0x40*m, n=0...15, m=0...63) DBU_RX

The redirection table has 64 entries in 16 registers.

Note: The content of the redirection tables is not defined following reset of the Memory Configuration registers. System software must initialize the table prior to enabling multiple receive queues. It might also update the redirection table during run time. Such updates of the table are not synchronized with the arrival time of received packets. Therefore, it is not guaranteed that a table update takes effect on a specific packet boundary.

Field	Bit(s)	Init.	Type	Description
ENTRY0	1:0	X	RW	Entry0 defines the RSS output index for hash value '4 4*n+0', where 'n' is the register index equal to 0...31.
RESERVED	7:2	0x0	RSV	Reserved.
ENTRY1	9:8	X	RW	Entry1 defines the RSS output index for hash value '4*n+1', where 'n' is the register index equal to 0...31.
RESERVED	15:10	0x0	RSV	Reserved.
ENTRY2	17:16	X	RW	Entry2 defines the RSS output index for hash value '4*n+2', where 'n' is the register index equal to 0...31.
RESERVED	23:18	0x0	RSV	Reserved.
ENTRY3	25:24	X	RW	Entry3 defines the RSS output index for hash value '4*n+3', where 'n' is the register index equal to 0...31.
RESERVED	31:26	0x0	RSV	Reserved.

7.2.2.9 Receive DMA Registers

7.2.2.9.1 Receive Descriptor Base Address Low - RDBAL[n] (0x00001000 + 0x40*n, n=0...63 and 0x0000D000 + 0x40*(n-64), n=64...127) DMA_RX

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 128-byte aligned block of data.

Field	Bit(s)	Init.	Type	Description
ZERO	6:0	0x0	RW	Ignored on writes. Returns 0 on reads.
RDBAL	31:7	X	RW	Receive Descriptor Base Address Low.

7.2.2.9.2 Receive Descriptor Base Address High - RDBAH[n] (0x00001004 + 0x40*n, n=0...63 and 0x0000D004 + 0x40*(n-64), n=64...127) DMA_RX

This register contains the upper 32 bits of the 64-bit Descriptor base address.

Field	Bit(s)	Init.	Type	Description
RDBAH	31:0	X	RW	Receive Descriptor Base Address [63:32].

7.2.2.9.3 Receive Descriptor Length - RDLEN[n] (0x00001008 + 0x40*n, n=0...63 and 0x0000D008 + 0x40*(n-64), n=64...127) DMA_RX

Note: Additional address(es): 0x0D008 + 0x40*(n-64), n=64...127.

Field	Bit(s)	Init.	Type	Description
LEN	19:0	0x0	RW	Descriptor Ring Length. This register sets the number of bytes allocated for descriptors in the circular descriptor buffer. It must be 128-byte aligned (7 LS bits must be set to zero). ***Note:*** Validated lengths up to 128 K (8 K descriptors).
RESERVED	31:20	0x0	RSV	Reserved. Reads as 0. Should be written as 0 for future compatibility.

7.2.2.9.4 Receive Descriptor Head - RDH[n] (0x00001010 + 0x40*n, n=0...63 and 0x0000D010 + 0x40*(n-64), n=64...127) DMA_RX

Note: Additional address(es): 0x0D010 + 0x40*(n-64), n=64...127.

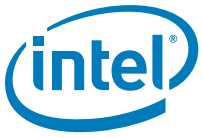
Field	Bit(s)	Init.	Type	Description
RDH	15:0	0x0	RO	Receive Descriptor Head. This register holds the head pointer for the receive descriptor buffer in descriptor units (16-byte datum). The RDH is controlled by hardware.
RESERVED	31:16	0x0	RSV	Reserved. Should be written as 0.

7.2.2.9.5 Split Receive Control Registers - SRRCTL[n] (0x00001014 + 0x40*n, n=0...63 and 0x0000D014 + 0x40*(n-64), n=64...127) DMA_RX

SRRCTL[0...15] are also mapped to address 0x02100... to maintain compatibility with the 82598.

Note: 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 010b, 011b, 100b or 101b.

Field	Bit(s)	Init.	Type	Description
BSIZEPACKET	4:0	0x2	RW	Receive Buffer Size for Packet Buffer. The value is in 1 KB resolution. Value can be from 1 KB to 16 KB. Default buffer size is 2 K. This field should not be set to 0. This field should be greater or equal to 2 in queues which RSC is enabled.
RESERVED	7:5	0x0	RSV	Reserved. Should be written as 0 to ensure future compatibility.
BSIZEHEADER	13:8	0x4	RW	Receive Buffer Size for Header Buffer. The value is in 64-byte resolution. Value can be from 64 bytes to 1024 bytes. The maximum supported header size is limited to 1023. Default buffer size is 256 bytes. This field must be greater than 0 if the value of DESCTYPE is greater or equal to 2. Values above 1024 bytes are reserved for internal use only.
RESERVED	21:14	0x0	RSV	Reserved.
RDMTS	24:22	0x0	RW	Receive Descriptor Minimum Threshold Size. An interrupt associated with this queue is asserted whenever the number of free descriptors is decreased to RDMTS * 64 (this event is considered as Rx ring buffer “almost empty”).



Field	Bit(s)	Init.	Type	Description
DESCTYPE	27:25	0x0	RW	Defines the descriptor type in Rx. 000b = Legacy. 001b = Advanced descriptor one buffer. 010b = Advanced descriptor header splitting. 011b = Reserved. 100b = Reserved. 101b = Advanced descriptor header splitting always use header buffer. 110b = Reserved. 111b = Reserved.
DROP_EN	28	0b	RW	Drop Enabled. If set to 1, 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 0 to ensure future compatibility.

7.2.2.9.6 Receive Descriptor Tail - RDT[n] (0x00001018 + 0x40*n, n=0...63 and 0x0000D018 + 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. Additional address(es): 0x0D018 + 0x40*(n-64), n=64...127

Note: Software should write an even number to the tail register, if the Packet Split feature is used. The tail pointer should be set to one descriptor beyond the last empty descriptor in Host Descriptor Ring.

Field	Bit(s)	Init.	Type	Description
RDT	15:0	0x0	RW	Receive Descriptor Tail.
RESERVED	31:16	0x0	RSV	Reserved. Reads as 0. Should be written as 0 for future compatibility.

7.2.2.9.7 Receive Descriptor Control - RXDCTL[n] (0x00001028 + 0x40*n, n=0...63 and 0x0000D028 + 0x40*(n-64), n=64...127) DMA_RX

Note: Additional address(es): 0x0D028 + 0x40*(n-64), n=64...127.

Field	Bit(s)	Init.	Type	Description
RLPML	13:0	0x0	RW	Long Packet Size Filter defined in bytes. Packets larger than the 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. ***Note:*** The packet size compared to RLMPL includes any optional timestamp added into the packet.
RESERVED	14	0b	RSV	Reserved. Software might Read and Write maintaining backward compatibility.
RLPML_EN	15	0b	RW	Enable Long Packet Size Filter (1b = enable).
RESERVED	22:16	0x00	RSV	Reserved. Software might Read and Write maintaining backward compatibility.
RESERVED	24:23	0x0	RSV	Reserved.

Field	Bit(s)	Init.	Type	Description
ENABLE	25	0b	RW	Receive Queue Enable. When set, this bit enables the operation of the specific receive queue. On read, gets the actual status of the queue (internal indication that the queue is actually enabled/ disabled).
RESERVED	29:26	0x00	RSV	Reserved.
VME	30	0b	RW	VLAN Mode Enable. 0 = Do not strip VLAN tag. 1 = Strip VLAN tag from received 802.1Q packets destined to this queue.
RESERVED	31	0b	RSV	Reserved.

7.2.2.9.8 RSC Control - RSCCTL[n] (0x0000102C + 0x40*n, n=0...63 and 0x0000D02C + 0x40*(n-64), n=64...127) DMA_RX

Note: Additional address(es): 0x0D02C + 0x40*(n-64), n=64...127.

Field	Bit(s)	Init.	Type	Description
RSCEN	0	0b	RW	RSC Enable. When the RSCEN bit is set, RSC is enabled on this queue.
RESERVED	1	0b	RSV	Reserved.
MAXDESC	3:2	0x0	RW	Maximum Descriptors per Large receive as follows: 00b = Maximum of 1 descriptor per Large Receive. 01b = Maximum of 4 descriptors per Large Receive. 10b = Maximum of 8 descriptors per Large Receive. 11b = Maximum of 16 descriptors per Large Receive. ***Note:*** MAXDESC * SRRCTL.*BSIZEPKT* must not exceed 64 KB minus 1, which is the maximum total length in the IP header, and must be larger than expected received MSS.
RESERVED	31:4	0x00	RSV	Reserved.

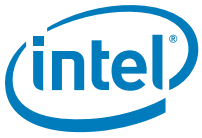
7.2.2.9.9 Split Receive Control Registers - SRRCTL_ALIAS[n] (0x00002100 + 0x4*n, n=0...15; RW) DMA_RX

SRRCTL[0...15] are also mapped to address 0x02100... to maintain compatibility with the 82598.

Fields definitions are the same as defined in Section 7.2.2.9.5.

7.2.2.9.10 Receive DMA Control Register - RDRXCTL (0x00002F00) DMA_RX

Field	Bit(s)	Init.	Type	Description
RESERVED	0:1	0x0	RSV	Reserved.
PSP	2	0b	RW	Pad Small Receive Packets. If this field is set, in virtualized operating mode, strip CRC (HLREG0.*RXCRCSTRP*) should be set.
DMAIDONE	3	0b	RO	DMA Init Done. When read as 1 indicates that the DMA init cycle is done (RO).
RESERVED	4	0b	RSV	Reserved.
RSC_PUSH_DIS	5	0b	RW	Disable coalescing of packets with PUSH flag asserted.
RESERVED	7:6	0x0	RSV	Reserved.
RESERVED	27:9	0x0	RSV	Reserved.
MBINTEN	28	0b	RW	Malicious Behavior Interrupt Enable for DMA Rx errors.



Field	Bit(s)	Init.	Type	Description
MDP_EN	29	0b	RW	Malicious Driver Protection Enable for DMA Rx. 0 = Mechanism is disabled. 1 = Mechanism is enabled.
RESERVED	31:30	0x0	RSV	Reserved.

7.2.2.9.11 Receive Control Register - RXCTRL (0x00003000) DBU_RX

Field	Bit(s)	Init.	Type	Description
RXEN	0	0b	RW	Receive Enable. When set to 0, packets are not received.
RESERVED	31:1	0x0	RSV	Reserved.

7.2.2.9.12 Receive Packet Buffer Size - RXPBSIZE[n] (0x00003C00 + 0x4*n, n=0...7) DBU_RX

Field	Bit(s)	Init.	Type	Description
RESERVED	9:0	0x0	RSV	Reserved.
SIZE	19:10	0x180	RW	Receive Packet Buffer Size for traffic class 'n', where 'n' is the register index. The size is defined in KB units and the default size of PB[0] is 384 KB. The default size of PB[1-7] is also 384 KB. Other possible setting of 4 x TCs is 0x60 (96 KB) for all PB[0-3] and 0x0 for PB[4-7]. See section 3.7.5.3.5 (Packet Buffer Size) for other optional settings with/without the flow director filters. ***Note:*** In any setting the RXPBSIZE[0] must always be enabled (greater than zero). Default value is 0x180 (384 KB).
RESERVED	31:20	0x0	RSV	Reserved.

7.2.2.10 Transmit Registers

7.2.2.10.1 Tx Packet Buffer Threshold - TXPBTHRESH[n] (0x00004950 + 0x4*n, n=0...7) DMA_TX

Field	Bit(s)	Init.	Type	Description
THRESH	9:0	0x96	RW	Threshold used for checking room place in the Tx packet buffer of TCn. Threshold is in KB units when the packet buffer is filled up with payload over that threshold; no more data read requests are sent. Default values: 0x96 (150 KB) for TXPBSIZE0 0x0 (0 KB) for TXPBSIZE1-7 It should be set to: Packet Buffer Size - MSS For example, if the packet buffer size is set to 20 KB in the corresponding TXPBSIZE.*SIZE* and if an MSS of 9.5 KB Jumbo frames is supported for TCn, it is set to 0xA (10 KB).
RESERVED	31:10	0x0	RSV	Reserved.

7.2.2.10.2 DMA Tx Control - DMATXCTL (0x00004A80) DMA_TX

Field	Bit(s)	Init.	Type	Description
TE	0	0b	RW	Transmit Enable. When set, this bit enables the transmit operation of the DMA-Tx.

Field	Bit(s)	Init.	Type	Description
RESERVED	2:1	0x0	RSV	Reserved
GDV	3	0b	RW	Global Double VLAN Mode. When set, this bit enables the Double VLAN mode. Should be set to the same value as CTRL_EXT.*EXTENDED_VLAN*.
RESERVED	4	1b	RW	Reserved.
MDP_EN	5	0b	RW	Malicious Driver Protection Enable for DMA Tx. 0 = Mechanism is disabled. 1 = Mechanism is enabled.
MBINTEN	6	0b	RW	Malicious Behavior interrupt enable for DMA Tx errors.
RESERVED	15:7	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.

7.2.2.10.3 DMA Tx TCP Flags Control Low - DTXTCPFLGL (0x00004A88) DMA_TX

This register holds the "mask" bits for the TCP flags in Tx segmentation (described in Section 7.2.4.7.1 and Section 7.2.4.7.2).

Field	Bit(s)	Init.	Type	Description
TCP_FLG_FIRST_SEG	11:0	0xFF6	RW	TCP Flags First Segment. Bits that make AND operation with the TCP flags at TCP header in the first segment.
RESERVED	15:12	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.

7.2.2.10.4 DMA Tx TCP Flags Control High - DTXTCPFLGH (0x00004A8C) DMA_TX

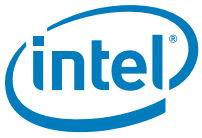
This register holds the "mask" bits for the TCP flags in Tx segmentation (described in Section 7.2.4.7.3).

Field	Bit(s)	Init.	Type	Description
TCP_FLG_LST_SEG	11:0	0xF7F	RW	TCP Flags Last Segment. Bits that make AND operation with the TCP flags at TCP header in the last segment.
RESERVED	31:12	0x00	RSV	Reserved.

7.2.2.10.5 Transmit Descriptor Base Address Low - TDBAL[n] (0x00006000 + 0x40*n, n=0...127) DMA_TX

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.	Type	Description
ZERO	6:0	0x0	RW	Ignored on writes. Returns 0 on reads.



Field	Bit(s)	Init.	Type	Description
TDBAL	31:7	X	RW	Transmit Descriptor Base Address Low.

7.2.2.10.6 Transmit Descriptor Base Address High - TDBAH[n] (0x00006004 + 0x40*n, n=0...127) DMA_TX

Note: This register contains the upper 32 bits of the 64-bit Descriptor base address.

Field	Bit(s)	Init.	Type	Description
TDBAH	31:0	X	RW	Transmit Descriptor Base Address [63:32].

7.2.2.10.7 Transmit Descriptor Length - TDLEN[n] (0x00006008 + 0x40*n, n=0...127) DMA_TX

Field	Bit(s)	Init.	Type	Description
LEN	19:0	0x0	RW	Descriptor Ring Length. This register sets the number of bytes allocated for descriptors in the circular descriptor buffer. It must be 128 -byte aligned (7 LS bits must be set to zero). ***Note:*** Validated lengths up to 128 K (8 K descriptors).
RESERVED	31:20	0x0	RSV	Reserved. Reads as 0. Should be written as 0.

7.2.2.10.8 Transmit Descriptor Head - TDH[n] (0x00006010 + 0x40*n, n=0...127) DMA_TX

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.	Type	Description
TDH	15:0	0x0	RO	Transmit Descriptor Head.
RESERVED	31:16	0x0	RSV	Reserved. Should be written with 0.

7.2.2.10.9 Transmit Descriptor Tail - TDT[n] (0x00006018 + 0x40*n, n=0...127) DMA_TX

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.	Type	Description
TDT	15:0	0x0	RW	Transmit Descriptor Tail.
RESERVED	31:16	0x0	RSV	Reserved. Reads as 0. Should be written as 0 for future compatibility.

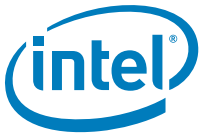
7.2.2.10.10 Transmit Descriptor Control - TXDCTL[n] (0x00006028 + 0x40*n, n=0...127) DMA_TX

Notes: This register controls the fetching and write back of transmit descriptors. The three threshold values are used to determine when descriptors are read from and written to host memory.

When *WTHRESH* = 0b, only descriptors with the RS bit set are written back.

For *PTHRESH* and *HTHRESH* recommended settings, see Section 7.2.3.4.

Field	Bit(s)	Init.	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 are in the on-chip buffer. If this number drops below *PTHRESH*, the algorithm considers pre-fetching descriptors from host memory. However, this fetch does not happen unless there are at least *HTHRESH* valid descriptors in host memory to fetch. ***Note:*** *HTHRESH* should be given a non-zero value each time *PTHRESH* is used.
RESERVED	7	0b	RSV	Reserved.
HTHRESH	14:8	0x00	RW	Host Threshold.
RESERVED	15	0b	RSV	Reserved.



Field	Bit(s)	Init.	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 a write-back threshold is 0b, descriptors are normally written back as soon as they are processed. *WTHRESH* must be written to a non-zero value to take advantage of the write back bursting capabilities. ***Note:*** When *WTHRESH* is set to a non-zero value, the software device driver should not set the RS bit in the Tx descriptors. When *WTHRESH* is set to zero, the software device driver should set the RS bit in the last Tx descriptors of every packet (if TSO is the last descriptor of the entire large send). ***Note:*** When head write back is enabled (TDWBAL[n].*HEAD_WB_EN* = 1b), the *WTHRESH* must be set to zero.
RESERVED	24:23	0x0	RSV	Reserved.
ENABLE	25	0b	RW	Transmit Queue Enable. When set, this bit enables the operation of a specific transmit queue. The default value for all queues is 0b. Setting this bit initializes all the internal registers of a specific queue. Until then, the state of the queue is kept and can be used for debug purposes. When disabling a queue, this bit is cleared only after all activity at the queue stopped. ***Note:*** When setting the global Tx enable DMATXCTL.*TE*, the Enable bit of Tx queue zero is enabled as well. ***Note:*** Following a write cycle by software, this bit is set by hardware only when the queue is actually enabled. When read, gets the actual status of the queue (internal indication that the queue is actually enabled/disabled).
SWFLSH	26	0b	RW	Transmit Software Flush. This bit enables software to trigger descriptor write-back flushing, independent of other conditions. This bit is self cleared by hardware.
RESERVED	31:27	0x0	RSV	Reserved.

7.2.2.10.11 Tx Descriptor Completion Write Back Address Low - TDWBAL[n] (0x00006038 + 0x40*n, n=0...127) DMA_TX

Field	Bit(s)	Init.	Type	Description
HEAD_WB_EN	0	0b	RW	Head Write-Back Enable. 0 = Head Write-back is disabled. 1 = Head Write-back is enabled. When *HEAD_WB_EN* is set, Tx descriptors are not written back.
RESERVED	1	0b	RW	Reserved.
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 00b, meaning that the actual address is QWORD aligned.

7.2.2.10.12 Tx Descriptor Completion Write Back Address High - TDWBAH[n] (0x0000603C + 0x40*n, n=0...127) DMA_TX

Field	Bit(s)	Init.	Type	Description
HEADWB_HIGH	31:0	0x0	RW	Highest 32 bits of Head Write-back memory location (for 64-bit addressing).

7.2.2.10.13 DMA Tx TCP Max Allow Size Requests - DTXMXSZRQ (0x00008100) DMA_TX

This register limits the total number of data bytes that might be in outstanding PCIe requests from the host memory. This allows requests to send low latency packets to be serviced in a timely manner, as this request is serviced right after the current outstanding requests are completed.

Field	Bit(s)	Init.	Type	Description
MAX_BYTES_NUM_REQ	11:0	0x10	RW	Max Allowed Number of Bytes Requests. The maximum allowed amount of 256 bytes outstanding requests. If the total size request is higher than the amount in the field no arbitration is done and no new packet is requested.
RESERVED	31:12	0x0	RSV	Reserved.

7.2.2.10.14 Multiple Transmit Queues Command Register - MTQC (0x00008120) DMA_TX

Notes: For permitted value and functionality of *VT_ENA*, and *NUM_TC_OR_Q*. For Tx queue assignment in VT mode, refer to Tx Queuing Schemes Table in Tx Queues Assignment Section.

This register can be modified only as part of the init phase.

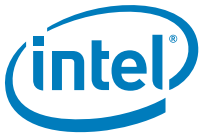
Field	Bit(s)	Init.	Type	Description
RESERVED	0	0b	RW	Reserved.
VT_ENA	1	0b	RW	Virtualization Enabled Mode. When set, the device supports either 16, 32, or 64 Pools. This bit should be set the same as PFVTCTL.*VT_ENA*.
NUM_TC_OR_Q	3:2	0x0	RW	Number of TCs or Number of Tx Queues per Pools. See functionality in the Tx Queues Assignment section
RESERVED	31:4	0x0	RSV	Reserved.

7.2.2.10.15 Transmit Packet Buffer Size - TXPBSIZE[n] (0x0000CC00 + 0x4*n, n=0...7) DBU_TX

Field	Bit(s)	Init.	Type	Description
RESERVED	9:0	0x0	RSV	Reserved.
SIZE	19:10	0xA0	RW	Transmit Packet Buffer Size of TCn. At default, only packet buffer 0 is enabled and TXPBSIZE values for TC 1-7 are meaningless. 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). The default value is 0xA0 (160 KB).
RESERVED	31:20	0x0	RSV	Reserved.

7.2.2.10.16 Manageability Transmit TC Mapping - MNGTXMAP (0x0000CD10) DBU_TX

Field	Bit(s)	Init.	Type	Description
MAP	2:0	0x0	RW	MAP value indicates the TC that the transmit Manageability traffic is routed to.



Field	Bit(s)	Init.	Type	Description
RESERVED	31:3	0x0	RSV	Reserved.

7.2.2.10.17 Tags Ethertypes - TAG_ETYPE (0x00017100) DMA_TX

Field	Bit(s)	Init.	Type	Description
ETAG_ETHERTYPE	15:0	0x893F	RW	The EtherType to use to identify E-tags.
RESERVED	31:16	0x0	RSV	Reserved.

7.2.2.11 Timers Registers

7.2.2.11.1 TCP Timer - TCPTIMER (0x0000004C) Target

Field	Bit(s)	Init.	Type	Description
DURATION	7:0	0x0	RW	Duration of the TCP interrupt interval, in ms.
KICKSTART	8	0b	RW	Counter Kick-start. 0 = No effect. 1 = Kick-starts the counter down-count from the initial value defined in *DURATION* field.
TCPCOUNTEN	9	0b	RW	TCP Count Enable 0 = TCP timer counting is disabled. 1 = TCP timer counting is enabled. Upon enabling, TCP counter counts from its internal state. If the internal state is equal to zero, down-count does not restart until *KICKSTART* is activated. If the internal state is not 0b, down-count continues from the internal state. This enables a pause of the counting for debug purposes.
TCPCOUNTFINISH	10	0b	RW	TCP Count Finish. This bit enables software to trigger a TCP timer interrupt, regardless of the internal state. 0 = No effect. 1 = Triggers an interrupt, and resets the internal counter to its initial value. Down-count does not restart until either *KICKSTART* is activated or *LOOP* is set.
LOOP	11	0b	RW	TCP Loop. 0 = TCP counter stops at zero value, and dies not re-start until *KICKSTART* is activated. 1 = TCP counter reloads duration each time it reaches zero, and continues down-counting from this point without kick-starting.
RESERVED	31:12	0x0	RSV	Reserved.

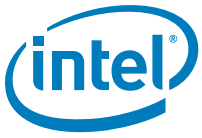
7.2.2.12 Flow Director Registers

These registers are used to control the flow director functionality.

7.2.2.12.1 Flow Director Filters Control Register - FDIRCTRL (0x0000EE00) DBU_RX

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.	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 (8K minus 1 signature filters or 2K minus 1 perfect match filters). 10b = 128 K (16K minus 1 signature filters or 4K minus 1 perfect match filters). 11b = 256 KB (32K minus 1 signature filters or 8K minus 1 perfect match filters).
RESERVED	2	0b	RSV	Reserved.
INIT_DONE	3	0b	RO	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. 0 = Hardware supports Signature filters according to the *PBALLOC*. 1 = Hardware supports perfect match filters according to the *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:*** The Flow Director filter Status and Error bits in the Extended Status and Error fields in the Rx descriptor are always enabled.
RESERVED	6	0b	RSV	Reserved.
REPORT_STATUS_ALWAYS	7	0b	RW	Report 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 1b 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.
DROP_NO_MATCH	15	0b	RW	If set, send to the *DROP_QUEUE* packets that where candidate for Flow Director and did not match any filters. Candidature of packets is defined in Candidate for Flow Director per mode table in Flow Director Filters section.
FLEX_OFFSET	20:16	0x0	RW	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.
FILTERMODE	23:21	0x0	RW	Filter Mode. Defines the fields on which the Flow director acts: 000b = IPMODE — L3/L4 tuple. 001b = MACVLANMODE — Acts on the MAC and VLAN 010b = Cloud: NVGRE — Based on TNI, inner MAC, inner VLAN. VXLAN — Based on VNI, inner MAC, inner VLAN. All other values are reserved.



Field	Bit(s)	Init.	Type	Description
MAX_LENGTH	27:24	0x0	RW	Max Linked List Length. This field defines the maximum recommended linked list associated with any Hash value. Packets that match filters that exceed *MAX_LENGTH* are reported with an active "Length" bit in the Extended Error field. In addition, "Drop" filters that exceed the *MAX_LENGTH* are posted to the Rx queue defined in the filter context rather than the *DROP_QUEUE* defined in this register. *MAX_LENGTH* is defined in units of 2 filters. Exceeding it reflects the addition of two more filters. ***Note:*** Software should set this field to a value that indicates exceptional long buckets. Supporting 32 K 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.

7.2.2.12.2 Flow Director Filters Source IPv6 - FDIRSIPV6[n] (0x0000EE0C + 0x4*n, n=0...2) DBU_RX

Field	Bit(s)	Init.	Type	Description
IP6SA	31:0	0x0	RW	3 LS DWords of the Source IPv6 address. The FDIRIPSA contains the MS DWord of the IP6 address. The LS byte of FDIRIPSA is first on the wire and the MS byte of FDIRSIPv6[2] is last on the wire In MACVLANMODE: • FDIRSIPv6_0[31:0], FDIRSIPv6_1[15:0] hold the destination MAC address value. • FDIRSIPv6_1[31:16], FDIRSIPv6_2[31:0] are reserved and should be set to 0x0.

7.2.2.12.3 Flow Director Filters IP SA - FDIRIPSA (0x0000EE18) DBU_RX

Field	Bit(s)	Init.	Type	Description
IP4SA	31:0	0x0	RW	Source IPv4 address or MS DWord of the Source IPv6 address. While the field is defined in Big Endian (LS byte is first on the wire). In MACVLANMODE, this field is reserved and should be set to 0x0.

7.2.2.12.4 Flow Director Filters IP DA - FDIRIPDA (0x0000EE1C) DBU_RX

Field	Bit(s)	Init.	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). In MACVLANMODE, this field is reserved and should be set to 0x0.

7.2.2.12.5 Flow Director Filters Port - FDIRPORT (0x0000EE20) DBU_RX

Field	Bit(s)	Init.	Type	Description
SOURCE	15:0	0x0	RW	Source Port Number. The field is defined in Little Endian (MS byte is first on the wire). In MACVLANMODE and cloud modes (NVGRE/VXLAN), this field is reserved and should be set to 0x0.
DESTINATION	31:16	0x0	RW	Destination Port Number. The field is defined in Little Endian (MS byte is first on the wire). In MACVLANMODE and cloud modes (NVGRE/VXLAN), this field is reserved and should be set to 0x0.

7.2.2.12.6 Flow Director Filters VLAN and FLEX bytes - FDIRVLAN (0x0000EE24) DBU_RX

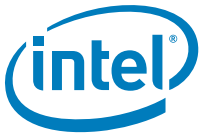
Field	Bit(s)	Init.	Type	Description
VLAN	15:0	0x0	RW	Vlan Tag. This field is defined in Little Endian (MS byte is first on the wire). The CFI bit must be set to Zero and it is not checked by hardware.
FLEX	31:16	0x0	RW	Flexible tuple data as defined by the Flex-Offset field in the FDIRCTRL register while the field is defined in Big Endian (LS byte is first on the wire).

7.2.2.12.7 Flow Director Filters Hash Signature - FDIRHASH (0x0000EE28) DBU_RX

Field	Bit(s)	Init.	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 whenever 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 SW-index for perfect match filters. In perfect match mode, the two MS bits should be zero.
RESERVED	31	0b	RSV	Reserved.

7.2.2.12.8 Flow Director Filters Command register - FDIRCMD (0x0000EE2C) DBU_RX

Field	Bit(s)	Init.	Type	Description
CMD	1:0	0x0	RW	Flow Director Filter Programming Command. 00b = No Action 01b = Add Flow 10b = Remove Flow 11b = Query Command. Following a command completion hardware clears the *CMD* field. In a query command, all other parameters are valid when the *CMD* field is zero.
FILTER_VALID	2	0b	RW	Valid Filter is found by the query command. This bit is set by the device following a Query Command completion.
FILTER_UPDATE	3	0b	RW	Filter Update Command. This bit is relevant only for “Add Flow” command and must be set to zero in any other commands. 0 = The filter parameters do not override existing ones if exist while setting only the collision bit. 1 = 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. 0 = The Destination IPv6 address should not match the IP6AT. 1 = The Destination IPv6 address should match the IP6AT. This field might never match local VM to VM traffic.
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
IPV6	7	0b	RW	IPv6 packet type when set to 1b and IPv4 packet type at 0b. Relevant only if FDIRM.*L3P* is not set.



Field	Bit(s)	Init.	Type	Description
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.
RESERVED	10:9	0x0	RSV	Reserved.
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 filter multiple times. In Signature based filtering it is set when software programs a filter with the same Hash and Signature multiple times. It should be cleared by software when it adds a Flow. It might be set by hardware when 2 flows collide with the same Hash and signature. During reception, this bit is reported on the Rx descriptor of packets that match the filter. See bit 7 for description of the Query-Type.
RESERVED	14:13	0x0	RSV	Reserved.
QUEUE_EN	15	0b	RW	Enable routing matched packet to the queue defined by the Rx-Queue. ***Note:*** Packets redirection to the FDIRCTRL.*DROP_QUEUE* is not gated by the *QUEUE_EN* bit.
RX_QUEUE	22:16	0x0	RW	Rx Queue Index. This field defines the absolute Rx queue index in all modes of operation (regardless of VT enablement).
TUNNEL_FILTER	23	0b	RSV	If set, the filter will be matched by NVGRE or VXLAN packets only. In this case, the L3 and L4 parameters programmed should be of the tunneled (inner) header.
POOL	29:24	0x0	RW	Pool number is meaningful when VT mode is enabled. When both VT is not enabled, this field must be set to 0 by software.
RESERVED	30	0b	RSV	Reserved.
FD_EXCEPTION	31	0b	RW	If set after a command was given, indicates a flow director exception occurred. This bit is a reflection of the EICR.*FLOW_DIRECTOR* bit. Should be set to zero when a new command is given.

7.2.2.12.9 Flow Director Filters Free - FDIRFREE (0x0000EE38) DBU_RX

Field	Bit(s)	Init.	Type	Description
FREE	15:0	0x0000	RW	Number of free (non programmed) filters in the Flow Director Filters logic.
RESERVED	31:16	0x0	RSV	Reserved.

7.2.2.12.10 Flow Director Filters DIPv4 Mask - FDIRDIP4M (0x0000EE3C) DBU_RX

Field	Bit(s)	Init.	Type	Description
IPM	31:0	0x0	RW	Mask Destination IPv4 address. Each bit set to 0b 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.

7.2.2.12.11 Flow Director Filters Source IPv4 Mask - FDIRSIP4M (0x0000EE40) DBU_RX

Field	Bit(s)	Init.	Type	Description
IPM	31:0	0x0	RW	Mask Source IPv4 address. Each bit set to 0b 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.

7.2.2.12.12 Flow Director Filters TCP Mask - FDIRTCPM (0x0000EE44) DBU_RX

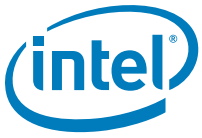
Field	Bit(s)	Init.	Type	Description
SPORTM	15:0	0x0	RW	Mask TCP Source Port. Each bit set to 0b means that the associated bit of the TCP Source Port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the TCP Source Port is ignored (masked out). ***Note:*** This register is swizzle as follows: ●Bit 0 in the mask affects bit 15 of the source port as defined in FDIRPORT.*SOURCE*. ●Bit 1 in the mask affects bit 14 in FDIRPORT.*SOURCE* and so on while bit 15 in the mask affects bit 0 in FDIRPORT.*SOURCE*.
DPORTM	31:16	0x0	RW	Mask TCP Destination Port. Each bit set to 0b means that the associated bit of the TCP destination Port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the TCP destination Port is ignored (masked out). ***Note:*** This register is swizzle the same as the FDIRTCPM.*SPORTM*.

7.2.2.12.13 Flow Director Filters UDP Mask - FDIRUDPM (0x0000EE48) DBU_RX

Field	Bit(s)	Init.	Type	Description
SPORTM	15:0	0x0	RW	Mask UDP Source Port. Each bit set to 0b means that the associated bit of the UDP Source Port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the UDP Source Port is ignored (masked out). ***Note:*** This register is swizzle the same as the FDIRTCPM.*SPORTM*.
DPORTM	31:16	0x0	RW	Mask UDP Destination Port. Each bit set to 0b means that the associated bit of the UDP destination Port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the UDP destination Port is ignored (masked out). ***Note:*** This register is swizzle the same as the FDIRTCPM.*SPORTM*.

7.2.2.12.14 Flow Director Filters Length - FDIRLEN (0x0000EE4C) DBU_RX

Field	Bit(s)	Init.	Type	Description
MAXLEN	5:0	0x0	RC	Longest Linked list of filters in the table. This field records the length of the longest linked list that is updated since the last time this register was read by the software. The longest bucket reported by this field includes *MAXLEN* + 1 filters.
RESERVED	7:6	0x0	RSV	Reserved.
BUCKET_LENGTH	13:8	0x0	RC	The length of the Linked list indicated by a query command. This field is valid following a query command completion.
RESERVED	15:14	0x0	RSV	Reserved.



Field	Bit(s)	Init.	Type	Description
MAXHASH	30:16	0x0	RC	The Lookup HASH value of the added filter that updated the value of the *MAXLEN* field in this register.
RESERVED	31	0b	RSV	Reserved.

7.2.2.12.15 Flow Director Filters Usage Statistics - FDIRUSTAT (0x0000EE50) DBU_RX

Field	Bit(s)	Init.	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 stuck 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 stuck at 0xFFFF and cleared on read.

7.2.2.12.16 Flow Director Filters Failed Usage Statistics - FDIRFSTAT (0x0000EE54) DBU_RX

Field	Bit(s)	Init.	Type	Description
FADD	7:0	0x0	RC	Number of Failed Added Filters due to no space in the filter table. The counter is stuck at 0xFF and cleared on read.
FREMOVE	15:8	0x0	RC	Number of Failed Removed Filters. The counter is stuck at 0xFF and cleared on read.
RESERVED	31:16	0x0	RSV	Reserved.

7.2.2.12.17 Flow Director Filters Match Statistics - FDIRMATCH (0x0000EE58) DBU_RX

Field	Bit(s)	Init.	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:*** This counter might include packets that match the L2 filters or 5 tuple filters or Syn filters even if they are enabled for queue assignment.

7.2.2.12.18 Flow Director Filters Miss Match Statistics - FDIRMISS (0x0000EE5C) DBU_RX

Field	Bit(s)	Init.	Type	Description
PCNT	31:0	0x0	RC	Number of packets that missed matched any Flow Director filter. The counter is stacked at 0xFF...F and cleared on read.

7.2.2.12.19 Flow Director Filters Lookup Table HASH Key - FDIRHKEY (0x0000EE68) DBU_RX

Field	Bit(s)	Init.	Type	Description
KEY	31:0	0x80000001	RW	Programmable Hash lookup table Key.

7.2.2.12.20 Flow Director Filters Signature HASH Key - FDIRSKEY (0x0000EE6C) DBU_RX

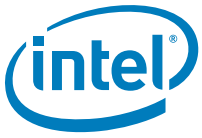
Field	Bit(s)	Init.	Type	Description
KEY	31:0	0x80800101	RW	Programmable Signature Key.

7.2.2.12.21 Flow Director Filters Other Mask - FDIRM (0x0000EE70) DBU_RX

Field	Bit(s)	Init.	Type	Description
VLANID	0	0b	RW	Mask VLAN ID tag. When set to 0b, the 12 bits of the VLAN ID tag are meaningful for the filtering functionality.
VLANP	1	0b	RW	Mask VLAN Priority tag. When set of 0b, the 3 bits of the VLAN Priority are meaningful for the filtering functionality.
POOL	2	0b	RW	Mask Pool. When set to 0b, the target Pool number is meaningful for the filtering functionality.
L4P	3	0b	RW	Mask L4 Protocol. When set to 0b, the UDP/TCP/SCTP protocol type is meaningful for the filtering functionality. ***Note:*** For the Flow Director filtering aspects, SCTP is treated as if it is TCP.
FLEX	4	0b	RW	Mask Flexible Tuple. When set to 0b, the 2 bytes of the Flexible Tuple are meaningful for the filtering functionality.
DIPV6	5	0b	RW	Mask Destination IPv6. When set to 0b, the compare against the IP6AT filter is meaningful for IPv6 packets.
L3P	6	0b	RW	Mask IP type comparison (In order to block all IP comparison, the *DIPV6* field, the FDIRIP6M and FDIRSIP4M and FDIRDIP4M should be set also).
RESERVED	31:7	0x0	RSV	Reserved.

7.2.2.12.22 Flow Director Filters IPv6 Mask - FDIRIP6M (0x0000EE74) DBU_RX

Field	Bit(s)	Init.	Type	Description
SIPM	15:0	0x0	RW	Mask Source IPv6 address. Each bit set to 0b means that the associated byte of the source IPv6 address is meaningful for the filtering functionality. Each bit set to 1b means that the associated byte of the source IPv6 address is ignored (masked out). The LS bit of this register matches the first byte on the wire. When working in MACVLAN mode, this field defines the mask to be used to compare the MAC address When working in VXLAN or NVGRE mode, should be used to allow comparison of TNI/VNI and inner MAC address.
DIPM	31:16	0x0	RW	Mask Destination IPv6 address. Each cleared bit means that the associated byte of the destination IPv6 address is meaningful for the filtering functionality. Each bit set to 1b means that the associated byte of the destination IPv6 address is ignored (masked out). The whole field is meaningful only for the Hash function and the Signature based filters. The *DIPV6* bit in the FDIRM register is meaningful for perfect match filters. The LS bit of this register matches the first byte on the wire.



7.2.2.12.23 Flow Director Filters SCTP Mask - FDIRSCTPM (0x0000EE78) DBU_RX

Field	Bit(s)	Init.	Type	Description
SPORTM	15:0	0x0	RW	Mask SCTP Source Port. Each bit set to 0b means that the associated bit of the SCTP Source Port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the SCTP Source Port is ignored (masked out). ***Note:*** This field is layout as follows: ●Bit 0 in the mask affects bit 15 of the source port as defined in FDIRPORT.*SOURCE*. ●Bit 1 in the mask affects bit 14 in FDIRPORT.*SOURCE*. ●.... ●Bit 15 in the mask affects bit 0 in FDIRPORT.*SOURCE*.
DPORTM	31:16	0x0	RW	Mask SCTP Destination Port. Each bit set to 0b means that the associated bit of the SCTP destination Port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the SCTP destination Port is ignored (masked out). ***Note:*** This field is layout as follow: ●Bit 0 in the mask affects bit 15 of the source port as defined in FDIRPORT.*DESTINATION*. ●Bit 1 in the mask affects bit 14 in FDIRPORT.*DESTINATION*. ●.... ●Bit 15 in the mask affects bit 0 in FDIRPORT.*DESTINATION*.

7.2.2.13 MAC Registers

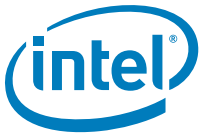
7.2.2.13.1 Highlander Control 0 Register - HLREG0 (0x00004240) MAC

Field	Bit(s)	Init.	Type	Description
RESERVED	1:0	11b	RSV	Reserved.
JUMBOEN	2	0b	RW	Jumbo Frame Enable. Allows frames up to the size specified in Reg MAXFRS (31:16). 0 = Disable jumbo frames (Default). 1 = Enable jumbo frames.
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 user. 0 = Transmit short frames with no padding. 1 = Pad frames (Default).
RESERVED	11	1b	RSV	Reserved. Must be set to 1b.
RESERVED	12	0b	RSV	Reserved.
RESERVED	13	1b	RSV	Reserved. Must be set to 1b.
RESERVED	14	0b	RSV	Reserved.
LPBK	15	0b	RW	Loopback. Turn on loopback where transmit data is sent back through receiver. 0 = Loopback disabled (Default). 1 = Loopback enabled.
MDCSPD	16	1b	RW	MDC Speed. High or low speed MDC clock frequency to PCS, XGXS, WIS, etc. MDCSPD — Freq at 10 GbE — Freq at 1 GbE — Freq at 100 Mb/s 0b — 2.4 MHz — 240 KHz — 240 KHz 1b — 24 MHz — 2.4 MHz — 240 KHz

Field	Bit(s)	Init.	Type	Description
CONTMDC	17	0b	RW	Continuous MDC. Turn off MDC between MDIO packets. 0 = MDC off between packets (default). 1 = Continuous MDC.
RESERVED	19:18	0x0	RSV	Reserved.
PREPEND	23:20	0x0	RW	Prepend Value. Number of 32-bit words, starting after the preamble and SFD, to exclude from the CRC generator and checker (default = 0x0)
RESERVED	26:24	0x0	RSV	Reserved.
RXLNGTHERREN	27	1b	RW	Rx Length Error Reporting. 0 = Disable reporting of all rx_length_err events. 1 = Enable reporting of rx_length_err events if length field < 0x0600.
RXPADSTRIPEN	28	0b	RW	Rx Padding Strip Enable. 0 = Do not strip padding from Rx packets with length field < 64 (Default). 1 = Strip padding from Rx packets with length field < 64 (debug only). ***Note:*** This functionality should be used as debug mode only. If Rx Pad Stripping is enabled, then the Rx CRC Stripping needs to be enabled as well.
RESERVED	31:29	0x0	RSV	Reserved.

7.2.2.13.2 Highlander Status 1 Register - HLREG1 (0x00004244) MAC

Field	Bit(s)	Init.	Type	Description
RESERVED	3:0	0x1	RSV	Reserved.
RESERVED	4	0b	RSV	Reserved.
RXERRSYM	5	0b	RO	Rx Error Symbol. 0 = Error symbol during Rx packet (latch high, clear on read). 1 = Error symbol received.
RXILLSYM	6	0b	RO	Rx Illegal Symbol. Illegal symbol during Rx packet (latch high, clear on read). 0 = No illegal symbol received (default). 1 = Illegal symbol received.
RXIDLERR	7	0b	RO	Rx Idle Error. No idle symbol during idle period (latch high, clear on read). 0 = No idle errors received (default). 1 = Idle error received.
RXLCLFLT	8	0b	RO	Rx Local Fault. Fault reported from PMD, PMA, or PCS (latch high, clear on read). 0 = No local fault (default). 1 = Local Fault is or was active.
RXRMTFLT	9	0b	RO	Rx Remote Fault. Link partner reported remote fault (latch high, clear on read). 0 = No remote fault (default). 1 = Remote fault is or was active.
RESERVED	31:10	0x0	RSV	Reserved.



7.2.2.13.3 Pause and Pace Register - PAP (0x00004248) MAC

Field	Bit(s)	Init.	Type	Description
RESERVED	15:0	0xFFFF	RSV	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) All other values are reserved.
RESERVED	31:20	0x0	RSV	Reserved.

7.2.2.13.4 Max Frame Size - MAXFRS (0x00004268) MAC

Field	Bit(s)	Init.	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 device uses a hard-wired value of 1518 for this field. The MFS does not include the 4 bytes of the VLAN header. Packets with VLAN header might be as large as *MFS* + 4. When Double VLAN is enabled the device adds 8 to the *MFS* for any packet. When E-Tags are enabled, the device adds 12 to the *MFS* for any packet. This value has no effect on transmit frames; it is the responsibility of software to limit the size of transmit frames. ***Note:*** Packets to/from MC are limited to 2KB even when Jumbo are enabled.

7.2.2.13.5 Link Status Register - LINKS (0x000042A4) MAC

Field	Bit(s)	Init.	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	Link Status. 0 = Link is currently down or link was down since last time read. 1 = Link is Up, and there was no link down from last time read. Self-cleared upon read if the link is low and set if the link is up.
RESERVED	26:8	0x0	RSV	Reserved.
NON_STANDARD_SPEED	27	0b	RO	If set, the *LINK_SPEED* field reflects non standard speeds (2.5/5 GbE).

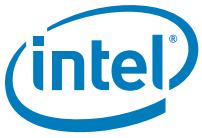
Field	Bit(s)	Init.	Type	Description
LINK_SPEED	29:28	0x0	RO	MAC Link Speed Status. If *NON_STANDARD_SPEED* = 0b: 00b = Reserved. 01b = 100 Mb/s 10b = 1 GbE 11b = 10 GbE If *NON_STANDARD_SPEED* = 1b: 00b - Reserved. 01b - 5 GbE 10b - Reserved. 11b - 2.5 GbE
LINK_UP	30	0b	RO	Link Up. 0 = Link is down. 1 = Link is up.
RESERVED	31	0b	RSV	Reserved.

7.2.2.13.6 MAC Manageability Control Register - MMNGC (0x000042D0) MAC

Field	Bit(s)	Init.	Type	Description
MNG_VETO	0	0b	RO	*MNG_VETO* (default 0) access read/write by the management controller, read only to the host. 0 = No specific constraints on link from management controller. 1 = Hold off any low-power link mode changes. This is done to avoid link loss and interrupting management traffic/activity.
RESERVED	31:1	0x0	RSV	Reserved.

7.2.2.13.7 MAC Control Register - MACC (0x00004330) MAC

Field	Bit(s)	Init.	Type	Description
FLU	0	0b	RW	Force Link Up. 0 = Normal mode. 1 = MAC is forced to the link up state regardless to the PHY link status.
MAC_RX2TX_LPBK_EN	1	0b	RW	Enable MAC Rx to Tx Loopback. 0 = No loopback (normal mode). 1 = Loopback enabled. Transmit path is driven from the receive path, at the MAC internal. XGMII interface.
SWIZZLE_TX_DATA	2	0b	RW	Swizzle the bytes in all 4 MAC internal Tx XGMII lanes. 0 = Swizzle disabled (normal mode). 1 = Swizzle enabled.
SWAP_TX_DATA	3	0b	RW	Swap the MAC internal Tx XGMII lanes. 0 = Swap disabled (normal mode). 1 = Swap enabled.
SWAP_TX_CONTROL	4	0b	RW	Swap the MAC internal Tx XGMII controls. 0 = Swap disabled (normal mode). 1 = Swap enabled.
SWIZZLE_RX_DATA	5	0b	RW	Swizzle the bytes in all 4 MAC internal Rx XGMII lanes. 0 = Swizzle disabled (normal mode). 1 = Swizzle enabled.
SWAP_RX_DATA	6	0b	RW	Swap the MAC internal Rx XGMII lanes. 0 = Swap disabled (normal mode). 1 = Swap enabled.



Field	Bit(s)	Init.	Type	Description
SWAP_RX_CONTROL	7	0b	RW	Swap the MAC internal Rx XGMII controls. 0 = Swap disabled (normal mode). 1 = Swap enabled.
FIFOHT	11:8	0xD	RW	FIFO High Threshold. Determines the high threshold of the MAC elastic FIFO.
FIFOLT	15:12	0x3	RW	FIFO Low Threshold. Determines the low threshold of the MAC elastic FIFO.
RESERVED	31:16	0x0	RSV	Reserved.

7.2.2.13.8 PHY Indirect Access Control - PHY_INDIRECT_CTRL (0x00011144) PCIe_GLUE

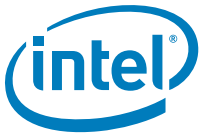
This register is used for indirect access to the PHY and PCS registers. It is used to set the address plus other controls.

Field	Bit(s)	Init.	Type	Description
ADDR	15:0	0x0	RW	16-bit address to be used for write or read transaction.
RESERVED	17:16	0b	RSV	Reserved.
RESP_STAT	19:18	0x0	RO	Response Status. 00b = Successful 01b = Unsuccessful / Not Supported 10b = Reserved 11b = Powered Down This field reflects the response status for the previously completed transaction. The value of this register is only meaningful if *BUSY* = 0b.
CMPL_ERR	27:20	0x0	RO	When the transaction is unsuccessful this field contains the error type. [20] = Wrong source SB-IOSF id [21] = Wrong destination SB-IOSF id [22] = EH not set [23] = Wrong SAI [24] = Wrong TAG [25] = Wrong OPCODE [26] = Data too long
PHY_SELECT	30:28	0x0	RW	Selects the destination PHY or PCS block within the attached HIP block/s. Supports up to 8 separate endpoints. Current supported values are: 000b = KR
BUSY	31	0b	RO	HW sets this bit when it starts to execute a transaction to the HIP. HW clears this bit when the transaction is completed.

7.2.2.13.9 PHY Indirect Access Data - PHY_INDIRECT_DATA (0x00011148) PCIe_GLUE

This register is used for indirect access to the PHY and PCS registers. A write to this register would trigger a write operation that would write the data written to this register based on the information in the PHY Indirect Ctrl register. A read from this register would trigger a read operation that would read data based on the information in the PHY Indirect Ctrl register.

Field	Bit(s)	Init.	Type	Description
DATA	31:0	0x0	RW	The indirect data to write or the data received following an indirect read.



7.2.2.14 Statistic Registers

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's; a small time-delay prior to reading the statistics might be required to avoid a potential mismatch between and interrupt and its cause.
- If RSC is enabled, statistics are collected before RSC is applied to the packets.
- If TSO is enabled, statistics are collected after segmentation.
- All byte (octet) counters composed of two registers can be fetched by two consecutive 32-bit accesses while reading the low 32-bit register first or a single 64-bit access.
- All receive statistic counters count the packets and bytes before coalescing by the RSC logic.
- All receive statistic counters in the filter unit (listed below) might count packets that might be dropped by the packet buffer or receive DMA. Same comment is valid for the byte counters associated with these packet counters: PRC64; PRC127; PRC255; PRC511; PRC1023; PRC1522; BPRC; MPRC; GPRC; RXNFGPC; RUC; ROC.

Statistics Hierarchy:

The following diagrams describe the relations between the packet flow and the different statistic counters.

- Figure 8.1, "Transmit Flow Statistics"
- Figure 8.2, "Receive Flow Statistics"

7.2.2.14.1 Queue Packets Received Count - QPRC[n] (0x00001030 + 0x40*n, n=0...15) DMA_RX

Field	Bit(s)	Init.	Type	Description
PRC	31:0	0x0	RC	Number of packets received for the Queue.

7.2.2.14.2 Queue Bytes Received Count Low - QBRC_L[n] (0x00001034 + 0x40*n, n=0...15) DMA_RX

Field	Bit(s)	Init.	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.

7.2.2.14.3 Queue Bytes Received Count High - QBRC_H[n] (0x00001038 + 0x40*n, n=0...15) DMA_RX

Field	Bit(s)	Init.	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.

7.2.2.14.4 Queue Packets Received Drop Count - QPRDC[n] (0x00001430 + 0x40*n, n=0...15) DMA_RX

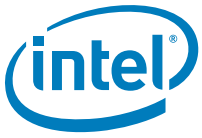
Field	Bit(s)	Init.	Type	Description
PRDC	31:0	0x0	RC	Number of receive packets dropped for the queue. Packets are dropped per queue in one of three cases: 1. Rx Queue is disabled in the RXDCTL[n] register (this includes packets sent to this queue due to the flow control drop no match mechanism or drop action). 2. No free descriptors in the Rx queue while hardware is set to *DROP_EN* in the SRRCTL[n] register or in the PFQDE register. 3. Packet size is larger than RLPML while *RLPML_EN* is set in the RXDCTL[n] register.

7.2.2.14.5 Receive Queue Statistic Mapping Registers - RQSMR[n] (0x00002300 + 0x4*n, n=0...31) DMA_RX

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].

Field	Bit(s)	Init.	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'.
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'.
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'.
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'.
RESERVED	31:28	0x0	RSV	Reserved.



7.2.2.14.6 Rx DMA Statistic Counter Control - RXDSTATCTRL (0x00002F40) DMA_RX

Field	Bit(s)	Init.	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 relates to the same queues that are directed to the QPRC[QSEL] counter as defined by the RQSMR[n] registers. 10000 = The counters relates to all Rx queues. All other values are reserved.
RESERVED	31:5	0x0	RSV	Reserved.

7.2.2.14.7 DMA Good Rx Packet Counter - RXDGPC (0x00002F50) DMA_RX

Field	Bit(s)	Init.	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.

7.2.2.14.8 DMA Good Rx Byte Counter Low - RXDGBCL (0x00002F54) DMA_RX

Field	Bit(s)	Init.	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.

7.2.2.14.9 DMA Good Rx Byte Counter High - RXDGBCH (0x00002F58) DMA_RX

Field	Bit(s)	Init.	Type	Description
GBCH	3:0	0x0	RC	High 4 bits of the 36-bit Byte Counter associated with RXDGBCL.
RESERVED	31:4	0x0	RSV	Reserved.

7.2.2.14.10 DMA Duplicated Good Rx Packet Counter - RXDDPC (0x00002F5C) DMA_RX

Field	Bit(s)	Init.	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.

7.2.2.14.11 DMA Duplicated Good Rx Byte Counter Low - RXDDBCL (0x00002F60) DMA_RX

Field	Bit(s)	Init.	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.

7.2.2.14.12 DMA Duplicated Good Rx Byte Counter High - RXDDBCH (0x00002F64) DMA_RX

Field	Bit(s)	Init.	Type	Description
GBCH	3:0	0x0	RC	High 4 bits of the 36-bit Byte Counter associated with RXDDBCL.
RESERVED	31:4	0x0	RSV	Reserved.

7.2.2.14.13 DMA Good Rx LPBK Packet Counter - RXLPBKPC (0x00002F68) DMA_RX

Field	Bit(s)	Init.	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 since this function is not supported for LPBK traffic.

7.2.2.14.14 DMA Good Rx LPBK Byte Counter Low - RXLPBKBCL (0x00002F6C) DMA_RX

Field	Bit(s)	Init.	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.

7.2.2.14.15 DMA Good Rx LPBK Byte Counter High - RXLPBKBCH (0x00002F70) DMA_RX

Field	Bit(s)	Init.	Type	Description
GBCH	3:0	0x0	RC	High 4 bits of the 36-bit Byte Counter associated with RXLPBKBCL.
RESERVED	31:4	0x0	RSV	Reserved.

7.2.2.14.16 DMA Duplicated Good Rx LPBK Packet Counter - RXDLPBKPC (0x00002F74) DMA_RX

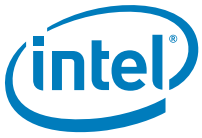
Field	Bit(s)	Init.	Type	Description
GPC	31:0	0x0	RC	Number of replicated or mirrored packets that meet the RXLPBKPC conditions. The sum of RXDLPBKPC and RXLPBKPC is the total good (non-erred) Rx packets from a local VM posted to the host. ***Note:*** The counter might count packets directed to ALL Rx queues or specific Rx queues as defined by the RXDSTATCTRL register.

7.2.2.14.17 DMA Duplicated Good Rx LPBK Byte Counter Low - RXDLPBKBCL (0x00002F78) DMA_RX

Field	Bit(s)	Init.	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.

7.2.2.14.18 DMA Duplicated Good Rx LPBK Byte Counter High - RXDLPBKBCH (0x00002F7C) DMA_RX

Field	Bit(s)	Init.	Type	Description
GBCH	3:0	0x0	RC	High 4 bits of the 36-bit Byte Counter associated with RXDLPBKBCL.
RESERVED	31:4	0x0	RSV	Reserved.



7.2.2.14.19 BMC2OS Packets Received by Host - B2OGPRC (0x00002F90) DMA_RX

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. Counter is cleared when read by driver. Counter is also cleared by PCIe reset and Software reset. When reaching maximum value counter does not wrap-around.

Field	Bit(s)	Init.	Type	Description
B2OGPRC	31:0	0x0	RC	BMC2OS packets received by host.

7.2.2.14.20 CRC Error Count - CRCERRS (0x00004000) STAT

Field	Bit(s)	Init.	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 in length (from <Destination Address> through <CRC>, inclusively). This registers counts all packets received, regardless of L2 filtering and receive enable. This register does not count packets with bad SFD, error control byte, or illegal code byte.

7.2.2.14.21 Illegal Byte Error Count - ILLERRC (0x00004004) STAT

Field	Bit(s)	Init.	Type	Description
IBEC	31:0	0x0	RC	Illegal Byte Error Count. Counts the number of receive packets with illegal bytes errors (i.e. there is an illegal symbol in the packet). This registers counts all packets received, regardless of L2 filtering and receive enablement.

7.2.2.14.22 Error Byte Count - ERRBC (0x00004008) STAT

Field	Bit(s)	Init.	Type	Description
EBC	31:0	0x0	RC	Error Byte Count. Counts the number of receive packets with Error bytes (i.e. there is an Error symbol in the packet). This registers counts all packets received, regardless of L2 filtering and receive enablement.

7.2.2.14.23 MAC short Packet Discard Count - MSPDC (0x00004010) STAT

Field	Bit(s)	Init.	Type	Description
MSPDC	31:0	0x0	RC	Number of MAC Short Packet Discard Packets received.

7.2.2.14.24 Bad SFD Count - MBSDC (0x00004018) STAT

Field	Bit(s)	Init.	Type	Description
MBSDC	31:0	0x0	RW	Counts the number of packets received with bad SFD. Does not count short packets not received

7.2.2.14.25 MAC Local Fault Count - MLFC (0x00004034) STAT

Field	Bit(s)	Init.	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 GbE.

7.2.2.14.26 MAC Remote Fault Count - MRFC (0x00004038) STAT

Field	Bit(s)	Init.	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 GbE.

7.2.2.14.27 LINK Down Counter - LINK_DN_CNT (0x0000403C) STAT

Counts the number of link down events in the MAC.

Field	Bit(s)	Init.	Type	Description
LINK_DN_CNT	31:0	0x0	RC	Counts the number of link down events in the MAC.

7.2.2.14.28 Receive Length Error Count - RLEC (0x00004040) STAT

Field	Bit(s)	Init.	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 does not match the packet length. To enable the receive length error count HLREG.*RXLNGTHERREN* bit needs to be set to 1b. This registers counts all packets received, regardless of L2 filtering and receive enablement.

7.2.2.14.29 Error Packets VLAN Mismatch Counter - ERR_PKT_VLAN_MIS_CNT (0x00004044) STAT

Counts the number of packets dropped due to VLAN filtering mismatch.

Field	Bit(s)	Init.	Type	Description
ERR_PKT_VLAN_MIS_CNT	31:0	0x0	RC	Counts the number of packets dropped due to VLAN filtering mismatch.

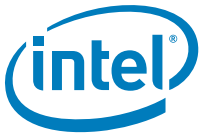
7.2.2.14.30 Error Packet Address Mismatch Counter - ERR_PKT_ADD_MIS_CNT (0x00004048) STAT

Counts the number of packets dropped due to address filtering mismatch.

Field	Bit(s)	Init.	Type	Description
ERR_PKT_ADD_MIS_CNT	31:0	0x0	RC	Counts the number of packets dropped due to address filtering mismatch.

7.2.2.14.31 Packets Received [64 Bytes] Count - PRC64 (0x0000405C) STAT

Field	Bit(s)	Init.	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.



7.2.2.14.32 Packets Received [65-127 Bytes] Count - PRC127 (0x00004060) STAT

Field	Bit(s)	Init.	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.

7.2.2.14.33 Packets Received [128-255 Bytes] Count - PRC255 (0x00004064) STAT

Field	Bit(s)	Init.	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.

7.2.2.14.34 Packets Received [256-511 Bytes] Count - PRC511 (0x00004068) STAT

Field	Bit(s)	Init.	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.

7.2.2.14.35 Packets Received [512-1023 Bytes] Count - PRC1023 (0x0000406C) STAT

Field	Bit(s)	Init.	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.

7.2.2.14.36 Packets Received [1024 to Max Bytes] Count - PRC1522 (0x00004070) STAT

Field	Bit(s)	Init.	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 Section 8.2.3.24.52). Due to changes in the standard for maximum frame size for VLAN tagged frames in 802.3, this device accepts packets which have a maximum length of 1522 bytes. The RMON statistics associated with this range has been extended to count 1522 byte long packets.

7.2.2.14.37 Good Packets Received Count - GPRC (0x00004074) STAT

Field	Bit(s)	Init.	Type	Description
GPRC	31:0	0x0	RO	Number of good (non-erred) Rx packets (from the network) that pass L2 filtering and has legal length as defined by the LongPacketEnable. This registers counts packets regardless of receive enable, this register does not count flow control packets.

7.2.2.14.38 Broadcast Packets Received Count - BPRC (0x00004078) STAT

Field	Bit(s)	Init.	Type	Description
BPRC	31:0	0x0	RO	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.

7.2.2.14.39 Multicast Packets Received Count - MPRC (0x0000407C) STAT

Field	Bit(s)	Init.	Type	Description
MPRC	31:0	0x0	RO	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.

7.2.2.14.40 Good Packets Transmitted Count - GPTC (0x00004080) STAT

Field	Bit(s)	Init.	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 in length (from <Destination Address> through <CRC>, inclusively). The register counts transmitted clear packets, secure packets and FC packets.

7.2.2.14.41 Good Octets Received Count Low - GORCL (0x00004088) STAT

Field	Bit(s)	Init.	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.

7.2.2.14.42 Good Octets Received Count High - GORCH (0x0000408C) STAT

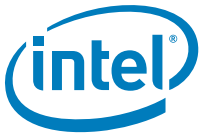
Field	Bit(s)	Init.	Type	Description
CNT_H	3:0	0x0	RC	Higher 4 bits of the "Good Octets Received" counter.
RESERVED	31:4	0x0	RSV	Reserved.

7.2.2.14.43 Good Octets Transmitted Count Low - GOTCL (0x00004090) STAT

Field	Bit(s)	Init.	Type	Description
CNT_L	31:0	0x0	RC	Lower 32 bits of the "Good Octets Transmitted" counter. See complete description in GOTCH register (Section 7.2.2.14.44).

7.2.2.14.44 Good Octets Transmitted Count High - GOTCH (0x00004094) STAT

Field	Bit(s)	Init.	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.



7.2.2.14.45 Receive Undersize Count - RUC (0x000040A4) STAT

Field	Bit(s)	Init.	Type	Description
RUC	31:0	0x0	RC	Receive Undersize Error. This register counts the number of received frames that are shorter than minimum size (64 bytes from <Destination Address> through <CRC>, inclusively), and had a valid CRC. This register counts packets regardless of L2 filtering and receive enablement.

7.2.2.14.46 Receive Fragment Count - RFC (0x000040A8) STAT

Field	Bit(s)	Init.	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.

7.2.2.14.47 Receive Oversize Count - ROC (0x000040AC) STAT

Field	Bit(s)	Init.	Type	Description
ROC	31:0	0x0	RC	Receive Oversize Error. This register counts the number of received frames that are longer than maximum size as defined by MAXFRS.*MFS* (from <Destination Address> through <CRC>, inclusively), and have valid CRC. This register counts packets regardless of L2 filtering and receive enablement.

7.2.2.14.48 Receive Jabber Count - RJC (0x000040B0) STAT

Field	Bit(s)	Init.	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.

7.2.2.14.49 Management Packets Received Count - MNGPRC (0x000040B4) STAT

Field	Bit(s)	Init.	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. SFD errors and short packets are not counted, except that packets dropped because the management receives a jumbo packet or because the receive FIFO is full are counted.

7.2.2.14.50 Management Packets Dropped Count - MNGPDC (0x000040B8) STAT

Field	Bit(s)	Init.	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: 1. The management receive FIFO is full or disabled or 2. A packet bigger than the maximal allowed size is received or 3. A packet length mismatch. The maximal allowed size is 1518 + the size of any recognized L2 header (VLAN, extended VLAN, E-tag, etc). Management packets include any packet directed to the manageability console (such as RMCP and ARP packets).

7.2.2.14.51 Total Octets Received Low - TORL (0x000040C0) STAT

Field	Bit(s)	Init.	Type	Description
CNT_L	31:0	0x0	RC	Lower 32 bits of the "Total Octets Received" counter. See complete description in the next TORH register (Section 7.2.2.14.52).

7.2.2.14.52 Total Octets Received High - TORH (0x000040C4) STAT

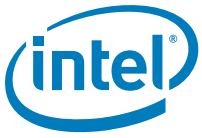
Field	Bit(s)	Init.	Type	Description
CNT_H	3:0	0x0	RC	Higher 4 bits of the "Total Octets Received" counter. The TORL and TORH registers make up a logical 36-bit counter of the total received octets (in the packets counted by the TPR counter). This register includes bytes received in a packet from the <Destination Address> field through the <CRC> field, inclusively.
RESERVED	31:4	0x0	RSV	Reserved.

7.2.2.14.53 Total Packets Received - TPR (0x000040D0) STAT

Field	Bit(s)	Init.	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. This register does not counts packets with SFD errors or packets which discarded by MAC.

7.2.2.14.54 Total Packets Transmitted - TPT (0x000040D4) STAT

Field	Bit(s)	Init.	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.



7.2.2.14.55 Packets Transmitted (64 Bytes) Count - PTC64 (0x000040D8) STAT

Field	Bit(s)	Init.	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.

7.2.2.14.56 Packets Transmitted [65-127 Bytes] Count - PTC127 (0x000040DC) STAT

Field	Bit(s)	Init.	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.

7.2.2.14.57 Packets Transmitted [128-255 Bytes] Count - PTC255 (0x000040E0) STAT

Field	Bit(s)	Init.	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.

7.2.2.14.58 Packets Transmitted [256-511 Bytes] Count - PTC511 (0x000040E4) STAT

Field	Bit(s)	Init.	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.

7.2.2.14.59 Packets Transmitted [512-1023 Bytes] Count - PTC1023 (0x000040E8) STAT

Field	Bit(s)	Init.	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.

7.2.2.14.60 Packets Transmitted [Greater than 1024 Bytes] Count - PTC1522 (0x000040EC) STAT

Field	Bit(s)	Init.	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.

7.2.2.14.61 Multicast Packets Transmitted Count - MPTC (0x000040F0) STAT

Field	Bit(s)	Init.	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.

7.2.2.14.62 Broadcast Packets Transmitted Count - BPTC (0x000040F4) STAT

Field	Bit(s)	Init.	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

7.2.2.14.63 XSUM Error Count - XEC (0x00004120) STAT

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.	Type	Description
XEC	31:0	0x0	RC	Number of Receive IPv4, TCP, UDP or SCTP XSUM Errors.

7.2.2.14.64 Priority XON Received Count - PXONRXCNT[n] (0x00004140 + 0x4*n, n=0...7) STAT

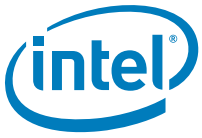
Note: These counters are similar to the PXONRXC[n] in the82598 that were in address 0x0CF00 + 4*n, n=0...7.

Field	Bit(s)	Init.	Type	Description
XONRXC	15:0	0x0	RC	Number of XON packets received per UP. Sticks to 0xFFFF.
RESERVED	31:16	0x0	RSV	Reserved.

7.2.2.14.65 Priority XOFF Received Count - PXOFFRXCNT[n] (0x00004160 + 0x4*n, n=0...7) STAT

Note: These counters are similar to the PXOFFRXC[n] in the82598 that were in address 0x0CF20 + 4*n, n=0...7.

Field	Bit(s)	Init.	Type	Description
XOFFRXC	15:0	0x0	RC	Number of XOFF packets received per UP. Sticks to 0xFFFF.
RESERVED	31:16	0x0	RSV	Reserved.



7.2.2.14.66 Total Unicast Packets Received (BMC copy) - BUPRC (0x00004180) STAT

This register counts the number of good (no errors) unicast packets received from the network. This register does not count unicast packets received that fail to pass address filtering. This register does not count packets counted by the Missed Packet Count (MPC) register. This register does not count flow control packets. Packets sent to the manageability engine are included in this counter. This register is available to the firmware only.

Field	Bit(s)	Init.	Type	Description
BUPRC	31:0	0x0	RC	Number of Unicast Packets Received.

7.2.2.14.67 BMC Total Multicast Packets Received - BMPRC (0x00004184) STAT

This register counts the same events as the MPRC register (Section 8.0.1.24.25) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	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.

7.2.2.14.68 Total Broadcast Packets Received (BMC copy) - BBPRC (0x00004188) STAT

This register counts the same events as the BPRC register (Section8.0.1.24.24) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	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.

7.2.2.14.69 Total Unicast Packets Transmitted (BMC copy) - BUPTC (0x0000418C) STAT

This register counts the number of unicast packets transmitted. This register is available to the firmware only.

Field	Bit(s)	Init.	Type	Description
BUPTC	31:0	0x0	RC	Number of Unicast Packets Transmitted.

7.2.2.14.70 BMC Total Multicast Packets Transmitted - BMPTC (0x00004190) STAT

This register counts the same events as the MPTC register (Section 8.0.1.24.67) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	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.

7.2.2.14.71 Total Broadcast Packets Transmitted (BMC copy) - BBPTC (0x00004194) STAT

This register counts the same events as the BPTC register (Section 8.0.1.24.68) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	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

7.2.2.14.72 BMC FCS Receive Errors - BCRCERRS (0x00004198) STAT

This register counts the same events as the CRCERRS register (Section 8.0.1.24.1) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	Type	Description
BCEC	31:0	0x0	RC	CRC Error Count. Counts the number of receive packets with CRC errors. For a packet to be counted in this register, it must be 64 bytes or greater in length (from <Destination Address> through <CRC>, inclusively). This registers counts all packets received, regardless of L2 filtering and receive enablement.

7.2.2.14.73 BMC Pause XON Frames Received - BXONRXC (0x0000419C) STAT

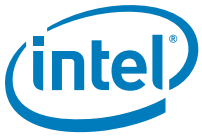
This register counts the same events as the XONRXC register (Section 8.0.1.24.10) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	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

7.2.2.14.74 Link XON Received Count - LXONRXCNT (0x000041A4) STAT

Note: This counter is similar to the LXONRXC in the 82598 that was in address 0x0CF60.

Field	Bit(s)	Init.	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.



7.2.2.14.75 Link XOFF Received Count - LXOFFRXCNT (0x000041A8) STAT

Note: This counter is similar to the LXOFFRXC in the 82598 that was in address 0x0CF68.

Field	Bit(s)	Init.	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.
RESERVED	31:16	0x0	RSV	Reserved.

7.2.2.14.76 Good Rx Non-Filtered Packet Counter - RXNFGPC (0x000041B0) STAT

Field	Bit(s)	Init.	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.

7.2.2.14.77 Good Rx Non-Filter Byte Counter Low - RXNFGBCL (0x000041B4) STAT

Field	Bit(s)	Init.	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.

7.2.2.14.78 Good Rx Non-Filter Byte Counter High - RXNFGBCH (0x000041B8) STAT

Field	Bit(s)	Init.	Type	Description
BCH	3:0	0x0	RC	High 4 bits of the 36-bit Byte Counter associated with RXNFGBCL.
RESERVED	31:4	0x0	RSV	Reserved.

7.2.2.14.79 BMC2OS Packets Sent by BMC - B2OSPC (0x000041C0) STAT

This register counts the total number of transmitted packets sent from the manageability path that were sent to host. This includes packets received by the host and packet dropped in the device due to congestion conditions. Counter is cleared when read by driver. Counter is also cleared by PCIe reset and Software reset. When reaching maximum value counter does not wrap-around.

Field	Bit(s)	Init.	Type	Description
B2OSPC	31:0	0x0	RC	BMC2OS packets sent by BMC.

7.2.2.14.80 OS2BMC packets received by BMC - O2BGPTC (0x000041C4) STAT

This register counts the total number of packets originating from the host that reached the NC-SI interface. Counter is cleared when read by driver. Counter is also cleared by PCIe reset and Software reset. When reaching maximum value counter does not wrap-around.

Field	Bit(s)	Init.	Type	Description
O2BGPTC	31:0	0x0	RC	OS2BMC Good Packets Transmitted Count. This includes packets sent from the host to the BMC or consumed by the internal manageability engine.

7.2.2.14.81 BMC Pause XOFF Frames Received - BXOFFRXC (0x000041E0) STAT

This register counts the same events as the XOFFRXC register (Section 8.0.1.24.12) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	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

7.2.2.14.82 BMC Pause XON Frames Transmitted - BXONTXC (0x000041E4) STAT

This register counts the same events as the XONTXC register (Section 8.0.1.24.9) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	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.
RESERVED	31:16	0x0	RC	Reserved.

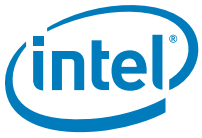
7.2.2.14.83 BMC Pause XOFF Frames Transmitted - BXOFFTXC (0x000041E8) STAT

This register counts the same events as the XOFFTXC register (Section 8.0.1.24.11) for the BMC usage. This register is available to the firmware only.

Field	Bit(s)	Init.	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.
RESERVED	31:16	0x0	RC	Reserved.

7.2.2.14.84 Sideband Receive Dropped Packet Count - B2OSDPC (0x000041F0) STAT

Field	Bit(s)	Init.	Type	Description
B2OSDPC	31:0	0x0	RW	Counts the number of packets received from the BMC interface receive that were dropped. The following causes can create a drop: • Memory buffer full. • Pause packet. • PT packet without SA match or requires enables. • MCTP drop (unsupported message type). • Length error. • Reject by specific interface (SMBus abort, MCTP reject, RMII reject). ***Note:*** This counter is shared between all ports, and counts events of packets dropped regardless of their destination port.



7.2.2.14.85 Fiber Channel CRC Error Count - FCCRC (0x00005118) RX_Filter

Field	Bit(s)	Init.	Type	Description
CRC_CNT	15:0	0x0	RC	FC CRC Count. Counts the number of packets with good Ethernet CRC and bad FC CRC.
RESERVED	31:16	X	RSV	Reserved.

7.2.2.14.86 Queue Packets Transmitted Count - QPTC_ALIAS[n] (0x00006030 + 0x40*n, n=0...15; RC) DMA_TX

Fields definitions are the same as defined in Section 7.2.2.14.88.

7.2.2.14.87 Transmit Queue Statistic Mapping Registers - TQSM[n] (0x00008600 + 0x4*n, n=0...31) DMA_TX

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.	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.

7.2.2.14.88 Queue Packets Transmitted Count - QPTC[n] (0x00008680 + 0x4*n, n=0...15) DMA_TX

These registers are mapped also to 0x06030... to maintain compatibility with the 82598.

Note: Additional address(es): 0x06030 + 0x40*n, n=0...15.

Field	Bit(s)	Init.	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.

7.2.2.14.89 Queue Bytes Transmitted Count Low - QBTC_L[n] (0x00008700 + 0x8*n, n=0...15) DMA_TX

Field	Bit(s)	Init.	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 64-bit read cycle. Each register is Read cleared. In addition, it sticks at 0xFF...F to avoid overflow.

7.2.2.14.90 Queue Bytes Transmitted Count High - QBTC_H[n] (0x00008704 + 0x8*n, n=0...15) DMA_TX

Field	Bit(s)	Init.	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.

7.2.2.14.91 Switch Security Violation Packet Count - SSVPC (0x00008780) DMA_TX

Field	Bit(s)	Init.	Type	Description
SSVPC	31:0	0x0	RC	Switch Security Violation Packet Count. This register counts all Tx packets dropped. For example, due to switch security violations such as SA or VLAN anti-spoof filtering or a packet that has (inner) VLAN that contradicts with PFVMVIR register definitions. Valid only in VMDq or IOV mode. This counter includes also traffic sent to the BMC and blocked due to security violations, or packets dropped due to malicious events detection.

7.2.2.14.92 DMA Good Tx Packet Counter - TXDGPC (0x000087A0) DMA_TX

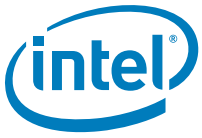
Field	Bit(s)	Init.	Type	Description
GPTC	31:0	0x0	RC	Number of Tx packets from host memory. This counter includes packets that are transmitted to the external network as well as packets that are transmitted only to local VMs. The later occurs only in VT mode when the local switch is enabled. Dropped packets counted in SSVPC register are not counted here.

7.2.2.14.93 DMA Good Tx Byte Counter Low - TXDGBCL (0x000087A4) DMA_TX

Field	Bit(s)	Init.	Type	Description
BCL	31:0	0x0	RC	The low 32 bits of the 36-bit byte counter of the Tx packets that match the TXDGPC. The counter counts all bytes posted by the host and the VLAN (if bytes are added by hardware). Dropped packets counted in SSVPC register are not counted here.

7.2.2.14.94 DMA Good Tx Byte Counter High - TXDGBCH (0x000087A8) DMA_TX

Field	Bit(s)	Init.	Type	Description
BCH	3:0	0x0	RC	High 4 bits of the 36-bit byte counter associated with TXDGBCL.
RESERVED	31:4	0x0	RSV	Reserved.



7.2.2.14.95 OS2BMC packets transmitted by host - O2BSPC (0x000087B0) DMA_TX

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 integrated 10 GbE LAN controller due to congestion conditions or due to anti spoof check. Counter is cleared when read by driver. Counter is also cleared by PCIe reset and Software reset. When reaching maximum value counter does not wrap-around.

Field	Bit(s)	Init.	Type	Description
O2BPC	31:0	0x0	RC	OS2BMC Packets Transmitted Count.

7.2.2.15 Wake-Up and Proxy Control Registers

7.2.2.15.1 Wake Up Control Register - WUC (0x00005800) RX_Filter

Note: The *PME_EN* and *PME_STATUS* bits are reset when global reset is 0. When AUX_PWR=0, these bits are also reset by the assertion of PE_RST_N.

Field	Bit(s)	Init.	Type	Description
RESERVED	0	0b	RSV	Reserved.
PME_EN	1	0b	RW	PME Enable. This bit is used by the software device driver to read the *PME_EN* bit of the Power Management Control/Status Register (PMCSR) without writing to the PCIe configuration space. Writing a 1b to this bit clears it. ***Note:*** Software should not modify this bit while PME Enablement is active.
PME_STATUS	2	0b	RW1C	PME Status. This bit is set when the integrated 10 GbE LAN controller receives a wake-up event. It is the same as the *PME_STATUS* bit in the Power Management Control/Status Register (PMCSR). Writing a 1b to this bit clears it. The *PME_STATUS* bit in the PMCSR is also cleared.
RESERVED	3	0b	RSV	Reserved.
WKEN	4	1b	RW	WKEN. This bit can be cleared to disable the PE_WAKE_N pin assertion (= 0) even if APM is enabled in the shared SPI Flash. In this case, PMCSR and WUS wake-up statuses are invalid. ***Note:*** This bit should not be cleared while in ACPI mode.
RESERVED	31:5	0x0	RSV	Reserved.

7.2.2.15.2 Wake Up Filter Control Register - WUFC (0x00005808) RX_Filter

Note: This register is used to enable each of the pre-defined and flexible filters for wake up support. A value of one means the filter is turned on, and a value of zero means the filter is turned off.

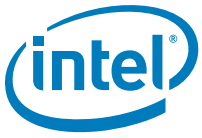
Field	Bit(s)	Init.	Type	Description
LNKC	0	0b	RW	Link Status Change Wake Up Enable.
MAG	1	0b	RW	Magic Packet Wake Up Enable.
EX	2	0b	RW	Directed Exact Wake Up Enable.
MC	3	0b	RW	Directed Multicast Wake Up Enable. Setting this bit does not enable broadcast packets, which are enabled by the BC bit in this register.
BC	4	0b	RW	Broadcast Wake Up Enable.
ARP	5	0b	RW	ARP/IPv4 Request Packet Wake Up Enable.

Field	Bit(s)	Init.	Type	Description
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. 0 = Ignore all TCO/management packets for wake up, except packets that meet the criteria defined in the MNGONLY register via the Host Enable field. For example, criteria intended for the host as well as to the MC). 1 = Ignore all TCO/management packets for wake up. While in normal operation this ignore is forwarded to the host as well as to the MC.
FLX0	16	0b	RW	Flexible Filter 0 Enable. Controls the usage of the FHFT[0] register (0x9000).
FLX1	17	0b	RW	Flexible Filter 1 Enable. Controls the usage of the FHFT[1] register (0x9100).
FLX2	18	0b	RW	Flexible Filter 2 Enable. Controls the usage of the FHFT[2] register (0x9200).
FLX3	19	0b	RW	Flexible Filter 3 Enable. Controls the usage of the FHFT[3] register (0x9300).
FLX4	20	0b	RW	Flexible Filter 4 Enable. Controls the usage of the FHFT[0] register (0x9600).
FLX5	21	0b	RW	Flexible Filter 5 Enable. Controls the usage of the FHFT[5] register (0x9700).
FLX6	22	0b	RW	Flexible Filter 6 Enable. Controls the usage of the FHFT[6] register (0x9800).
FLX7	23	0b	RW	Flexible Filter 7 Enable. Controls the usage of the FHFT[7] register (0x9900).
RESERVED	30:24	0x0	RSV	Reserved.
FW_RST_WK	31	0b	RW	Enable Wake on Firmware Reset Assertion. When set, a firmware reset causes a system wake up that enables the software driver to resend proxying information to firmware.

7.2.2.15.3 Wake Up Status Register - WUS (0x00005810) RX_Filter

Note: 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 global reset is de-asserted or when cleared by the driver.

Field	Bit(s)	Init.	Type	Description
LNKC	0	0b	RW1C	Link Status Changed. This bit can be set even if another event is already set.
MAG	1	0b	RW1C	Magic Packet Received.
EX	2	0b	RW1C	Directed Exact Packet Received. The packet's address matched one of the 16 pre-programmed exact values in the Receive Address registers.
MC	3	0b	RW1C	Directed Multicast Packet Received. The packet was a multicast packet that's hashed to a value that corresponds to a 1 bit in the Multicast Table Array.
BC	4	0b	RW1C	Broadcast Packet Received.
ARP	5	0b	RW1C	ARP/IPv4 Request Packet Received.
IPV4	6	0b	RW1C	Directed IPv4 Packet Received.



Field	Bit(s)	Init.	Type	Description
IPV6	7	0b	RW1C	Directed IPv6 Packet Received.
MNG	8	0b	RW1C	Indicates that a manageability event that should cause a PME to happen.
RESERVED	15:9	0x0	RSV	Reserved.
FLX0	16	0b	RW1C	Flexible Filter 0 Match.
FLX1	17	0b	RW1C	Flexible Filter 1 Match.
FLX2	18	0b	RW1C	Flexible Filter 2 Match.
FLX3	19	0b	RW1C	Flexible Filter 3 Match.
FLX4	20	0b	RW1C	Flexible Filter 4 Match.
FLX5	21	0b	RW1C	Flexible Filter 5 Match.
FLX6	22	0b	RW1C	Flexible Filter 6 Match.
FLX7	23	0b	RW1C	Flexible Filter 7 Match.
RESERVED	30:24	0x0	RSV	Reserved.
FW_RST_WK	31	0b	RW1C	Wake Due to Firmware Reset Assertion Event. When set to 1b, indicates that asserting a firmware reset causes the system wake up so the software driver can re-send proxying information to firmware. This bit can be set even if another event is already set.

7.2.2.15.4 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.	Type	Description
V40	0	0b	RW	IPv4 Address 0 Valid. Loaded from shared SPI Flash.
V41	1	0b	RW	IPv4 Address 1 Valid.
V42	2	0b	RW	IPv4 Address 2 Valid.
V43	3	0b	RW	IPv4 Address 3 Valid.
RESERVED	15:4	0x0	RSV	Reserved.
V60	16	0b	RW	IPv6 Address 0 Valid.
V61	17	0b	RW	IPv6 Address 1 Valid.
V62	18	0b	RW	IPv6 Address 2 Valid.
V63	19	0b	RW	IPv6 Address 3 Valid.
RESERVED	31:20	0x0	RSV	Reserved.

7.2.2.15.5 IPv4 Address Table - IP4AT[n] (0x00005840 + 0x8*n, n=0...3) RX_Filter

4 x IPv4 addresses for ARP/IPv4 Request packet and Directed IPv4 packet wake up. IPv4[0] is loaded from MIPAF words in the shared SPI Flash.

Field	Bit(s)	Init.	Type	Description
IPV4ADDR	31:0	X	RW	IPv4 Address 'n', 'n' = 0...3.

7.2.2.15.6 IPv6 Address Table - IP6AT[n] (0x00005880 + 0x4*n, n=0...3) RX_Filter

First IPv6 addresses for Neighbor Discovery packet filtering and Directed IPv6 packet wake up.

Field	Bit(s)	Init.	Type	Description
IPV6ADDR	31:0	X	RW	4 x Register IPv6 filter. Register 'n' contains bytes '4*n' up to '4*n+3' of the IPv6 address. LS byte of register '0' is first on the wire.

7.2.2.15.7 IPv6 Address Table Extended - IP6AT_EXT[n] (0x00005990 + 0x4*n, n=0...11) RX_Filter

3 x IPv6 addresses for Neighbor Discovery packet filtering.

Field	Bit(s)	Init.	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 '0' is first on the wire.

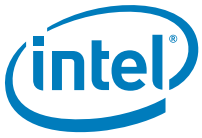
7.2.2.15.8 Wake Up Packet Memory (128 Bytes) - WUPM[n] (0x00005A00 + 0x4*n, n=0...31) RX_Filter

Field	Bit(s)	Init.	Type	Description
WUPD	31:0	X	RO	Wake Up Packet Data. This register is used to store the first 128 bytes of the wake up packet for software retrieval after the system wakes up. It should not be cleared by any reset including master reset.

7.2.2.15.9 Proxying Status Register - PROXYS (0x00005F60) RX_Filter

This register is used to record statistics about all Proxying packets received. If a packet matches multiple criteria then multiple bits could be set. Writing a 1b to any bit clears that bit. This register is not cleared when RST# is asserted. It is only cleared when global reset is deasserted or when cleared by the software device driver. Noted additional packets are received that matches one of the wake-up filters, after the original wake-up packet is received, the PROXYS register is updated with the matching filters accordingly.

Field	Bit(s)	Init.	Type	Description
RESERVED	1:0	0x0	RSV	Reserved.
EX	2	0b	RW1C	Exact Packet Received.
RESERVED	4:3	0x0	RSV	Reserved.
ARP_DIRECTED	5	0b	RW1C	ARP Request Packet with IP4AT Filter Match Received. When set to 1b, indicates a match on any ARP request packet that passed main filtering and Target IP address also matches one of the valid IP4AT filters.
RESERVED	8:6	0x0	RSV	Reserved.
NS	9	0b	RW1C	IPV6 Neighbor Solicitation Received. When set to 1b, indicates a match on NS packet that passed main filtering.
NS_DIRECTED	10	0b	RW1C	IPV6 Neighbor Solicitation with Directed DA Match Received. When set to 1b, indicates a match on NS packet and Target IP address also matches IPV6AT filter.
ARP	11	0b	RW1C	ARP Request Packet Received. When set to 1b, indicates a match on ARP request packet that passed main filtering.



Field	Bit(s)	Init.	Type	Description
MLD	12	0b	RW1C	IPv6 Multicast Listener Discovery (MLD) Packet Received. When set to 1b, indicates a match on any of the following MLD packet types that passed main filtering: • Multicast Listener Query (ICMPv6 Type = decimal 130). Defined in MLDv1 and MLDv2. • Multicast Listener Report (ICMPv6 Type = decimal 131). Defined in MLDv1 and MLDv2. • Version 2 Multicast Listener Report Message (ICMPv6 Type = decimal 143). Defined in MLDv2 only.
RESERVED	31:13	0x0	RSV	Reserved.

7.2.2.15.10 Proxying Filter Control Register - PROXYFC (0x00005F64) RX_Filter

This register is used to enable each of the pre-defined filters for Proxying support. This register is not cleared when RST# is asserted. It is only cleared when global reset is deasserted or when cleared by the software device driver.

Field	Bit(s)	Init.	Type	Description
PPROXYE	0	0b	RW	Port Proxying Enable. When set to 1b, Proxying of packets is enabled when device is in D3 low power state. Proxy information and requirements is passed by Software driver to Firmware via the Admin queue.
RESERVED	1	0b	RSV	Reserved.
EX	2	0b	RW	Exact Proxying Enable.
RESERVED	4:3	0x0	RSV	Reserved.
ARP_DIRECTED	5	0b	RW	ARP Request Packet and IP4AT Match Proxy Enable. If set to 1b, forward to Management for proxying on match of any ARP request packet that passed main filtering and Target IP address also matches one of the valid IP4AT filters.
RESERVED	8:6	0x0	RSV	Reserved.
NS	9	0b	RW	IPV6 Neighbor Solicitation Proxy Enable. If set to 1b, forward to Management for proxying on match of any NS packet (ICMPv6 type 135) that passed main filtering.
NS_DIRECTED	10	0b	RW	IPV6 Neighbor Solicitation and Directed DA Match Proxy Enable. If set to 1b, forward to Management for proxying on match of NS packet and Target IP address also matches valid IPV6AT filter.
ARP	11	0b	RW	ARP Request Packet Proxy Enable. If set to 1b, forward to Management for proxying on match of any ARP request packet that passed main filtering.
MLD	12	0b	RW	IPv6 Multicast Listener Discovery (MLD) Proxy Enable. If set to 1b, forward to Management for Proxying on match of any of the following MLD packet types that passed main filtering: • Multicast Listener Query (ICMPv6 Type = decimal 130). Defined in MLDv1 and MLDv2. • Multicast Listener Report (ICMPv6 Type = decimal 131). Defined in MLDv1 and MLDv2. • Version 2 Multicast Listener Report Message (ICMPv6 Type = decimal 143). Defined in MLDv2 only.
RESERVED	14:13	0x0	RSV	Reserved.
NOTCO	15	0b	RW	Ignore TCO/management packets for proxying. 0 = Ignore only TCO/management packets for Proxying that meet the criteria defined in the MNGONLY register (I.e. are intended only for the BMC and not the Host). 1 = Ignore any TCO/management packets for Proxying, even if in normal operation it is forwarded to the Host in addition to the BMC.

Field	Bit(s)	Init.	Type	Description
RESERVED	31:16	0x0	RSV	Reserved.

7.2.2.15.11 Filter DW Even - FHFT_FILTER_DW_EVEN[n,m] (0x00009000 + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x00009600 + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3) RX_Filter

Field	Bit(s)	Init.	Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	Even DW of the flex filter.

7.2.2.15.12 Filter DW Odd - FHFT_FILTER_DW_ODD[n,m] (0x00009004 + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x00009604 + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3) RX_Filter

Field	Bit(s)	Init.	Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	Odd DW of the flex filter.

7.2.2.15.13 Filter Mask - FHFT_FILTER_MASK[n,m] (0x00009008 + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x00009608 + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3) RX_Filter

Field	Bit(s)	Init.	Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	Bit mask for the 8 bytes of the filter (bit per byte).
RESERVED	31:8	0x0	RSV	Reserved.

7.2.2.15.14 Filter Length - FHFT_FILTER_LENGTH[n,m] (0x0000900C + 0x10*n + 0x100*m, n=0...15, m=0...3 and 0x0000960C + 0x10*(n-16) + 0x100*m, n=16...31, m=0...3) RX_Filter

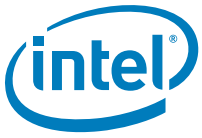
Field	Bit(s)	Init.	Type	Description
LENGTH	7:0	0x0	RW	Flex Filter Length. The length field is only valid on the last DW of the filter. All other length fields are reserved.
RESERVED	31:8	0x0	RSV	Reserved.

7.2.2.16 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 shared SPI Flash by the manageability firmware.

7.2.2.16.1 Management VLAN TAG Value - MAVTV[n] (0x00005010 + 0x4*n, n=0...7) RX_Filter

Field	Bit(s)	Init.	Type	Description
VID	11:0	0x0	RW	Contain the VLAN ID that should be compared with the incoming packet if the corresponding bit in MFVAL.*VLAN* is set.
RESERVED	31:12	0x0	RSV	Reserved.



7.2.2.16.2 Management Flex UDP/TCP Ports - MFUTP[n] (0x00005030 + 0x4*n, n=0...7) RX_Filter

Note: Each 32-bit register (n=0,...,7) refers to two port filters (register 0 refers to ports 0&1, register 2 refers to ports 2&3, etc.). SCTP packets do not match the MFUTP filters. MFUTP filters are programmed in Network order.

Field	Bit(s)	Init.	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.

7.2.2.16.3 BMC IP Address Register - BMCIP[n] (0x00005050 + 0x4*n, n=0...3) RX_Filter

These registers contains the BMC IP Address table.

Field	Bit(s)	Init.	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, BMCIP 0...2 shall be written with zeros. ***Note:*** Field is defined in Big Endian (LS byte is first on the wire).

7.2.2.16.4 BMC IP Valid Register - BMCIPVAL (0x00005060) RX_Filter

This register indicates the type of IP address stored in the IPVAL register and indicates if a valid address is stored.

Field	Bit(s)	Init.	Type	Description
IPADDR_TYPE	0	0b	RW	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.

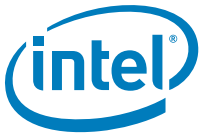
7.2.2.16.5 Manageability Decision Filters Ext - MDEF_EXT[n] (0x00005160 + 0x4*n, n=0...7) RX_Filter

Field	Bit(s)	Init.	Type	Description
L2_ETHERTYPE_AND	3:0	0x0	RW	L2 EtherType. Controls the inclusion of L2 EtherType filtering in the manageability filter decision (AND section).
L2_ETHERTYPE_OR	7:4	0x0	RW	L2 EtherType. Controls the inclusion of L2 EtherType filtering in the manageability filter decision (OR section).
FLEX_PORT	23:8	0x0	RW	Flex Port. Controls the inclusion of Flex port filtering in the manageability filter decision (OR section). Bit 8 corresponds to flex port 0, etc.
FLEX_TCO	24	0b	RW	Flex TCO. Controls the inclusion of Flex TCO filtering in the manageability filter decision (OR section). Bit 24 corresponds to Flex TCO filter 0. ***Note:*** Supported only for Network traffic.

Field	Bit(s)	Init.	Type	Description
ND_135	25	0b	RW	Neighbor Discovery. Controls the inclusion of Neighbor Solicitation Neighbor Discovery filtering in the manageability filter decision (OR section). ***Notes:*** • Supported only for Network traffic. • Neighbor Discovery types supported by this bit is 0x87 (135d) - Neighbor Solicitation
ND_136	26	0b	RW	Neighbor Discovery. Controls the inclusion of Neighbor Advertisement Neighbor Discovery filtering in the manageability filter decision (OR section). ***Notes:*** • Supported only for Network traffic. • Neighbor Discovery types supported by this bit is 0x88 (136d) - Neighbor Advertisement
ND_137	27	0b	RW	Neighbor Discovery. Controls the inclusion of Redirect Neighbor Discovery filtering in the manageability filter decision (OR section). ***Notes:*** • Supported only for Network traffic. • Neighbor Discovery types supported by this bit is 0x89 (137d) - Redirect
RESERVED	28	0b	RSV	Reserved.
MLD	29	0b	RW	MLD. Control the inclusion of MLD packets. These are ICMPv6 packets with the following types: 130, 131, 132, 143.
APPLY_TO_NETWORK_TRAFFIC	30	0b	RW	Apply to Network Traffic. 0 = Do not apply this decision filter to traffic received from the network. 1 = Apply this decision filter to traffic received from the network.
APPLY_TO_HOST_TRAFFIC	31	0b	RW	Apply to Host Traffic. 0 = This decision filter does not apply to traffic received from the host. 1 = This decision filter applies to traffic received from the host.

7.2.2.16.6 Management Ethernet Type Filters - METF[n] (0x00005190 + 0x4*n, n=0...3) RX_Filter

Field	Bit(s)	Init.	Type	Description
ETYPE	15:0	0x0	RW	EtherType value to be compared against the L2 EtherType field in the Rx packet. ***Note:*** Appears in Little Endian order (high byte first on the wire).
RESERVED	29:16	0x0	RSV	Reserved.
POLARITY	30	0b	RW	0b =Positive filter — Filter enters the decision filters if a match occurred. 1b =Negative filter —Filter enters the decision filters if a match did not occur.
RESERVED	31	0b	RSV	Reserved.



7.2.2.16.7 Management Control Register - MANC (0x00005820) RX_Filter

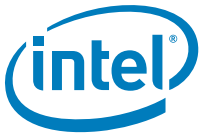
Field	Bit(s)	Init.	Type	Description
FC_DISCARD	0	0b	RW	FC Discard. 0 = Apply filtering rules to packets with Flow Control EtherType. 1 = Discard packets with Flow Control EtherType. ***Note:*** Flow Control EtherType is 0x8808
NCSI_DISCARD	1	0b	RW	NCSI Discard. 0 = Apply filtering rules to packets with NC-SI EtherType. 1 = Discard packets with NC-SI EtherType. ***Note:*** NC-SI EtherType is 0x88F8
RESERVED	16:2	0x0	RSV	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.
RESERVED	22:20	0x0	RW	Reserved
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:*** This capability is not provided for tunneled packets.
EN_IPV4_FILTER	24	0b	RW	Enable IPv4 Address Filters. 0 = These bits store a single IPv6 filter. 1 = The last 128 bits of the MIPAF register are used to store 4 IPv4 addresses for IPv4 filtering.
FIXED_NET_TYPE	25	0b	RW	Fixed Next Type. 0 = Both tagged and untagged packets might be forwarded to manageability engine. 1 = Only packets matching the net type defined by the *NET_TYPE* field will pass to manageability.
NET_TYPE	26	0b	RW	Net Type 0 = Pass only untagged packets. 1 = 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. 0 = The BMC can not communicate with the OS. 1 = The BMC can communicate with the OS. When cleared, the BMC traffic is not forwarded to the OS, even if the Host MAC address filter and VLANs (RAH/L, MTA, VFTA and PFVLVF registers) indicate that it should. When cleared the OS traffic is not forwarded to the BMC even if the decision filters indicates it should. This bit does not impact the BMC to Network traffic. ***Note:*** This bit can change while the host is sending or receiving traffic.
EN_BMC2NET	29	0b	RW	Enable BMC to Network and Network to BMC traffic. 0 = The BMC can not communicate with the network. 1 = The BMC can communicate with the network When cleared the BMC traffic is not forwarded to the network and the network traffic is not forwarded to the BMC even if the decision filters indicates it should. This bit does not impact the host to BMC traffic. ***Note:*** This bit can change while the host is sending or receiving traffic.
PROXYEN	30	0b	RW	Set by Firmware to indicate that Proxy offload is supported,
RESERVED	31	0b	RSV	Reserved.

7.2.2.16.8 Manageability Only Traffic - MNGONLY (0x00005864) RX_Filter

Field	Bit(s)	Init.	Type	Description
EXCLUSIVE_TO_MNG	7:0	0x0	RW	Exclusive to MNG. When set, indicates that packets forwarded by the manageability filters to manageability are not sent to the host. Bits 0...7 correspond to decision rules defined in registers MDEF[0...7] and MDEF_EXT[0...7].
RESERVED	31:8	0x0	RSV	Reserved.

7.2.2.16.9 Manageability Decision Filters - MDEF[n] (0x00005890 + 0x4*n, n=0...7) RX_Filter

Field	Bit(s)	Init.	Type	Description
EXACT_AND	3:0	0x0	RW	Exact. Controls the inclusion of Exact MAC address 0 to 3 in the manageability filter decision (AND section). Bit 0 corresponds to exact MAC address 0 (MMAL0 and MMAH0), etc.
BROADCAST_AND	4	0b	RW	Broadcast. Controls the inclusion of broadcast address filtering in the manageability filter decision (AND section).
VLAN_AND	12:5	0x0	RW	VLAN. Controls the inclusion of VLAN tag 0 to 7, respectively, in the manageability filter decision (AND section). Bit 5 corresponds to VLAN tag 0, etc.
IPV4_ADDRESS	16:13	0x0	RW	IPv4 Address. Controls the inclusion of IPV4 address 0 to 3 (MIPAF[3,n]) respectively, in the manageability filter decision (AND section). Bit 13 corresponds to IPV4 address 0, etc. ***Notes:*** • This field is relevant only if MANC.*EN_IPv4_FILTER* is set. • Supported only for Network traffic.
IPV6_ADDRESS	20:17	0x0	RW	IPv6 Address. Controls the inclusion of IPV6 address 0 to 3, respectively (MIPAF[0:3,n]), in the manageability filter decision (AND section). Bit 17 corresponds to IPV6 address 0, etc, ***Notes:*** • Bit 20 is relevant only if MANC.*EN_IPv4_FILTER* is cleared. • Supported only for Network traffic.
EXACT_OR	24:21	0x0	RW	Exact. Controls the inclusion of exact MAC address 0 to 3 in the manageability filter decision (OR section). Bit 21 corresponds to exact MAC address 0 (MMAL0 and MMAH0), etc.
BROADCAST_OR	25	0b	RW	Broadcast. Controls the inclusion of broadcast address filtering in the manageability filter decision (OR section).
MULTICAST_AND	26	0b	RW	Multicast. Controls the inclusion of Multicast address filtering in the manageability filter decision (AND section). Broadcast packets are not included by this bit.
ARP_REQUEST	27	0b	RW	ARP Request. Controls the inclusion of ARP Request filtering in the manageability filter decision (OR section). Note: Supported only for Network traffic.
ARP_RESPONSE	28	0b	RW	ARP Response - Controls the inclusion of ARP Response filtering in the manageability filter decision (OR section). ***Note:*** Supported only for Network traffic.



Field	Bit(s)	Init.	Type	Description
ND_134	29	0b	RW	Neighbor Discovery. Controls the inclusion of Router Advertisement Neighbor Discovery filtering in the manageability filter decision (OR section). ***Notes:*** • Supported only for Network traffic. • Neighbor Discovery types supported by this bit is 0x86 (134d) - Router Advertisement
PORT_0X298	30	0b	RW	Port 0x298. Controls the inclusion of Port 0x298 filtering in the manageability filter decision (OR section). ***Note:*** Supported only for Network traffic.
PORT_0X26F	31	0b	RW	Port 0x26F. Controls the inclusion of Port 0x26F filtering in the manageability filter decision (OR section). ***Note:*** Supported only for Network traffic.

7.2.2.16.10 Manageability IP Address Filter - MIPAF[n,m] (0x000058B0 + 0x4*n + 0x10*m, n=0...3, m=0...3) RX_Filter

Field	Bit(s)	Init.	Type	Description
IP_ADDR	31:0	X	RW	Manageability IP Address Filters. For each n, m, m=0...3, n=0...3, where MANC.*EN_IPv4_FILTER* = 0, MIPAF[m,n] register holds DW 'n' of IPv6 filter 'm' (4 x IPv6 filters). For each n, m, m=0...3, n=0...3, where MANC.*EN_IPv4_FILTER* = 1, MIPAF[m,n] registers for m=0,1,2 is the same as the previous case (3 x IPv6 filters). And MIPAF[3,n] registers holds IPv4 filter 'n' (4 x IPv4 filters). ***Note:*** These registers appear in Big Endian order (LS byte, LS address is first on the wire).

7.2.2.16.11 Manageability Ethernet MAC Address Low - MMAL[n] (0x00005910 + 0x8*n, n=0...3) RX_Filter

Field	Bit(s)	Init.	Type	Description
MMAL	31:0	X	RW	Manageability Ethernet MAC Address Low. The lower 32 bits of the 48-bit Ethernet MAC Address. ***Note:*** Appear in Big Endian order (LS byte of MMAL is first on the wire).

7.2.2.16.12 Manageability Ethernet MAC Address High - MMAH[n] (0x00005914 + 0x8*n, n=0...3) RX_Filter

Field	Bit(s)	Init.	Type	Description
MMAH	15:0	X	RW	Manageability Ethernet MAC Address High. The upper 16 bits of the 48-bit Ethernet MAC Address. ***Note:*** Appear in Big Endian order (MS byte of MMAH is last on the wire).
RESERVED	31:16	0x0	RSV	Reserved. Reads as 0. Ignored on write.

7.2.2.16.13 FTFT Filter DW Even Words - FTFT_FILTER_EVEN[n] (0x00009400 + 0x10*n, n=0...15) RX_Filter

The Flexible TCO Filter Table registers (FTFT) contains a 128B 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 DW entries FTFT_FILTER_ODD and FTFT_FILTER_EVEN), where each 2 Dews are accompanied by an 8-bit mask (FTFT_FILTER_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. The FTFT_FILTER_LENGTH register indicates the number of bytes to compare.

Notes: The mask field must be 8 bytes aligned even if the length field is not 8 bytes aligned as hardware implementation compares 8 bytes at a time so it should get extra masks until the end of the next quad word. Any mask bit that is located after the length should be set to 0 indicating no comparison should be done.

In case the actual length which is defined by the length field register and the mask bits is not 8 bytes aligned there might be a case that a packet which is shorter then the actual required length passes the flexible filter. This might happen due to comparison of up to 7 bytes that come after the packet but are not a real part of the packet.

The last DW of each filter contains a length field defining the number of bytes from the beginning of the packet compared by this filter. If actual packet length is less than length specified by this field, the filter fails. Otherwise, it depends on the result of actual byte comparison. The value should not be greater than 128.

FTFT registers are configured by firmware.

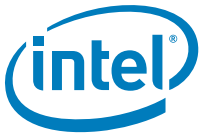
Field	Bit(s)	Init.	Type	Description
FHFT_FILTER0_DW0	31:0	X	RW	ODD DW filter value.

7.2.2.16.14 FTFT Filter DW Odd Words - FTFT_FILTER_ODD[n] (0x00009404 + 0x10*n, n=0...15) RX_Filter

Field	Bit(s)	Init.	Type	Description
FHFT_FILTER0_DW1	31:0	X	RW	Even DW filter value.

7.2.2.16.15 FTFT Filter Mask - FTFT_FILTER0_MASK[n] (0x00009408 + 0x10*n, n=0...15) RX_Filter

Field	Bit(s)	Init.	Type	Description
FHFT_FILTER0_MASK	7:0	X	RW	Mask for the 8 bytes of the filter (bit per byte).
RESERVED	31:8	0x0	RSV	Reserved.



7.2.2.16.16 FTFT Filter Length - FTFT_FILTER0_LENGTH[n] (0x0000940C + 0x10*n, n=0...15) RX_Filter

Field	Bit(s)	Init.	Type	Description
LENGTH	7:0	0x0	RW	Length field is valid only on the last DW of the Filter all other fields are reserved.
RESERVED	31:8	0x0	RSV	Reserved.

7.2.2.17 Manageability (ARC subsystem) HOST Interface

Host interface to the ARC subsystem is described in the Manageability Host Interface Section in the Manageability chapter.

7.2.2.17.1 Host ARC Data RAM - ARCRAM[n] (0x00015800 + 0x4*n, n=0...447) MNG

Field	Bit(s)	Init.	Type	Description
RAM_SPACE	31:0	X	RW	RAM Space area that spans on the CSR space: 0x15800 - 0x15EFC.

7.2.2.17.2 HOST Interface Control Register - HICR (0x00015F00) MNG

Field	Bit(s)	Init.	Type	Description
EN	0	0b	RW	Enable. When set it indicates that a RAM area is provided for device driver accesses. This bit is read only for the device driver.
C	1	0b	RW	Command. The device driver sets this bit when it has finished putting a command block in the ARC internal data ram. This bit should be cleared by the firmware when the command's processing is completed. Setting this bit causes an interrupt to the ARC.
SV	2	0b	RW	Status Valid. Indicates that there is a valid status in CSR area that the device driver can read. 0 = Status valid. 1 = Status not valid. The value of the bit is valid only when the *C* bit is cleared. Only the device driver reads this bit.
SMB	3	0b	RW	Semaphore Bit. This bit is set when this register is read by the HOST driver and cleared when the HOST driver writes zero to it. This bit can be used as semaphore between the two LAN drivers in the device. This bit is reset at PCIe Reset.
SWSMB	4	0b	RW	Software/Firmware Semaphore Software bit. This bit is set only by the HOST software (Read only to the firmware). The bit is not set if bit 8 in the Software Status register (CSR 0x15F10) is set. This bit can be used by the software and firmware to synchronize mutual resources.This bit is cleared by hardware on a core reset or the manageability watch dog expires scenario.
RESERVED	31:5	0x0	RSV	Reserved.

7.2.2.17.3 Firmware Resets Count - FWRESETCNT (0x00015F40) MNG

Field	Bit(s)	Init.	Type	Description
FWRESETCNT	31:0	0x0	RW	Firmware Resets Count. Updated by Hardware. Stuck at 0xFFFF,FFFF

7.2.2.17.4 Software Semaphore Register - SWSM (0x00015F70) MNG

Software Semaphore, between Hosts/Host Processes.

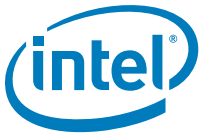
Field	Bit(s)	Init.	Type	Description
SMBI	0	0b	RWS	Set on HOST Read, Can be cleared on HOST Write, Not Set. Also cleared on PCI Immediate Inband Reset.
RESERVED	31:1	0x0	RSV	Reserved.

7.2.2.17.5 Firmware Semaphore Register - FWSM (0x00015F74) MNG

This register is shared by both LAN ports.

Note: This register should be written only by the manageability firmware. The device driver should only read this register. The firmware ignores the shared SPI Flash semaphore in operating system hung states. Bits 15:0 are cleared on firmware reset.

Field	Bit(s)	Init.	Type	Description
RESERVED	0	0b	RSV	Reserved.
FLASH_UPDATE_ENABLED	1	0b	RW	If set, the Flash update functionality is active.
PASS_THROUGH_ENABLED	2	0b	RW	If set, the pass through functionality is active.
PROXY_ENABLED	3	0b	RW	If set, the proxy functionality is active.
HOST_INTERFACE_ENABLED	4	0b	RW	If set, the host interface functionality is active.
FWSM	5	0b	RW	NVM Recovery Mode Defines the operation mode of firmware: 0b = Normal operation mode. 1b = NVM recovery mode.
RESERVED	14:6	0x0	RSV	Reserved.
FW_VAL_BIT	15	0b	RW	Firmware Valid Bit. Hardware clears this bit in reset de-assertion so software can know firmware modes (bits 1-4) are invalid. Firmware should set it to 1b when it is ready (end of boot sequence).
RESERVED	18:16	0x0	RW	Reserved



Field	Bit(s)	Init.	Type	Description
EXT_ERR_IND	24:19	0x0	RW	External Error Indication. Firmware uses this register to store the reason that the firmware has reset/clock gated (such as shared SPI Flash, patch corruption). Possible values: 000000b = No error. 000001b = Unspecified error. 000010b = No manageability (No shared SPI Flash). 000011b = TCO isolate mode active. 000101b = Shadow RAM dump fault. 000110b = Bad Flash contents. (for FW/RO update). 001010b = Shared SPI Flash CRC error in "Test Configuration" module. 001011b = Shared SPI Flash CRC error in "Common FW Parameters" module. 001100b = Shared SPI Flash CRC error in "PT LAN 0 configuration" module. 001101b = shared SPI Flash CRC error in "Sideband configuration" module. 001110b = Shared SPI Flash CRC error in "Flexible TCO filter configuration" module. 001111b = Shared SPI Flash CRC error in "PT LAN 1 configuration" module. 010000b = Shared SPI Flash CRC error in "OEM support structure" module. 010100b = Management memory parity/ECC error. 010101b = SR ECC error. 0x3F = Reserved (max error value). All other values are reserved.
RESERVED	25	0b	RSV	Reserved.
PHY0_CONFIG__ERR_IND	26	0b	RW	PHY0 Configuration Error Indication. Set to 1b by firmware when it fails to configure PHY of LAN0. Cleared by firmware upon successful configuration of PHY of LAN0.
PHY1_CONFIG__ERR_IND	27	0b	RW	PHY1 Configuration Error indication. Set to 1b by firmware when it fails to configure PHY of LAN1. Cleared by firmware upon successful configuration of PHY of LAN1
RESERVED	30:28	0x0	RSV	Reserved.
FACTORY_MAC_ADDRESS_RESTORED	31	0b	RW	When set, indicates to software that the factory MAC address and the current MAC addresses are identical after the last power-up event. This bit is cleared by the device at power up.

7.2.2.17.6 Software-Firmware Synchronization - SW_FW_SYNC (0x00015F78) MNG

Each bit represent different software semaphore agreed between software and firmware as follows. Bits 4:0 and bits 10:12 are owned by software while bits 9:5 and bits 13:15 are owned by firmware. Note that hardware does not lock access to these bits.

Note: This register is shared by both LAN ports. See SW and FW synchronization Section for more details on software and firmware synchronization.

Field	Bit(s)	Init.	Type	Description
SW_NVM_SM	0	0b	RW	If set, shared SPI Flash access is owned by software.
SW_PHY0_SM	1	0b	RW	If set, PHY 0 access is owned by software.

Field	Bit(s)	Init.	Type	Description
SW_PHY1_SM	2	0b	RW	If set, PHY 1access is owned by software.
SW_MAC_CSR_SM	3	0b	RW	If set, MAC CSR access is owned by software.
RESERVED	4	0b	RSV	Reserved.
FW_NVM_SM	5	0b	RW	If set, shared SPI Flash access is owned by firmware.
FW_PHY0_SM	6	0b	RW	If set, PHY 0 access is owned by firmware.
FW_PHY1_SM	7	0b	RW	If set, PHY 1access is owned by firmware.
FW_MAC_CSR_SM	8	0b	RW	If set, MAC CSR access is owned by firmware.
NVM_UPDATE_STARTED	9	0b	RW	If set, shared SPI Flash update started. Software should not write to shared SPI Flash.
SW_MNG_SM	10	0b	RW	If set, Manageability host interface is owned by software.
SW_I2C0_SM	11	0b	RW	If set, i2c of port 0 is owned by SW.
SW_I2C1_SM	12	0b	RW	If set, i2c of port 1 is owned by SW.
FW_I2C0_SM	13	0b	RW	If set, i2c of port 0 is owned by FW.
FW_I2C1_SM	14	0b	RW	If set, i2c of port 1 is owned by FW.
RESERVED	30:15	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 or Firmware driver that reads this register and gets the value of zero for this bit locks the access to this register until it clears this bit.

7.2.2.17.7 Software-Firmware Synchronization - SW_FW_SYNC_MIRR (0x00015F7C; RW) MNG

Each bit represent different software semaphore agreed between software and firmware as follows. Bits 4:0 and bits 10:12 are owned by software while bits 9:5 and bits 13:15 are owned by firmware. Note that hardware does not lock access to these bits

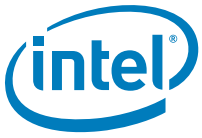
Fields definitions are the same as defined in Section 7.2.2.17.6.

7.2.2.18 Time Sync (IEEE 1588) Registers

7.2.2.18.1 Time Sync SDP Configuration Register - TSSDP (0x0000003C) Target

This register defines the assignment of SDP pins to the time sync auxiliary capabilities.

Field	Bit(s)	Init.	Type	Description
AUX0_SDP_SEL	1:0	0x0	RW	Select one of the SDPs to serve as the trigger for auxiliary timestamp (AUXSTMPL0 and AUXSTMPH0 registers) 00b = SDP0 is assigned. 01b = SDP1 is assigned. 10b = SDP2 is assigned. 11b = SDP3 is assigned.
AUX0_TS_SDP_EN	2	0b	RW	When set, indicates that one of the SDPs can be used as an external trigger to Aux timestamp 0. ***Note:*** If this bit is set to one of the SDP pins, the corresponding pin should be configured to input mode using SPD_DIR.



Field	Bit(s)	Init.	Type	Description
AUX1_SDP_SEL	4:3	0x0	RW	Select one of the SDPs to serve as the trigger for auxiliary time stamp 1 (in AUXSTMPL1 and AUXSTMPH1 registers) 00b = SDP0 is assigned. 01b = SDP1 is assigned. 10b = SDP2 is assigned. 11b = SDP3 is assigned.
AUX1_TS_SDP_EN	5	0b	RW	When set, indicates that one of the SDPs can be used as an external trigger to Aux timestamp 1. ***Note:*** If this bit is set to one of the SDP pins, the corresponding pin should be configured to input mode using SPD_DIR)
TS_SDP0_SEL	7:6	0x0	RW	SDP0 Allocation to Tsync event. When *TS_SDP0_EN* is set, these bits select the Tsync event that is routed to SDP0: 00b = Target Time 0 is output on SDP0. 01b = Target Time 1 is output on SDP0. 10b = Freq Clock 0 is output on SDP0. 11b = Freq Clock 1 is output on SDP0.
TS_SDP0_EN	8	0b	RW	When set, indicates that SDP0 is assigned to Tsync.
TS_SDP1_SEL	10:9	0x0	RW	SDP1 Allocation to Tsync event. When *TS_SDP1_EN* is set, these bits select the Tsync event that is routed to SDP1. 00b = Target Time 0 is output on SDP1. 01b = Target Time 1 is output on SDP1. 10b = Freq Clock 0 is output on SDP1. 11b = Freq Clock 1 is output on SDP1.
TS_SDP1_EN	11	0b	RW	When set indicates that SDP1 is assigned to Tsync.
TS_SDP2_SEL	13:12	0x0	RW	SDP2 Allocation to Tsync event. When *TS_SDP2_EN* is set, these bits select the Tsync event that is routed to SDP2. 00b = Target Time 0 is output on SDP2. 01b = Target Time 1 is output on SDP2. 10b = Freq Clock 0 is output on SDP2. 11b = Freq Clock 1 is output on SDP2.
TS_SDP2_EN	14	0b	RW	When set, indicates that SDP2 is assigned to Tsync.
TS_SDP3_SEL	16:15	0x0	RW	SDP3 Allocation to Tsync event. When *TS_SDP3_EN* is set, these bits select the Tsync event that is routed to SDP3. 00b = Target Time 0 is output on SDP3. 01b = Target Time 1 is output on SDP3. 10b = Freq Clock 0 is output on SDP3. 11b = Freq Clock 1 is output on SDP3.
TS_SDP3_EN	17	0b	RW	When set, indicates that SDP3 is assigned to Tsync.
RESERVED	31:18	0x0	RSV	Reserved. Write 0, ignore on read.

7.2.2.18.2 Rx Message Type Register Low - RXMTRL (0x00005120) RX_Filter

Field	Bit(s)	Init.	Type	Description
CTRLT	7:0	0x0	RW	V1 Control to Timestamp.
MSGT	15:8	0x0	RW	V2 Message ID to Timestamp.
UDPT	31:16	0x13F	RW	UDP Port Number to Timestamp.

7.2.2.18.3 Rx Time Sync Control Register - TSYNCRXCTL (0x00005188) RX_Filter

Field	Bit(s)	Init.	Type	Description
RXTT	0	0b	RO	Rx Timestamp Valid. Set to 1b when a valid value for Rx timestamp is captured in the Rx timestamp register. Cleared by read of Rx timestamp register RXSTMPH.
TYPE	3:1	0x0	RW	Type of Packets to Timestamp. 000b = Time stamp L2 (V2) packets only (Sync or Delay_req depends on RXMTRL.*MSGT* and packets with message ID 2 and 3). 001b = Time stamp L4 (V1) packets only (Sync or Delay_req depends on RXMTRL.*CTRLT*). 010b = Time stamp V2 (L2 and L4) packets (Sync or Delay_req depends on RXMTRL.*MSGT* and packets with message ID 2 and 3). 011b = Reserved. 100b = Time stamp all packets. 101b = Time stamp V2 packets in which message ID bit 3 is zero, which means time stamp all event packets. 110b = Reserved. 111b = Reserved.
EN	4	0b	RW	Enable Rx Timestamp in TXSTMPH/L registers. 0 = Time stamping to TXSTMPH/L disabled. 1 = Time stamping to TXSTMPH/L enabled.
RESERVED	22:5	0x0	RSV	Reserved.
TSIP_UT_EN	23	0b	RW	Defines if untagged packets are appended a timestamp.
TSIP_UP_EN	31:24	0x0	RW	Defines which UP timestamp is appended to the received packet (per UP bitmap). For example, to require a timestamp on packets received with UP=0 or UP=3, bits 24 and 26 should be set.

7.2.2.18.4 Rx Timestamp High - RXSTMPH (0x000051A4) RX_Filter

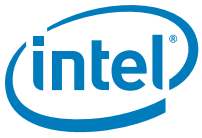
Field	Bit(s)	Init.	Type	Description
RXSTMPH	31:0	0x0	RO	Rx timestamp MSB value. Defined in seconds.

7.2.2.18.5 Rx Timestamp Low - RXSTMPL (0x000051E8) RX_Filter

Field	Bit(s)	Init.	Type	Description
RXSTMPL	29:0	0x0	RO	Rx Timestamp LSB value. Defined in ns units. The value in this field wraps around at 999999999 decimal
RESERVED	31:30	0x0	RO	Reserved.

7.2.2.18.6 Tx Time Sync Control Register - TSYNCTXCTL (0x00008C00) SEC_TX

Field	Bit(s)	Init.	Type	Description
TXTT	0	0b	ROS	Tx Timestamp Valid. Set to 1b when a valid value for Tx timestamp is captured in the Tx timestamp register. Cleared by read of Tx timestamp register TXSTMPH.
RESERVED	3:1	0x0	RSV	Reserved.
EN	4	0b	RW	Enable Tx Timestamp. 0 = Time stamping disabled. 1 = Time stamping enabled.
RESERVED	31:5	0x0	RSV	Reserved.



7.2.2.18.7 Tx Timestamp Value Low - TXSTMPL (0x00008C04) SEC_TX

Field	Bit(s)	Init.	Type	Description
TXSTMPL	29:0	0x0	RO	Tx Timestamp LSB value. Defined in ns units. The value in this field wraps around at 999999999 decimal.
RESERVED	31:30	0x0	RO	Reserved.

7.2.2.18.8 Tx Timestamp Value High - TXSTMPH (0x00008C08) SEC_TX

Field	Bit(s)	Init.	Type	Description
TXSTMPH	31:0	0x0	RO	Tx Timestamp MSB value. Defined in seconds.

7.2.2.18.9 System Time Register Low - SYSTIMEL (0x00008C0C) SEC_TX

Field	Bit(s)	Init.	Type	Description
STL	29:0	0x0	RW	System Time LSB register, defined in ns. Writes of values above a second (above 0x3B9AC9FF) are ignored
RESERVED	31:30	0x0	RW	Reserved.

7.2.2.18.10 System Time Register High - SYSTIMEH (0x00008C10) SEC_TX

Field	Bit(s)	Init.	Type	Description
STH	31:0	0x0	RW	System Time MSB register, defined in seconds.

7.2.2.18.11 Increment Attributes Register - TIMEINCA (0x00008C14) SEC_TX

Field	Bit(s)	Init.	Type	Description
INCVALUE	30:0	0x0	RW	Increment Value to the SYSTIME registers on each 12.5 ns.
ISGN	31	0b	RW	Increment sign. 0 = Each 12.5 ns cycle add to SYSTIM a value of 12.5 ns + *INCVALUE* * 2^-32 ns. 1 = Each 12.5 ns cycle add to SYSTIM a value of 12.5 ns - *INCVALUE* * 2^-32 ns.

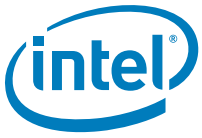
7.2.2.18.12 Time Adjustment Offset Register - TIMADJ (0x00008C18) SEC_TX

Field	Bit(s)	Init.	Type	Description
TADJL	30:0	0x00	RW	Time adjustment value (defined in ns units).
SIGN	31	0b	RW	Sign (0b=?+?, 1b =?-?).

7.2.2.18.13 TimeSync Auxiliary Control Register - TSAUXC (0x00008C20) SEC_TX

Field	Bit(s)	Init.	Type	Description
EN_TT0	0	0b	RW	Enable Target Time 0. Enable bit is set by software to 1b to enable pulse or level change generation as a function of the TSAUXC.*PLSG0* bit.
EN_TT1	1	0b	RW	Enable Target Time 1. Enable bit is set by software to 1b, to enable pulse or level change generation as a function of the TSAUXC.*PLSG* bit.

Field	Bit(s)	Init.	Type	Description
EN_CLK0	2	0b	RW	Enable Configurable Frequency Clock 0. Clock is generated according to frequency defined in the FREQOUT0 register on the SDP pin (0 to 3) that has both of the following: 1. TSSDP.*TS_SDPx_SEL* field with a value of 10b. 2. TSSDP.*TS_SDPx_EN* value of 1b.
SAMP_AUT0	3	0b	SC	When setting the *SAMP_AUT0* flag, the SYSTIMEL/H registers are latched to the AUXSTMPL0/ AUXSTMPH0 registers. Then the *SAMP_AUT0* flag is auto-cleared by the hardware.
ST0	4	0b	RW	Start Clock 0 Toggle on Target Time 0. Enable Clock 0 toggle only after target time 0, that is defined in the TRGTTIML0 and TRGTTIMH0 registers, has passed.
EN_CLK1	5	0b	RW	Enable Configurable Frequency Clock 1. Clock is generated according to frequency defined in the FREQOUT1 register on the SDP pin (0 to 3) that has both of the following: 1. TSSDP.TS_SDPx_SEL field with a value of 11b. 2. TSSDP.TS_SDPx_EN value of 1b.
SAMP_AUT1	6	0b	SC	When setting the *SAMP_AUT1* flag, the SYSTIMEL/H registers are latched to the AUXSTMPL1/ AUXSTMPH1 registers. Then the *SAMP_AUT1* flag is auto-cleared by the hardware.
ST1	7	0b	RW	Start Clock 1 Toggle on Target Time 1. Enable Clock 1 toggle only after Target Time 1, that is defined in the TRGTTIML1 and TRGTTIMH1 registers, has passed.
EN_TS0	8	0b	RW	Enable hardware time stamp 0. Enable Time stamping occurrence of change in SDP pin into the AUXSTMPL0 and AUXSTMPH0 registers. SDP pin (0 to 3) is selected for time stamping, if the SDP pin is selected via the TSSDP.*AUX0_SDP_SEL* field and the TSSDP.*AUX0_TS_SDP_EN* bit is set to 1b.
AUTT0	9	0b	RW	Auxiliary Timestamp Taken. 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 Timestamp Taken. This is a read only bit field. At 0b the Auxiliary Time Stamp 1 is enabled. This bit is set to 1b by hardware following a level change of the input SDP. Reading the AUXSTMPH1 register clears this bit.
RESERVED	16:12	0x0	RSV	Reserved.
PLSG0	17	0b	RW	Use Target Time 0 to generate start of pulse and Target Time 1 to generate end of pulse. SDP pin selected to drive pulse or level change is set according to the TSSDP.*TS_SDPx_SEL* field with a value of 00b and TSSDP.*TS_SDPx_EN* bit with a value of 1b. 0 = Target Time 0 generates change in SDP level. 1 = Target time 0 generates start of pulse on SDP pin. ***Note:*** Pulse or level change is generated when TSAUXC.*EN_TT0* is set to 1b.
RESERVED	29:18	0x0	RSV	Reserved.
RESERVED	30	1b	RSV	Reserved.
DISABLE_SYSTIME	31	1b	RW	Disable SYSTIME count operation. 0 = SYSTIME timer activated. 1 = SYSTIME timer disabled. Value of SYSTIMEH, SYSTIMEL and SYSTIMER remains constant.



7.2.2.18.14 Target Time Register 0 Low - TRGTTIMEL0 (0x00008C24) SEC_TX

Field	Bit(s)	Init.	Type	Description
TTL	29:0	0x0	RW	Target Time 0 LSB value. Defined in ns units.
RESERVED	31:30	0x0	RW	Reserved.

7.2.2.18.15 Target Time Register 0 High - TRGTTIMEH0 (0x00008C28) SEC_TX

Field	Bit(s)	Init.	Type	Description
TTH	31:0	0x0	RW	Target Time 0 MSB value. Defined in seconds.

7.2.2.18.16 Target Time Register 1 Low - TRGTTIMEL1 (0x00008C2C) SEC_TX

Field	Bit(s)	Init.	Type	Description
TTL	29:0	0x0	RW	Target Time 1 LSB value. Defined in ns units.
RESERVED	31:30	0x0	RW	Reserved.

7.2.2.18.17 Target Time Register 1 High - TRGTTIMEH1 (0x00008C30) SEC_TX

Field	Bit(s)	Init.	Type	Description
TTH	31:0	0x0	RW	Target Time 1 MSB value. Defined in seconds.

7.2.2.18.18 Frequency Out 0 Control Register - FREQOUT0 (0x00008C34) SEC_TX

Field	Bit(s)	Init.	Type	Description
CHCT	29:0	0x0	RW	Clock Out Half Cycle Time. Defines the Half Cycle time of Clock 0 in ns units. When clock output is enabled, permitted values are any value larger than 25 and any value below 1 second.
RESERVED	31:30	0x0	RSV	Reserved.

7.2.2.18.19 Frequency Out 1 Control Register - FREQOUT1 (0x00008C38) SEC_TX

Field	Bit(s)	Init.	Type	Description
CHCT	29:0	0x0	RW	Clock Out Half Cycle Time. Defines the Half Cycle time of Clock 1 in ns units. When clock output is enabled, permitted values are any value larger than 13 and up to including 999,999,900 decimal (slightly below 1 second).
RESERVED	31:30	0x0	RSV	Reserved.

7.2.2.18.20 Auxiliary Time Stamp 0 Register Low - AUXSTMPL0 (0x00008C3C) SEC_TX

Field	Bit(s)	Init.	Type	Description
TST_LOW	29:0	0x0	RO	Auxiliary Time Stamp 0 LSB value. Defined in ns units.
RESERVED	31:30	0x0	RO	Reserved.

7.2.2.18.21 Auxiliary Time Stamp 0 Register High - AUXSTMPH0 (0x00008C40) SEC_TX

Field	Bit(s)	Init.	Type	Description
TST_HI	31:0	0x0	RO	Auxiliary Time Stamp 0 MSB value. Defined in seconds.

7.2.2.18.22 Auxiliary Time Stamp 1 Register Low - AUXSTMPL1 (0x00008C44) SEC_TX

Field	Bit(s)	Init.	Type	Description
TST_LOW	29:0	0x0	RO	Auxiliary Time Stamp 1 LSB value Defined in ns units.
RESERVED	31:30	0x0	RO	Reserved.

7.2.2.18.23 Auxiliary Time Stamp 1 Register High - AUXSTMPH1 (0x00008C48) SEC_TX

Field	Bit(s)	Init.	Type	Description
TST_HI	31:0	0x0	RO	Auxiliary Time Stamp 1 MSB value. Defined in seconds.

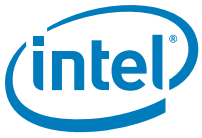
7.2.2.18.24 System Time Register Residue - SYSTIMR (0x00008C58) SEC_TX

Field	Bit(s)	Init.	Type	Description
SYSTIMR	31:0	0x0	RW	System Time Residue value (defined in 2^(-32) ns resolution).

7.2.2.18.25 Time Sync Interrupt Cause Register - TSICR (0x00008C60) SEC_TX

Note: Value of register is always read as 0x0. Once ICR.*Time_Sync* is set, internal value of register \r\n should be cleared by write 1b to all bits or cleared by read to enable reception of an additional \r\nICR.*Time_Sync* interrupt.

Field	Bit(s)	Init.	Type	Description
SYS_WARP	0	0b	RW1C	SYSTIMEL Warp Around. Set when SYSTIME ns counter warps around (SYSTIMEL). Warp around occurrence can be used by Software to update software time. This event should happen every second.
TXTS	1	0b	RW1C	Transmit Time Stamp, Set when new timestamp is loaded into TXSTMP register
RXTS	2	0b	RW1C	Receive Time Stamp. Set when new timestamp is loaded into RXSTMP register
TT0	3	0b	RW1C	Target Time 0 Trigger. Set when Target Time 0 (TRGTTIML/H0) trigger occurs.
TT1	4	0b	RW1C	Target Time 1 Trigger. Set when Target Time 1 (TRGTTIML/H1) trigger occurs.
AUTT0	5	0b	RW1C	Auxiliary Timestamp 0 Taken. Set when new timestamp is loaded into AUXSTMP 0 (auxiliary timestamp 0) register.
AUTT1	6	0b	RW1C	Auxiliary Timestamp 1 Taken. Set when new timestamp is loaded into AUXSTMP 1 (auxiliary timestamp 1) register.
TADJ	7	0b	RW1C	Time Adjust 0 Done: Set when Time Adjust to clock out 0 or 1 completed
RESERVED	31:8	0x0	RSV	Reserved. Write 0, ignore on read.



7.2.2.18.26 Time Sync Interrupt Mask Register - TSIM (0x00008C68) SEC_TX

Field	Bit(s)	Init.	Type	Description
SYS_WARP	0	0b	RW	SYSTIM Warp Around Mask. 0 = No Interrupt generated when TSICR.*SYS_WARP* is set. 1 = Interrupt generated when TSICR.*SYS_WARP* is set.
TXTS	1	0b	RW	Transmit Time Stamp Mask. 0 = No Interrupt generated when TSICR.*TXTS* is set. 1 = Interrupt generated when TSICR.*TXTS* is set.
RXTS	2	0b	RW	Receive Time Stamp Mask. 0 = No Interrupt generated when TSICR.*RXTS* is set. 1 = Interrupt generated when TSICR.*RXTS* is set.
TT0	3	0b	RW	Target Time 0 Trigger Mask. 0 = No Interrupt generated when TSICR.*TT0* is set. 1 = Interrupt generated when TSICR.*TT0* is set.
TT1	4	0b	RW	Target Time 1 Trigger Mask. 0 = No Interrupt generated when TSICR.*TT1* is set. 1 = Interrupt generated when TSICR.*TT1* is set.
AUTT0	5	0b	RW	Auxiliary Timestamp 0 Taken Mask. 0 = No Interrupt generated when TSICR.*AUTT0* is set. 1 = Interrupt generated when TSICR.*AUTT0* is set.
AUTT1	6	0b	RW	Auxiliary Timestamp 1 Taken Mask. 0 = No Interrupt generated when TSICR.*AUTT1* is set. 1 = Interrupt generated when TSICR.*AUTT1* is set.
TADJ	7	0b	RW	Time Adjust 0 Done Mask. 0 = No Interrupt generated when TSICR.*TADJ* is set. 1 = Interrupt generated when TSICR.*TADJ* is set.
RESERVED	31:8	0x0	RSV	Reserved. Write 0, ignore on read.

7.2.2.19 Virtualization PF Registers

7.2.2.19.1 PF VFLR Events Clear - PFVFLREC[n] (0x00000700 + 0x4*n, n=0...1) Target

Field	Bit(s)	Init.	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.

7.2.2.19.2 PF Mailbox Interrupt Causes Register - PFMBICR[n] (0x00000710 + 0x4*n, n=0...3) Target

Each register handles 16 VFs as defined below.

Field	Bit(s)	Init.	Type	Description
VFREQ	15:0	0x0	RW1C	Each bit in the *VFREQ* field is set when VF number (16*n+j) writes a message in its mailbox. While 'n' is the register index, n=0...3 and 'j' is the index of the bits in the VFREQ, j=0...15.
VFACK	31:16	0x0	RW1C	Each bit in the VFACK field is set when VF number (16*n+j) acknowledges 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.

7.2.2.19.3 PF Mailbox Interrupt Mask Register - PFMBIMR[n] (0x00000720 + 0x4*n, n=0...1) Target

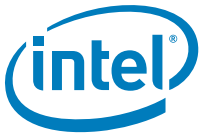
Field	Bit(s)	Init.	Type	Description
VFIM	31:0	0xFF	RW	Bit j - Mailbox indication from VF # (32*n+j) might cause an interrupt to the PF.

7.2.2.19.4 PF Queue Drop Enable Register - PFQDE (0x00002F04) DMA_RX

Field	Bit(s)	Init.	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 (i.e., if either of the bits is set, a packet received when no descriptor is available is dropped).
HIDE_VLAN	1	0b	RW	If this bit is set, the RXDCTL.*VME* setting is ignored, the VLAN is always stripped, and a value of zero is written in the RDESC.VLAN tag and in the RDESC.STATUS.VP fields of the received descriptor.
STRIP_TAG	2	0b	RW	If this bit is set, the E-tag is stripped from the packet. If only one type of tag should be stripped, the other tag Ethertype should be invalidated using ETAG_ETYPE.*VALID* field.
RESERVED	3	0b	RSV	Reserved. Refer to WE and RE bits of this register.
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 2:0 are written into the relevant queue context. Bit 3 is reserved. This bit should never be set together with the *RE* bit in this register.
RE	17	0b	RW	Read Enable. When this bit is set, the content of bits 2:0 are read from the relevant queue context. Bit 3 is reserved. This bit should never be set together with the *WE* bit in this register.
RESERVED	31:18	0x0	RSV	Reserved.

7.2.2.19.5 Last Malicious VM - RX - LMVM_RX (0x00002FA4) DMA_RX

Field	Bit(s)	Init.	Type	Description
MALICIOUS_QUEUE	6:0	0x0	RW	The Rx queue on which the malicious behavior reported in LVMMC was detected. The queue number is the absolute number in the PF space.
RESERVED	16:7	0x0	RSV	Reserved.
MAL_PF	17	0b	RW	Malicious Driver behavior detected on current PF. 0 = Malicious event was on a queue belonging to a VF. 1 = Malicious event was on a queue belonging to the PF.
RESERVED	31:18	0x0	RSV	Reserved.



7.2.2.19.6 Last VM Misbehavior Cause - RX - LVMMC_RX (0x00002FA8) DMA_RX

Bits in LVMMC_RX register define the cause for blocking the malicious queue that was reported in the LMVM_RX.*MALICIOUS_QUEUE* field when RDRXCTL.*MDP_EN* is set. Refer to "Interrupt on Misbehavior of VM (Malicious Driver Detection)" Section for details of the different bits.

Field	Bit(s)	Init.	Type	Description
INV_MACC	0	0b	RC	Invalid Memory Access. A PCIe DMA access initiated by a VF ended with Unsupported Request (UR) or Completer Abort (CA), or a read access was blocked due to out of range address. When a Malicious DMA access is detected the Rx queue (VF) that initiated the access is disabled and corresponding *WQBR_RX* bit is set.
INVALID_RXQ_CONTEXT	1	0b	RC	An invalid receive queue context was detected when queue was enabled or an attempt to change the static part of the queue context on the fly was detected.
RESERVED	31:2	0x0	RSV	Reserved.

7.2.2.19.7 Wrong Queue Behavior Register - Rx - WQBR_RX[n] (0x00002FB0 + 0x4*n, n=0...3) DMA_RX

Field	Bit(s)	Init.	Type	Description
WVBR	31:0	0x0	RW1C	Bitmap indicating against which Rx queue a malicious action was taken. The queue is released only by a reset of the VF or by clearing the corresponding bit in WQBR_RX register.

7.2.2.19.8 PF Mailbox - PFMAILBOX[n] (0x00004B00 + 0x4*n, n=0...63) Target

Field	Bit(s)	Init.	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 reads 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 *VFREQ* and *VFACK* fields of the PFMBICR register.
RESERVED	31:5	0x0	RSV	Reserved.

7.2.2.19.9 Filter Local Packets Low - PFFLPL (0x000050B0) RX_Filter

Field	Bit(s)	Init.	Type	Description
FLP	31:0	0x0	RW	Filter local packets. Filter incoming packets whose MAC source address matches one of the LAN port DA MAC addresses. If the SA of the received packet matches one of the DA in the RAH/RAL registers, then the VM tied to this DA does not receive the packet. Other VMs can still receive it. (Bit per Pool)

7.2.2.19.10 Filter Local Packets High - PFFLPH (0x000050B4) RX_Filter

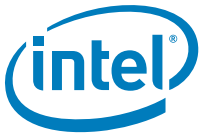
Field	Bit(s)	Init.	Type	Description
FLP	31:0	0x0	RW	Filter Local Packets. Filter incoming packets whose MAC source address matches one of the LAN port DA MAC addresses. If the SA of the received packet matches one of the DA in the RAH/ RAL registers, the VM tied to this DA does not receive the packet. Other VMs can still receive it. (Bit per Pool)

7.2.2.19.11 PF VM Tx Switch Loopback Enable - PFVMTXSW[n] (0x00005180 + 0x4*n, n=0...1) RX_Filter

Field	Bit(s)	Init.	Type	Description
LLE	31:0	0x0	RW	Local Loopback Enable. For each register 'n', and bit 'i', where i=0..31, enables local loopback for pool 32*n+1. 0 = The packet is dropped. 1 = A packet originating from a specific pool and destined to the same pool is allowed to be looped back.

7.2.2.19.12 PF Virtual Control Register - PFVTCTL (0x000051B0) RX_Filter

Field	Bit(s)	Init.	Type	Description
VT_ENA	0	0b	RW	Virtualization Enabled Mode. 0 = Rx traffic is handled internally as if it belongs to VF zero while VF zero is enabled. 1 = When set, the integrated 10 GbE LAN controller supports either 16, 32, or 64 Pools. This bit should be set the same as MTQC.*VT_ENA*.
RESERVED	6:1	0x0	RSV	Reserved.
DEF_PL	12:7	0x0	RW	Default Pool. Pool assignment for packets that do not pass any pool queuing decision. Enabled by the *DIS_DEF_POOL* bit.
RESERVED	15:13	0x0	RSV	Reserved.
POOLING_MODE	17:16	0x0	RW	Pooling Mode. 00b = Pool select by MAC address 01b = Pool select by MAC address or E-tag 10b = Reserved 11b = Reserved
RESERVED	28:18	0x0	RSV	Reserved.
DIS_DEF_POOL	29	0b	RW	Disable Default Pool. Determines the behavior of an Rx packet that does not match any Rx filter and is therefore not allocated a destination pool. 0 = Packet is assigned to the Default Pool (see DEF_PL above). 1 = Packet is dropped.
RPL_EN	30	0b	RW	Replication Enable when set to 1b. When set, MRQC.*MRQE* should be set to one of the
RESERVED	31	0b	RSV	Reserved.



7.2.2.19.13 PF VF Receive Enable - PFVFRE[n] (0x000051E0 + 0x4*n, n=0...1) RX_Filter

This register is reset on common reset cases and on per-function reset cases. Respective bits per VF are reset on VFLR, BME bit clear or on VF software reset. See VF Receive Enable -- PFVFRE / VF Transmit Enable -- PFVFTE for more details.

Field	Bit(s)	Init.	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.

7.2.2.19.14 PF VM VLAN Insert Register - PFVMVIR[n] (0x00008000 + 0x4*n, n=0...63) DMA_TX

Field	Bit(s)	Init.	Type	Description
PORT_VLAN_ID	15:0	0x0	RW	Port VLAN tag to insert if the *VLANA* field = 01b.
RESERVED	26:16	0x0	RSV	Reserved.
TAGA	28:27	0x0	RW	Tag Action. 00b = Do not insert any outer tag. 01b = Insert an E-tag. 10b = Reserved. 11b = Reserved.
RESERVED	29	0b	RSV	Reserved.
VLANA	31:30	0x0	RW	VLAN action. 00b = Use descriptor command. 01b = Always insert Default VLAN. 10b = Never insert VLAN. 11b = Reserved.

7.2.2.19.15 Last VM Misbehavior Cause - TX - LVMMC_TX (0x00008108) DMA_TX

Bits in LVMMC_TX register define the cause for blocking the malicious queue that was reported in the LMVM_TX.*MALICIOUS_QUEUE* field when DMATXCTL.*MDP_EN* is set. Refer to "Interrupt on Misbehavior of VM (Malicious Driver Detection)" Section for details of the different bits.

Note: Only the first malicious event is registered for each packet. Therefore, if a bit is not set, it does not mean that this event didn't occur, only that another malicious behavior was detected first.

Field	Bit(s)	Init.	Type	Description
MAC_HEADER	0	0b	RC	Illegal MAC header size.
IPV4_HEADER	1	0b	RC	Illegal IPv4 header size.
IPV6_HEADER	2	0b	RC	Illegal IPv6 header size.
WRONG_MAC_IP	3	0b	RC	Wrong MAC +IP header size
TCP_LSO	4	0b	RC	Illegal TCP header was detected in a large send operation.
IPSEC_OFFLOAD	5	0b	RC	A VF requested an IPSec offload.
UDP_LSO	6	0b	RC	Illegal UDP header was detected in a large send operation.
STCP_CS	7	0b	RC	Illegal STCP header was detected in an SCTP checksum offload operation.
SIZE	8	0b	RC	Illegal packet size (> 15.5K).
RESERVED	9	0b	RSV	Reserved.
OFF_ILL	10	0b	RC	Illegal offload request.

Field	Bit(s)	Init.	Type	Description
SCTP_ALIGNED	11	0b	RC	SCTP CRC request of non 4 byte aligned data.
ZERO_MSS	12	0b	RC	A request for large send with zero MSS was detected.
CONTEXT_IN_PACKET	13	0b	RC	A context descriptor was detected in the middle of a packet.
LSO_MORE_THAN_4	14	0b	RC	Large send with more than 4 header buffers misbehavior was detected.
OOS_SSO	15	0b	RC	Out of sync during Single Send.
OOS_LSO	16	0b	RC	Out of sync during Large Send.
SSO_UDP	17	0b	RC	Wrong parameter of headers for UDP SSO
SSO_TCP	18	0b	RC	Wrong parameter of headers for TCP SSO
INV_MACC	19	0b	RC	Invalid Memory Access A PCIe DMA access initiated by a VF ended with Unsupported Request (UR) or Completer Abort (CA), or was prevented from being sent if out of range. When a Malicious DMA access is detected the Tx queue (VF) that initiated the access is disabled and corresponding *WQBR_TX* bit is set.
DESC_TYPE	20	0b	RC	Wrong descriptor type (other than 2,3).
WRONG_NULL	21	0b	RC	Null without EOP.
NO_EOP	22	0b	RC	Packet without EOP (i.e. bigger than the ring size).
CONTEXT_BURST	23	0b	RC	Contiguous context descriptor burst that exceeds 3 Contexts.
RESERVED	24	0b	RSV	Reserved.
MAC_VLAN_SPOOF	25	0b	RC	A MAC spoof or VLAN spoof attempt was detected.
VLAN_IERR	26	0b	RC	VLAN Insertion Error. VLE bit set in TX descriptor when VMVIR[n].*VLANA* register field is not 0 causing drop of TX packets.
LEGACY_DESC_IOV	27	0b	RC	A legacy description in IOV was detected.
RESERVED	28	0b	RC	Reserved.
INVALID_TXQ_CONTEXT	29	0b	RC	An invalid transmit queue context was detected when queue was enabled or an attempt to change the static part of the queue context on the fly was detected.
RESERVED	31:30	0x0	RSV	Reserved.

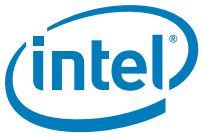
7.2.2.19.16 PF VF Transmit Enable - PFVFTE[n] (0x00008110 + 0x4*n, n=0...1) DMA_TX

This register is reset on common reset cases and on per-function reset cases. Respective bits per VF are reset on VFLR, BME bit clear or on VF software reset. See VF Receive Enable -- PFVFRE / VF Transmit Enable -- PFVFTE for more details.

Field	Bit(s)	Init.	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.

7.2.2.19.17 Last Malicious VM - TX - LMVM_TX (0x00008124) DMA_TX

Field	Bit(s)	Init.	Type	Description
MALICIOUS_QUEUE	6:0	0x0	RW	The Tx queue on which the malicious behavior reported in LVMMC was detected. The queue number is the absolute number in the PF space.
RESERVED	16:7	0x0	RSV	Reserved.



Field	Bit(s)	Init.	Type	Description
MAL_PF	17	0b	RW	Malicious Driver behavior detected on current PF. 0 = Malicious event was on a queue belonging to a VF. 1 = Malicious event was on a queue belonging to the PF.
RESERVED	31:18	0x0	RSV	Reserved.

7.2.2.19.18 Wrong Queue Behavior Register - Tx - WQBR_TX[n] (0x00008130 + 0x4*n, n=0...3) DMA_TX

Field	Bit(s)	Init.	Type	Description
WVBR	31:0	0x0	RW1C	Bitmap indicating against which Tx queue an anti spoof action or malicious action was taken. The queue is released only by a reset of the VF or by clearing the corresponding bit in the WQBR_TX register

7.2.2.19.19 PFVF Anti Spoof Control - PFVFSPOOF[n] (0x00008200 + 0x4*n, n=0...7) DMA_TX

Field	Bit(s)	Init.	Type	Description
MACAS	7:0	0x00	RW	For each register 'n', and bit 'i', where i=0..7, enables anti-spoofing filter on Ethernet MAC Addresses for VF(8*n+i).
VLANAS	15:8	0x00	RW	For each register 'n', and bit '8+i', where i=0..7, enables anti-spoofing filter on VLAN tag for VF(8*n+i). ***Note:*** If *VLANAS* is set for a specific pool, then the respective *MACAS* bit must be set as well.
ETHERTYPEAS	23:16	0x0	RW	For each register 'n', and bit '16+i', where i=0..7, enables anti-spoofing filter on Ethertype for VF(8*n+i).
ETHERTYPELB	31:24	0x0	RW	For each register 'n', and bit '24+i', where i=0..7, enables Loop-Back filter on Ethertype for VF(8*n+i).

7.2.2.19.20 PF DMA Tx General Switch Control - PFDTXGSWC (0x00008220) DMA_TX

Field	Bit(s)	Init.	Type	Description
LBE	0	0b	RW	Enables VMDQ Loopback.
RESERVED	31:1	0x0	RSV	Reserved.

7.2.2.19.21 PF VM 0:31 Error Count Mask - PFVMECM0 (0x00008790) DMA_TX

Field	Bit(s)	Init.	Type	Description
FILTER	31:0	0x0	RW	Defines if a packet dropped from pools 0 to 31, respectively, is counted in the SSVPC counter.

7.2.2.19.22 PF VM 32:63 Error Count Mask - PFVMECM1 (0x00008794) DMA_TX

Field	Bit(s)	Init.	Type	Description
FILTER	31:0	0x0	RW	Defines if a packet dropped from pools 32 to 63, respectively, is counted in the SSVPC counter.

7.2.2.19.23 PF VM L2Control Register - PFVML2FLT[n] (0x0000F000 + 0x4*n, n=0...63) RX_Filter

This register controls per VM Inexact L2 Filtering.

Field	Bit(s)	Init.	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.

7.2.2.19.24 PF VM VLAN Pool Filter - PFVLVF[n] (0x0000F100 + 0x4*n, n=0...63) RX_Filter

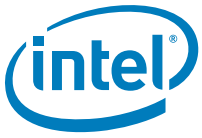
Software should initialize these registers before transmit and receive are enabled.

Field	Bit(s)	Init.	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.

7.2.2.19.25 PF VM VLAN Pool Filter Bitmap - PFVLVFB[n] (0x0000F200 + 0x4*n, n=0...127) RX_Filter

Software should initialize these registers before transmit and receive are enabled.

Field	Bit(s)	Init.	Type	Description
POOL_ENA	31:0	X	RW	Pool Enable bit array. Each couple of registers '2*n' and '2*n+1' enables routing of packets that match a PFVLVF[n] filter to a Pool list. Each bit when set, enables packet reception with the associated Pools as described: Bit 'i' in register '2*n' is associated with POOL 'iTab'. Bit 'i' in register '2*n+1' is associated with POOL '32+i'.



7.2.2.19.26 PF Unicast Table Array - PFUTA[n] (0x0000F400 + 0x4*n, n=0...127) RX_Filter

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 start of work.

Field	Bit(s)	Init.	Type	Description
BIT_VECTOR	31:0	X	RW	Word wide bit vector specifying 32 bits in the unicast destination address filter table.

7.2.2.19.27 PF Mirror Rule Control - PFMRCTL[n] (0x0000F600 + 0x4*n, n=0...3) RX_Filter

This register defines mirroring rules for each of 4 destination pools.

Field	Bit(s)	Init.	Type	Description
VPME	0	0b	RW	Virtual Pool Mirroring Enable. Enables mirroring of certain pools as defined in the PFMRVM registers.
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.

7.2.2.19.28 PF Mirror Rule VLAN - PFMRVLAN[n] (0x0000F610 + 0x4*n, n=0...7) RX_Filter

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 (e.g. register 0 corresponds to VLAN filters 31:0, while register 4 corresponds to VLAN filters 63:32).

Field	Bit(s)	Init.	Type	Description
VLAN	31:0	0x0	RW	Bitmap listing which VLANs participate in the mirror rule.

7.2.2.19.29 PF Mirror Rule Pool - PFMRVM[n] (0x0000F630 + 0x4*n, n=0...7) RX_Filter

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 (e.g. register 0 corresponds to pools 31:0, while register 4 corresponds to pools 63:32).

Field	Bit(s)	Init.	Type	Description
POOL	31:0	0x0	RW	Bitmap listing which pools participate in the mirror rule.

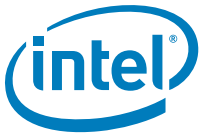
7.2.2.19.30 PF VM Tag Insert Register - PFVMTIR[n] (0x00017000 + 0x4*n, n=0...63) DMA_TX

Field	Bit(s)	Init.	Type	Description
PORT_TAG_ID	31:0	0x0	RW	Port tag to insert if PFVVMIR.*TAGA* field = 01b or 10b. TAGA = 01b = Insert E-tag (i.e. EtagEtype-PFVMTIR[31:0]-0x0000). TAGA = 10b = Reserved. TAGA = Other = Ignore this field.

7.2.2.20 Power Management

7.2.2.20.1 DMA Coalescing Control Register - DMACR (0x00002400) DMA_RX

Field	Bit(s)	Init.	Type	Description
DMACWT	15:0	0x20	RW	DMA Coalescing Watchdog Timer. When in DMA coalescing, defines the upper limit in 40.960 µs units between arrival of coalescing exit event condition and actual exit from DMA coalescing. The programmed value should be non zero.
HIGH_PRI_TC	23:16	0x0	RW	High Priority Traffic Class. This field defines which TC is considered high priority and a packet received for this TC will cause exit from DMA Coalescing. When a TC bit is set it indicates this TC is high priority. Bit 16 = TC0 Bit 17 = TC1 ... Bit 23 = TC7
RESERVED	27:24	0x0	RSV	Reserved.
EN_MNG_IND	28	1b	RW	Enable Management Indications for DMAC operation. When set, DMA Coalescing functionality will be affected from the MCTP management buffer status indications.
RESERVED	29	0b	RSV	Reserved.
LX_COALESCING_INDICATION	30	0b	RW	Lx Coalescing Indication. Defines whether to move in/out of DMA coalescing when the P-IOSF ISM moves in/out of IDLE state. 0 = DMA coalescing can also start when P-IOSF is not IDLE. DMA coalescing stops when any TLP transactions are executed on the P-IOSF. 1 = DMA coalescing conditions are met only when P-IOSF ISM is in IDLE state.
DMAC_EN	31	0b	RW	DMA Coalescing Enable. 0 = Disable DMA Coalescing. 1 = Enable DMA Coalescing.



7.2.2.20.2 DMA Coalescing Time to Lx Request - DMCTLX (0x00002404) DMA_RX

Field	Bit(s)	Init.	Type	Description
TTLX	11:0	0x20	RW	Time to LX Request. Controls the time between detection of DMA Coalescing condition to actual entry into DMA Coalescing state. Timer counts is in 1.28 µs intervals. ***Notes:*** • Timer adds delay to decision on when to enter DMA Coalescing state. • For link speed of 10 GbE, minimum value should be 0x1. For link speed of 1 GbE, minimum value should be 0x2. For link speed of 100 Mb/s, minimum value should be 0x20. This limitation is enforced by the hardware.
RESERVED	31:12	0x0	RSV	Reserved. Write 0. Ignore on read.

7.2.2.20.3 DMA Coalescing Threshold - DMCTH[n] (0x00003300 + 0x4*n, n=0...7) DBU_RX

Field	Bit(s)	Init.	Type	Description
DMACRXT	8:0	0x20	RW	DMA Coalescing Receive Threshold Rx PB TC[n]. This value defines the DMA coalescing Receive threshold in 1 Kilobyte units. When amount of data in internal receive buffer exceeds *DMACRXT*[n] value, DMA coalescing is stopped. Value written to the field should take into account: • The latency tolerance requirements (LTR) sent over the PCIe and XOFF receive threshold values, to avoid needless generation of flow control packets when in DMA coalescing operating mode and flow control is enabled. • The maximum size of the receive buffer.
RESERVED	31:9	0x0	RSV	Reserved.

7.2.2.20.4 EEE TX LPI Count - TLPIC (0x000041F4) STAT

This register counts EEE TX LPI entry events. A EEE TX LPI event occurs when the transmitter enters EEE (IEEE802.3az) LPI state. This register only increments if transmits are enabled and EEE operation is enabled.

Field	Bit(s)	Init.	Type	Description
ETLPIC	31:0	0x0	RC	Number of EEE TX LPI events. Register cleared on read. Register does not wrap around at 0xFFFFFFFF.

7.2.2.20.5 EEE RX LPI Count - RLPIC (0x000041F8) STAT

This register counts EEE RX LPI entry events. A EEE RX LPI event occurs when the receiver detects link partner entry into EEE (IEEE802.3az) LPI state. This register only increments if receives are enabled and EEE operation is enabled.

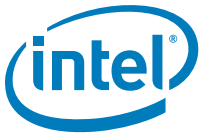
Field	Bit(s)	Init.	Type	Description
ERLPIC	31:0	0x0	RC	Number of EEE RX LPI events. Register cleared on read. Register does not wrap around at 0xFFFFFFFF.

7.2.2.20.6 Energy Efficient Ethernet (EEE) Setup Register - EEE_SU (0x00004380) MAC

Field	Bit(s)	Init.	Type	Description
DTW_MIN	7:0	0x0	RW	Time to add to minimum Tw_sys_tx value defined in IEEE802.3az clause 78.5. This additional time should be configured according to the link speed/type and is in addition to the specification definition: • For 1000BASE-T operation (16.5 usec) • For 100BASE-TX operation (30 usec) • For 10GBASE-T operation (7.36 usec for Case-1 and 4.48 usec for Case-2) The minimum Tw_sys_tx value defines the duration where no data will be transmitted following move out of EEE LPI TX state. Time to add defined in this field is expressed in 0.1 microsecond resolution. ***Note:*** The idle time value defined by this field plus the Tw_sys_tx value defined in IEEE802.3az clause 78.5 is used when moving out of EEE TX LPI state to transmit flow control frames even if value specified in EEER.Tw_system field is higher.
RESERVED	15:8	0x0	RSV	Reserved.
TW_WAKE_MIN	21:16	0x0	RW	Minimum time, expressed in 1 µs, between sending a request to move into EEE TX LPI and sending a request to move back to active state. ***Note:*** If conditions to exit LPI during the *TW_WAKE_MIN* interval cease to exist than device will not move out of TX LPI after timer has expired.
RESERVED	23:22	0x0	RW	Reserved
TX_LU_LPI_DLY	25:24	0x3	RW	Delay to enable entry of TX EEE LPI state following Link-up indication. 00b = No delay 01b = 10 ms 10b = 100 ms 11b = 1 Second ***Note:*** IEEE802.3az clause 78.1.2.1 defines delay of 1 second following link-up.
TEEE_DLY	31:26	0x2	RW	TX EEE LPI Entry Delay Field defines delay to EEE entry once conditions to enter EEE LPI are detected. Field resolution is 1 µs. ***Note:*** If conditions to enter LPI during the TEEE_DLY interval cease to exist device will not enter TX LPI and continue normal operation.

7.2.2.20.7 Energy Efficient Ethernet (EEE) STATUS - EEE_STAT (0x00004398) MAC

Field	Bit(s)	Init.	Type	Description
RESERVED	28:0	0x0	RSV	Reserved. Write 0. Ignore on read.
EEE_NEG	29	0b	RO	EEE support Negotiated on link. 0 = EEE operation not supported on link. 1 = EEE operation supported on link.
RX_LPI_STATUS	30	0b	RO	RX Link in LPI state. 0 = RX in Active state. 1 = RX in LPI state.
TX_LPI_STATUS	31	0b	RO	TX Link in LPI state. 0 = TX in Active state. 1 = TX in LPI state.

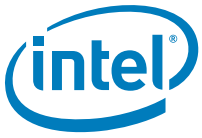


7.2.2.20.8 Energy Efficient Ethernet (EEE) Register - EEER (0x000043A0) MAC

Field	Bit(s)	Init.	Type	Description
TW_SYSTEM	15:0	0x0	RW	Time expressed in microseconds that no data will be transmitted following move from EEE TX LPI link state to Link Active state. Field holds the Transmit Tw_sys_tx value negotiated during EEE LLDP negotiation. ***Notes:*** • If value is lower than minimum Tw_sys_tx value defined in IEEE802.3az clause 78.5 then interval where no data is transmitted following move out of EEE TX LPI state defaults to minimum Tw_sys_tx. • Following link disconnect or Auto-negotiation value of this field returns to default value, until SW re-negotiates new Tw_sys_tx value via EEE LLDP. • Fast Retrain and Local/Remote Fault indication are not considered link disconnect and do not cause the field to return to the default value. • When transmitting flow control frames device waits the minimum time defined in the IEEE802.3az standard before transmitting the flow control packet. device does not wait the *TW_SYSTEM* time following exit of LPI before transmitting the flow control frame.
TX_LPI_EN	16	0b	RW	Enable entry into EEE LPI on TX path. 0 = Disable entry into EEE LPI on TX path. 1 = Enable entry into EEE LPI on TX path. ***Notes:*** • Even when TX_LPI_EN is 1b device will not enable entry into TX LPI state for at least 1 second following the change of link_status to OK as defined in IEEE802.3az clause 78.1.2.1. • Even if the TX_LPI_EN bit is set, device will initiate entry into TX EEE LPI link state only if EEE support at the link speed was negotiated during Auto-negotiation.
RX_LPI_EN	17	1b	RW	Enable entry into EEE LPI on RX path. 0 = Disable entry into EEE LPI on RX path. 1 = Enable entry into EEE LPI on RX path. ***Note:*** Even if the RX_LPI_EN bit is set, device will recognize entry into RX EEE LPI link state only if EEE support at the link speed was negotiated during Auto-negotiation.
RESERVED	27:18	0x0	RSV	Reserved. Write 0. Ignore on read.
EEE_FRC_AN	28	0b	RW	Force EEE Auto-negotiation. When bit is set to 1 enables EEE operation in internal MAC logic even if link partner does not support EEE. ***Note:*** Should be set to 1b to enable testing of EEE operation via MAC loop-back
RESERVED	31:29	0x0	RSV	Reserved. Write 0. Ignore on read.

7.2.2.20.9 Latency Tolerance Reporting (LTR) Control - LTRC (0x00011708) PCIe

Field	Bit(s)	Init.	Type	Description
SLTRV	9:0	0x5	RW	Snoop Latency Value. Along with the Max Snoop Latency Scale field, this register specifies the maximum snoop latency this function wants to request. Software should set this only if it is set to a lower number than the platforms maximum supported latency.
SSCALE	12:10	0x2	RW	Snoop Latency Scale. This field provides a scale for the value contained within the Snoop Latency Value field. Encoding: 000b = Value times 1 ns 001b = Value times 32 ns 010b = Value times 1,024 ns 011b = Value times 32,768 ns 100b = Value times 1,048,576 ns 101b = Value times 33,554,42 ns 110b = Not permitted. 111b = Not permitted The SW driver is advised to use the same scale as configured in the LTR capability.
RESERVED	14:13	0x0	RSV	Reserved. Write 0. Ignore on read.
LTRS_REQUIREMENT	15	0b	RW	LTR Snoop Requirement. 0 = No Latency requirements in Snoop memory access. 1 = Latency tolerance in Snoop memory access specified in the LTRC.*SLTRV* field.
NSLTRV	25:16	0x5	RW	No Snoop Latency Value. Along with the Max No Snoop Latency Scale field, this register specifies the maximum no snoop latency this function wants to request. Software should set this only if it is set to a lower number than the platforms maximum supported latency.
NSSCALE	28:26	0x2	RW	No Snoop Latency Scale. This field provides a scale for the value contained within the No Snoop Latency Value field. Encoding: 000b = Value times 1 ns 001b = Value times 32 ns 010b = Value times 1,024 ns 011b = Value times 32,768 ns 100b = Value times 1,048,576 ns 101b = Value times 33,554,432 ns 110b = Not permitted 111b = Not permitted The SW driver is advised to use the same scale as configured in the LTR capability.
RESERVED	29	0b	RSV	Reserved. Write 0. Ignore on read.
LTR_SEND	30	0b	RW	Indication to send an LTR message. This bit is set by the FW (indicates HW to send LTR message update to the system) and cleared by HW after the message was sent. This bit is not sent as part of the LTR message.
LTRNS_REQUIREMENT	31	0b	RW	LTR No Snoop Requirement. 0 = No Latency requirements in No Snoop memory access. 1 = Latency tolerance in No Snoop memory access specified in the LTRC.*NSLTRV* field.



7.2.2.20.10 DMA Coalescing Management Threshold - DMCMNGTH (0x00015F20) MNG

Field	Bit(s)	Init.	Type	Description
RESERVED	3:0	0x0	RSV	Reserved.
DMCMNGTHR	19:4	0x100	RW	DMA Coalescing Management Threshold. BMC TX DMAC Threshold. This value defines the DMA coalescing management threshold in 16-byte units. When amount of empty space in internal transmit buffer exceeds *DMCMNGTHR* value, DMA coalescing is stopped and PCIe moves to L0 state. ***Note:*** If value is 0x0 condition to move out of DMA Coalescing due to passing the DMA Coalescing Management Threshold level is disabled.
RESERVED	31:20	0x0	RSV	Reserved.

7.2.2.21 VF Registers Mapping in the PF space

7.2.2.21.1 VF Control Register - VFCTRL[n] (0x00000300 + 0x4*n, n=0...63; WO) Target

This register array is the mapping of the VFs VFCTRL registers.

Fields definitions are the same as defined in Section 7.3.3.1.1.

7.2.2.21.2 VF Extended Interrupt Cause - VFEICR[n] (0x00000B00 + 0x4*n, n=0...63; RC/W1C) Interrupt

Fields definitions are the same as defined in Section 7.3.3.2.1.

7.2.2.21.3 VF Extended Interrupt Cause Set - VFEICS[n] (0x00000C00 + 0x4*n, n=0...63; WO) Interrupt

Fields definitions are the same as defined in Section 7.3.3.2.2.

7.2.2.21.4 VF Extended Interrupt Mask Set/Read - VFEIMS[n] (0x00000D00 + 0x4*n, n=0...63; RWS) Interrupt

Fields definitions are the same as defined in Section 7.3.3.2.3.

7.2.2.21.5 VF Extended Interrupt Mask Clear - VFEIMC[n] (0x00000E00 + 0x4*n, n=0...63; WO) Interrupt

Fields definitions are the same as defined in Section 7.3.3.2.4.

7.2.2.21.6 VF Good Packets Received Count - VFGPRC[n] (0x0000101C + 0x40*n, n=0...63; RW) DMA_RX

PF mirror of VFGPRC.

Fields definitions are the same as defined in Section 7.3.3.5.1.

7.2.2.21.7 VF Good Octets Received Count Low - VFGORC_LSB[n] (0x00001020 + 0x40*n, n=0...63; RW) DMA_RX

PF mirror of VFGORC_LSB.

Fields definitions are the same as defined in Section 7.3.3.5.2.

7.2.2.21.8 VF Multiple Receive Queues Command Register - VFMRQC[n] (0x00003400 + 0x4*n, n=0...63; RW) DBU_RX

PF mirror of VFMRQC registers of the VFs.

Fields definitions are the same as defined in Section 7.3.3.3.10.

7.2.2.21.9 VF Mailbox - VFMAILBOX[n] (0x00004C00 + 0x4*n, n=0...63; RW) Target

This register array is the mapping of the VFMAILBOX registers of the VFs.

Fields definitions are the same as defined in Section 7.3.3.1.5.

7.2.2.21.10 VF Extended Interrupt Auto Mask Enable - VFEIAM[n] (0x00004D00 + 0x4*n, n=0...63; RW) Interrupt

Fields definitions are the same as defined in Section 7.3.3.2.5.

7.2.2.21.11 VF Interrupt Vector Allocation Registers Misc - VFIVAR_MISC[n] (0x00004E00 + 0x4*n, n=0...63; RW) Interrupt

These registers maps the mailbox interrupt into MSI-X vector (PF mirror). See Mapping of Interrupt Causes Section for more details.

Fields definitions are the same as defined in Section 7.3.3.2.7.

7.2.2.21.12 VF Good Packets Transmitted Count - VFGPTC[n] (0x00008300 + 0x4*n, n=0...63; RO) STAT

PF mirror of VFGPTC.

Fields definitions are the same as defined in Section 7.3.3.5.5.

7.2.2.21.13 VF Good Octets Transmitted Count LSB - VFGOTC_LSB[n] (0x00008400 + 0x8*n, n=0...63; RO) STAT

PF mirror of VFGOTC_LSB.

Fields definitions are the same as defined in Section 7.3.3.5.6.

7.2.2.21.14 VF Good Octets Transmitted Count MSB - VFGOTC_MSB[n] (0x00008404 + 0x8*n, n=0...63; RO) STAT

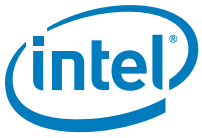
PF mirror of VFGOTC_MSB.

Fields definitions are the same as defined in Section 7.3.3.5.7.

7.2.2.21.15 VF Multicast Packets Received Count - VFMPRC[n] (0x0000D01C + 0x40*n, n=0...63; RO) DMA_RX

PF mirror of VFMPRC.

Fields definitions are the same as defined in Section 7.3.3.5.4.



7.2.2.21.16 VF Good Octets Received Count High - VFGORC_MSB[n] (0x0000D020 + 0x40*n, n=0...63; RW) DMA_RX

PF mirror of VFGORC_MSB.

Fields definitions are the same as defined in Section 7.3.3.5.3.

7.2.2.21.17 VF Interrupt Vector Allocation Registers - VFIVAR[n] (0x00012500 + 0x4*n, n=0...63; RW) Interrupt

These registers map VF interrupt causes into MSI-X vectors (PF mirror). See Mapping of Interrupt Causes Section for more details.

Fields definitions are the same as defined in Section 7.3.3.2.6.

7.2.2.21.18 PF Mailbox Memory - PFMBMEM[n,m] (0x00013000 + 0x4*n + 0x40*m, n=0...15, m=0...63; RW) Target

Mailbox Memory for PF and VF Driver Communications. The mailbox size for each VM is 64 bytes accessed by 16 x 32-bit registers. Locations can be accessed as 32- or 64-bit words. This is the mapping of this memory in the PF space.

Fields definitions are the same as defined in Section 7.3.3.1.4.

7.2.2.22 MNG_IOSF_SB

7.2.2.22.1 MNG SB-IOSF Request Control-Status - MNGSB_MSGCTL (0x00010020) MNG

IOSF Sideband Message Control Status Register.

Field	Bit(s)	Init.	Type	Description
HDR_DWS	1:0	0x0	RW	The number of header DWords to send. Should be 0x3 for SPI access.
RESERVED	7:2	0x0	RSV	Reserved.
EXP_RDW	16:8	0x0	RW	The number of DWords to expect in the response. The number DATA DWs per DMA message. The number to add to address (DW1) in Auto-Increment or DMA Modes.
RESERVED	25:17	0x0	RSV	Reserved.
MSG_MODE	27:26	0x0	RW	00b = Normal 01b = Register 10b = Auto Increment 11b = DMA
TOKEN_MODE	29:28	0x0	RW	00b = Normal 0 b1 = (Get)/Keep Token 10b = Reserved 11b = Reserved
BARCLR	30	0b	WO	Writing a 1b to this bit clears the buffer of the BAR interface, should be done anytime a FLASH WRITE or ERASE is executed (with setting CMDV). Can be set alone also.
CMDV	31	0b	SC	Command (Message/Response) is in progress. Set to send an IOSF request/action. Cleared by HW when response is received.

7.2.2.22.2 MNG SB-IOSF Response Control-Status - MNGSB_RSPCTL (0x00010024) MNG

IOSF Sideband Response Control Status Register.

Field	Bit(s)	Init.	Type	Description
DMA_MSG_DWORDS	8:0	0x0	RW	The number DATA DWs per DMA sideband transaction.
RESERVED	25:9	0x0	RSV	Reserved.
RSP_MODE	27:26	0x0	RW	
RESERVED	29:28	0x0	RSV	Reserved.
RSP_BAD_LEN	30	0b	RO	Response does not equal expected length.
RSP_ERR	31	0b	RO	Response Field not SUCESS.

7.2.2.22.3 MNG SB-IOSF DMA Address - MNGSB_DADD (0x00010030) MNG

IOSF Sideband DMA Internal Address (Byte Address).

Field	Bit(s)	Init.	Type	Description
ADDR	31:0	0x0	RW	This is the internal address in Firmware memory where the data being written is placed in (must be DW aligned).

7.2.2.22.4 MNG SB-IOSF Auto-Inc Counter - MNGSB_DCNT (0x00010034) MNG

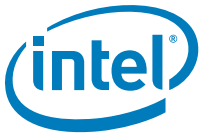
IOSF Sideband Auto-Increment DWord Count Register.

Field	Bit(s)	Init.	Type	Description
BYTE_CNT	31:0	0x0	RW	The size of the DMA transfer. Must be a multiple of 64 bytes.

7.2.2.22.5 MNG SB-IOSF Request Hdr0 - MNGSB_WHDR0 (0x000100F4) MNG

IOSF SB Message 1st Word.

Field	Bit(s)	Init.	Type	Description
RESERVED	11:0	0x0	RSV	Reserved.
DEST_SEL	15:12	0x0	RW	*DEST_SEL* is used to indicate to the HW which destination the message should be sent to: DEST_SEL = 0 is enabled only in un-secure mode and allows FW full control of the SB IOSF message header except for the SAI value DEST_SEL = 1 indicates SPI DEST_SEL = 2 thru 15 are reserved at this point. DEST_SEL = 16 thru 31 generates a dummy response.
OPCODE_SEL	23:16	0x0	RW	Selects the opcodes straps, When DEST_SEL =0 is used and enabled This field is used as the raw OPCODE value When DEST_SEL = 1 this selects per-set SPI access OPCODE 0x1 = SPI Read 0x2 = SPI Write 0x4 = SPI Erase 0x6 = CRRd 0x7 = CRWr All other values are reserved.



Field	Bit(s)	Init.	Type	Description
TAG	30:24	0x0	RW	TAG is a value that the IP may use to tag its messages. The value is taken as is fro this register without HW modification. At this point TAG should always be ZERO.
RSV	31	0b	RSV	Not used, replaced by HW in Messages

7.2.2.22.6 MNG SB-IOSF Request Hdr1 - MNGSB_WHDR1 (0x000100F8) MNG

IOSF SB Message 2nd Word.

Field	Bit(s)	Init.	Type	Description
ADDR	31:0	0x0	RW	For Flash RD/WR this is the actual memory address (must be DW aligned). For Flash Erase this is the actual memory address (must be 4KB aligned). For DMA RD/WR this is the actual memory address (must be 64B aligned).

7.2.2.22.7 MNG SB-IOSF Request Hdr2 - MNGSB_WHDR2 (0x000100FC) MNG

IOSF SB Message 3rd Word.

Field	Bit(s)	Init.	Type	Description
LENGTH	31:0	0x0	RW	For Flash RD/WR this is the actual length in bytes (MAX is 0x40). For Flash Erase the only valid value is 0x10 which is 4 KB. For DMA RD/WR this must be 0x40.

7.2.2.22.8 MNG SB-IOSF Write Data - MNGSB_WDATA (0x00010100) MNG

IOSF SB Write FIFO (4th + Message Words).

Field	Bit(s)	Init.	Type	Description
DATA	31:0	0x0	RW	Only used in regular Flash Write (non DMA). Will contain the data to be written.

7.2.2.22.9 MNG SB-IOSF Response Hdr0 - MNGSB_RHDR0 (0x000102FC) MNG

IOSF SB Response 1st Word.

Field	Bit(s)	Init.	Type	Description
DESTINATION	7:0	0x0	RW	
SOURCE	15:8	0x0	RW	
OPCODE	23:16	0x0	RW	
TAG	26:24	0x0	RW	
RESPONSE	29:27	0x0	RW	
RSVD	30	0b	RSV	
EH	31	0b	RW	Extended Header.

7.2.2.22.10 MNG SB-IOSF Read Data - MNGSB_RDATA (0x00010300) MNG

IOSF SB Read Data FIFO (2nd + Response Words).

Field	Bit(s)	Init.	Type	Description
DATA	31:0	0x0	RW	When opcode is: Flash Rd/RegRd then Read Data (FIFO). Otherwise, reserved

7.2.3 BAR3 Registers Summary

Table 7.4. BAR3 Registers Summary

Offset / Alias Offset	Abbreviation	Name	Block	Page
MSI-X Table Registers				
0x00000000 + 0x10*n, n=0...255	MSIXTADD[n]	MSI-X Table Entry Lower Address	MSIX	438
0x00000004 + 0x10*n, n=0...255	MSIXTUADD[n]	MSI-X Table Entry Upper Address	MSIX	438
0x00000008 + 0x10*n, n=0...255	MSIXTMSG[n]	MSI-X Table Entry Message	MSIX	439
0x0000000C + 0x10*n, n=0...255	MSIXVCTRL[n]	MSI-X Table Entry Vector Control	MSIX	439

7.2.4 Detailed Register Description - PF BAR3

7.2.4.1 MSI-X Table Registers

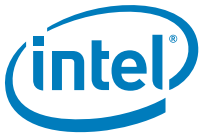
The MSI-X capability is described in Section 9.3.8. The MSI-X table is described in Section 9.3.8.2 and the Pending Bit Array (PBA) is described in Section 9.3.8.3. These registers are located in the MSI-X BAR.

7.2.4.1.1 MSI-X Table Entry Lower Address - MSIXTADD[n] (0x00000000 + 0x10*n, n=0...255) MSIX

Field	Bit(s)	Init.	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.

7.2.4.1.2 MSI-X Table Entry Upper Address - MSIXTUADD[n] (0x00000004 + 0x10*n, n=0...255) MSIX

Field	Bit(s)	Init.	Type	Description
MSIXTUADD	31:0	0x0	RW	Message Upper Address. System-specified message upper address bits. If this field is zero, Single Address Cycle (SAC) messages are used. If this field is non-zero, Dual Address Cycle (DAC) messages are used. This field is read/write.



7.2.4.1.3 MSI-X Table Entry Message - MSIXTMSG[n] (0x00000008 + 0x10*n, n=0...255) MSIX

Field	Bit(s)	Init.	Type	Description
MSIXTMSG	31:0	0x0	RW	Message Data. System-specified message data. For MSI-X messages, the contents of this field from an MSI-X table entry specifies the data driven on AD[31:0] during the memory write transaction's data phase. This field is read/write.

7.2.4.1.4 MSI-X Table Entry Vector Control - MSIXVCTRL[n] (0x0000000C + 0x10*n, n=0...255) MSIX

Field	Bit(s)	Init.	Type	Description
MSIXVCTRL	0	1b	RW	Mask Bit. When this bit is set, the function is prohibited from sending a message using this MSI-X table entry. However, any other MSI-X table entries programmed with the same vector are still capable of sending an equivalent message unless they are also masked. This bit's state after reset is 1b (entry is masked). This bit is read/write.
RESERVED	31:1	0x0	RSV	Reserved. After reset, the state of these bits must be 0b. However, for potential future use, software must preserve the value of these reserved bits when modifying the value of other Vector Control bits. If software modifies the value of these reserved bits, the result is undefined.

7.3 Device Registers - VF

7.3.1 VF Registers mapping in the PF space

Abbreviation	Virtual Address	Physical Address
VFTPH_RXCTRL	0x0000100C + 0x40*n, n=0...7	0x00002200 + 0x4*n, n=0...15
VFSRRCTL	0x00001014 + 0x40*n, n=0...7	0x00002100 + 0x4*n, n=0...15

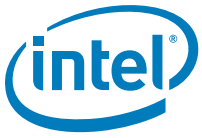
7.3.2 BAR0 Registers Summary

Table 7.5. BAR0 Registers Summary

Offset / Alias Offset	Abbreviation	Name	Block	Page
General Control Registers - VF				
0x00000000	VFCTRL	VF Control Register	Target	441
0x00000008	VFSTATUS	VF Device Status Register	Target	441
0x00000010	VFLINKS	Link Status Register	MAC	441
0x00000200 + 0x4*n, n=0...15	VFMBMEM[n]	VF Mailbox Memory	Target	441
0x000002FC	VFMAILBOX	VF Mailbox	Target	441
Interrupt Registers - VF				
0x00000100	VFEICR	VF Extended Interrupt Cause	Interrupt	442
0x00000104	VFEICS	VF Extended Interrupt Cause Set	Interrupt	442
0x00000108	VFEIMS	VF Extended Interrupt Mask Set/Read	Interrupt	442

Table 7.5. BAR0 Registers Summary (Continued)

Offset / Alias Offset	Abbreviation	Name	Block	Page
0x0000010C	VFEIMC	VF Extended Interrupt Mask Clear	Interrupt	442
0x00000114	VFEIAM	VF Extended Interrupt Auto Mask Enable	Interrupt	442
0x00000120 + 0x4*n, n=0...3	VFIVAR[n]	VF Interrupt Vector Allocation Registers	Interrupt	442
0x00000140	VFIVAR_MISC	VF Interrupt Vector Allocation Registers Misc	Interrupt	443
0x00000148	VFPBACL	VF MSI-X PBA Clear	PCIe	443
0x00000180 + 0x4*n, n=0...1	VFRSCINT[n]	VF RSC Enable Interrupt	Interrupt	443
0x00000820 + 0x4*n, n=0...1	VFEITR[n]	VF Extended Interrupt Throttle Registers	Interrupt	443
Receive Registers - VF				
0x00000300	VFPSRTYPE	VF Replication Packet Split Receive Type Register	DBU_RX	444
0x00001000 + 0x40*n, n=0...7	VFRDBAL[n]	VF Receive Descriptor Base Address Low	DMA_RX	444
0x00001004 + 0x40*n, n=0...7	VFRDBAH[n]	VF Receive Descriptor Base Address High	DMA_RX	444
0x00001008 + 0x40*n, n=0...7	VFRDLEN[n]	VF Receive Descriptor Length	DMA_RX	444
0x00001010 + 0x40*n, n=0...7	VFRDH[n]	VF Receive Descriptor Head	DMA_RX	444
0x00001014 + 0x40*n, n=0...7	VFSRRCTL[n]	VF Split and Replication Receive Control Registers	DMA_RX	444
0x00001018 + 0x40*n, n=0...7	VFRDT[n]	VF Receive Descriptor Tail	DMA_RX	444
0x00001028 + 0x40*n, n=0...7	VFRXDCTL[n]	VF Receive Descriptor Control	DMA_RX	444
0x0000102C + 0x40*n, n=0...7	VFRSCCTL[n]	VF RSC Control	DMA_RX	444
0x00003000	VFMRQC	VF Multiple Receive Queues Command Register	DBU_RX	444
0x00003100 + 0x4*n, n=0...9	VFRSSRK[n]	VF RSS Random Key Register	DBU_RX	445
0x00003200 + 0x4*n, n=0...15	VFRETA[n]	VF Redirection Table	DBU_RX	445
Transmit Registers - VF				
0x00002000 + 0x40*n, n=0...7	VFTDBAL[n]	VF Transmit Descriptor Base Address Low	DMA_TX	445
0x00002004 + 0x40*n, n=0...7	VFTDBAH[n]	VF Transmit Descriptor Base Address High	DMA_TX	445
0x00002008 + 0x40*n, n=0...7	VFTDLEN[n]	VF Transmit Descriptor Length	DMA_TX	445
0x00002010 + 0x40*n, n=0...7	VFTDH[n]	VF Transmit Descriptor Head	DMA_TX	445
0x00002018 + 0x40*n, n=0...7	VFTDT[n]	VF Transmit Descriptor Tail	DMA_TX	446
0x00002028 + 0x40*n, n=0...7	VFTXDCTL[n]	VF Transmit Descriptor Control	DMA_TX	446
0x00002038 + 0x40*n, n=0...7	VFTDWBAL[n]	VF Tx Descriptor Completion Write Back Address Low	DMA_TX	446
0x0000203C + 0x40*n, n=0...7	VFTDWBAH[n]	VF Tx Descriptor Completion Write Back Address High	DMA_TX	446
Statistic Register Descriptions - VF				
0x0000101C	VFGPRC	VF Good Packets Received Count	DMA_RX	446
0x00001020	VFGORC_LSB	VF Good Octets Received Count Low	DMA_RX	446
0x00001024	VFGORC_MSB	VF Good Octets Received Count High	DMA_RX	446
0x00001034	VFMPRC	VF Multicast Packets Received Count	DMA_RX	447
0x0000201C	VFGPTC	VF Good Packets Transmitted Count	STAT	447
0x00002020	VFGOTC_LSB	VF Good Octets Transmitted Count LSB	STAT	447
0x00002024	VFGOTC_MSB	VF Good Octets Transmitted Count MSB	STAT	447



7.3.3 Detailed Register Description - VF BAR0

7.3.3.1 General Control Registers - VF\

7.3.3.1.1 VF Control Register - VFCTRL (0x00000000) Target

Field	Bit(s)	Init.	Type	Description
RESERVED	25:0	0x0	RSV	Reserved.
RST	26	0b	WO	VF Reset. This bit performs a reset of the queue enable and the interrupt registers of the VF.
RESERVED	31:27	0x0	RSV	Reserved.

7.3.3.1.2 VF Device Status Register - VFSTATUS (0x00000008; RO) Target

This register is a mirror of the PF status register.

Fields definitions are the same as defined in Section 7.2.2.1.2.

7.3.3.1.3 Link Status Register - VFLINKS (0x00000010; RO) MAC

This register is the mapping of the PF's LINKS register.

Fields definitions are the same as defined in Section 7.2.2.13.5.

7.3.3.1.4 VF Mailbox Memory - VFMBMEM[n] (0x00000200 + 0x4*n, n=0...15) Target

Mailbox Memory for PF and VF Driver Communications. The mailbox size for each VM is 64 bytes accessed by 16 x 32-bit registers. Locations can be accessed as 32- or 64-bit words.

Field	Bit(s)	Init.	Type	Description
MAILBOX_DATA	31:0	X	RW	Mailbox data is composed of 16 x 4-byte registers.

7.3.3.1.5 VF Mailbox - VFMAILBOX (0x000002FC) Target

This register is cleared by VLFR (excepted to RSTI bit).

Field	Bit(s)	Init.	Type	Description
REQ	0	0b	WO	Request for PF Ready. Setting this bit, causes an interrupt to the PF. This bit always reads as 0b. Setting this bit sets the corresponding bit in the VFREQ field in PFMBICR register.
ACK	1	0b	WO	PF message received. Setting this bit, causes an interrupt to the PF. This bit always reads as 0b. Setting this bit sets the corresponding bit in the *VFACK* field in PFMBICR register.
VFU	2	0b	RW	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.

Field	Bit(s)	Init.	Type	Description
RSTI	6	1b	RO	Indicates that the PF reset the shared resources and the reset sequence is in progress. This bit is not affected by VFLR.
RSTD	7	0b	RC	Indicates that a PF software reset completed.
RESERVED	31:8	0x0	RSV	Reserved.

7.3.3.2 Interrupt Registers - VF

7.3.3.2.1 VF Extended Interrupt Cause - VFEICR (0x00000100) Interrupt

Field	Bit(s)	Init.	Type	Description
MSIX	2:0	0x0	RW1C	Indicates an interrupt cause mapped to MSI-X vectors 2:0.
RESERVED	31:3	0x0	RSV	Reserved.

7.3.3.2.2 VF Extended Interrupt Cause Set - VFEICS (0x00000104) Interrupt

Field	Bit(s)	Init.	Type	Description
MSIX	2:0	0x0	WO	Sets to the corresponding EICR bit of MSI-X vectors 2:0.
RESERVED	31:3	0x0	RSV	Reserved.

7.3.3.2.3 VF Extended Interrupt Mask Set/Read - VFEIMS (0x00000108) Interrupt

Field	Bit(s)	Init.	Type	Description
MSIX	2:0	0x0	RWS	Sets the Mask bit for the corresponding EICR bit of MSI-X vectors 2:0.
RESERVED	31:3	0x0	RSV	Reserved.

7.3.3.2.4 VF Extended Interrupt Mask Clear - VFEIMC (0x0000010C) Interrupt

Field	Bit(s)	Init.	Type	Description
MSIX	2:0	0x0	WO	Clears the Mask bit for the corresponding EICR bit of MSI-X vectors 2:0.
RESERVED	31:3	0x0	RSV	Reserved.

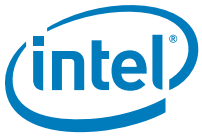
7.3.3.2.5 VF Extended Interrupt Auto Mask Enable - VFEIAM (0x00000114) Interrupt

Field	Bit(s)	Init.	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.

7.3.3.2.6 VF Interrupt Vector Allocation Registers - VFIVAR[n] (0x00000120 + 0x4*n, n=0...3) Interrupt

These registers map VF interrupt causes into MSI-X vectors. See Mapping of Interrupt Causes Section for more details.

Field	Bit(s)	Init.	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).



Field	Bit(s)	Init.	Type	Description
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).
RESERVED	30:25	0x0	RSV	Reserved.
INT_ALLOC_VAL_3	31	0b	RW	Valid bit for INT_Alloc[3].

7.3.3.2.7 VF Interrupt Vector Allocation Registers Misc - VFIVAR_MISC (0x00000140) Interrupt

This register maps the mailbox interrupt into MSI-X vector. See Mapping of Interrupt Causes Section for more details.

Field	Bit(s)	Init.	Type	Description
INT_ALLOC_0	1:0	X	RW	Defines the MSI-X vector assigned to the mailbox 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.

7.3.3.2.8 VF MSI-X PBA Clear - VFPBACL (0x00000148) PCIe

Field	Bit(s)	Init.	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.

7.3.3.2.9 VF RSC Enable Interrupt - VFRSCINT[n] (0x00000180 + 0x4*n, n=0...1; RW) Interrupt

Fields definitions are the same as defined in Section 7.2.2.6.20.

7.3.3.2.10 VF Extended Interrupt Throttle Registers - VFEITR[n] (0x00000820 + 0x4*n, n=0...1; RW) Interrupt

Fields definitions are the same as defined in Section 7.2.2.6.4.

7.3.3.3 Receive Registers - VF

7.3.3.3.1 VF Replication Packet Split Receive Type Register - VFPSRTYPE (0x00000300; RW) DBU_RX

Fields definitions are the same as defined in Section 7.2.2.8.17.

7.3.3.3.2 VF Receive Descriptor Base Address Low - VFRDBAL[n] (0x00001000 + 0x40*n, n=0...7; RW) DMA_RX

Fields definitions are the same as defined in Section 7.2.2.9.1.

7.3.3.3.3 VF Receive Descriptor Base Address High - VFRDBAH[n] (0x00001004 + 0x40*n, n=0...7; RW) DMA_RX

Fields definitions are the same as defined in Section 7.2.2.9.2.

7.3.3.3.4 VF Receive Descriptor Length - VFRDLEN[n] (0x00001008 + 0x40*n, n=0...7; RW) DMA_RX

Fields definitions are the same as defined in Section 7.2.2.9.3.

7.3.3.3.5 VF Receive Descriptor Head - VFRDH[n] (0x00001010 + 0x40*n, n=0...7; RO) DMA_RX

Fields definitions are the same as defined in Section 7.2.2.9.4.

7.3.3.3.6 VF Split and Replication Receive Control Registers - VFSRRCTL[n] (0x00001014 + 0x40*n, n=0...7; RW) DMA_RX

Fields definitions are the same as defined in Section 7.2.2.9.5.

7.3.3.3.7 VF Receive Descriptor Tail - VFRDT[n] (0x00001018 + 0x40*n, n=0...7; RW) DMA_RX

Fields definitions are the same as defined in Section 7.2.2.9.6.

7.3.3.3.8 VF Receive Descriptor Control - VFRXDCTL[n] (0x00001028 + 0x40*n, n=0...7; RW) DMA_RX

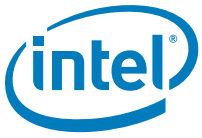
Fields definitions are the same as defined in Section 7.2.2.9.7.

7.3.3.3.9 VF RSC Control - VFRSCCTL[n] (0x0000102C + 0x40*n, n=0...7; RW) DMA_RX

Fields definitions are the same as defined in Section 7.2.2.9.8.

7.3.3.3.10 VF Multiple Receive Queues Command Register - VFMRQC (0x00003000) DBU_RX

Field	Bit(s)	Init.	Type	Description
RSSE	0	0b	RW	RSS Enable. This bit enables the VF RSS mechanism. 0 = VF RSS disabled. 1 = VF RSS enable.



Field	Bit(s)	Init.	Type	Description
RESERVED	15:1	0x0	RSV	Reserved.
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; 0x0. ***Note:*** On Tunnel packets IPv4-IPv6, only the IPv4 header might be used for the RSS filtering.

7.3.3.3.11 VF RSS Random Key Register - VFRSSRK[n] (0x00003100 + 0x4*n, n=0...9; RW) DBU_RX

PF mirror of VFRSSK registers of the VFs.

Fields definitions are the same as defined in Section 7.2.2.8.25.

7.3.3.3.12 VF Redirection Table - VFRETA[n] (0x00003200 + 0x4*n, n=0...15; RW) DBU_RX

VF version of VFRETA registers. The redirection table has 64 entries in 16 registers.

Fields definitions are the same as defined in Section 7.2.2.8.26.

7.3.3.4 Transmit Registers - VF

7.3.3.4.1 VF Transmit Descriptor Base Address Low - VFTDBAL[n] (0x00002000 + 0x40*n, n=0...7; RW) DMA_TX

Fields definitions are the same as defined in Section 7.2.2.10.5.

7.3.3.4.2 VF Transmit Descriptor Base Address High - VFTDBAH[n] (0x00002004 + 0x40*n, n=0...7; RW) DMA_TX

Fields definitions are the same as defined in Section 7.2.2.10.6.

7.3.3.4.3 VF Transmit Descriptor Length - VFTDLEN[n] (0x00002008 + 0x40*n, n=0...7; RW) DMA_TX

Fields definitions are the same as defined in Section 7.2.2.10.7.

7.3.3.4.4 VF Transmit Descriptor Head - VFTDH[n] (0x00002010 + 0x40*n, n=0...7; RO) DMA_TX

Fields definitions are the same as defined in Section 7.2.2.10.8.

7.3.3.4.5 VF Transmit Descriptor Tail - VFTDT[n] (0x00002018 + 0x40*n, n=0...7; RW) DMA_TX

Fields definitions are the same as defined in Section 7.2.2.10.9.

7.3.3.4.6 VF Transmit Descriptor Control - VFTXDCTL[n] (0x00002028 + 0x40*n, n=0...7; RW) DMA_TX

Fields definitions are the same as defined in Section 7.2.2.10.10.

7.3.3.4.7 VF Tx Descriptor Completion Write Back Address Low - VFTDWBAL[n] (0x00002038 + 0x40*n, n=0...7; RW) DMA_TX

Fields definitions are the same as defined in Section 7.2.2.10.11.

7.3.3.4.8 VF Tx Descriptor Completion Write Back Address High - VFTDWBAH[n] (0x0000203C + 0x40*n, n=0...7; RW) DMA_TX

Fields definitions are the same as defined in Section 7.2.2.10.12.

7.3.3.5 Statistic Register Descriptions - VF

Registers in this section are RO by VF and RW by PF. Statistics are reset by PF clearing the register.

7.3.3.5.1 VF Good Packets Received Count - VFGPRC (0x0000101C) DMA_RX

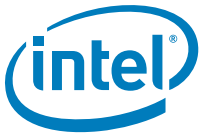
Field	Bit(s)	Init.	Type	Description
VFGPRC	31:0	0x0	RW	Number of good packets received for this VF (of any length). This counter includes loopback packets or replications of multicast packets. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.

7.3.3.5.2 VF Good Octets Received Count Low - VFGORC_LSB (0x00001020) DMA_RX

Field	Bit(s)	Init.	Type	Description
VFGORC_LSB	31:0	0x0	RW	Number of good octets received (32 LSB of a 36-bit counter) by the queues allocated to this VF. The counter includes loopback packets or replications of multicast packets. This register includes bytes received in a packet from the <Destination Address> field through the <CRC> field, inclusively. Octets are counted on the VF interface rather than on the network interface. For example, LinkSec octets are not counted. Bytes of RSC are counted before coalescing. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.

7.3.3.5.3 VF Good Octets Received Count High - VFGORC_MSB (0x00001024) DMA_RX

Field	Bit(s)	Init.	Type	Description
VFGORC_MSB	3:0	0x0	RW	Number of good octets received (4 MSB of a 36-bit counter) by the queues allocated to this VF. See the VFGORC_LSB register description for more details. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xF to 0x0.
RESERVED	31:4	0x0	RSV	Reserved.



7.3.3.5.4 VF Multicast Packets Received Count - VFMPRC (0x00001034) DMA_RX

Field	Bit(s)	Init.	Type	Description
VFMPRC	31:0	0x0	RO	Number of multicast good packets received by this VF (of any length) that pass the Ethernet MAC Address filtering (excluding broadcast packets). The counter does not count received flow control packets. This register increments only if receives are enabled. This register does not count packets counted by the Missed Packet Count (MPC) register. This counter also includes loopback packets or replications of multicast packets. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.

7.3.3.5.5 VF Good Packets Transmitted Count - VFGPTC (0x0000201C) STAT

Field	Bit(s)	Init.	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.

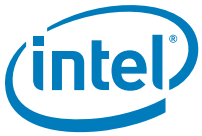
7.3.3.5.6 VF Good Octets Transmitted Count LSB - VFGOTC_LSB (0x00002020) STAT

Field	Bit(s)	Init.	Type	Description
VFGOTC_LSB	31:0	0x0	RO	Number of good octets transmitted (32 LSB of a 36-bit counter) by the queues allocated to this VF. T his register includes bytes transmitted in a packet from the <Destination Address> field through the <CRC> field, inclusively. This register counts octets of the packets counted by the VFGPTC register. Octets are counted on the VF interface rather than on the network interface. For example, LinkSec octets are not counted. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000.

7.3.3.5.7 VF Good Octets Transmitted Count MSB - VFGOTC_MSB (0x00002024) STAT

Field	Bit(s)	Init.	Type	Description
VFGOTC_MSB	3:0	0x0	RO	Number of good octets transmitted (4 MSB of a 36-bit counter) by the queues allocated to this VF. See the description of the VFGOTC-LSB register for more details. The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xF to 0x0.
RESERVED	31:4	0x0	RSV	Reserved.

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



8.0 PCIe Programming Interface

8.1 Overview

The integrated 10 GbE LAN controller 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 the shared SPI Flash settings as detailed in the following sections):

- Mandatory PCI configuration registers - Section 8.2.2
- Legacy PCI capabilities - Section 8.2.3
  - Power management capabilities
  - MSI / MSI-X capabilities
  - Vital Product Data (VPD) capability
- PCIe extended capabilities - Section 8.2.4
  - Advanced Error Reporting (AER)
  - Serial ID
  - Alternate requester ID.
  - Single root IOV (SR-IOV)
  - Latency Tolerance Reporting (LTR)
  - Access Control Services (ACS)
  - Secondary PCIe

8.1.1 PCIe Configuration Space in an Integrated I/O Interface Connected System

When the integrated Integrated 10 GbE LAN Controller is connected using the integrated I/O interface and not an actual physical PCIe link, the PCIe configurations are subject to some modifications. One such example are the PCIe link registers that are not relevant anymore and should reflect a static setup.

In the following sections these differences are highlighted where relevant.

8.1.2 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.
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 shared SPI Flash. 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.

8.2 PCIe Register Map

8.2.1 PCIe Configuration Space Summary

Table 8.1 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 shared SPI Flash)
- Magenta fields indicate hard-coded values.
- Other fields contain RW attributes.

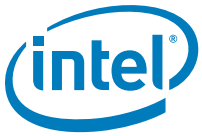


Table 8.1. PCI Configuration Registers Map - LAN Functions

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 (0x0)
	0x10	Base Address Register 0			
	0x14	Base Address Register 1			
	0x18	Base Address Register 2			
	0x1C	Base Address Register 3			
	0x20	Base Address Register 4			
	0x24	Base Address Register 5			
	0x28	CardBus CIS pointer (0x0000)			
	0x2C	Subsystem ID		Subsystem Vendor ID	
	0x30	Expansion ROM Base Address			
	0x34	Reserved			Cap Ptr (0x40)
	0x38	Reserved			
	0x3C	Max Latency (0x00)	Min Grant (0x00)	Interrupt Pin (0x01...0x04)	Interrupt Line (0x00)
Power Management Capability	0x40	Power Management Capabilities		Next Pointer (0x50)	Capability ID (0x01)
	0x44	Data	Bridge Support Extensions	Power Management Control & Status	
PCI / PCIe Capabilities	0x50...0x67	MSI Capability			
	0x70...0x7B	MSI-X Capability			
	0xA0...0xDB	PCIe Capability			
	0xE0...0xE7	VPD Capability			
Extended PCIe Configuration	0x100...0x12B	AER Capability			
	0x140...0x14B	Serial ID Capability			
	0x150...0x157	ARI Capability			
	0x160...0x19F	SR-IOV Capability			
	Reserved	Reserved			
	0x1B0...0x1B7	ACS Capability			
	0x1C0...0x1C7	LTR Capability			
	0x1D0...0x1E3	Secondary PCIe Extended Capability			

Table 8.2. 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 (0x0)
	0x10	Base Address Register 0			
	0x14	Base Address Register 1 (0x0)			
	0x18	Base Address Register 2 (0x0)			
	0x1C	Base Address Register 3 (0x0)			
	0x20	Base Address Register 4 (0x0)			
	0x24	Base Address Register 5 (0x0)			
	0x28	CardBus CIS pointer (0x0000)			
	0x2C	Subsystem Device ID		Subsystem Vendor ID	
	0x30	Expansion ROM Base Address (0x0)			
	0x34	Reserved			Cap Ptr (0x40)
	0x38	Reserved			
	0x3C	Max Latency (0x00)	Min Grant (0x00)	Interrupt Pin (0x00)	Interrupt Line (0x00)
PCI / PCIe Capabilities	0x40...0x47	Power Management Capability			
	0xA0...0xDB	PCIe Capability			
	0xE0...0xE7	VPD Capability			
Extended PCIe Configuration	0x100...0x12B	AER Capability			
	0x140...0x14B	Serial ID Capability			
	0x1B0...0x1B7	ACS Capability			

8.2.1.1 Sharing Among PCI Functions

The integrated 10 GbE LAN controller supports multiple PCI functions. As each function exposes a PCIe configuration space, each register and each field is either shared among the functions or is replicated per each PCI function. In each section a table lists configuration sharing of the registers described in this section. Also, the description of each field describes special considerations regarding configuration sharing.

8.2.2 Mandatory PCI Configuration Registers

Table 8.3. Configuration Sharing of PCI Configuration Space

Field	Sub-field	Shared?	Replicated?	Comments
Vendor ID	Vendor ID	x		
Device ID	Device ID		x	
Command Register	I/O Access Enable		x	Issue UR per PF if disabled.

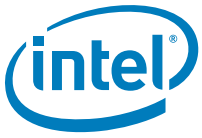


Table 8.3. Configuration Sharing of PCI Configuration Space (Continued)

Field	Sub-field	Shared?	Replicated?	Comments
	Memory Access Enable		x	Issue UR per PF if disabled.
	Bus Master Enable		x	
	Parity Error Response		x	Enables certain error reporting per PF.
	SERR# Enable		x	Controls error reporting per PF.
	Interrupt Disable		x	Selection of interrupt method per PF.
Status Register	Interrupt Status		x	
	Capabilities List	x		Hardwired to 1b.
	Data Parity Reported / Master Data Parity Error		x	Reports poisoned packets per PF.
	Signaled Target Abort		x	Reports completer abort per PF.
	Received Target Abort		x	Reports receiving a completer abort per PF.
	Received Master Abort		x	Reports receiving an UR per PF.
	Signaled System Error		x	Reports Fatal / non-fatal message per PF.
	Detected Parity Error		x	Reports receiving a poisoned TLP per PF.
Revision Register		x		
Class Code Register			x	Per function type.
Cache Line Size Register			x	Does not affect device behavior.
Latency Timer		x		Hardwired to 00h in PCIe.
Header Type Register		x		
BIST		x		Not supported (0x00).
BARs	Memory BAR		x	
	I/O BAR		x	
	MSI-X BAR		x	See MSI-X capability.
I/O BAR Mapping	IOADDR, IODATA		x	
Subsystem Vendor ID		x		
Subsystem ID		x		
Expansion ROM			x	
Cap_Ptr Register		x		
Interrupt Line Register			x	Just store the register value.
Interrupt Pin Register			x	Separate interrupt number (A-D) per PF.
Min. Grant		x		
Max Latency		x		

8.2.2.1 Vendor ID Register (0x0; RO)

This is a read-only register that has the same value for all PCI functions. It identifies unique Intel products. The value of this field is loaded from the PCI_VENDORID register loaded from shared SPI Flash. The default value, if not loaded, is 0x8086.

8.2.2.2 Device ID Register (0x2; RO)

This is a read-only register that identifies individual the integrated 10 GbE LAN controller PCI functions. Both ports have the same default value equals as defined in the following table, and can be auto-loaded from the shared SPI Flash during initialization with different values for each port as well as the dummy function (See Section 3.3 for dummy function relevance).

The device ID values available for different the integrated 10 GbE LAN controller SKUs are listed in the following table:

Note: At initialization this register is loaded from the shared SPI Flash if the PCI_CAPSUP.LOAD_DEV_ID bit is set. In which case the value of each PF is loaded from the PCI_PFDEVID field in the shared SPI Flash.

Table 8-4. Device IDs per SKUs/Connection type

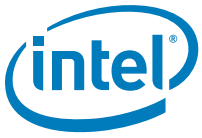
Integrated 10 GbE LAN Controller Product	Device ID[1]	Branding String
Silicon default (10 GbE SKU)	0x1306	Intel® Ethernet Connection X553 (X553)
Silicon default, 1 GbE SKU	0x1307	X553
VF (generic hypervisor)	0x15C5	X553 Virtual Function
VF (Microsoft* Hyper-V)	0x15B4	X553 Virtual Function
KR backplane, 10 GbE maximum	0x15C2	X553 10 GbE Backplane
KX backplane, 2.5 GbE	0x15C3	X553 1 GbE Backplane
SFP+ (via native SFI)	0x15C4	X553 10 GbE SFP+
SFP+ (via Inphi* CS4227)	0x15CE	X553 10 GbE SFP+
10GBASE-T (via Intel® Ethernet Connection X557-AT)	0x15C8	X553/X557-AT
1GBASE-T (via Marvell* 1543), 10 GbE SKU	0x15E4	X553 1 GbE
1GBASE-T (via Marvell 1543), 1 GbE SKU	0x15E5	X553 1 GbE
SGMII Backplane - 10 GbE	0x15C6	X553 10 GbE
SGMII Backplane - 1 GbE	0x15C7	X553 1 GbE

1. Odd device IDs indicates 10 GbE disabled SKUs.

8.2.2.3 Command Register (0x4; RW)

Shaded bits are not used by this implementation and are hard wired to 0b. Each function has its own Command register. Unless explicitly specified, functionality is the same in both functions.

Bit(s)	Initial Value	Description
0	0b	I/O Access Enable. If I/O BAR is not requested, this bit is hard-wired to zero.
1	0b	Memory Access Enable.
2	0b	Enable Mastering, also named Bus Master Enable (BME). • LAN functions RW field. • Dummy function RO as zero field.
3	0b	Special Cycle Monitoring – Hard-wired to 0b.
4	0b	MWI Enable – Hard-wired to 0b.



Bit(s)	Initial Value	Description
5	0b	Palette Snoop Enable – Hard-wired to 0b.
6	0b	Parity Error Response.
7	0b	Wait Cycle Enable – Hard-wired to 0b.
8	0b	SERR# Enable.
9	0b	Fast Back-to-Back Enable – Hard-wired to 0b.
10	0b	Interrupt Disable. When set, devices are prevented from generating legacy interrupt messages.
15:11	0b	Reserved.

8.2.2.4 Status Register (0x6; RO)

Shaded bits are not used by this implementation and are hard-wired to 0b. Each function has its own Status register. Unless explicitly specified, functionality is the same in both functions.

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 integrated 10 GbE LAN controller 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 – Hardwire to 0b.
6	0b		Reserved.
7	0b		Fast Back-to-Back Capable – Hardwire to 0b.
8	0b	RW1C	Data Parity Reported.
10:9	00b		DEVSEL Timing – Hardwire 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 integrated 10 GbE LAN controller.

8.2.2.5 Revision Register (0x8; RO)

The default revision ID of this device is based on a hard-strapped value.

The value of the rev ID is a logic XOR between the default value and the value in the shared SPI Flash PCIe Device Revision ID *(PCI_REVID)*.

After power on this value can also be overridden using a sideband integrated I/O interface message.

Note that LAN 0 and LAN 1 functions have the same revision ID.

8.2.2.6 Class Code Register (0x9; RO)

The class code is a read-only value that identifies the device functionality according to the value of the *Storage Class* bit in the shared SPI Flash *PCI_CLASS* shared SPI Flash register.

- Class Code = 0x020000 (Ethernet Adapter) if shared SPI Flash->*Storage Class* = 0b
- Class Code = 0x010000 (SCSI Storage device) if shared SPI Flash->*Storage Class* = 1b

In the dummy function the class code equals to 0xFF0000.

8.2.2.7 Cache Line Size Register (0xC; RW)

This field is implemented by PCIe devices as a read/write field for legacy compatibility purposes but has no impact on any PCIe device functionality. The default value is zero.

8.2.2.8 Latency Timer (0xD; RO)

Not used. Hardwire to 0b.

8.2.2.9 Header Type Register (0xE; RO)

This indicates if a device is single- or multi-function. If a single LAN function is the only active one 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 8.5 lists the different options to set the header type field.

Table 8.5. Header Type Settings

Lan 0 Enable	Lan 1 Enable	Cross-Mode Enable	Header Type Expected Value
0	0	X	N/A (no function)
1	0	0	0x00
0	1	0	0x80 (dummy exist)
1	1	X	0x80 (dual function)
1	0	1	0x80 (dummy exist)
0	1	1	0x00

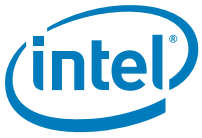
8.2.2.10 Subsystem Vendor ID Register (0x2C; RO)

This value can be loaded automatically from the shared SPI Flash at power up or reset. A value of 0x8086 is the default for this field at power up if the shared SPI Flash does not respond or is not programmed. All functions are initialized to the same value.

8.2.2.11 Subsystem ID Register (0x2E; RO)

This value can be loaded automatically from the shared SPI Flash at power up with a default value of 0x0000.

PCI Function	Default Value	Shared SPI Flash Address
LAN Functions	0x0000	0x0B



8.2.2.12 Cap_Ptr Register (0x34; RO)

The *Capabilities Pointer* field (Cap_Ptr) is an 8-bit field that provides an offset in the integrated 10 GbE LAN controller's PCI configuration space for the location of the first item in the capabilities linked list. The integrated 10 GbE LAN controller 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.

8.2.2.13 Interrupt Line Register (0x3C; RW)

Read/write register programmed by software to indicate which of the system interrupt request lines the integrated 10 GbE LAN controller's interrupt pin is bound to. Refer to the PCI definition for more details. Each PCI function has its own register.

8.2.2.14 Interrupt Pin Register (0x3D; RO)

Read-only register. LAN 0 / LAN 1 — A value of 0x1...0x4 indicates that this function implements a legacy interrupt on INTA#...INTD# respectively. Loaded from the *PCI_CNF* shared SPI Flash word per function. For dummy function the returned value is 0x0 (Function uses no legacy interrupt Message).

Note: If only a single device/function of the integrated 10 GbE LAN controller component is enabled, this value is ignored and the I*nterrupt Pin* field of the enabled device reports INTA# usage.

8.2.2.15 Max_Lat and Min_Gnt (0x3E;RO)

not used. Hardwired to 0b.

For Dummy functions this register is RO - zero.

8.2.2.16 Memory and IO Base Address Registers (0x10...0x27; RW)

Base Address Registers (BARs) are used to map the integrated 10 GbE LAN controller register space of the device functions. The integrated 10 GbE LAN controller has a memory BAR, I/O BAR and MSI-X BAR described in Table 8.6. The BARs location and sizes are listed in the Table 8.6 and Table 8.7. The fields within each BAR are then listed in Table 8.7.

Table 8.6. Integrated 10 GbE LAN Controller 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 *PCI_LBARCTRL* register. Address 0 in the Flash device is mapped to address 256 K in the memory BAR. When the usable Flash size + CSR space are smaller than the memory BAR, 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 shared SPI Flash at word PCIe Control 3 – Offset 0x07.
MSI-X BAR	The MSI-X vectors and Pending Bit Array (PBA) structures are accessed as direct memory mapped offsets from the MSI-X BAR. Software can access Dword entities.

Table 8.7. 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 = Reserved. 10b = 64-bit BAR
Prefetch Memory	3	RO	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 shared SPI Flash 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 size. Initial value of the RW fields is 0x0. The size of the memory BAR is described in Table 8.8.
			Mapping Window: RO bits
			Memory CSR + Flash BAR size depends on *PCI_LBARCTRL.FLSize and PCI_LBARCTRL.CSRSize fields.*: 17:4 for 256 KB, 18:4 for 512 KB, and so on...
			MSI-X space is 16 KB.: 13:4
			I/O space size is 32 bytes.: 4:0

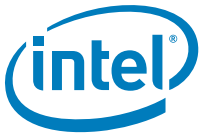
Table 8.8. Usable Flash Size and CSR Mapping Window Size

FLSize	CSRSize	Resulted CSR + Flash BAR Size	Flash Region Size	Usable Flash Space
000b-100b	X	Reserved		
101b	0	2 MB	2 MB	2 MB minus 256 KB
101b	1	4 MB	2 MB	2 MB

8.2.2.17 Expansion ROM Base Address Register (0x30; RW)

This register is used to define the address and size information for boot-time access to the expansion ROM module in the Flash memory. It is enabled by the *PCI_LBARCTRL.EXROM_DIS* register field. This register returns a zero value for functions without an expansion ROM window and for dummy functions.

Field	Bit(s)	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 *PCI_LBARCTRL.EXROM_BAR_SIZE* field



8.2.3 PCI Capabilities

The first entry of the PCI capabilities link list is pointed to by the Cap_Ptr register. Table 8.9 lists the capabilities supported by the integrated 10 GbE LAN controller.

Table 8.9. 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 8.10. PCI Capabilities for Dummy Function

Address	Item	Next Pointer
0x40:47	PCI Power Management	0x50
0xA0:DB	PCIe Capabilities	0x00

8.2.3.1 PCI Power Management Capability

All fields are reset at full power up. All fields except PME_En and PME_Status are reset after exiting from the D3cold state. If AUX power is not supplied, the PME_En and PME_Status fields also reset after exiting from the D3cold state. Refer to the detailed description for registers loaded from the shared SPI Flash 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	

Table 8.11 lists sharing of the power management capability registers among the different PCI functions.

Table 8.11. Sharing the Power Management Capability Registers

Field	Sub-field	Shared?	Replicated?	Comments
Capability ID		x		
Next Pointer			x	
Power Management Capabilities	PME_Support	x		
	D2_Support	x		
	D1_Support	x		
	AUX Current	x		
	DSI	x		
	PME Clock	x		Hardwired to 0b.

Table 8.11. Sharing the Power Management Capability Registers (Continued)

Field	Sub-field	Shared?	Replicated?	Comments
	Version	x		
Power Management Control / Status	PME_Status		x	
	Data_Scale	x		
	Data_Select		x	
	PME_En		x	
	No_Soft_Reset	x		
	PowerState		x	
Data Register			x	

8.2.3.1.1 Capability ID Register (0x40; RO)

This field equals 0x01 indicating the linked list item as being the PCI Power Management registers.

8.2.3.1.2 Next Pointer Register (0x41; RO)

This field provides an offset to the next capability item in the capability list. This field equals for both LAN ports to 0x50 pointing to the MSI capability. In dummy function, it equals to 0xA0 pointing to the PCIe capabilities.

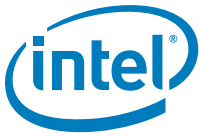
8.2.3.1.3 Power Management Capabilities — PMC Register (0x42; RO)

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 integrated 10 GbE LAN controller does not support the D2 state.
9	0b	RO	D1_Support – The integrated 10 GbE LAN controller does not support the D1 state.
8:6	000b	RO	AUX Current – Required current defined in the Data register.
5	1b	RO	DSI – The integrated 10 GbE LAN controller 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. Hardwire to 0b.
2:0	011b	RO	Version – The integrated 10 GbE LAN controller complies with the PCI PM specification revision 1.2.

8.2.3.1.4 Power Management Control / Status Register — PMCSR (0x44; RW)

This register (listed in the following table) is used to control and monitor power management events in the integrated 10 GbE LAN controller. 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) if the Data_Select field is set to 0, 3, 4, 7, (or 8 for function 0 in multi-function device). Otherwise, it equals 00b.
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 writable only when power management is enabled via the shared SPI Flash.
8	0b at power up	RWS	PME_En. If power management is enabled in the shared SPI Flash, writing a 1b to this register enables wake up.
7:4	0000b	RO	Reserved.
3	1b	RO	No_Soft_Reset. This bit is always set to 1b to indicate that the integrated 10 GbE LAN controller does not perform an internal reset upon transition from D3hot to D0 via software control of the *PowerState* bits. Configuration context is kept as part of the transition from the D3hot to the D0 state, thus a full re-initialization sequence of the configuration space is not needed to return the integrated 10 GbE LAN controller 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.

8.2.3.1.5 PMCSR_BSE Bridge Support Extensions Register (0x46; RO)

This register is not implemented in the integrated 10 GbE LAN controller; values set to 0x00.

8.2.3.1.6 Data Register (0x47; RO)

This optional register is used to report power consumption and heat dissipation. The reported register is controlled by the *Data_Select* field in the PMCSR; the power scale is reported in the *Data_Scale* field in the *PMCSR*. The data for this field is loaded from the shared SPI Flash via the *PCI_PWRDATA* register if power management is enabled in the shared SPI Flash or with a default value of 0x00. The values for the integrated 10 GbE LAN controller's functions are as follows (the relevant column is selected based on the value of the Data_Select field):

Function	D0 (Consume/ Dissipate)	D3 (Consume/ Dissipate)	Common	Data_Scale
Data_Select	(0x0/0x4)	(0x3/0x7)	(0x8)	
0	PCI_PWRDATA.D0_POWER	PCI_PWRDATA.D3_POWER	Function zero of a multi-function device: PCI_PWRDATA.COMM_POWER Single-function device: 0x00	01b
1	PCI_PWRDATA.D0_POWER	PCI_PWRDATA.D3_POWE	0x00	01b

Note: For other Data_Select values the Data register output is reserved (0b).

8.2.3.2 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			

Table 8.12 lists configuration sharing of the MSI Capability registers among the different PCI functions.

Table 8.12. Configuration sharing of the MSI Capability

Field	Sub-field	Shared?	Replicated?	Comments
Capability ID		x		
Next Pointer			x	
Message Control	MSI Enable		x	
	Multiple Messages Capable	x		
	Multiple Message Enable		x	
	64-bit Capable	x		
	MSI per-vector masking	x		
Message Address Low			x	
Message Address High			x	
Message Data			x	
Mask Bits			x	
Pending Bits			x	

8.2.3.2.1 Capability ID Register (0x50; RO)

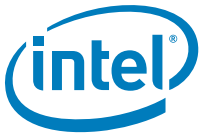
This field equals 0x05 indicating that the linked list item as being the MSI registers.

8.2.3.2.2 Next Pointer Register (0x51; RO)

This field provides an offset to the next capability item in the capability list. Its value of 0x70 and points to MSI-X capability.

8.2.3.2.3 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 integrated 10 GbE LAN controller generates an MSI for interrupt assertion instead of INTx signaling.
3:1	000b	RO	Multiple Messages Capable. The integrated 10 GbE LAN controller indicates a single requested message per function.
6:4	000b	RW	Multiple Message Enable. Since the integrated 10 GbE LAN controller requests a single vector in the *Multiple Message Capable* field, Software is expected to write 000b to this field.
7	1b	RO	64-bit Capable. A value of 1b indicates that the integrated 10 GbE LAN controller is capable of generating 64-bit message addresses.
8	1b[1]	RO	MSI per-vector masking. A value of 0b indicates that the integrated 10 GbE LAN controller is not capable of per-vector masking. A value of 1b indicates that the integrated 10 GbE LAN controller 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 shared SPI Flash.

8.2.3.2.4 Message Address Low Register (0x54; RW)

Written by the system to indicate the lower 32 bits of the address to use for the MSI memory write transaction. The lower two bits always return 0b regardless of the write operation.

8.2.3.2.5 Message Address High Register (0x58; RW)

Written by the system to indicate the upper 32 bits of the address to use for the MSI memory write transaction.

8.2.3.2.6 Message Data Register (0x5C; RW)

Written by the system to indicate the lower 16 bits of the data written in the MSI memory write Dword transaction. The upper 16 bits of the transaction are written as 0b.

8.2.3.2.7 Mask Bits Register (0x60; RW)

The Mask Bits and Pending Bits registers enable software to disable or defer message sending on a per-vector basis. As the integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller is prohibited from sending MSI messages.
31:1	0x0	RO	Reserved

8.2.3.2.8 Pending Bits Register (0x64; RW)

Bits	Default	RW	Description
0	0b	RO	If set, the integrated 10 GbE LAN controller has a pending MSI message.
31:1	0x0	RO	Reserved

8.2.3.3 MSI-X Capability

Note: This capability is not available for dummy functions.

More than one MSI-X capability structure per function is prohibited while a function is permitted to have both an MSI and an MSI-X capability structure.

In contrast to the MSI capability structure, which directly contains all of the control/status information for the function's vectors, the MSI-X capability structure instead points to an MSI-X table structure and an MSI-X Pending Bit Array (PBA) structure, each residing in memory space.

Each structure is mapped by a BAR belonging to the function that begins at 0x10 in the configuration space. A BAR Indicator Register (BIR) indicates which BAR and a Qword-aligned offset indicates where the structure begins relative to the base address associated with the BAR. The BAR is 64-bit, but must map to the memory space. A function is permitted to map both structures with the same BAR or map each structure with a different BAR.

The MSI-X table structure (Section 8.2.3.4) typically contains multiple entries, each consisting of several fields: *Message Address*, *Message Upper Address*, *Message Data*, and *Vector Control*. Each entry is capable of specifying a unique vector.

The PBA structure [MSI-X Table Offset Register (0x74; RW)] contains the function's pending bits, one per table entry, organized as a packed array of bits within Qwords. The last Qword is not necessarily fully populated.

To request service using a given MSI-X table entry, a function performs a Dword memory write transaction using:

- The contents of the *Message Data* field entry for data
- The contents of the *Message Upper Address* field for the upper 32 bits of the address
- The contents of the *Message Address* field entry for the lower 32 bits of the address

A memory read transaction from the address targeted by the MSI-X message produces undefined results.

The MSI-X table and MSI-X PBA are permitted to co-reside within a naturally aligned 4 KB address range, though they must not overlap with each other.

MSI-X table entries and *Pending* bits are each numbered 0 through N-1, where N-1 is indicated by the *Table Size* field in the MSI-X Message Control register. For a given arbitrary MSI-X table entry K, its starting address can be calculated with the formula:

Entry starting address = Table base + K*16

For the associated *Pending* bit K, its address for Qword access and bit number within that Qword can be calculated with the formulas:

Qword address = PBA base + (K div 64)*8

Qword bit# = K mod 64

Software that chooses to read *Pending* bit K with Dword accesses can use these formulas:

Dword address = PBA base + (K div 32)*4

Dword bit# = K mod 32

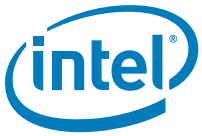


Table 8.13. MSI-X Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x70	Message Control (0x00090)		Next Pointer (0xA0)	Capability ID (0x11)
0x74	Table Offset			
0x78	PBA Offset			

Table 8.14 lists configuration sharing of the MSI-X Capability registers among the different PCI functions.

Table 8.14. Configuration sharing of the MSI-X Capability

Field	Sub-field	Shared?	Replicated?	Comments
Capability ID		x		
Next Pointer			x	
Message Control	Table Size	x		
Function Mask			x	
MSI-X Enable			x	
MSI-X Table Offset	Table BIR		x	
	Table Offset		x	
MSI-X Pending Bit Array	PBA BIR		x	
	PBA Offset		x	
MSI-X Table			x	
MSI-X PBA Structure			x	

8.2.3.3.1 Capability ID Register (0x70; RO)

This field equals 0x11 indicating that the linked list item as being the MSI-X registers.

8.2.3.3.2 Next Pointer Register (0x71; RO)

This field provides an offset to the next capability item in the capability list. Its value of 0xA0 points to PCIe capability.

8.2.3.3.3 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 integrated 10 GbE LAN controller supports up to 64 different interrupt vectors per function. This field is loaded from the PCI_CNF2.MSI_X_PF_N register field.
13:11	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.
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.

8.2.3.3.4 MSI-X Table Offset Register (0x74; RW)

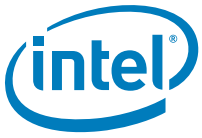
These register fields are listed in the following table.

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.

8.2.3.3.5 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.



8.2.3.4 MSI-X Table Structure

Dword3 — MSIXTVCTRL	Dword2 — MSIXTMSG	Dword1 — MSIXTUADD	Dword0 — MSIXTADD	Entry Number	BAR 3 — Offset
Vector Control	Msg Data	Msg Upper 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.

See Section 8.2.2.5 for details of the MSI-X registers in BAR 3 of the PF.

8.2.3.5 VPD Registers

Note: This capability is not available for dummy functions.

The integrated 10 GbE LAN controller supports access to a VPD structure stored in the shared SPI Flash 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 shared SPI Flash 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			

Table 8.15 lists configuration sharing of the VPD Capability registers among the different PCI functions.

Table 8.15. Configuration sharing of the VPD Capability

Field	Shared?	Replicated?	Comments
Capability ID	x		
Next Pointer	x		
VPD Address/F		x	
VPD Data		x	

8.2.3.5.1 Capability ID Register (0xE0; RO)

This field equals 0x3 indicating the linked list item as being the VPD registers.

8.2.3.5.2 Next Pointer Register (0xE1; RO)

Offset to the next capability item in the capability list. A 0x00 value indicates that it is the last item in the capability-linked list.

8.2.3.5.3 VPD Address Register (0xE2; RW)

Word-aligned byte address of the VPD area in the shared SPI Flash 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 shared SPI Flash 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 shared SPI Flash. The VPD address and data should not be modified before the action is done.

8.2.3.5.4 VPD Data Register (0xE4; RW)

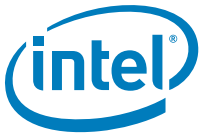
VPD read/write data.

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.

8.2.3.6 PCIe Capability

The integrated 10 GbE LAN controller 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	



Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0xCC	Reserved			
0xD0	Link Status 2		Link Control 2	
0xD4	Reserved			
0xD8	Reserved		Reserved	

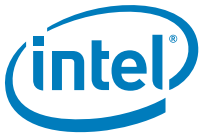
Table 8.15 lists configuration sharing of the PCIe Capability registers among the different PCI functions.

Table 8.16. Configuration Sharing of the PCIe Capability

Field	Sub-field	Shared?	Replicated?	Comments
Capability ID		x		
Next Pointer			x	
PCIe Capabilities		x		
Device Capabilities	Max Payload Size Supported	x		
	Phantom Functions Supported	x		Not supported.
	Extended Tag Field Supported	x		
	Endpoint L0s Acceptable Latency	x		
	Endpoint L1 Acceptable Latency	x		
	Function Level Reset Capability	x		
Device Control	Correctable Error Reporting Enable		x	
	Non-Fatal Error Reporting Enable		x	
	Fatal Error Reporting Enable		x	
	Unsupported Request Reporting Enable		x	
	Enable Relaxed Ordering		x	
	Max Payload Size		x	Use minimum of all configured values. In ARI mode, use value in function 0.
	Extended Tag field Enable		x	
	Auxiliary Power PM Enable		x	Same policy for all PFs (Logical OR of the PFs' bits).
	Enable No Snoop		x	
	Max Read Request Size		x	Use minimum of all configured values.
	Initiate Function Level Reset		x	
Device Status	Correctable Detected		x	
	Non-Fatal Error Detected		x	

Table 8.16. Configuration Sharing of the PCIe Capability (Continued)

Field	Sub-field	Shared?	Replicated?	Comments
	Fatal Error Detected		x	
	Unsupported Request Detected	x		
	Aux Power Detected	x		
	Transactions Pending		x	
Link Capabilities	Supported Link Speeds	x		
	Max Link Width	x		
	Active State Link PM Support	x		
	L0s Exit Latency	x		
	L1 Exit Latency	x		
	Clock Power Management	x		
	Port Number	x		
Link Control	Active State Link PM Control		x	Same policy for all PFs (Logical AND of the PFs' bits). In ARI mode, use value in function 0.
	Read Completion Boundary (RCB)		x	
	Common Clock Configuration		x	Same policy for all PFs (Logical AND of the PFs' bits). In ARI mode, use value in function 0.
	Extended Sync		x	Same policy for all PFs (Logical OR of the PFs' bits).
Link Status	Current Link Speed	x		
	Negotiated Link Width	x		
	Slot Clock Configuration	x		
Device Capabilities 2	Completion Timeout Ranges Supported	x		
	Completion Timeout Disable Supported	x		
	LTR Mechanism Supported	x		
	Extended Fmt Field Supported	x		
	OBFF Supported	x		
Device Control 2	Completion Timeout Value		x	Completion timeout decision per PF or use the largest configured value among PFs.
	Completion Timeout Disable		x	Completion timeout mechanism enabled per PF.
	IDO Request Enable		x	
	IDO Completion Enable		x	
	LTR Mechanism Enable		x	PF0 only. RsvdP on other functions.
	OBFF Enable		x	PF0 only. RsvdP on other functions.
Link Capabilities 2		x		
Link Control 2		x		PF0 only. RsvdP on other functions.
Link Status 2		x		



8.2.3.6.1 Capability ID Register (0xA0; RO)

This field equals 0x10 indicating that the linked list item as being the PCIe Capabilities registers.

8.2.3.6.2 Next Pointer Register (0xA1; RO)

Offset to the next capability item in the capability list. Its value of 0xE0 points to the VPD structure. If VPD is disabled or for a dummy function, a value of 0x00 value indicates that it is the last item in the capability-linked list.

8.2.3.6.3 PCIe Capabilities Register (0xA2; RO)

The PCIe Capabilities register identifies PCIe device type and associated capabilities. This is a read-only register identical to all functions.

Bits	Default	RW	Description
3:0	0010b	RO	Capability Version. Indicates the PCIe capability structure version. The integrated 10 GbE LAN controller supports PCIe version 2 (also loaded from the *PCI_CAPSUP.PCIE_VER* bit in the shared SPI Flash).
7:4	0000b	RO	Device/Port Type. Indicates the type of PCIe functions. This field's default value is based on a hard-strap. 0 - 0000b: Integrated PCIe endpoint 1 - 1001b: Integrated RC endpoint ***Note:*** Should be always 0000b.
8	0b	RO	Slot Implemented. The integrated 10 GbE LAN controller does not implement slot options. Therefore, this field is hard wired to 0b.
13:9	00000b	RO	Interrupt Message Number. The integrated 10 GbE LAN controller does not implement multiple MSI per function. As a result, this field is hard wired to 0x0.
15:14	00b	RO	Reserved.

8.2.3.6.4 Device Capabilities Register (0xA4; RO)

This register identifies the PCIe device specific capabilities. It is a read-only register with the same value for the two LAN functions and for all other functions.

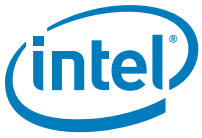
Bits	Rd/Wr	Default	Description
2:0	RO	000b (128 bytes)	Max Payload Size Supported. This field indicates the maximum payload that The integrated 10 GbE LAN controller can support for TLPs.
4:3	RO	00b	Phantom Function Supported. Not supported by the integrated 10 GbE LAN controller.
5	RO	0b	Extended Tag Field Supported. Maximum supported size of the *Tag* field. The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller can withstand due to the transition from L0s state to the L0 state. All functions share the same value loaded from the shared SPI Flash PCIe Init Configuration 1 bits [8:6]. A value of 011b equals 512 ns.
11:9	RO	110b	Endpoint L1 Acceptable Latency. This field indicates the acceptable latency that the integrated 10 GbE LAN controller 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 shared SPI Flash.

Bits	Rd/Wr	Default	Description
12	RO	0b	Attention Button Present. Hard wired in the integrated 10 GbE LAN controller to 0b for all functions.
13	RO	0b	Attention Indicator Present. Hard wired in the integrated 10 GbE LAN controller to 0b for all functions.
14	RO	0b	Power Indicator Present. Hard wired in the integrated 10 GbE LAN controller to 0b for all functions.
15	RO	1b	Role Based Error Reporting. Hard wired in the integrated 10 GbE LAN controller to 1b for all functions.
17:16	RO	000b	Reserved 0b.
25:18	RO	0x00	Slot Power Limit Value. Used in upstream ports only. Hardwired in the integrated 10 GbE LAN controller to 0x00 for all functions.
27:26	RO	00b	Slot Power Limit Scale. Used in upstream ports only. Hardwired in the integrated 10 GbE LAN controller 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.

8.2.3.6.5 Device Control Register (0xA8; RW)

This register controls the PCIe specific parameters. Note that there is a dedicated register per each function.

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 integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller functions. As a receiver, the integrated 10 GbE LAN controller must handle TLPs as large as the set value. As a transmitter, the integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller.
9	RW	0b	Phantom Functions Enable. Not implemented in the integrated 10 GbE LAN controller.
10	RWS	0b	Auxiliary Power PM Enable. When set, enables the integrated 10 GbE LAN controller to draw AUX power independent of PME AUX power. The integrated 10 GbE LAN controller is a multi-function device, therefore allowed to draw AUX power if at least one of the functions has this bit set.
11	RW	0b	Enable No Snoop.No-snoop is not used by the integrated 10 GbE LAN controller.



Bits	RW	Default	Description
14:12	RW	010b	Max Read Request Size. This field sets maximum read request size for the integrated 10 GbE LAN controller as a requester. 000b = 128 bytes. 001b = 256 bytes. 010b = 512 bytes. 011b = 1024 bytes. 100b = 2048 bytes. 101b = 4096 bytes. 110b = Reserved. 111b = Reserved.
15	RW	0b	Initiate FLR – A write of 1b initiates FLR to the function. The value read by software from this bit is always 0b.

8.2.3.6.6 Device Status Register (0xAA; RW1C)

This register provides information about PCIe device specific parameters. Note that there is a dedicated register per each function.

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 integrated 10 GbE LAN controller received an unsupported request. This field is identical in all functions. The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller has ANY transactions pending. (transactions include completions for any outstanding non-posted request for all used traffic classes).
15:6	RO	0x00	Reserved.

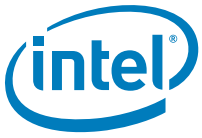
8.2.3.6.7 Link Capabilities Register (0xAC; RO)

This register identifies PCIe link-specific capabilities. This is a read-only register identical to all functions.

Note: A root complex integrated endpoint MUST not implement this register and return a value of zero when accessed.

Bits	RW	Discrete Default	Integrated EP[1] Default	Description
3:0	RO	0001b	0001b	Supported Max Link Speed. This field indicates the supported Link speed(s) of the associated link port. Defined encodings are: 0001b = 2.5 GT/s link speed supported. 0010b = 5 GT/s and 2.5 GT/s link speeds supported. 0011b = 8 GT/s and 5 GT/s and 2.5 GT/s link speeds supported.

Bits	RW	Discrete Default	Integrated EP[1] Default	Description
9:4	RO	0x04 - 17x17 0x08 - 25x25	0x1	Max Link Width. Indicates the maximum link width. The integrated 10 GbE LAN controller supports a x1, x2, x4 and x8-link width. This field is loaded from the PCIe Analog Configuration shared SPI Flash module by interpreting the masked lanes, with a default value of eight lanes for the 25x25 package and four for the 17x17 package. Defined encoding: 000000b = Reserved. 000001b = x1. 000010b = x2. 000100b = x4. 001000b = x8.
11:10	RO	11b	11b	Active State Link PM Support. Indicates the level of the active state of power management supported in the integrated 10 GbE LAN controller. 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 shared SPI Flash Act_Stat_PM_Sup field in the shared SPI Flash PCIe Init Configuration 3 offset 0x3.
14:12	RO	101b (1 µs – 2 µs)	000b	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 shared SPI Flash.
17:15	RO	100b (8 µs — 16 µs)	000b	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 shared SPI Flash.
18	RO	0	0	Clock Power Management.
19	RO	0	0	Surprise Down Error Reporting Capable. Hard wired to 0b.
20	RO	0	0	Data Link Layer Link Active Reporting Capable.
21	RO	0	0	Link Bandwidth Notification Capability. Hard wired to 0b.
22	RO	1b	0	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	0	Reserved.



Bits	RW	Discrete Default	Integrated EP[1] Default	Description
31:24	HwInit	0x0	Strap	Port Number. The PCIe port number for the given PCIe link. This field is set in the link training phase. **For an integrated EP this value is based on a strapping wire.**

1. End Point (EP). By default, the integrated 10 GbE LAN controller is designed with an integrated EP.

8.2.3.6.8 Link Control Register (0xB0; RO)

This register controls PCIe link specific parameters. There is a dedicated register per each function.

Note: A root complex integrated endpoint MUST not implement this register and return a value of zero when accessed.

Bits	RW	Discrete Default	Integrated EP[1] Default	Description
1:0	RW	00b	xxb	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 integrated 10 GbE LAN controller. 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). **For an integrated EP:** R/W scratch pad (no effect on agent).
2	RO	0b	0b	Reserved.
3	RO	0b	0b	Read Completion Boundary. **For an integrated EP:** R/W.
4	RO	0b	0b	Link Disable. Reserved for endpoint devices. Hardwired to 0b.
5	RO	0b	0b	Retrain Clock. Not applicable for endpoint devices. Hardwire to 0b. **For an integrated EP:** Software writes this bit, but it must always return 0 on reads (no effect on agent).
6	RW	0b	xb	Common Clock Configuration. When set, indicates that the integrated 10 GbE LAN controller and the component at the other end of the link are operating with a common reference clock. A value of 0b indicates that they are operating with an asynchronous clock. In ARI mode, the common clock configuration is determined solely by the field in function 0 (even when it is a dummy function) while it is meaningless in the other function(s). **For an integrated EP:** R/W scratch pad (no effect on agent).
7	RW	0b	xb	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. **For an integrated EP:** R/W scratch pad (no effect on agent).
8	RO	0b	0b	Enable Clock Power Management. Not supported - hardwired to 0b. **For an integrated EP:** Same.
9	RW	0b		Reserved. Returns the value that was written.

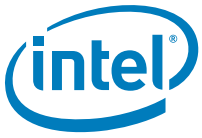
Bits	RW	Discrete Default	Integrated EP[1] Default	Description
10	RO	0b	0b	Link Bandwidth Management Interrupt Enable. Not supported in the integrated 10 GbE LAN controller. Hardwired to 0. **For an integrated EP:** Hardwire to 0.
11	RO	0b	0b	Link Autonomous Bandwidth Interrupt Enable. Not supported in the integrated 10 GbE LAN controller. Hardwired to 0b. **For an integrated EP:** Hardwire to 0b.
15:12	RO	0x0	0x0	Reserved

1. End Point (EP). By default, the integrated 10 GbE LAN controller is designed with an integrated EP.

8.2.3.6.9 Link Status Register (0xB2; RO)

This register provides information about PCIe Link specific parameters. This is a read only register identical to all functions.

Bits	RW	Discrete Default	Integrated EP[1] Default	Description
3:0	RO	X	001b	Current Link Speed. This field indicates the negotiated link speed of the given PCIe link. Defined encodings are: 0001b = 2.5 GT/s PCIe link. 0010b = 5 GT/s PCIe link. 0011b = 8 GT/s PCIe link. All other encodings are reserved. The value in this field is undefined when the Link is not up. **For an integrated EP:** Hardwire to 001b.
9:4	RO	X	001b	Negotiated Link Width. Indicates the negotiated width of the link. Relevant encodings for the integrated 10 GbE LAN controller are: 000001b = x1. 000010b = X2. 000100b = x4. 001000b = x8. The value in this field is undefined when the Link is not up. **For an integrated EP:** Hardwire to 001b.
10	RO	0b	0b	Undefined.
11	RO	0b	0b	Link Training. Indicates that link training is in progress. This field is not applicable and is reserved for endpoint devices, and is hardwired to 0b. **For an integrated EP:** Hardwire to 0b.
12	HwInit	1b	1b	Slot Clock Configuration. When set, indicates that the integrated 10 GbE LAN controller uses the physical reference clock that the platform provides at the connector. This bit must be cleared if the integrated 10 GbE LAN controller uses an independent clock. The *Slot Clock Configuration* bit is loaded from the *Slot_Clock_Cfg* shared SPI Flash bit. **For an integrated EP:** Hardwire to 1b.
13	RO	0b	0b	Data Link Layer Link Active. Not supported in the integrated 10 GbE LAN controller. Hardwire to 0b. **For an integrated EP:** Hardwire to 0b.
14	RO	0b	0b	Link Bandwidth Management Status. Not supported in the integrated 10 GbE LAN controller. Hardwire to 0b. **For an integrated EP:** Hardwire to 0b.



Bits	RW	Discrete Default	Integrated EP[1] Default	Description
15	RO	0b	0b	Link Autonomous Bandwidth Status. This bit is not applicable and is reserved for endpoints. **For an integrated EP:** Hardwire to 0b.

1. End Point (EP). By default, the integrated 10 GbE LAN controller is designed with an integrated EP.

The following registers are supported only if the capability version is two and above.

8.2.3.6.10 Device Capability 2 Register (0xC4; RO)

This register identifies the PCIe device-specific capabilities. It is a read-only register with the same value for both LAN functions.

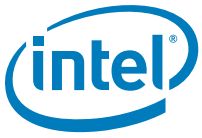
Bits	RW	Default	Description
3:0	RO	1111b	Completion Timeout Ranges Supported. This field indicates the integrated 10 GbE LAN controller's support for the optional completion timeout programmability mechanism. Four time value ranges are defined: • Range A: 50 µs to 10 ms. • Range B: 10 ms to 250 ms. • 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 integrated 10 GbE LAN controller supports. • 0000b = Completion timeout programming not supported. The integrated 10 GbE LAN controller 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.
10:6	RO	0x00	Not supported - hardwired to 0x00.
11	RO	1b	LTR Mechanism Supported – A value of 1b indicates support for the optional Latency Tolerance Reporting (LTR) mechanism. For a multi-function device associated with an upstream port, each function must report the same value for this bit. **Note:** Value loaded from LTR_EN bit in the shared SPI Flash.
17:12	RO	0x0	Reserved.
19:18	RO	00b	OBFF Supported. 00b = OBFF Not Supported. 01b = OBFF supported using message signaling only. 10b = OBFF supported using WAKE# signaling only. 11b = OBFF supported using WAKE# and Message signaling.

Bits	RW	Default	Description
20	RO	0b	Extended Fmt Field Supported - If Set, the function supports the 3-bit definition of the Fmt field. If Clear, the function supports a 2-bit definition of the Fmt field. Not supported by this device.
21	RO	0b	End-End TLP Prefix Supported – Indicates whether End-End TLP Prefix support is offered by a Function. Not supported by this device.
23:22	RsvdP	0b	Max End-End TLP Prefixes – Indicates the maximum number of End-End TLP Prefixes supported by this function. Reserved for this device.
31:24	RO	0x00	Reserved.

8.2.3.6.11 Device Control 2 Register (0xC8; RW)

This register controls the PCIe specific parameters. Note that there is a dedicated register per each function.

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.
7:6	RO	00b	Not supported - hardwired to 00b.



Bits	RW	Default	Description
8	RW	0b	IDO Request Enable – If this bit is set, the function is permitted to set the ID-Based Ordering (*IDO*) bit (Attribute[2]) of requests it initiates. Default value of this bit is 0b.
9	RW	0b	IDO Completion Enable – If this bit is set, the function is permitted to set the ID-Based Ordering (*IDO*) bit (Attribute[2]) of completions it returns. Default value of this bit is 0b.
10	RW / RsvdP	0b	LTR Mechanism Enable – When set to 1b, this bit enables upstream ports to send LTR messages. For a multi-function device, the bit in function 0 is RW, and only function 0 controls the component's link behavior. In all other functions of that device, this bit is RsvdP. If the *LTR_EN* bit in the shared SPI Flash is 0b, then this bit is RO with a value of 0b. Default value of this bit is 0b.
12:11	RO	00b	Reserved.
14:13	RsvdP	00b	OBFF Enable. 00b = Disabled. 01b = Enabled using Message signaling [Variation A]. 10b = Enabled using Message signaling [Variation B]. 11b = Enabled using WAKE# signaling.
15	RsvdP	0b	End-End TLP Prefix Blocking. Not applicable to endpoints.

8.2.3.6.12 Link Capabilities 2 Register (0xCC)

Bits	RW	Default	Description
0	RO	0b	Reserved.
7:1	RO	0x01	Supported Link Speeds Vector – This field indicates the supported Link speed(s) of the associated Port. For each bit, a value of 1b indicates that the corresponding Link speed is supported; otherwise, the Link speed is not supported. Bit definitions are: Bit 1 - 2.5 GT/s. Bit 2 - 5.0 GT/s. Bit 3 - 8.0 GT/s. Bits 7:4 - RsvP. Multi-function devices associated with the same upstream port must report the same value in this field for all Functions. This field is loaded from shared SPI Flash and is reflected in the *PCI_LINKCAP* register.
8	RO	0b	Crosslink Supported – When set to 1b, this bit indicates that the associated port supports cross links. It is recommended that this bit be set in any port that supports cross links even though doing so is only required for ports that also support operating at 8.0 GT/s or higher Link speeds.
31:9	RO	0x00	Reserved.

8.2.3.6.13 Link Control 2 Register (0xD0; RWS)

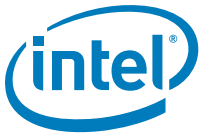
All RW fields in this register affect the device behavior only through function 0. In function 1 these fields are reserved read as zeros.

Note: **For an integrated EP, this register should be RO and return a value of ZERO always.**

Bits	RW	Default	Description
3:0	RWS (func 0) / RsvdP (else)	0001b (func 0) 0000b (else)	Target Link Speed. This field is used to set the target compliance mode speed when software is using the *Enter Compliance* bit to force a link into compliance mode. The encoding is the binary value of the bit in the Supported Link Speeds Vector (in the Link Capabilities 2 register) that corresponds to the desired target link speed. All other encodings are Reserved. For example, 5.0 GT/s corresponds to bit 2 in the Supported Link Speeds Vector, so the encoding for a 5.0 GT/s target Link speed in this field is 0010b. If a value is written to this field that does not correspond to a speed included in the *Supported Link Speeds* field, the result is undefined. The default value of this field is the highest link speed supported by the integrated 10 GbE LAN controller (as reported in the *Supported Link Speeds* field of the Link Capabilities register).
4	RWS (func 0) / RsvdP (else)	0b	Enter Compliance. Software is permitted to force a link to enter compliance mode at the speed indicated in the *Target Link Speed* field by setting this bit to 1b in both components on a link and then initiating a hot reset on the link. The default value of this field following a fundamental reset is 0b.
5	RWS (func 0) / RsvdP (else)	0b	Hardware Autonomous Speed Disable. When set to 1b, this bit disables hardware from changing the link speed for reasons other than attempting to correct unreliable link operation by reducing link speed.
6	RO	0b	Selectable De-Emphasis. This bit is not applicable and reserved for endpoints.
9:7	RWS (func 0) / RsvdP (else)	000b	Transmit Margin. This field controls the value of the non de emphasized voltage level at the Transmitter pins. Encodings: 000b = Normal operating range. 001b = 800-1200 mV for full swing and 400-700 mV for half-swing. 010b = (n-1) — Values must be monotonic with a non-zero slope. The value of n must be greater than 3 and less than 7. At least two of these must be below the normal operating range of n: 200-400 mV for full-swing and 100-200 mV for half-swing. 111b= (n) reserved.
10	RWS (func 0) / RsvdP (else)	0b	Enter Modified Compliance. When this bit is set to 1b, the device transmits modified compliance pattern if the LTSSM enters Polling.Compliance state. The default value of this bit is 0b.
11	RWS (func 0) / RsvdP (else)	0b	Compliance SOS-When set to 1b, the LTSSM is required to send SOS periodically in between the (modified) compliance patterns. This bit is applicable when the Link is operating at 2.5 GT/s or 5 GT/s data rates only. The default value of this bit is 0b.
15:12	RWS (func 0) / RsvdP (else)	0x0	Compliance Preset/De-emphasis. For 8.0 GT/s Data Rate: This field sets the Transmitter Preset in Polling.Compliance state if the entry occurred due to the Enter Compliance bit being 1b. For 5.0 GT/s Data Rate: This field sets the de-emphasis level in Polling.Compliance state if the entry occurred due to the Enter Compliance bit being 1b. When the Link is operating at 2.5 GT/s, the setting of this bit field has no effect. Defined Encodings are: 0001b -3.5 dB. 0000b -6 dB. For a multi-function device associated with an upstream port, the bit field in function 0 is of type RWS, and only function 0 controls the component's link behavior. In all other functions of that device, this bit field is of type RsvdP. This bit field is intended for debug, and compliance testing purposes. The default value of this field is 0000b.

8.2.3.7 Link Status 2 Register (0xD2; RW)

Note: For an integrated EP, this register should be RO and return a value of ZERO always.



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.
1	ROS (Func 0) RsvdZ (Func 1)	0b	Equalization Complete – When set to 1b, this bit indicates that the transmitter equalization procedure has completed.
2	ROS (Func 0) RsvdZ (Func 1)	0b	Equalization Phase 1 Successful. When set to 1b, this bit indicates that phase 1 of the transmitter equalization procedure has successfully completed.
3	ROS (Func 0) RsvdZ (Func 1)	0b	Equalization Phase 2 Successful. When set to 1b, this bit indicates that phase 2 of the transmitter equalization procedure has successfully completed.
4	ROS (Func 0) RsvdZ (Func 1)	0b	Equalization Phase 3 Successful. When set to 1b, this bit indicates that phase 3 of the transmitter equalization procedure has successfully completed.
5	RW1C (Func 0) RsvdZ (Func 1)	0b	Link Equalization Request. This bit is set by hardware to request the link equalization process to be performed on the link. This bit is available only in function zero.
15:6	RsvdZ	0x00	Reserved.

8.2.4 PCIe Extended Configuration Space

PCIe configuration space is located in a flat memory-mapped address space. PCIe extends the configuration space beyond the 256 bytes available for PCI to 4096 bytes. The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller supports the following PCIe extended capabilities:

Table 8.17. Extended Capabilities list

Capability	Offset	Next Header	Section
Advanced Error Reporting Capability	0x100	Any of the below / 0x000	8.2.4.1
Serial Number	0x140	Any of the below / 0x000[1]	8.2.4.2
Alternative RID Interpretation (ARI)	0x150	Any of the below / 0x000[1]	8.2.4.3
IOV support[2]	0x160	Any of the below / 0x000	8.2.4.4

Table 8.17. Extended Capabilities list

Capability	Offset	Next Header	Section
Access Control Services (ACS)[3]	0x1B0	Any of the below / 0x000	8.2.4.5
Latency Tolerance Reporting (LTR)	0x1C0	Any of the below / 0x000	8.2.4.6
Secondary PCIe	0x1D0	0x000	8.2.4.7

1. Depends on shared SPI Flash settings enabling the ARI/IOV structures.
2. In a dummy function, the IOV structure is not exposed.
3. When in Single function mode (function 1 is disabled), the ACS capability is not exposed.

8.2.4.1 Advanced Error Reporting Capability (AER)

The PCIe advanced error reporting capability is an optional extended capability to support advanced error reporting. The tables that follow list the PCIe advanced error reporting extended capability structure for PCIe devices.

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x100	Next Capability offset	Version (0x1)	AER Capability ID (0x0001)	
0x104	Uncorrectable Error Status			
0x108	Uncorrectable Error Mask			
0x10C	Uncorrectable Error Severity			
0x110	Correctable Error Status			
0x114	Correctable Error Mask			
0x118	Advanced Error Capabilities and Control Register			
0x11C... 0x128	Header Log			

Table 8.18 lists configuration sharing of the AER Capability registers among the different PCI functions.

Table 8.18. Configuration Sharing of the AER Capability

Field	Sub-field	Shared?	Replicated?	Comments
Enhanced Capability Header Register	Extended Capability ID	x		
	Capability Version	x		
	Next Capability Offset		x	
Uncorrectable Error Status			x	
Uncorrectable Error Mask			x	
Uncorrectable Error Severity			x	
Correctable Error Status			x	
Correctable Error Mask			x	
Advanced Error Capabilities and Control	First Error Pointer		x	
	ECRC Generation Capable	x		
	ECRC Generation Enable		x	ECRC insertion is per PF.

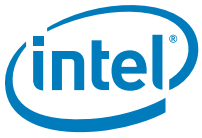


Table 8.18. Configuration Sharing of the AER Capability (Continued)

Field	Sub-field	Shared?	Replicated?	Comments
	ECRC Check Capable	x		
	ECRC Check Enable		x	
Header Log			x	

8.2.4.1.1 Advanced Error Reporting Enhanced Capability Header Register (0x100; RO)

Bit Location	Attribute	Default Value	Description
15:0	RO	0x0001	Extended Capability ID. PCIe extended capability ID indicating advanced error reporting capability.
19:16	RO	0x2	Version Number. PCIe advanced error reporting extended capability version number.
31:20	RO	See description	Next Capability Offset. Next PCIe extended capability offset. See Table 8.17 and Table 8.10 for possible values of the next capability offset.

8.2.4.1.2 Uncorrectable Error Status Register (0x104; RW1CS)

The Uncorrectable Error Status register reports error status of individual uncorrectable error sources on a PCIe device. An individual error status bit that is set to 1b indicates that a particular error occurred; software can clear an error status by writing a 1b to the respective bit. Register is cleared by Global Reset.

Bit Location	Attribute	Default Value	Description
0	RO	0b	Reserved.
3:1	RsvdP	0b	Reserved.
4	RW1CS	0b	Data Link Protocol Error Status.
5	RO	0b	Reserved.
11:6	RsvdP	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. Not supported. Hardwired to 0b.
25:22	RO	0x0	Not Supported.
31:26	RsvdP	0b	Reserved.

8.2.4.1.3 Uncorrectable Error Mask Register (0x108; RWS)

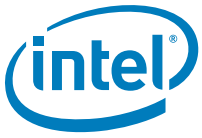
The Uncorrectable Error Mask register controls reporting of individual uncorrectable errors by device to the host bridge via a PCIe error message. A masked error (respective bit set in mask register) is not reported to the host bridge by an individual device. Note that there is a mask bit per bit of the Uncorrectable Error Status register.

Bit Location	Attribute	Default Value	Description
0	RO	0b	Reserved.
3:1	RsvdP	0b	Reserved.
4	RWS	0b	Data Link Protocol Error Mask.
5	RO	0b	Reserved.
11:6	RsvdP	0b	Reserved.
12	RWS	0b	Poisoned TLP Mask.
13	RWS	0b	Flow Control Protocol Error Mask.
14	RWS	0b	Completion Timeout Mask.
15	RWS	0b	Completer Abort Mask.
16	RWS	0b	Unexpected Completion Mask.
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.
25:22	RO	0x0	Not Supported.
31:26	RsvdP	0b	Reserved.

8.2.4.1.4 Uncorrectable Error Severity Register (0x10C; RWS)

The Uncorrectable Error Severity register controls whether an individual uncorrectable error is reported as a fatal error. An uncorrectable error is reported as fatal when the corresponding error bit in the severity register is set. If the bit is cleared, the corresponding error is considered non-fatal.

Bit Location	Attribute	Default Value	Description
0	RO	0b	Reserved.
3:1	RsvdP	0b	Reserved.
4	RWS	1b	Data Link Protocol Error Severity.
5	RO	0b	Reserved.
11:6	RsvdP	000000b	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.



Bit Location	Attribute	Default Value	Description
19	RWS	0b	ECRC Error Severity.
20	RWS	0b	Unsupported Request Error Severity.
21	RO	0b	ACS Violation Severity.
25:22	RO	0x0	Not Supported.
31:26	RsvdP	0x0	Reserved.

8.2.4.1.5 Correctable Error Status Register (0x110; RW1CS)

The Correctable Error Status register reports error status of individual correctable error sources on a PCIe device. When an individual error status bit is set to 1b it indicates that a particular error occurred; software can clear an error status by writing a 1b to the respective bit. Register is cleared by Global Reset.

Bit Location	Attribute	Default Value	Description
0	RW1CS	0b	Receiver Error Status.
5:1	RsvdZ	0b	Reserved.
6	RW1CS	0b	Bad TLP Status.
7	RW1CS	0b	Bad DLLP Status.
8	RW1CS	0b	REPLAY_NUM Rollover Status.
11:9	RsvdZ	0b	Reserved.
12	RW1CS	0b	Replay Timer Timeout Status.
13	RW1CS	0b	Advisory Non-Fatal Error Status.
15:14	RO	0b	Reserved.

8.2.4.1.6 Correctable Error Mask Register (0x114; RWS)

The Correctable Error Mask register controls reporting of individual correctable errors by device to the host bridge via a PCIe error message. A masked error (respective bit set in mask register) is not reported to the host bridge by an individual device. There is a mask bit per bit in the Correctable Error Status register.

Bit Location	Attribute	Default Value	Description
0	RWS	0b	Receiver Error Mask.
5:1	RsvdP	0b	Reserved.
6	RWS	0b	Bad TLP Mask.
7	RWS	0b	Bad DLLP Mask.
8	RWS	0b	REPLAY_NUM Rollover Mask.
11:9	RsvdP	0b	Reserved.
12	RWS	0b	Replay Timer Timeout Mask.
13	RWS	1b	Advisory Non-Fatal Error Mask. This bit is set by default to enable compatibility with software that does not comprehend Role-Based Error Reporting.
15:14	RO	0b	Reserved.

8.2.4.1.7 Advanced Error Capabilities and Control Register (0x118; RO)

Bit Location	Attribute	Default Value	Description
4:0	ROS	0b	Vector pointing to the first recorded error in the Uncorrectable Error Status register. This is a read-only field that identifies the bit position of the first uncorrectable error reported in the Uncorrectable Error Status register.
5	RO	1b	ECRC Generation Capable. If set, this bit indicates that the function is capable of generating ECRC. This bit is loaded from shared SPI Flash. It is reflected in the PCI_CAPSUP register.
6	RWS	0b	ECRC Generation Enable. When set, ECRC generation is enabled.
7	RO	1b	ECRC Check Capable. If set, this bit indicates that the function is capable of checking ECRC. This bit is loaded from shared SPI Flash. It is reflected in the PCI_CAPSUP register.
8	RWS	0b	ECRC Check Enable.When set Set, ECRC checking is enabled.
9	RO	0b	Multiple Header Recording Capable. Not Supported. hardwired to 0b.
10	RO	0b	Multiple Header Recording Enable. Not Supported. hardwired to 0b.
11	RsvdP	0b	TLP Prefix Log Present. Not supported. hardwired to 0b.
15:12	RsvdP	0x0	Reserved.

8.2.4.1.8 Header Log Register (0x11C:128; RO)

The header log register captures the header for the transaction that generated an error. This register is 16 bytes.

Bit Location	Attribute	Default Value	Description
127:0	ROS	0b	Header of the packet in error (TLP or DLLP).

8.2.4.2 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.

The integrated 10 GbE LAN controller implements this capability on all the functions and returns the same value in both.

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x140	Next Capability Offset	Version (0x1)	Serial ID Capability ID (0x0003)	
0x144	Serial Number Register (Lower Dword)			
0x148	Serial Number Register (Upper Dword)			

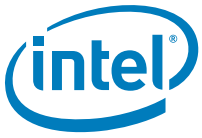


Table 8.19 lists configuration sharing of the Serial ID Capability registers among the different PCI functions.

Table 8.19. Configuration Sharing of the Serial Number Capability

Field	Sub-field	Shared?	Replicated?	Comments
Enhanced Capability Header Register	Extended Capability ID	x		
	Capability Version	x		
	Next Capability Offset		x	
Serial Number Registers		x		

8.2.4.2.1 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	See description	RO	Next Capability Offset. This field contains the offset to the next PCIe capability structure or 0x000 if no other items exist in the linked list of capabilities. See Table 8.17.and Table 8.10 for possible values of the next capability offset.

8.2.4.2.2 Serial Number Registers (0x144:0x148; RO)

The Serial Number register is a 64-bit field that contains the IEEE defined 64-bit Extended Unique Identifier (EUI-64*). The register at offset 0x144 holds the lower 32 bits and the register at offset 0x148 holds the higher 32 bits. The following figure details the allocation of register fields in the Serial Number register. 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 shared SPI Flash (not the same field that is loaded from shared SPI Flash into the *RAL* and *RAH* registers). It is reflected in the *PCI_SERL* and *PCI_SERH* registers. The default value in case of no shared SPI Flash is 0x0.

Note: The official document that defines EUI-64* is: http://standards.ieee.org/regauth/oui/tutorials/EUI64.html

8.2.4.3 Alternate Routing ID Interpretation (ARI) Capability Structure

In order to allow more than eight functions per endpoint without requesting an internal switch, as is usually needed in virtualization scenarios, the PCI-SIG defines a new capability that allows a different interpretation of the *Bus*, *Device*, and *Function* fields. The capability is exposed when the PCI_CAPSUP.ARI_EN bit is set from shared SPI Flash.

The ARI capability structure is as follows:

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x150	Next Capability Offset	Version (0x1)	ARI Capability ID (0x000E)	
0x154	ARI Control Register		ARI Capabilities	

Table 8.20 lists configuration sharing of the ARI Capability registers among the different PCI functions.

Table 8.20. Configuration sharing of the ARI Capability

Field	Sub-field	Shared?	Replicated?	Comments
Enhanced Capability Header Register	Extended Capability ID	x		

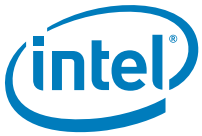


Table 8.20. Configuration sharing of the ARI Capability

Field	Sub-field	Shared?	Replicated?	Comments
	Capability Version	x		
	Next Capability Offset		x	
ARI capability Register	Next Function Pointer		x	

8.2.4.3.1 PCIe ARI Header Register (0x150; RO)

Field	Bit(s)	Initial Value	Access	Description
ID	15:0	0x000E	RO	PCIe Extended Capability ID. PCIe extended capability ID for the alternative RID interpretation.
Version	19:16	1b	RO	Capability Version. This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 0x1 for this version of the specification.
Next Capability Offset	31:20	See description	RO	Next Capability Offset.This field contains the offset to the next PCIe extended capability structure. See Table 8.17. and Table 8.10 for possible values of the next capability offset.

8.2.4.3.2 PCIe ARI Capabilities and Control Register (0x154; RO)

Field	Bit(s)	Initial Value	Access	Description
Reserved	0	0b	RO	Not supported in the integrated 10 GbE LAN controller.
Reserved	1	0b	RO	Not supported in the integrated 10 GbE LAN controller.
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.
Reserved	16	0b	RO	Not supported in the integrated 10 GbE LAN controller.
Reserved	17	0b	RO	Not supported in the integrated 10 GbE LAN controller.
Reserved	19:18	00b	RO	Reserved.
Reserved	22:20	0b	RO	Not supported in the integrated 10 GbE LAN controller.
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.

8.2.4.4 IOV Capability Structure

Note: This capability structure is not exposed in a dummy function.

This is a structure used to support the SR-IOV capabilities reporting and control. The capability is exposed when the *PCI_CAPSUP.IOV_EN* bit is set from shared SPI Flash and the *PCI_CNF2.NUM_VFS* is non-zero.

The following tables show the implementation of this structure in the integrated 10 GbE LAN controller.

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x160	Next Capability Offset (0x0)	Version (0x1)	IOV Capability ID (0x0010)	
0x164	SR IOV Capabilities			
0x168	SR IOV Status		SR IOV Control	
0x16C	Total VFs (RO)		Initial VF (RO)	
0x170	Reserved	Function Dependency Link (RO)	Num VF (RW)	
0x174	VF Stride (RO)		First VF Offset (RO)	
0x178	VF Device ID		Reserved	
0x17C	Supported Page Size (0x553)			
0x180	System Page Size (RW)			
0x184	VF BAR0 — Low (RW)			
0x188	VF BAR0 — High (RW)			
0x18C	VF BAR2 (RO)			
0x190	VF BAR3- Low (RW) (64 bit)			
0x194	VF BAR3 High (RW) (64 bit)			
0x198	VF BRA5 (RO)			
0x19C	VF Migration State Array Offset (RO)			

Table 8.21 lists configuration sharing of the SR-IOV Capability registers among the different PCI functions.

Table 8.21. Configuration Sharing of the SR-IOV Capability

Field	Sub-field	Shared?	Replicated?	Comments
Enhanced Capability Header Register	Extended Capability ID	x		
	Capability Version	x		
	Next Capability Offset		x	
SR-IOV Capabilities	VF Migration Capable	x		Not supported.
	ARI Capable Hierarchy Preserved		x	PF0 only. RO zero in all other functions.
	VF Migration Interrupt Message Number			Not supported.
SR-IOV Control	VF Enable		x	
	Memory Space Enable		x	
	ARI Capable Hierarchy	x		PF0 only. RO zero in all other functions.
InitialVFs			x	
TotalVFs			x	
Num VFs			x	
Function Dependency Link			x	Each PF indicates its PF number here.

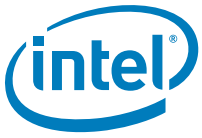


Table 8.21. Configuration Sharing of the SR-IOV Capability (Continued)

Field	Sub-field	Shared?	Replicated?	Comments
First VF Offset			x	
VF Stride			x	
VF Device ID			x	
Supported Page Size		x		
System Page Size			x	
VF BARs			x	

8.2.4.4.1 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 offset	31:20	See description	RO	Next Capability Offset. This field contains the offset to the next PCIe extended capability structure or 0x000 if no other items exist in the linked list of capabilities. See Table 8.17. and Table 8.10 for possible values of the next capability offset.

8.2.4.4.2 PCIe SR-IOV Capabilities Register (0x164; RO)

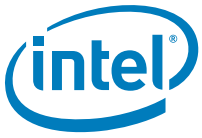
Field	Bit(s)	Initial Value	Access	Description
Reserved	0	0b	RO	Not supported in the integrated 10 GbE LAN controller.
ARICHP	1	1b/0b[1]	RO	ARI Capable Hierarchy Preserved - If set, the *ARI Capable Hierarchy* bit is preserved across certain power state transitions.
Reserved	20:2	0x0	RO	Reserved.
Reserved	31:21	0x0	RO	Not supported in the integrated 10 GbE LAN controller.

1. Set on first function where SR-IOV is enabled (see *PCI_CAPSUP.IOV_EN* bit) and Read Only Zero in the other PF.

8.2.4.4.3 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 integrated 10 GbE LAN controller.
Reserved	2	0b	RO	Not supported in the integrated 10 GbE LAN controller.
VF MSE	3	0b	RW	Memory Space Enable for Virtual Functions. VF MSE controls memory space enable for all VFs associated with this PF as with the *Memory Space Enable* bit in a functions PCI command register. The default value for this bit is 0b. When VF Enable is 1, virtual function memory space access is permitted only when VF MSE is Set. VFs must follow the same error reporting rules as defined in the base specification if an attempt is made to access a virtual functions memory space when VF Enable is 1 and VF MSE is zero. Implementation Note: Virtual functions memory space cannot be accessed when VF Enable is zero. Thus, VF MSE is don't care when VF Enable is zero, however, software might choose to set VF MSE after programming the VF BARn registers, prior to setting VF Enable to 1b.
ARI Capable Hierarchy	4	0b	RW (first function where SR-IOV is enabled) ROS (other function)[1]	ARI Capable Hierarchy. The integrated 10 GbE LAN controller 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 8.2.4.4.2) or *No_Soft_Reset* is set, a power state transition of this PF from D3hot to D0 does not affect the value of this bit.
Reserved	15:5	0x0	RO	Reserved.
Reserved	16	0b	RO	Not implemented in the integrated 10 GbE LAN controller.
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.



8.2.4.4.4 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 integrated 10 GbE LAN controller 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 shared SPI Flash. Reflected in *PCI_CNF2.TOTAL_VFS*.

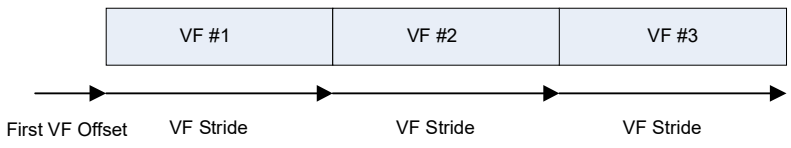
8.2.4.4.5 PCIe SR-IOV Num VFs Register (0x170; RW)

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 integrated 10 GbE LAN controller 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.

8.2.4.4.6 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	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.



8.2.4.4.7 PCIe SR-IOV VF Device ID Register (0x178; RO)

Field	Bit(s)	Initial Value	Access	Description
DEVID	31:16	See Section 8.2.2.2	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 shared SPI Flash.
Reserved	15:0	0x0	RO	Reserved.

8.2.4.4.8 PCIe SR-IOV Supported Page Size Register (0x17C; RO)

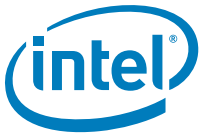
Field	Bit(s)	Initial Value	Access	Description
Supported page Size	31:0	0x553	RO	For PFs that supports the stride-based BAR mechanism, this field defines the supported page sizes. This PF supports a page size of 2^(n+12) if bit n is set. For example, if bit 0 is Set, the Endpoint (EP) supports 4KB page 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.

8.2.4.4.9 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.

8.2.4.4.10 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 shared SPI Flash. It is reflected in the *PCI_VFSUP* register.
Prefetch Mem	3	0b*	RO	0b = Non-prefetchable space. 1b = Prefetchable space. This bit is loaded from the shared SPI Flash. It is reflected in the *PCI_VFSUP* register.
Memory Address Space	31:4	0x0	RW	Which bits are RW bits and which are RO to 0x0 depend on the memory mapping window size. The size is a maximum between 16 KB and the page size.



8.2.4.4.11 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.

8.2.4.4.12 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.

8.2.4.4.13 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 shared SPI Flash. It is reflected in the *PCI_VFSUP* register.
Prefetch Mem	3	0b*	RO	0b = Non-prefetchable space. 1b = Prefetchable space. This bit is loaded from theshared SPI Flash. It is reflected in the *PCI_VFSUP* register.
Memory Address Space	31:4	0x0	RW	Which bits are RW bits and which are RO to 0x0 depend on the memory mapping window size. The size is a maximum between 16 KB and page size.

8.2.4.4.14 PCIe SR-IOV BAR 3 — High Register (0x194; RW)

Field	Bit(s)	Initial Value	Access	Description
BAR4 — MSB	31:0	0x0	RW	MSB part of BAR3.

8.2.4.4.15 PCIe SR-IOV BAR 5 Register (0x198; RO)

Field	Bit(s)	Initial Value	Access	Description
BAR2	31:0	0x0	RO	This BAR is not used.

8.2.4.4.16 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 integrated 10 GbE LAN controller.
Reserved	31:3	0x0	RO	Not implemented in the integrated 10 GbE LAN controller.

8.2.4.5 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 and is not exposed in Single function mode (when function 1 is disabled). The capability is exposed when the PCI_CAPSUP.ACS_EN bit is set from shared SPI Flash.

The ACS capability structure is shared and exposed to all PFs.

The following table lists the PCIe ACS extended capability structure for PCIe devices.

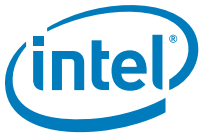
Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x1B0	Next Capability Offset	Version (0x1)	ACS Capability ID (0x0D)	
0x1B4	ACS Control Register (0x0)		ACS Capability Register (0x0)	

8.2.4.5.1 ACS CAP ID (0x1B0; 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	See description	Next Capability Pointer. See Table 8.17. and Table 8.10 for possible values of the next capability offset.

8.2.4.5.2 ACS Control and Capabilities (0x1B4; RO)

Bit Location	Attribute	Default Value	Description
0	RO	0b	ACS Source Validation (V). Hardwired to zero, not supported in the integrated 10 GbE LAN controller.
1	RO	0b	ACS Translation Blocking (B). Hardwired to zero, not supported in the integrated 10 GbE LAN controller.
2	RO	0b	ACS P2P Request Redirect (R). Hardwired to zero, not supported in the integrated 10 GbE LAN controller.
3	RO	0b	ACS P2P Completion Redirect (C). Hardwired to zero, not supported in the integrated 10 GbE LAN controller.
4	RO	0b	ACS Upstream Forwarding (U). Hardwired to zero, not supported in the integrated 10 GbE LAN controller.
5	RO	0b	ACS P2P Egress Control (E). Hardwired to zero, not supported in the integrated 10 GbE LAN controller.
6	RO	0b	ACS Direct Translated P2P (T). Hardwired to zero, not supported in the integrated 10 GbE LAN controller.
7	Rsrv	0b	Reserved.
15:8	RO	0x0	Egress Control Vector Size. Hardwired to zero, not supported in the integrated 10 GbE LAN controller.



Bit Location	Attribute	Default Value	Description
16	RO	0b	ACS Source Validation Enable (V). Hardwired to zero, not supported in the integrated 10 GbE LAN controller.
17	RO	0b	ACS Translation Blocking Enable (B). Hardwired to zero, not supported in the integrated 10 GbE LAN controller.
18	RO	0b	ACS P2P Request Redirect Enable (R). Hardwired to zero, not supported in the integrated 10 GbE LAN controller.
19	RO	0b	ACS P2P Completion Redirect Enable (C). Hardwired to zero, not supported in the integrated 10 GbE LAN controller.
20	RO	0b	ACS Upstream Forwarding Enable (U). Hardwired to zero, not supported in the integrated 10 GbE LAN controller.
21	RO	0b	ACS P2P Egress Control Enable (E). Hardwired to zero, not supported in the integrated 10 GbE LAN controller.
22	RO	0b	ACS Direct Translated P2P Enable (T). Hardwired to zero, not supported in the integrated 10 GbE LAN controller.
31:23	Rsrv	0b	Reserved.

8.2.4.6 Latency Tolerance Reporting (LTR) Capability Structure

The LTR capability is an optional extended capability that enables software to provide platform latency information to devices with upstream ports (endpoints and switches). This capability structure is required if the device supports LTR. The capability is exposed when the *PCI_CAPSUP.LTR_EN* bit is set from shared SPI Flash.

The LTR capability structure is implemented only in function 0 even when function 0 is a dummy function and controls the component's link behavior on behalf of all the functions of the device.

The following table lists the PCIe LTR extended capability structure for PCIe devices.

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x1C0	PCIe Extended Capability Header			
0x1C4	Max No-Snoop Latency Register		Max Snoop Latency Register	

8.2.4.6.1 LTR Extended Capability Header (0x1C0; RO)

Bit Location	Attribute	Default Value	Description
15:0	RO	0x18	PCIe Extended Capability ID. PCIe extended capability ID indicating LTR capability.
19:16	RO	0x1	Capability Version. PCIe LTR extended capability version number.
31:20	RO	See Description	Next Capability Offset. See Table 8.17. and Table 8.10 for possible values of the next capability offset.

8.2.4.6.2 Max Snoop Latency Register (0x1C4; RW)

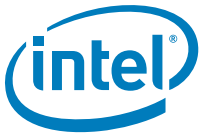
Bit Location	Attribute	Default Value	Description
9:0	RW	0x0	Maximum Snoop Latency Value. Along with the *Max Snoop Latency Scale* field, this register specifies the maximum snoop latency that a device is permitted to request. Software should set this to the platform's maximum supported latency or less. Field is also an indicator of the platform maximum latency, should an endpoint send up LTR latency values with the requirement bit not set.
12:10	RW	0x0	Max Snoop Latency Scale. This field provides a scale for the value contained within the *Maximum Snoop Latency Value* field. Encoding: 000b = Value times 1 ns. 001b = Value times 32 ns. 010b = Value times 1,024 ns. 011b = Value times 32,768 ns. 100b = Value times 1,048,576 ns. 101b = Value times 33,554,432 ns. 110b - 111b = Not permitted.
15:13	RW	0x0	Reserved.

8.2.4.6.3 Max No-Snoop Latency Register (0x1C6; RW)

Bit Location	Attribute	Default Value	Description
9:0	RW	0x0	Maximum No-Snoop Latency Value. Along with the *Max No-Snoop Latency Scale* field, this register specifies the maximum no-snoop latency that a device is permitted to request. Software should set this to the platform's maximum supported latency or less. Field is also an indicator of the platforms maximum latency, should an endpoint send up LTR Latency Values with the Requirement bit not set.
12:10	RW	0x0	Max No-Snoop Latency Scale. This field provides a scale for the value contained within the *Maximum No-Snoop Latency Value* field. Encoding: 000b = Value times 1 ns. 001b = Value times 32 ns. 010b = Value times 1,024 ns. 011b = Value times 32,768 ns. 100b = Value times 1,048,576 ns. 101b = Value times 33,554,432 ns. 110b-111 = Not permitted.
15:13	RW	0x0	Reserved.

8.2.4.7 Secondary PCIe Extended Capability

The secondary PCIe extended capability structure is required for all ports and RCRBs that support a Link speed of 8.0 GT/s or higher. For multi-function upstream ports, this capability must be implemented in function 0 and must not be implemented in other functions. The capability is exposed when the PCI_CAPSUP.SEC_EN bit is set from shared SPI Flash.



The following table lists the secondary PCIe extended capability structure for PCIe devices.

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x1D0	PCIe Extended Capability Header			
0x1D4	Link Control 3 Register			
0x1D8	Lane Error Status Register			
0x1DC	Equalization Control Register (Sized by Maximum Link Width)			
0x1E0	Equalization Control Register (Sized by Maximum Link Width)			

8.2.4.7.1 Secondary PCIe Extended Capability Header (0x1D0)

Bit Location	Attribute	Default Value	Description
15:0	RO	0x19	PCIe Extended Capability ID. This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. PCIe extended capability ID for the secondary PCIe extended capability is 0x0019.
19:16	RO	0x1	Capability Version. This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 0x1 for this version of the specification.
31:20	RO	See Description	Next Capability Offset. See Table 8.17. and Table 8.10 for possible values of the next capability offset.

8.2.4.7.2 Link Control 3 Register (0x1D4)

Bit Location	Attribute	Default Value	Description
0	RW/RsvdP	0b	Perform Equalization. When this bit is 1b and a 1b is written to the Link Retrain register with *Target Link Speed* set to 8.0 GT/s, the downstream port must perform transmitter equalization. This bit is RW for upstream ports when crosslink supported is 1b. This bit is not applicable and is RsvdP for upstream ports when the crosslink supported bit is 0b. The default value is 0b.
1	RW/RsvdP	0b	Link Equalization Request Interrupt Enable. When set, this bit enables the generation of interrupt to indicate that the link equalization Request bit has been set. This bit is RW for upstream ports when crosslink supported is 1b. This bit is not applicable and is RsvdP for upstream ports when the crosslink supported bit is 0b. The default value for this bit is 0b.
31:2	RsvdZ	0x00	Reserved

8.2.4.7.3 Lane Error Status Register (0x1D8)

The Lane Error Status register consists of a 32-bit vector, where each bit indicates if the corresponding PCI Express Lane detected an error.

Bit Location	Attribute	Default Value	Description
7:0	RW1CS	0x00	Lane Error Status Bits. Each bit indicates if the corresponding lane detected a lane-based error. A value of 1b indicates that a lane based-error was detected on the corresponding lane number. The default value of this field is 0b. This field is intended for debug purposes only.
31:8	RsvdZ	0x00	Reserved.

8.2.4.7.4 Lane Equalization Control Register (0x1DC: 0x1E3)

The Equalization Control register consists of control fields required for per Lane equalization and number of entries in this register are sized by Maximum Link Width.

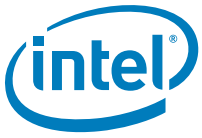
15	0
Lane (0) Equalization Control Register	
Lane (1) Equalization Control Register	
...	
Lane (Maximum Link Width - 1) Equalization Control Register	

Table 8.22. Lane [(Maximum Link Width – 1):0] Equalization Control Register

Bit Location	Attribute	Default Value	Description
3:0	RsvdP	0000b	Downstream Port Transmitter Preset. For an upstream port if crosslink supported is 0b, this field is RsvdP.
6:4	RsvdP	000b	Downstream Port Receiver Preset Hint. For an upstream port if crosslink supported is 0b, this field is RsvdP.
11:8	RO	1111b	Upstream Port Transmitter Preset. Field contains the transmit preset value sent or received during link equalization. Since crosslink is not supported, the field is intended for debug and diagnostics. It contains the value captured from the associated lane during link equalization. Field is RO. The default value is 1111b.
14:12	HwInit/RO	111b	Upstream Port Receiver Preset Hint. This field contains the receiver preset hint value sent or received during link equalization. Field usage varies as follows: Since crosslink is not supported, this field is intended for debug and diagnostics. It contains the value captured from the associated lane during link equalization. Field is RO. The default value is 111b.
15	Rsvd	0b	Reserved.

8.2.5 Driver Forward Compatibility Register (0x94; RO)

This register is an advisory register that returns a fixed value indicating the type of software device driver class needed for the integrated 10 GbE LAN controller.



Bit(s)	R/W	Initial Value	Description
15:0	RO	0xFFFF	Pointer to admin. queues. A value of 0xFFFF indicates that the integrated 10 GbE LAN controller does not support admin. queues.
19:16	RO	0x0	Status. Not implemented in the integrated 10 GbE LAN controller.
31:20	RO	0xA0A	Signature indicating a driver class register.

8.2.6 CSR Access Via Configuration Address Space

The registers described in this section are not part of the PCIe standard configuration and can be used to access the CSR space before memory BARs are allocated. When this mechanism is used, there is no need to expose an I/O BAR for pre boot operation.

Note: These registers are not available for dummy functions.

8.2.6.1 IOADDR Register (0x98; R/W)

This is a read/write register. Each function has its own IOADDR register. Functionality is the same in all functions. Register is cleared at power-up (Global Reset) 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 I/O 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 integrated 10 GbE LAN controller. Otherwise, accesses to the IODATA register are don't-cares (writes are discarded silently, reads return arbitrary results).

1. In the event that the *PCI_CAPSUP.CSR_CONF_EN* bit is cleared, accesses to the IOADDR register via the configuration address space is ignored and has no effect on the register and the CSRs referenced by the IOADDR register.

8.2.6.2 IODATA Register (0x9C; R/W)

This is a read/write register. Each function has its own IODATA register. Functionality is the same in all functions. Register is cleared at power-up (Global Reset) 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.

8.3 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 lists the behavior of each register in the VF configuration space.

Table 8.23. VF PCIe Configuration Space

Section	Offset	Name	VF Behavior	Notes
PCI Mandatory Registers	0	Vendor ID	RO — 0xFFFF	
	2	Device ID	RO — 0xFFFF	
	4	Command	Per VF	See Section 8.3.1.1.
	6	Status	Per VF	See Section 8.3.1.2.
	8	RevisionID	RO as PF	
	9	Class Code	RO as PF	
	C	Cache Line Size	RO — 0x0	
	D	Latency Timer	RO — 0x0	
	E	Header Type	RO — 0x0	
	F	Reserved	RO — 0x0	
	10 — 27	BARs	RO — 0x0	Emulated by VMM.
	28	CardBus CIS	RO — 0x0	Not used.
	2C	Sub Vendor ID	RO as PF	
	2E	Sub System	RO as PF	
	30	Expansion ROM	RO — 0x0	Emulated by VMM.
	34	Cap Pointer	RO — 0x70	Next = MSI-X capability.
	3C	Int Line	RO — 0x0	
	3D	Int Pin	RO — 0x0	
	3E	Max Lat/Min Gnt	RO — 0x0	
MSI-X Capability	70	MSI-X Header	RO — 0xA011	Next = PCIe capability.
	72	MSI-x Message Control	per VF	See Section 8.3.2.1.
	74	MSI-X table Address	RO — as PF	See Section 8.3.2.1.2.
	78	MSI-X PBA Address	RO	See Section 8.3.2.1.3.
PCIe Capability	A0	PCIe Header	RO — 0x0010	Next = Last capability.
	A2	PCIe Capabilities	RO — as PF	
	A4	PCIe Dev Cap	RO — as PF	
	A8	PCIe Dev Ctrl	RW	As PF apart from FLR — See Table 8.3.2.2.1.
	AA	PCIe Dev Status	per VF	See Table 8.3.2.2.2.
	AC	PCIe Link Cap	RO — as PF	
	B0	PCIe Link Ctrl	RO — 0x0	
	B2	PCIe Link Status	RO — 0x0	
	C4	PCIe Dev Cap 2	RO — as PF	
	C8	PCIe Dev Ctrl 2	RO — 0x0	
	D0	PCIe Link Ctrl 2	RO — 0x0	
	D2	PCIe Link Status 2	RO — 0x0	

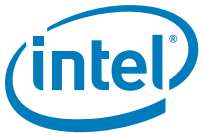


Table 8.23. VF PCIe Configuration Space (Continued)

Section	Offset	Name	VF Behavior	Notes
AER Capability	100	AER — Header	RO — 0x15010001	Next = ARI structure.
	104	AER — Uncorr Status	per VF	See Section 8.3.3.1.1.
	108	AER — Uncorr Mask	RO — 0x0	
	10C	AER — Uncorr Severity	RO — 0x0	
	110	AER — Corr Status	Per VF	See Section 8.3.3.1.2.
	114	AER — Corr Mask	RO — 0x0	
	118	AER — Cap/Ctrl	RO as PF	
	11C — 128	AER — Error Log	Shared two logs for all VFs	Same structure as in PF. In case of overflow, the header log is filled with ones.
ARI Capability	150	ARI — Header	0x1A01000E	
	154	ARI — Cap/Ctrl	RO — 0X0	
ACS Capability	1B0	ACS - Header	RO - 0x0001000D	Last extended capability.
	1B4	ACS - Capability	RO - 0x00000000	

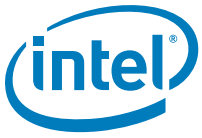
8.3.1 Mandatory Configuration Space

8.3.1.1 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.

8.3.1.2 VF Status Register (0x6; RW)

Bits	Initial Value	Rd/Wr	Description
2:0	0x0	RO	RSV: Reserved.
3	0b	RO	IS: Interrupt Status. Hardwired to 0b.
4	1b	RO	NC: New Capabilities. Indicates that the integrated 10 GbE LAN controller VFs implement extended capabilities. The integrated 10 GbE LAN controller VFs implement a capabilities list, to indicate that it supports MSI-X and PCIe extensions.
5	0b	RO	66E: 66 MHz Capable. Hardwired to 0b.
6	0b	RO	RSV: Reserved.
7	0b	RO	FB2BC: Fast Back-to-Back Capable. Hardwired to 0b.
8	0b	RW1C	MPERR: Data Parity Reported.
10:9	00b	RO	DEVSEL: DEVSEL Timing. Hardwired 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.



8.3.2 PCI Capabilities

8.3.2.1 MSI-X Capability

The only registers with a different layout than the PF for MSI-X, is the control register.

Note: The message address and data registers in enhanced mode use the first MSI-X entry of each VF in the regular MSI-X table.

See Section 8.3.5.1 for details of the MSI-X registers in BAR 3 of the VF.

8.3.2.1.1 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 shared SPI Flash. This field is loaded from the *PCI_CNF2.MSI_X_VF_N* register field.

8.3.2.1.2 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	0x3	RO	Table BIR. Indicates which one of a function's BARs, beginning at 0x10 in the configuration space, is used to map the function's MSI-X table into the memory space. BIR values: 0...5 correspond to BARs 0x10...0x24, respectively. BIR equals 3 to indicate BAR 3 and 4 are used for MSI-X vectors.

8.3.2.1.3 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	0x3	RO	PBA BIR. Indicates which one of a function's BARs, located beginning at 0x10 in configuration space, is used to map the function's MSI-X PBA into memory space. BIR values: 0...5 correspond to BARs 0x10...0x24, respectively. BIR equals 3 to indicate BAR 3 and 4 are used for MSI-X PBA

8.3.2.2 PCIe Capability Registers

The device control and device status registers have some fields which are specific per VF.

8.3.2.2.1 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.
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 integrated 10 GbE LAN controller.
9	RO	0b	Not implemented in the integrated 10 GbE LAN controller.
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.

8.3.2.2.2 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 integrated 10 GbE LAN controller received an unsupported request. This field is identical in all functions. The integrated 10 GbE LAN controller 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.

8.3.3 PCIe Extended Capabilities

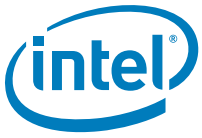
8.3.3.1 AER Registers

The following registers in the AER capability have a different behavior in a VF function.

Unlike the PF AER registers, these registers are not sticky since the VF is reset on FLR and on in-band reset.

8.3.3.1.1 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).



Bit Location	Attribute	Default Value	Description
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.
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.

8.3.3.1.2 Correctable Error Status Register (0x110; RW1C)

The Correctable Error Status register reports error status of individual correctable error sources on a PCIe device. When an individual error status bit is set to 1b it indicates that a particular error occurred; software can clear an error status by writing a 1b to the respective bit.

Bit 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.

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

9.0 System Manageability

Network management is an important requirement in today's networked computer environment. Software-based management applications provide the ability to administer systems while the operating system is functioning in a normal power state (not in a pre-boot state or powered-down state). The Intel® Out of Band Management fill the management void that exists when the operating system is not running or fully functional. This is accomplished by providing mechanisms by which manageability network traffic can be routed to and from a Management Controller (BMC).

This section describes the supported management interfaces and hardware configurations for platform system management. It describes the interfaces to an external BMC, the partitioning of platform manageability among system components, and the functionality provided by the integrated 10 GbE LAN controller in each platform configuration.

9.1 Pass-Through (PT) Functionality

Pass-Through (PT) is the term used when referring to the process of sending and receiving Ethernet traffic over the sideband interface. The integrated 10 GbE LAN controller has the ability to route Ethernet traffic to the host operating system as well as the ability to send Ethernet traffic over the sideband interface to an external BMC. See Figure 9-1.

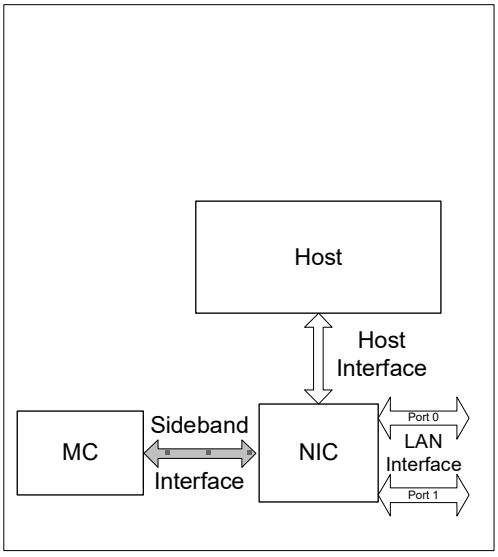


Figure 9-1. Sideband Interface

The sideband interface provides a mechanism by which the integrated 10 GbE LAN controller can be shared between the host and the BMC. By providing this sideband interface, the BMC can communicate with the LAN without requiring a dedicated Ethernet controller. The integrated 10 GbE LAN controller supports these sideband interfaces:

- SMBus
- NC-SI

The usable bandwidth for either direction is up to 1 Mb/s when using SMBus and 100 Mb/s for the NC-SI interface. When working over the integrated I/O interface, the bandwidth is limited only by the link's bandwidth and the the integrated 10 GbE LAN controller processing capabilities and can sustain any network bandwidth. Only one mode of sideband can be active at any given time. The configuration is done using an NVM setting.

The maximum packet size supported for traffic received from the LAN to the BMC is 1518 byte and additional VLAN or E-tag. For traffic from the BMC to the LAN the maximal supported packet size is 1536 bytes including all tags.

Note: In MCTP mode, SMBus interface can receive MCTP commands. For example, the MCTP enumeration process can be done over SMBus. However, only the SMBus can receive NC-SI commands or pass-through traffic.

9.1.1 Supported Topologies

The integrated 10 GbE LAN controller can support some management topologies. The following connections are available:

- Single connection via legacy SMBus (See Section 9.5).
- Single connection via NC-SI over RMII (See Section 9.6)
- Single connection via NC-SI over MCTP. This connection can be over the SMBus. This connection can be used either for pass through or for control only (see Section 9.7).

The topology used is defined in the *Redirection Sideband Interface* field in the Common Firmware Parameters 1 shared SPI Flash word and is common to all the ports in the integrated 10 GbE LAN controller.

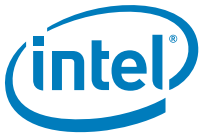
9.1.2 Pass Through Packet Routing

When an Ethernet packet reaches the integrated 10 GbE LAN controller, it is examined and compared to a number of configurable filters. These filters are configurable by the BMC and include, but not limited to, filtering on:

- MAC Address
- IP Address
- UDP/IP Ports
- VLAN Tags
- EtherType

If the incoming packet matches any of the configured filters, it is passed to the BMC. Otherwise it is not passed.

The packet filtering process is described in Section 9.3.



9.2 Components of the Sideband Interface

There are two components to a sideband interface:

- Physical Layer
- Logical Layer

9.2.1 Physical Layer

This is the electrical connection between the integrated 10 GbE LAN controller and BMC.

9.2.1.1 SMBus

The SMBus physical layer is defined by the SMBus specification. The interface is made up of two connections: data and clock. There is also an optional third connection: the alert line. This line is used by the integrated 10 GbE LAN controller to notify the BMC that there is data available for reading. Refer to the SMBus specification for details.

The SMBus can run at three speeds: 100 KHz (standard SMBus), or 400 KHz (I^2C fast mode) or 1 MHz (I^2C fast mode plus).The speed used is selected by the *SMBus Connection Speed* in *SMBus Notification Timeout and Flags* NVM word.

9.2.1.1.1 PEC Support

SMBus transactions can be protected by using Packet Error Code (PEC). Packet Error Checking, whenever applicable, is implemented by appending a PEC byte at the end of each message transfer. The PEC byte is a CRC8 calculation on all the message bytes.

PEC is added in transmit and expected in receive for the following SMBus packets:

- ARP packets
- MCTP over SMBus transactions.

For ARA cycles and legacy SMBus transactions, a PEC is not expected.

The following table lists the behavior of the integrated 10 GbE LAN controller in each PEC configured mode for transactions directly handled by the hardware upon reception of packets with or without PEC.

Table 9-1. SMBus PEC Modes[1]

Transaction/Function		Target PEC Mode	
SMBus Transaction (Relative to the Integrated 10 GbE LAN Controller)	**Integrated 10 GbE LAN Controller PEC Mode**	**PEC Enabled**	**PEC Disabled**
Master Write[2]	Enabled	(A) Target ACKs the PEC byte.	(A) Target NACKs the PEC byte.
Master Write[2]	Disabled	(A) Target receives stop before expected PEC byte.	(A) PEC byte is not expected.
Slave Write[3]	Enabled	(A) Target ACKs last data byte; PEC byte is NACKed.	(A) Target NACKs last data byte; No PEC byte is written by a slave.

Table 9-1. SMBus PEC Modes[1]

Transaction/Function		Target PEC Mode	
SMBus Transaction (Relative to the Integrated 10 GbE LAN Controller)	**Integrated 10 GbE LAN Controller PEC Mode**	**PEC Enabled**	**PEC Disabled**
Slave Write[3]	Disabled	(A) Target ACKs last data byte; PEC byte is 0xFF.	(A) Target NACKs last data byte and generates a stop after that.
Slave Read[4]	Enabled	(A) Target sends PEC byte; PEC byte is ACKed by the slave.	(A) Target does not send PEC byte and generates a stop after that.
Slave Read[4]	Disabled	(R) Target sends PEC byte; PEC byte is NACKed by the slave.	(A) Target does not send PEC byte and generates a stop after that.

1. (A) - Accept Transaction (R) - Reject Transaction.
2. Used in Legacy SMBus write commands (direct receive) and in MCTP over SMBus (transmitted transactions.
3. Used in Legacy SMBus Read commands.
4. Used in Legacy SMBus mode (alert/async-notify) and in MCTP over SMBus (received transactions).

Note: In both SMBus ARP and MCTP, the specification indicates that PEC must be used. However, if PEC is not used by the master, the transaction is still accepted and processed by the integrated 10 GbE LAN controller.

The PEC behavior is controlled by the *SMBus transaction PEC* bit in the SMBus Notification Timeout and Flags NVM word: If this bit is set, PEC is added for master SMBus write transactions. a PEC is added to slave read transactions and can be received in slave write transaction. If this bit is cleared, PEC is not added to master write or slave read transactions, a slave write transaction with PEC is dropped. This bit should be set for MCTP mode and should be cleared in legacy SMBus mode.

9.2.1.2 NC-SI

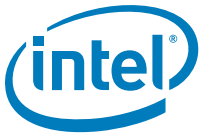
The integrated 10 GbE LAN controller uses the DMTF standard sideband interface. This interface consists of six lines for transmission and reception of Ethernet packets and two optional lines for arbitration among more than one physical network controller.

The physical layer of NC-SI is very similar to the RMII interface, although not an exact duplicate. Refer to the NC-SI specification for details of the differences.

9.2.2 Logical Layer

9.2.2.1 Legacy SMBus

The protocol layer for SMBus consists of commands the BMC issues to configure filtering for the integrated 10 GbE LAN controller management traffic and the reading and writing of Ethernet frames over the SMBus interface. There is no industry standard protocol for sideband traffic over SMBus. The protocol layer for SMBus on the integrated 10 GbE LAN controller is Intel proprietary. The Legacy SMBus protocol is described in Section 9.5.



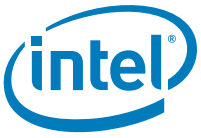
9.2.2.2 NC-SI

The DMTF also defines the protocol layer for the NC-SI interface. NC-SI compliant devices are required to implement a minimum set of commands. The specification also provides a mechanism for vendors to add additional capabilities through the use of OEM commands. Intel OEM NC-SI commands for the integrated 10 GbE LAN controller are discussed in Section 9.6.3. For information on base NC-SI commands, see the NC-SI specification.

NC-SI traffic can run on top of three different physical layers:

1. NC-SI physical layer as described in Section 9.2.1.2.
2. MCTP over SMBus. As previously described, this layer supports Sx modes.

The integrated 10 GbE LAN controller exposes one NC-SI package with two channels, one per port. The integrated 10 GbE LAN controller implement a type C NC-SI interface (Single package, common bus buffers and shared RX queue) as described in section 5.2 of the NC-SI specification.



9.2.2.2.1 Package ID setting

The package ID can be set either from the NVM *Package ID* field in the NC-SI Configuration 1 NVM word or from an SDP pin. If set from SDP, the Package ID is {0,SDP0_0 value, 0}. The mode used is set by the *NC-SI Package ID from SDP* field in the NC-SI Configuration 2 NVM word. Note that when the package ID is set from the SDP pin, SDP0_0 should be set as input is *ESDP.SDP0_IODIR* field.

The internal channel ID matches the lowest numbered PCIe function number through which this port is exposed to the host.

9.2.2.2.2 Channel ID mapping

The mapping of the channels to physical ports is according to the NC-SI Channel to Port Mapping NVM word if the *NC-SI Channel to Port Mapping.Table Valid* bit is set. If this bit is not set, the following algorithm should be used:

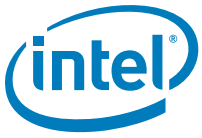
```
CHANNEL_ID = 0

If (FACTPS. LAN_FUNCTION_SEL == 0 )  // no swap

    For (x = 0 to 1) {

        PORT = x;

        If (FACTPS.LANx_VALID) NC-SI_Channel[PORT] = CHANNEL_ID++;

    }

Else  // swap

    For (x = 0 to 1) {

        PORT =  1 - x;

        If (FACTPS.LANx_VALID) NC-SI_Channel[PORT] = CHANNEL_ID++;

    }
```

9.3 Packet Filtering

Since both the host operating system and BMC use the integrated 10 GbE LAN controller to send and receive Ethernet traffic, there needs to be a mechanism by which incoming Ethernet packets can be identified as those that should be sent to the BMC rather than the host operating system.

There are two different types of filtering available. The first is filtering based upon the MAC address. With this filtering, the BMC has at least one dedicated MAC address and incoming Ethernet traffic with the matching MAC address(es) are passed to the BMC. This is the simplest filtering mechanism to utilize and it allows an BMC to receive all types traffic (including, but not limited to, IPMI, NFS, HTTP etc).

The other mechanism available utilizes a highly configurable mechanism by which packets can be filtered using a wide range of parameters. Using this method, the BMC can share a MAC address (and IP address, if desired) with the host OS and receive only specific Ethernet traffic. This method is useful if the BMC is only interested in specific traffic, such as IPMI packets.



9.3.1 Manageability Receive Filtering

This section describes the manageability receive packet filtering flow. Packet reception by the integrated 10 GbE LAN controller can generate one of the following results:

- Discarded
- Sent to host memory
- Sent to the external BMC
- Sent to both the BMC and host memory

The decisions regarding forwarding of packets to the host and to the BMC are separate and are configured through two sets of registers. However, the BMC might define some types of traffic as exclusive. This traffic is forwarded only to the BMC, even if it passes the filtering process of the host. These types of traffic are defined using the MNGONLY register.

An example of packets that might be necessary to send exclusively to the BMC might be specific TCP/UDP ports of a shared MAC address or a MAC address dedicated to the BMC. If the BMC configures the manageability filters to send these ports to the BMC, it should configure the settings to not send them to the host, otherwise, these ports will be received and handled by the host operating system.

The BMC controls the types of packets that it receives by programming receive manageability filters. The following filters are accessible to the BMC:

Table 9-2. Filters Accessible to BMC

Filters	Functionality	When Reset?
Filters Enable	General configuration of the manageability filters.	
Manageability Only	Enables routing of packets exclusively to the manageability.	
Manageability Decision Filters [7:0]	Configuration of manageability decision filters.	Power Good Reset.
MAC Address [3:0]	Four unicast MAC manageability addresses.	Power Good Reset and Firmware Reset.
VLAN Filters [7:0]	Eight VLAN tag values.	Power Good Reset and Firmware Reset.
UDP/TCP Port Filters [15:0]	16 destination port values.	Power Good Reset and Firmware Reset
Flexible 128 bytes TCO Filter	Length and values for one flex TCO filter.	Power Good Reset and Firmware Reset
IPv4 and IPv6 Address Filters [3:0]	IP address for manageability filtering.	Power Good Reset and Firmware Reset

All filtering capabilities are available on both the NC-SI and legacy SMBus interfaces. However, in NC-SI mode, in order to program part of the capabilities, the Intel OEM commands described in Section 9.6.3 should be used.

All filters are reset only on Internal Power On Reset. Register filters that enable filters or functionality are also reset by firmware reset in NC-SI mode. These registers can be loaded from the NVM following a reset if the *Enable NVM* configuration bit in *Common Manageability Parameters 2[9]* NVM word is set.

The high-level structure of manageability filtering is done using two steps.

1. The packet is parsed and fields in the header are compared to programmed filters.
2. A set of decision filters are applied to the result of the first step.

Some general rules apply:

- Fragmented packets are passed to manageability but not parsed beyond the IP header.

- Packets with L2 errors (CRC, alignment, etc.) are not forwarded to the BMC.
- Packets longer than 2KB are filtered out.
- The filtering relates only to the outer L3/L4 header. In tunneled packets, the inner L2, L3 and L4 header parameters are not considered for manageability filtering.

The following sections describe the manageability filtering, followed by the final filtering rules.

The filtering rules are created by programming the decision filters as described in Section 9.3.4.

9.3.2 L2 Filters

9.3.2.1 MAC and VLAN Filters

The manageability MAC filters allow comparison of the destination MAC address to one of 4 filters defined in the MMAH and MMAL registers.

The VLAN filters allow comparison of the 12 bit VLAN tag to one of 8 filters defined in the MAVTV registers.

9.3.2.2 EtherType Filters

Manageability L2 EtherType filters enable filtering of received packets based on the Layer 2 EtherType field. The L2 type field of incoming packets is compared against the EtherType filters programmed in the Manageability EtherType Filter (METF; up to 4 filters); the result is incorporated into decision filters.

Each manageability EtherType filter can be configured as pass (positive) or reject (negative) using a polarity bit. In order for the reverse polarity mode to be effective and block certain type of packets, the EtherType filter should be part of all the enabled decision filters.

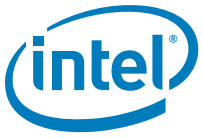
An examples for usage of L2 EtherType filters is to determine the destination of 802.1X control packets. The 802.1X protocol is executed at different times in either the management controller or by the host. L2 EtherType filters are used to route these packets to the proper agent.

In addition to the flexible EtherType filters, the integrated 10 GbE LAN controller supports 2 fixed EtherType filters used to block NC-SI control traffic (0x88F8) and flow control traffic (0x8808) from reaching the manageability interface. The NC-SI EtherType is used for communication between the management controller on the NC-SI link and the integrated 10 GbE LAN controller. Packets coming from the network are not expected to carry this EtherType and such packets are blocked to prevent attacks on the management controller. Flow control packets should be consumed by the MAC and as such are not expected to be forwarded to the management interface.

9.3.3 L3/L4 Filtering

The manageability filtering stage combines checks done at previous stages with additional L3/L4 checks to make a the decision on whether to route a packet to the BMC. The following sections describe the manageability filtering done at layers L3/L4 and final filtering rules.

Note: The L3 filters will not match tunneled fields (L3, L4 or ARP).



9.3.3.1 ARP Filtering

ARP filtering — The integrated 10 GbE LAN controller supports filtering of ARP request packets (initiated externally) and ARP responses (to requests initiated by the BMC).

In legacy SMBus mode, the ARP filters can be used as part of the ARP off load described in Section 9.5.4. ARP off load is not specifically available when using NC-SI. However, the general filtering mechanism is utilized to filter incoming ARP traffic as requested using the Enable Broadcast Filtering NC-SI command.

In order to limit the reception of ARP packets to the ARP packets dedicated to this station (ARP target IP = BMC IP), the ARP request/response filter can be bind to specific IP address, by setting both the ARP request/response and the IP AND bits in an MDEF filter, as the IP bit is set also if there is a match on the target IP (the *TPA* field in the ARP packet) of an ARP request or an ARP response.

Note: If the OR section of the MDEF is all cleared and one of the IPv4 address are set, then ARP packets matching the IP address passes the filter. If these packets should be dropped, then an OR Ethertype filter with a value of 0x0800 (IPv4) should be added.

9.3.3.2 Neighbor Discovery Filtering and MLD

The integrated 10 GbE LAN controller supports filtering of the following ICMPv6 packets.

Neighbor discovery packets:

1. 0x86 (134d) - Router Advertisement.
2. 0x87 (135d) - Neighbor Solicitation.
3. 0x88 (136d) - Neighbor Advertisement.
4. 0x89 (137d) - Redirect.

MLD packets:

1. 0x82 (130d) - MLD Query
2. 0x83 (131d) - MLDv1 Report
3. 0x84 (132d) - MLD Done
4. 0x8F (143d) - MLDv2 Report

The neighbor discovery packets has dedicated enables for each type in the decision filters. For MLD, a single enable controls the forwarding of all the MLD packets. This means that either all the MLD packets types are selected for reception or none of them.

9.3.3.3 RMCP Filtering

The integrated 10 GbE LAN controller supports filtering by fixed destination port numbers, port 0x26F and port 0x298. These ports are IANA reserved for RMCP.

In SMBus mode, there are filters that can be enabled for these ports. When using NC-SI, they are not specifically available. However, the general filtering mechanism can be utilized to filter incoming ARP traffic.

9.3.3.4 Flexible Port Filtering

The integrated 10 GbE LAN controller implements 16 flex destination port filters. The integrated 10 GbE LAN controller directs packets whose L4 destination port matches to the BMC. The BMC must ensure that only valid entries are enabled in the decision filters.

9.3.3.5 IP Address Filtering

The integrated 10 GbE LAN controller supports filtering by destination IP address using IPv4 and IPv6 address filters. These are dedicated to manageability. The integrated 10 GbE LAN controller provides four IPv6 address filters or three IPv6 addresses and four IPv4 address filters.

The IPv4 match also rises for ARP packets for which the target IP matches the IP address in the MIPAF4 register.

9.3.3.6 Checksum Filtering

If bit MANC.EN_XSUM_FILTER is set, the integrated 10 GbE LAN controller directs packets to the BMC only if they match all other filters previously described as well as pass L3/L4 checksum (if it exists).

The integrated 10 GbE LAN controller be instructed to direct packets to the BMC only if they pass the L3/L4 checksum (if they exist) in addition to matching other filters previously described.

Enabling the XSUM filter when using the SMBus interface is accomplished by setting the *Enable XSUM Filtering to Manageability* bit. This is done using the Update Management Receive Filter Parameters command. See Section 9.5.11.1.6.

To enable the XSUM filtering when using NC-SI, use the Enable Checksum Offloading command. See Section 9.6.3.15.

9.3.4 Flexible 128-byte Filter

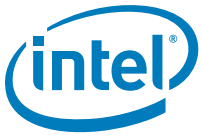
The integrated 10 GbE LAN controller provides one flex TCO filter. This filter looks for a pattern match within the first 128 bytes of the packet.

Flex filters are temporarily disabled when read from or written to by the host. Any packet received during a read or write operation is dropped. Filter operation resumes once the read or write access completes.

9.3.4.1 Flexible Filter Structure

The filter is composed of the following fields:

1. Flexible Filter length — This field indicates the number of bytes in the packet header that should be inspected. The field also indicates the minimal length of packets inspected by the filter. Packet below that length will not be inspected. Valid values for this field are: 8*n, where n=1…16.
2. Data — This is a set of up to 128 bytes comprised of values that header bytes of packets are tested against.



3. Mask — This is a set of 128 bits corresponding to the 128 data bytes that indicate for each corresponding byte if is tested against its corresponding byte. The general filter is 128 bytes that the BMC configures; all of these bytes may not be needed or used for the filtering, so the mask is used to indicate which of the 128 bytes are used for the filter.

Each filter tests the first 128 bytes (or less) of a packet, where not all bytes must necessarily be tested.

9.3.4.2 TCO Filter Programming

Programming each filter is done using the following commands (NC-SI or SMBus) in a sequential manner:

1. Filter Mask and Length — This command configures the following fields:
   a. Mask — A set of 16 bytes containing the 128 bits of the mask. Bit 0 of the first byte corresponds to the first byte on the wire.
   b. Length — A 1-byte field indicating the length.
2. Filter Data — The filter data is divided into groups of bytes. as follows:

Group	Test Bytes
0x0	0-29
0x1	30-59
0x2	60-89
0x3	90-119
0x4	120-127

Each group of bytes need to be configured using a separate command, where the group number is given as a parameter. The command has the following parameters:

   a. Group number — A 1-byte field indicating the current group addressed
   b. Data bytes — Up to 30 bytes of test-bytes for the current group

9.3.5 Configuring Manageability Filters

There are a number of pre-defined filters that are available for the BMC to enable, such as ARPs and IPMI ports 0x298/0x26F. These are generally enabled by setting the appropriate bit within the MANC register using specific commands.

For more advanced filtering needs, the BMC has the ability to configure a number of configurable filters. It is a two-step process to use these filters. They must first be configured and then enabled.

9.3.5.1 Manageability Decision Filters

Manageability Decision Filters (MDEF) are a set of eight filters, each with the same structure. The filtering rule for each decision filter is programmed by the BMC and defines which of the L2, VLAN, Ethertype and L2/L3 filters participate in decision making. Any packet that passes at least one rule is directed to manageability and possibly to the host.

The inputs to each decision filter are:

- Packet passed a valid management L2 exact address filter.
- Packet is a broadcast packet.
- Packet has a VLAN header and it passed a valid manageability VLAN filter.
- Packet matched one of the valid IPv4 or IPv6 manageability address filters.
- Packet is a multicast packet.
- Packet passed ARP filtering (request or response).
- Packet passed neighbor solicitation filtering.
- Packet passed MLD filtering.
- Packet passed 0x298/0x26F port filter.
- Packet passed a valid flex port filter.
- Packet passed a valid flex TCO filter.
- Packet passed or failed an L2 EtherType filter.
- Packet passed or failed flow control or NC-SI L2 EtherType discard filter.

The structure of each decision filter is shown in Figure 9-2. A boxed number indicates that the input is conditioned by a mask bit defined in the MDEF register and MDEF_EXT register for this rule. Decision filter rules are as follows:

- At least one bit must be set in a register. If all bits are cleared (MDEF/MDEF_EXT = 0x0000), then the decision filter is disabled and ignored.
- All enabled AND filters must match for the decision filter to match. An AND filter not enabled in the MDEF/MDEF_EXT registers is ignored. If an AND filter is preceded by a OR filter, then at least one of the enabled OR inputs must match for the filter to pass.
- If no OR filter is enabled in the register, the OR filters are ignored in the decision (the filter might still match).
- If one or more OR filters are enabled in the register, then at least one of the enabled OR filters must match for the decision filter to match.

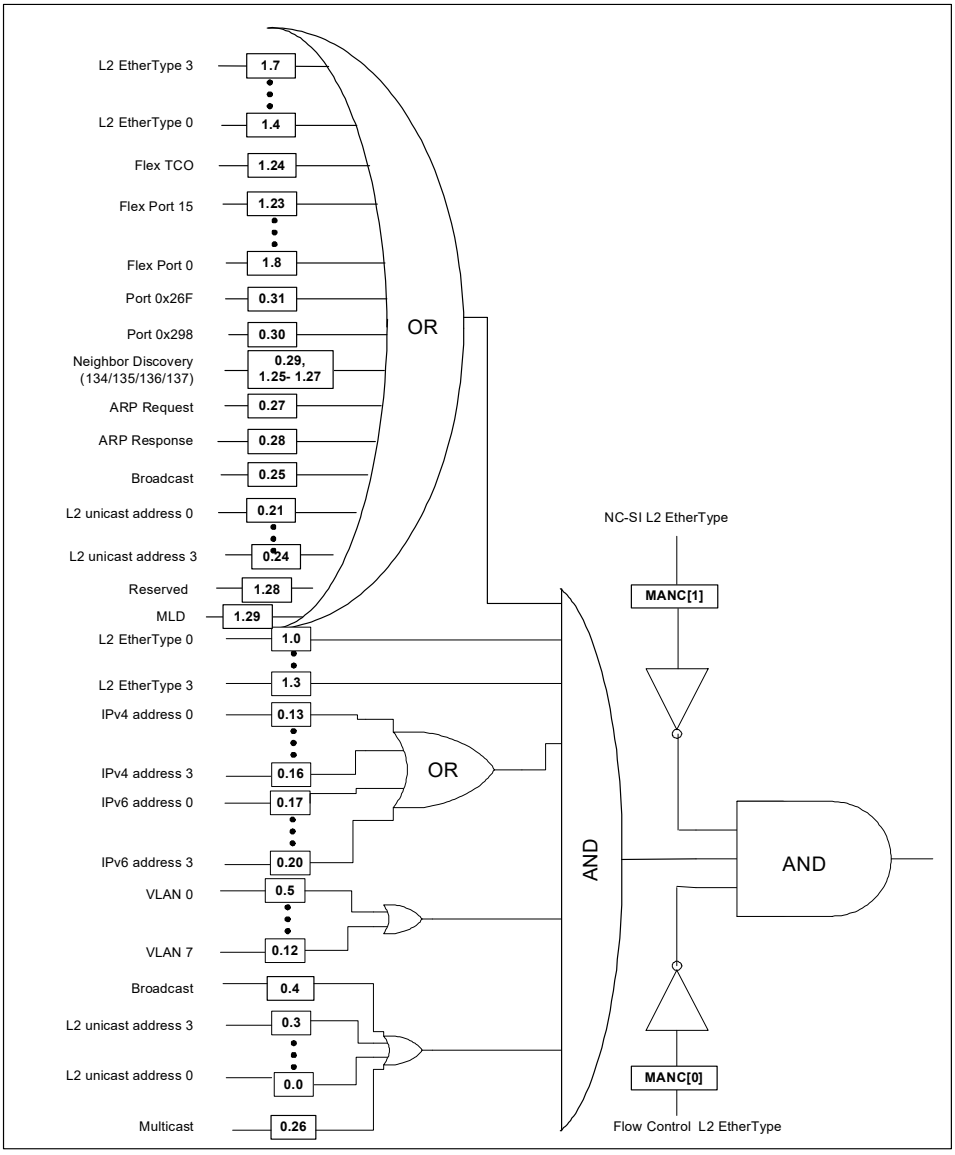


Figure 9-2. Manageability Decision Filters

A decision filter (for any of the eight filters) defines which of the above inputs is enabled as part of a filtering rule. The BMC programs two 32-bit registers per rule (MDEF[7:0] and MDEF_EXT[7:0]) with the settings as described in Section 8.2.2.20.6 and Section 8.2.2.20.7. A set bit enables its corresponding filter to participate in the filtering decision.

In addition to the controls previously described, the *MDEF_EXT.apply_to_host_traffic* and *MDEF_EXT.apply_to_network_traffic* bits defines which traffic is compared to this filter. At least one of these bits must be set for the filter to be valid.

If the *MDEF_EXT.apply_to_host_traffic* bit is set, the traffic from the host is candidate for this filter. If the *MDEF_EXT.apply_to_network_traffic* bit is set, the traffic from the network is candidate for this filter. If both bits are set, this filter is applied to all traffic.

9.3.5.2 Exclusive Traffic

The decisions regarding forwarding of packets to the host for LAN traffic or to the LAN for host traffic are independent from the management decision filters. However, the BMC might define some types of traffic as exclusive. The behavior for such traffic is defined by the using the bits corresponding to the decision filter in the *MNGONLY* register (one bit per each of the eight decision rules) and the *MDEF_EXT.apply_to_host_traffic* and *MDEF_EXT.apply_to_network_traffic* bits. Table 9-3 lists the behavior in each case. If one or more filters match the traffic and at least one of the filters is set as exclusive, the traffic is treated as exclusive.

Table 9-3. Exclusive traffic behavior

	Filter Match		Filter Don't Match
traffic source	**MNGONLY = 0**	**MNGONLY = 1**	**N/A**
From network	Traffic is forwarded to the manageability. Traffic is forwarded to the host according to host filtering	Traffic is forwarded only to manageability.	Traffic is forwarded to the host according to host filtering.
From host	Traffic is forwarded to the manageability and to the LAN	Traffic is forwarded only to manageability.	Traffic is forwarded to the LAN.

Any traffic matching any of the configurable filters (see Section 9.3.5.1) can be used as filters to pass traffic to the host.

Table 9-4. *MNGONLY* Register Description and Usage

Bits	Description	Default
0	Decision Filter 0	Determines if packets that have passed decision filter 0 are sent exclusively to the manageability path.
1	Decision Filter 1	Determines if packets that have passed decision filter 1 are sent exclusively to the manageability path.
2	Decision Filter 2	Determines if packets that have passed decision filter 2 are sent exclusively to the manageability path.
3	Decision Filter 3	Determines if packets that have passed decision filter 3 are sent exclusively to the manageability path.
4	Decision Filter 4	Determines if packets that have passed decision filter 4 are sent exclusively to the manageability path.
5	Unicast and Mixed	NC-SI mode: Determines if unicast and mixed packets are sent exclusively to the manageability path. SMBus mode: Determines if packets that have passed decision filter 5 are sent exclusively to the manageability path.

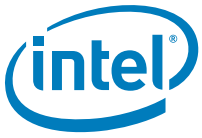


Table 9-4. *MNGONLY* Register Description and Usage (Continued)

6	Global Multicast	NC-SI mode: Determines if multicast packets are sent exclusively to the manageability path. SMBus mode: Determines if packets that have passed decision filter 6 are sent exclusively to the manageability path.
7	Broadcast	NC-SI mode: Determines if broadcast packets are sent exclusively to the manageability path. SMBus mode: Determines if ARP packets are sent exclusively to the manageability path.
31:8	Reserved	Reserved.

When using the SMBus interface, the BMC enables these filters by issuing the Update Management Receive Filter Parameters command (see Section 9.5.11.1.6) with the parameter of 0x0F.

The MNGONLY is also configurable when using NC-SI using the Set Intel Filters — Manageability Only Command (see Section 9.6.3.7.2).

All manageability filters are controlled by the BMC only and not by the LAN device driver.

9.3.5.3 Global Controls

On top of the MDEF filters, the MANC register contains some global controls applied to all the packets in order to be candidate for manageability filtering:

- Receive Enable bits:
  - The *RCV_TCO_EN* field controls the reception of manageability traffic. It should be set only if one of the following bits is set also.
  - The *EN_BMC2OS* bit controls the reception of manageability traffic from the host.
  - The *EN_BMC2NET* bit controls the reception of manageability traffic from the network.
- VLAN filtering: In order to support the NC-SI VLAN modes the following controls are provided:
  - The *FIXED_NET_TYPE* field controls if only VLAN tagged or VLAN un-tagged traffic is received. If this bit is cleared both types are received. If it is set, only the type described by the *NET_TYPE* field is accepted.
  - If set, the NET_TYPE field indicates that only VLAN tagged traffic is received, if cleared only packets without VLAN is accepted. This field is validated by the *FIXED_NET_TYPE* field.

  Both fields relates to the inner VLAN.

9.3.6 Filtering Programming Interfaces

The integrated 10 GbE LAN controller provides multiple options to program the forwarding filters, depending on the interface used and the level of flexibility needed. The following table lists the different options and points to the description of the relevant commands.

Table 9-5. Filtering Programming Interfaces

Interface	Flexible/ Abstract	Description
NC-SI (over RMII or over MCTP)	Abstract (dedicated MAC address)	The regular NC-SI commands can be used to allow forwarding based on a dedicated MAC address. The list of supported commands can be found in Section 9.6.2.1. When using these commands, one of the two other modes can be used to add finer grain filtering.
	Flexible	This interface described in most of the subsections of Section 9.6.3.4. It uses the packet reduction commands to reduce the forwarding scope of the filters set by the regular NC-SI commands and the packet addition commands to add new packet types to the forwarding rules.
SMBus	Abstract	The Set Common filter command (Section 9.5.11.1.7) can be used to set the most common filters. When using this commands the flexible filtering interface should not be used. When sending this command, all previous filtering requests are cleared.
	Flexible	The Update MNG RCV Filter Parameters commands (Section 9.5.11.1.6) can be used to define the exact filtering rules to be applied.

9.3.7 Possible Configurations

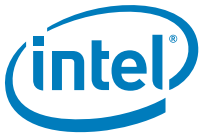
This section describes ways of using management filters. Actual usage might vary.

9.3.7.1 Dedicated MAC Packet Filtering

- Select one of the eight rules for dedicated MAC filtering.
- Load host MAC address to one of the management MAC address filters and set the appropriate bit in field 3:0 of the MDEF register.
- Set other bits to qualify which packets are allowed to pass through. For example:
  - Set bit 5 in MDEF to qualify with the first manageability VLAN.
  - Set relevant bits 13 to 20 in MDEF to qualify with a match to one of the IP addresses.
  - Set any L3/L4 bits (bits 27 to 31 in MDEF and bits 16 to 23 in MDEF_EXT) to qualify with any of a set of L3/L4 filters.

9.3.7.2 Broadcast Packet Filtering

- Select one of the eight rules for broadcast filtering.
- Set bit 25 in MDEF of the decision rule to enforce broadcast filtering.
- Set other bits to qualify which broadcast packets are allowed to pass through. For example:
  - Set bit 5 in MDEF to qualify with the first manageability VLAN.
  - Set relevant bits 13 to 20 in MDEF to qualify with a match to one of the IP addresses.
  - Set any L3/L4 bits (bits 27 to 31 in MDEF and bits 16 to 23 in MDEF_EXT) to qualify with any of a set of L3/L4 filters.



9.3.7.3 VLAN Packet Filtering

- Select one of the eight rules for VLAN filtering.
- Set bit 5 to 12 in MDEF to qualify with the relevant manageability VLANs.
- Set other bits to qualify which VLAN packets are allowed to pass through. For example:
  - Set any L3/L4 bits (bits 27 to 31 in MDEF and bits 16 to 23 in MDEF_EXT) to qualify with any of a set of L3/L4 filters.

9.3.7.4 IPv6 Filtering

IPv6 filtering is done using the following IPv6-specific filters:

- IP Unicast filtering — requires filtering for Link Local address and a Global address. Filtering setup might depend on whether or not the MAC address is shared with the host or dedicated to manageability:
  - Dedicated MAC address (for example, dynamic address allocation with DHCP does not support multiple IP addresses for one MAC address). In this case, filtering can be done at L2 using two dedicated unicast MAC filters.
  - Shared MAC address (for example, static address allocation sharing addresses with host). In this case, filtering needs to be done at L3, requiring two IPv6 address filters, one per address.
- A neighbor Discovery filter — The integrated 10 GbE LAN controller supports IPv6 neighbor Discovery protocol. Since the protocol relies on multicast packets, the integrated 10 GbE LAN controller supports filtering of these packets. IPv6 multicast addresses are translated into corresponding Ethernet multicast addresses in the form of 33-33-xx-xx-xx-xx, where the last 32 bits of address are taken from the last 32 bits of the IPv6 multicast address. As a result, two direct MAC filters can be used to filter IPv6 solicited-node multicast packets as well as IPv6 all node multicast packets.

9.3.7.5 Receive Filtering with Shared IP

When using the Legacy SMBus interface or the MCTP interface, it is possible to share the host MAC and IP address with the BMC. This functionality is also available when using base NC-SI using Intel OEM commands.

When the BMC shares the MAC and IP address with the host, receive filtering is based on identifying specific flows through port allocation. The following setting might be used when using the legacy SMBus interface:

- Select one of the eight rules.
- Set a manageability dedicated MAC filter to the host MAC address and set the matching bit (0-3) in the MDEF register.
- If VLAN is used for management, load one or more management VLAN filters and set the matching bit (5- 12) in the MDEF register.

ARP filter/neighbor discovery filter is enabled when the BMC is responsible for handling the ARP protocol. Set bit 27 or bit 28 in the MDEF register for this functionality.

In NC-SI over MCTP, dedicated commands are used to allow shared IP filtering.

9.3.8 Determining Manageability MAC Address

If the BMC needs to use a dedicated MAC address or configure the automatic ARP response mechanism (only available in SMBus mode), it might be beneficial for the BMC to be able to determine the MAC address used by the host.

Both the NC-SI and SMBus interfaces provide an Intel OEM command to read the System MAC address.

A possible use for this is that the MAC address programmed at manufacturing time does not increment by one each time, but rather by two. In this way, the BMC can read the system MAC address and add one to it and be guaranteed of a unique MAC address.

Determining the IP address being used by the host is beyond the scope of this document.

9.4 OS-to-BMC Traffic

9.4.1 Overview

Traditionally, the communication between a host and the local BMC is not handled through the network interface and requires a dedicated interface such as an IPMI KCS interface. The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller 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 on the pass-through interface, the integrated 10 GbE LAN controller checks if it should be forwarded to the network, the host, or both. Figure 9-3 describes the flow for OS-to-BMC traffic for the NC-SI over RMII case. It is not supported in legacy SMBus mode.

The OS-to-BMC flow can be enabled using the *OS2BMC enable* field for the relevant port in the OS 2 BMC configuration structure of the shared SPI Flash.

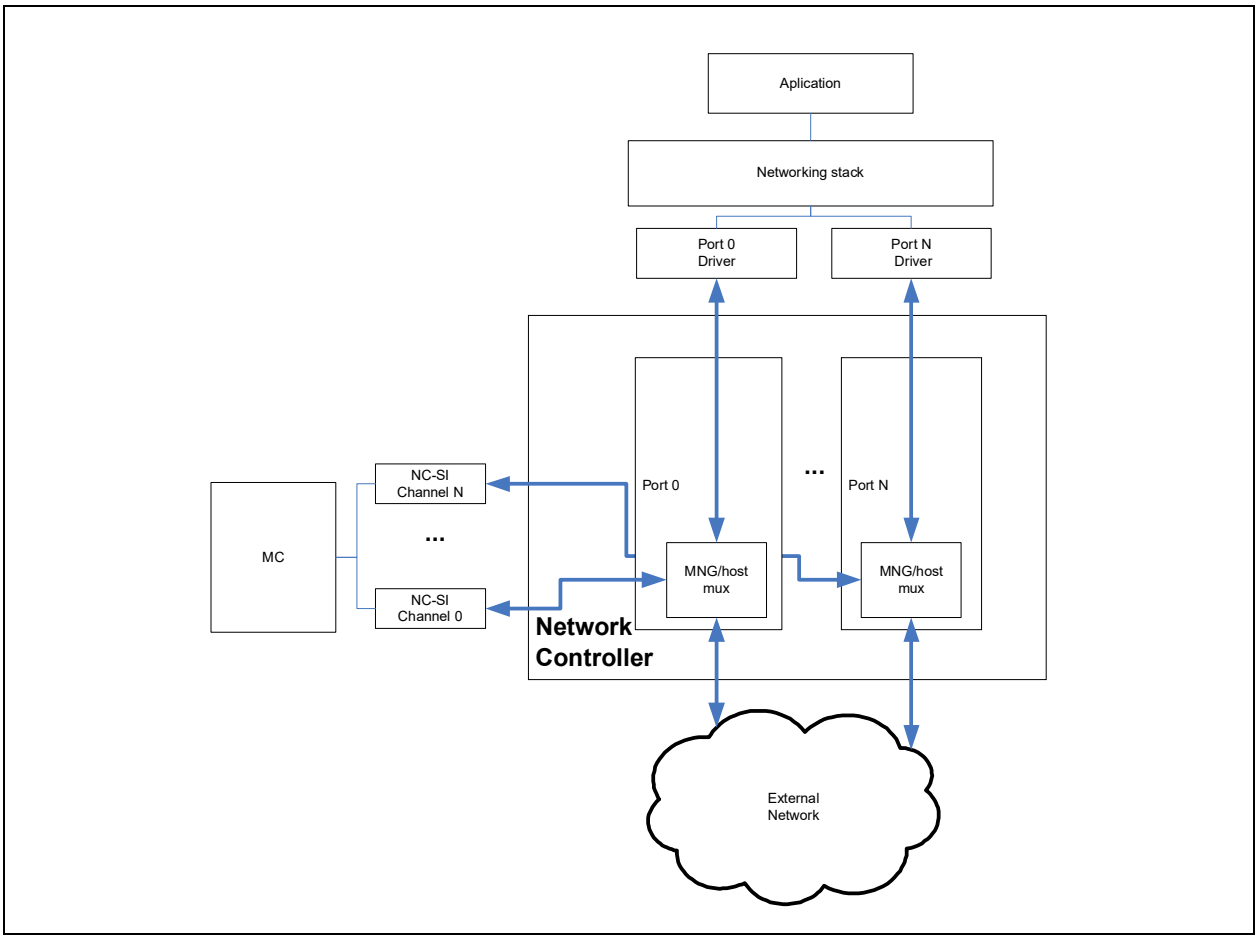


Figure 9-3. OS-to-BMC Block Diagram

The OS-to-BMC flow is enabled only for ports enabled by the NC-SI Enable Channel command or via the *OS to BMC Enable* field for the relevant port in the OS-to-BMC configuration structure of the NVM.

OS-to-BMC traffic must comply with NC-SI specifications and is therefore limited to maximum sized frames of 1536 bytes (in both directions).

9.4.2 Filtering

When OS-to-BMC traffic is enabled, the filters used for network to BMC traffic are also used for OS-to-BMC traffic. Traffic considered as exclusive to the BMC (Relevant bit in MNGONLY is set) is also considered as exclusive to the BMC when sent from the Host and not forwarded to the network.

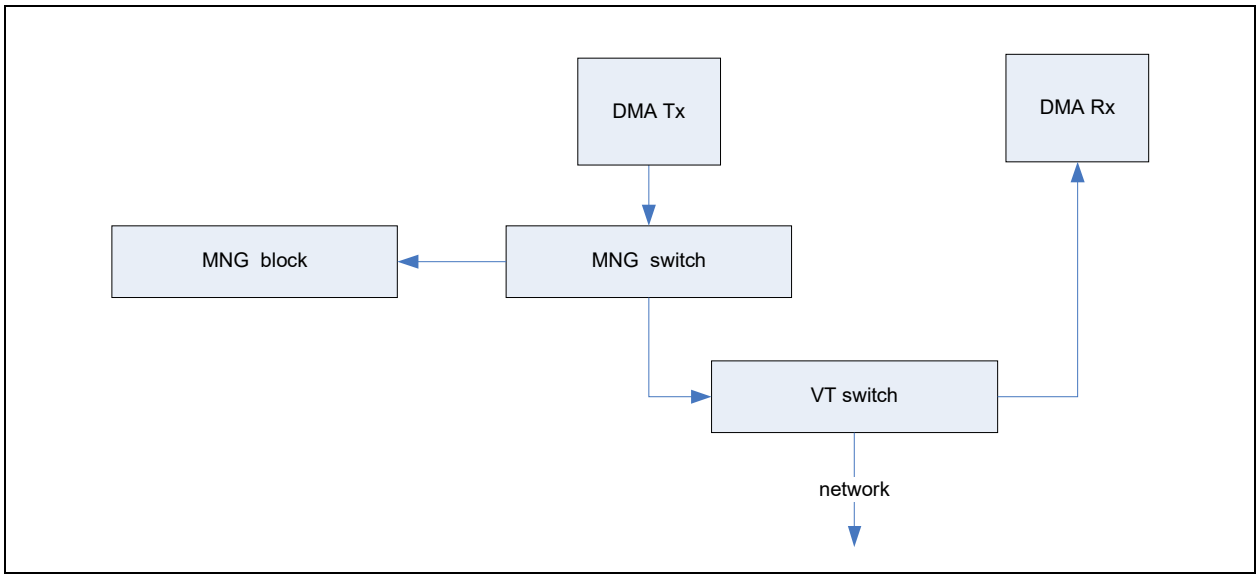


Figure 9-4. OS-to-BMC and VM-to-VM Filtering

9.4.2.1 Handling of OS-to-BMC Packets

All the regular transmit offloads are also available for OS-to-BMC packets.

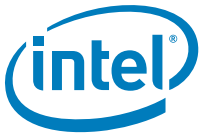
9.4.2.2 BMC-to-OS Filtering

When OS-to-BMC 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 6.1.2). 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 operating system packets:

- If *BMC to net* is disabled, all the traffic from the BMC is sent to the host.
- If *BMC to host* is disabled, all the traffic from the BMC is sent to the network.
- If both *BMC to net* and *BMC to host are enabled,* the packet are forwarded only according to the destination MAC address and VLAN tag. Unicast packets that matches one of the exact filters (RAH/RAL) are sent only to the host. Other packets that pass the L2 host filtering is sent to both the host and the network. Packets that do not pass the L2 host filtering are sent only to the network.

Note: In virtualization mode, if packet is received by host only due to default pooling (*PFVTCTL.Dis_Def_Pool* bit is cleared), then it is not sent to the host (even if *BMC to net* is disabled).



9.4.2.3 Queuing of Packets Received From the BMC

Packets received from the BMC are queued in the default queue.

9.4.2.4 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. Packet received from the BMC are identified by the *RDESC.STATUS.BMC* bit.

The following list describes which offloads are available for BMC packets:

- CRC is checked and removed on the BMC packets.
- The BMC packets are not detected as time sync packet. The *RDESC.STATUS.TS* always clears for these packets.
- In systems where the double VLAN feature is enabled (*CTRL_EXT.EXTENDED_VLAN* is set), the VEXT bit is valid for BMC packets and reflects the presence of a tag with an Ethertype matching the value in *EXVET* register.

Note: In systems that uses double VLAN, the BMC is expected to send all packets (apart from NC-SI commands) with the outer VLAN included. Failing to do so might cause corruptions 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.

9.4.3 Blocking of Network-to-BMC Flow

In some systems the BMC might have its own private connection to the network and might use the integrated 10 GbE LAN controller port only for the OS-to-BMC traffic. In this case, the BMC-to-network flow should be blocked while enabling the OS-to-BMC and OS-to-network flows.

This can be done by clearing the *MANC.EN_BMC2NET* bit for the relevant port. The BMC can control this functionality using the Enable Network to BMC flow and Disable Network to BMC flow NC-SI OEM commands. This can also be controlled using the *Network to BMC disable* field in the NVM OS2BMC Configuration Structure.

Notes: When network to BMC flow is blocked and OS-to-BMC flow is enabled, all the traffic from the BMC is sent to the operating system without any check. The operating system traffic filtering is still done using the regular decision filters.

The NC-SI channel should not be enabled for receive or transmit before at least one of the EN_BMC2NET or EN_BMC2OS fields is set, unless used for AEN transmissions only. In this case, the channel might be enabled for receive, but all receive filters should be cleared.

9.4.4 OS-to-BMC and Flow Control

The traffic between the host and manageability uses the same buffers as any loopback traffic. Thus it flows through the transmit buffer and then through the receive buffer. If the transmit buffer is flow controlled, then the host to BMC traffic is also stopped. If the receive buffer is full, the traffic is dropped or the transmit is stopped according to the flow control policy of this traffic class.

Packets received to the manageability (either from host or from network) might be dropped if the manageability internal buffers are full.

9.4.5 Statistics

Packets sent from the OS 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 integrated 10 GbE LAN controller supports the following statistical counters that measure just the BMC-to-OS and OS-to-BMC traffic:

- O2BGPTC: OS-to-BMC packets received by BMC
- O2BSPC: OS-to-BMC packets transmitted by operating system and received by manageability buffer.
- B2OSPC: BMC-to-OS packets sent by BMC
- B2OGPRC: BMC-to-OS packets received by the operating system.

The software device driver can use these statistics to count packets dropped by the integrated 10 GbE LAN controller during the transfer between the operating system and the BMC or the LAN and the BMC as follows:

- Dropped packets in BMC-to-OS path = *B2OSPC - B2OGPRC*
- Dropped packet in BMC-to-LAN path = *B2OSDPC - (B2OSPC - B2OGPRC)*
- Dropped packets in OS-to-BMC path = *O2BSPC - O2BGPTC*
- Dropped packets in LAN-to-BMC path = *MNGPDC*

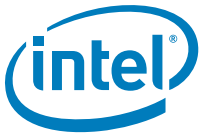
See Section 6.1.7.2 and Section 6.2.6.2 for details of the statistics hierarchy.

9.4.6 OS-to-BMC Enablement

The integrated 10 GbE LAN controller supports the unified network software model for OS-to-BMC traffic, where the OS-to-BMC traffic is shared with the regular traffic. In this model, there is no need for a special configuration of the operating system networking stack or the BMC stack, but if the link is down, then the OS-to-BMC communication is stopped.

In order to enable OS-to-BMC either:

- Enable OS2BMC in the port traffic type field in the Traffic type Parameters NVM word for the relevant port.
- Send an Enable Network-to-BMC Command



When OS2BMC is enabled, the operating system must avoid sending packets longer than 1.5 KB to the BMC.

9.5 SMBus Pass-Through Interface

SMBus is the system management bus defined by Intel. It is used in personal computers and servers for low-speed system management communications. This section describes how the SMBus interface operates in legacy pass-through mode.

9.5.1 General

The SMBus sideband interface includes standard SMBus commands used for assigning a slave address and gathering device information as well as Intel proprietary commands used specifically for the pass-through interface.

9.5.2 Pass-Through Capabilities

This section details manageability capabilities the integrated 10 GbE LAN controller provides while in SMBus mode. Pass-through traffic is carried by the sideband interface as described in Section 9.1.

These services are not available in NC-SI mode.

When operating in SMBus mode, in addition to exposing a communication channel to the LAN for the BMC, the integrated 10 GbE LAN controller provides the following manageability services to the BMC:

- ARP handling — The integrated 10 GbE LAN controller can be programmed to auto-ARP replying for ARP request packets to reduce the traffic over the BMC interconnect.
- Default configuration of filters by NVM - When working in SMBus mode, the default values of the manageability receive filters can be set according to the PT LAN and flex TCO NVM structures.

9.5.3 Port to SMBus Mapping

The integrated 10 GbE LAN controller is presented on the SMBus manageability link as two different devices (for example, via two different SMBus addresses on which each device is connected to a different LAN port). There is no logical connection between the devices.

The fail-over between the LAN ports is done by the BMC (by sending/receiving packets through different devices). The status report to the BMC, ARP handling, DHCP, and other pass-through functionality are unique for each port and configured by the BMC.

9.5.4 Automatic Ethernet ARP Operation

The integrated 10 GbE LAN controller can offload the Ethernet Address Resolution Protocol (ARP) for the BMC in order to reduce the bandwidth required on the SMBus link.

Automatic Ethernet ARP parameters are loaded from the NVM when the integrated 10 GbE LAN controller is powered up or configured through the sideband management interface. The following parameters should be configured in order to enable ARP operation:

- ARP auto-reply enabled
- ARP IP address (to filter ARP packets)
- ARP MAC addresses (for ARP responses)

These are all configurable over the sideband interface using the advanced version of the Receive Enable command.

When an ARP request packet is received and ARP auto-reply is enabled, the integrated 10 GbE LAN controller checks the targeted IP address (after the packet has passed L2 checks and ARP checks). If the targeted IP matches the IP configuration for the integrated 10 GbE LAN controller, it replies with an ARP response.

The integrated 10 GbE LAN controller responds to ARP request targeted to the ARP IP address with the configured ARP MAC address. In case that there is no match, the integrated 10 GbE LAN controller silently discards the packets. If the integrated 10 GbE LAN controller is not configured to do auto-ARP response, it can be configured to forward the ARP packets to the BMC (which can respond to ARP requests).

When the external BMC uses the same IP and MAC address of the OS, the ARP operation should be coordinated with the host operating system.

Note: If sharing the MAC and IP with the host operating system is possible, the integrated 10 GbE LAN controller provides the ability to read the stem MAC address, allowing the BMC to share the MAC address. There is no mechanism however provided by the integrated 10 GbE LAN controller to read the IP address. The host operating system (or an agent within) and BMC must coordinate the sharing of IP addresses.

9.5.5 SMBus Transactions

This section gives a brief overview of the SMBus protocol. Following is an example for a format of a typical SMBus transaction.

Table 9-6. Typical SMBus Transaction

1	7	1	1	8	1	8	1	1
S	Slave Address	Wr	A	Command	A	PEC	A	P
	1100 001	0	0	0000 0010	0	[Data Dependent]	0	

The top row of the table identifies the bit length of the field in a decimal bit count. The middle row (bordered) identifies the name of the fields used in the transaction. The last row appears only with some transactions, and lists the value expected for the corresponding field. This value can be either hexadecimal or binary.

The SMBus controller is a master for some transactions and a slave for others. The differences are identified in this document.

Shorthand field names are listed in Table 9-7 and are fully defined in the SMBus specification.

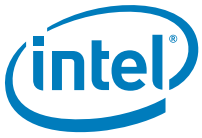


Table 9-7. Shorthand Field Names

Field Name	Definition
S	SMBus START Symbol
P	SMBus STOP Symbol
PEC	Packet Error Code
A	ACK (Acknowledge)
N	NACK (Not Acknowledge)
Rd	Read Operation (Read Value = 1b)
Wr	Write Operation (Write Value = 0b)

9.5.5.1 SMBus Addressing

The SMBus is presented as up to two SMBus devices on the SMBus (two addresses). All pass-through functionality is duplicated on the SMBus address, where each SMBus address is connected to a different LAN port. Note that it is not permitted to configure multiple ports to the same SMBus address. When a LAN function is disabled, the corresponding SMBus address is not presented to the BMC.

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

In addition to the SMBus address values, all parameters of the SMBus (channel selection, address mode, and address enable) can be set only through NVM configuration. Note that the NVM is read at the integrated 10 GbE LAN controller's power up and resets.

9.5.5.2 SMBus ARP Functionality

The integrated 10 GbE LAN controller supports the SMBus ARP protocol as defined in the SMBus 2.0 specification. The integrated 10 GbE LAN controller is a persistent slave address device so its SMBus address is valid after power-up and loaded from the NVM. The integrated 10 GbE LAN controller supports all SMBus ARP commands defined in the SMBus specification both general and directed.

SMBus ARP capability can be disabled through the NVM.

9.5.5.3 SMBus ARP Flow

SMBus ARP flow is based on the status of two flags:

- AV (Address Valid): This flag is set when the integrated 10 GbE LAN controller has a valid SMBus address.
- AR (Address Resolved): This flag is set when the integrated 10 GbE LAN controller SMBus address is resolved (SMBus address was assigned by the SMBus ARP process).

These flags are internal Integrated 10 GbE LAN Controller flags and are not exposed to external SMBus devices.

Since the integrated 10 GbE LAN controller is a Persistent SMBus Address (PSA) device, the AV flag is always set, while the AR flag is cleared after power up until the SMBus ARP process completes. Since AV is always set, the integrated 10 GbE LAN controller always has a valid SMBus address.

When the SMBus master needs to start an SMBus ARP process, it resets (in terms of ARP functionality) all devices on SMBus by issuing either Prepare to ARP or Reset Device commands. When the integrated 10 GbE LAN controller accepts one of these commands, it clears its AR flag (if set from previous SMBus ARP process), but not its AV flag (the current SMBus address remains valid until the end of the SMBus ARP process).

Clearing the AR flag means that the integrated 10 GbE LAN controller responds to SMBus ARP transactions that are issued by the master. The SMBus master issues a Get UDID command (general or directed) to identify the devices on the SMBus. The integrated 10 GbE LAN controller always responds to the Directed command and to the General command only if its AR flag is not set.

After the Get UDID, The master assigns the integrated 10 GbE LAN controller SMBus address by issuing an Assign Address command. The integrated 10 GbE LAN controller checks whether the UDID matches its own UDID and if it matches, it switches its SMBus address to the address assigned by the command (byte 17). After accepting the Assign Address command, the AR flag is set and from this point (as long as the AR flag is set), the integrated 10 GbE LAN controller does not respond to the Get UDID General command. Note that all other commands are processed even if the AR flag is set. If the address changed, from the one previously stored in the NVM, The integrated 10 GbE LAN controller stores the SMBus address that was assigned in the SMBus ARP process in the NVM, so at the next power up, it returns to its assigned SMBus address. This process uses the NVM update flow described in Section 2.4.2.1.

Figure 9-5 shows the integrated 10 GbE LAN controller SMBus ARP flow.

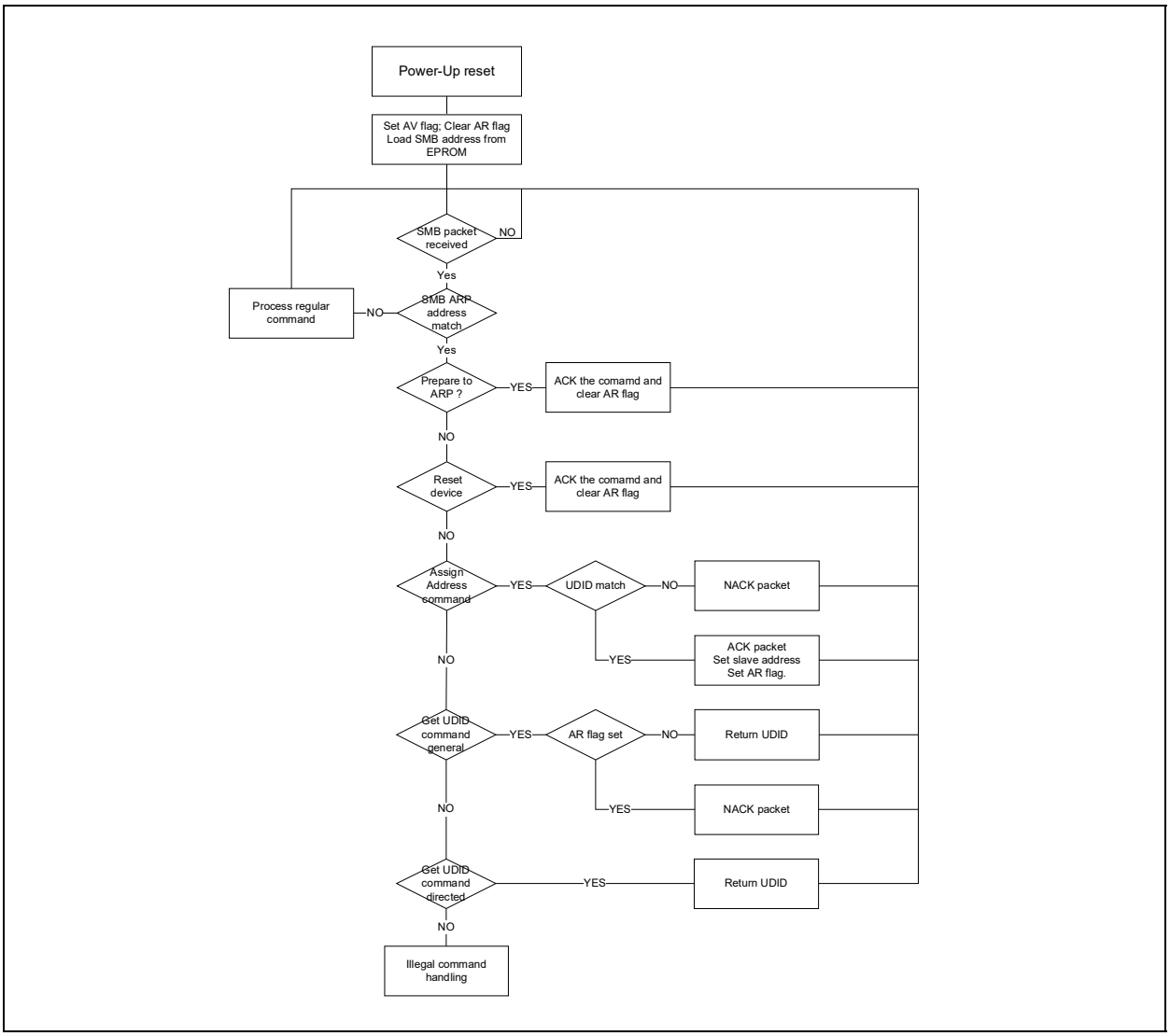


Figure 9-5. SMBus ARP Flow

9.5.5.4 SMBus ARP UDID Content

The UDID provides a mechanism to isolate each device for the purpose of address assignment. Each device has a unique identifier. The 128-bit number is comprised of the following fields:

Table 9-8. UDID

1 Byte	1 Byte	2 Bytes	2 Bytes	2 Bytes	2 Bytes	2 Bytes	4 Bytes
Device Capabilities	Version/ Revision	Vendor ID	Device ID	Interface	Subsystem Vendor ID	Subsystem Device ID	Vendor Specific ID
See notes that follow	See notes that follow	See notes that follow	See notes that follow	0x0004/ 0x0024	0x0000	0x0000	See notes that follow
MSB							LSB

Where:

Vendor ID:	The device manufacturer's ID as assigned by the SBS Implementers' Forum or the PCI SIG. Constant value: 0x8086
Device ID:	The device ID as assigned by the device manufacturer (identified by the Vendor ID field). Constant value: see the defined silicon default value in Section 8.2.2.2.
Interface:	Identifies the protocol layer interfaces supported over the SMBus connection by the device. Bits 3:0 = 0x4 indicates SMBus Version 2.0 Bit 5 (ASF bit) = 1 in MCTP mode.
Subsystem Fields:	These fields are not supported and return zeros.

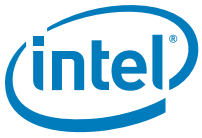
Device Capabilities: Dynamic and Persistent Address, *PEC Support* bit:

7	6	5	4	3	2	1	0
Address Type		Reserved (0)	Reserved (0)	Reserved (0)	Reserved (0)	Reserved (0)	PEC Supported
0b	1b	0b	0b	0b	0b	0b	0/1b[1]
MSB							LSB

1. The value is set according to the *SMBus Transaction PEC* bit in the NVM.

Version/Revision: UDID Version 1, Silicon Revision:

7	6	5	4	3	2	1	0
Reserved (0)	Reserved (0)	UDID Version			Silicon Revision ID ***Note:*** The revision ID reported here is 0x0 for A0/A1 and 0x1 for B0, and does not match the value reported in the PCIe config space.		
0b	0b	001b					
MSB							LSB



Vendor Specific ID: Four LSB bytes of the device Serial Number combined with the port number. The Serial Number is taken from the NVM and is reflected in the PCI_SERL and PCI_SERH registers.

1 Byte	1 Byte	1 Byte	1 Byte
Port Number[1], MAC Address, Byte 3 LSB	MAC Address, Byte 2	MAC Address, Byte 1	MAC Address, Byte 0
MSB			LSB

1. The port number is either one or two bits according to the number of port in the SoC.

9.5.5.5 SMBus ARP and Multi-port

The integrated 10 GbE LAN controller responds as two SMBus devices having two sets of AR/AV flags (one for each port). The integrated 10 GbE LAN controller responds two time to the SMBus ARP master, once each for each port. All SMBus addresses are taken from the SMBus ARP address word of the NVM.

Note that the Unique Device Identifier (UDID) is different for the different ports in the version ID field. The integrated 10 GbE LAN controller first respond as port 0, and only when an address is assigned, then start responding as port 1 to the Get UDID command.

9.5.5.6 Concurrent SMBus Transactions

The SMBus interface is single threaded. Thus, concurrent SMBus transactions are not permitted. Once a transaction is started, it must be completed before additional transaction can be initiated.

A transaction is defined as:

- All the SMBus commands used to receive a packet.
- All the SMBus commands used to send a packet.
- The read and write SMBus commands used as part of read parameters described in Section 9.5.11.2.
- The single write SMBus commands described in Section 9.5.11.1.

9.5.6 SMBus Notification Methods

The integrated 10 GbE LAN controller supports three methods of notifying the BMC that it has information that needs to be read by the BMC:

- SMBus alert - Refer to Section 9.5.6.1.
- Asynchronous notify - Refer to Section 9.5.6.2.
- Direct receive - refer to Section 9.5.6.3.

The notification method used by the integrated 10 GbE LAN controller can be configured from the SMBus using the Receive Enable command (Section 9.5.11.1.3). The default method is set by the NVM in the Notification Method field in LAN Receive Enable 1.

Note: The SMBus notification method used must be the same for all ports.

The following events cause the integrated 10 GbE LAN controller to send a notification event to the BMC:

- Receiving a LAN packet that is designated to the BMC.
- Firmware was reset and requires re-initialization.
- Receiving a Request Status command from the BMC initiates a status response.
- The integrated 10 GbE LAN controller is configured to notify the BMC upon status changes (by setting the EN_STA bit in the Receive Enable command) and one of the following events happen:
  - TCO Command Aborted
  - Link Status changed
  - Power state change

There can be cases where the BMC is hung and not responding to the SMBus notification. The integrated 10 GbE LAN controller has a time-out value (defined in the NVM) to avoid hanging while waiting for the notification response. If the BMC does not respond until the time out expires, the notification is de-asserted and all pending data is silently discarded.

Note that the SMBus notification time-out value can only be set in the NVM. The BMC cannot modify this value.

9.5.6.1 SMBus Alert and Alert Response Method

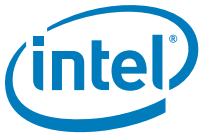
The SMBus Alert# (SMBALERT_N) signal is an additional SMBus signal that acts as an asynchronous interrupt signal to an external SMBus master. The integrated 10 GbE LAN controller asserts this signal each time it has a message that it needs the BMC to read and if the chosen notification method is the SMBus alert method. Note that the SMBus alert method is an open-drain signal which means that other devices besides the integrated 10 GbE LAN controller can be connected on the same alert pin. As a result, the BMC needs a mechanism to distinguish between the alert sources.

The BMC can respond to the alert by issuing an ARA Cycle command to detect the alert source device. The integrated 10 GbE LAN controller responds to the ARA cycle with its own SMBus slave address (if it was the SMBus alert source) and de-asserts the alert when the ARA cycle completes. Following the ARA cycle, the BMC issues a read command to retrieve the integrated 10 GbE LAN controller message.

Some BMCs do not implement the ARA cycle transaction. These BMCs respond to an alert by issuing a Read command to the integrated 10 GbE LAN controller (0xC0/0xD0 or 0xDE). The integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller is the source of the SMBus Alert, it replies the read transaction and then de-asserts the alert after the command byte of the read transaction.

Note: In SMBus Alert mode, the SMBALERT_N pin is used for notification. Each port generate alerts on events that are independent of each other.
If two ports have events to notify, the second alert is asserted only after the first event is handled.

The ARA cycle is an SMBus receive byte transaction to SMBus Address 0001-100b. Note that the ARA transaction does not support PEC. The ARA transaction format is as follows:



1	7	1	1	8	1	1	1
S	Alert Response Address	Rd	A	Slave Device Address		A	P
	0001 100	1	0	Manageability Slave SMBus Address	0	1	

9.5.6.2 Asynchronous Notify Method

When configured using the asynchronous notify method, the integrated 10 GbE LAN controller acts as a SMBus master and notifies the BMC of one of the events listed in Section 9.5.6 by issuing a modified form of the write word transaction. The asynchronous notify transaction SMBus address and data payload is configured using the Receive Enable command (Section 9.5.11.1.3) or using the NVM defaults. Note that the asynchronous notify is not protected by a PEC byte.

1	7	1	1	7	1	1	8	1	8	1	1
S	Target Address	Wr	A	Sending Device Address		A	Data Byte Low	A	Data Byte High	A	P
	BMC Slave Address	0	0	MNG Slave SMBus Address	0	0	Interface	0	Alert Value	0	

The target address and data byte low/high are taken from the Receive Enable command or NVM configuration.

If the BMC does not read the status in the delay defined by the *SMBus Notification Timeout* NVM field the status word is cleared and the packet is dropped.

9.5.6.3 Direct Receive Method

If configured, the integrated 10 GbE LAN controller has the capability to send a message it needs to transfer to the external BMC as a master over the SMBus instead of alerting the BMC and waiting for it to read the message.

The message format follows. Note that the command that is used is the same command that is used by the external BMC in the Block Read command. The opcode that the integrated 10 GbE LAN controller puts in the data is also the same as it put in the Block Read command of the same functionality. The rules for the *F* and *L* flags (bits) are also the same as in the Block Read command.

1	7	1	1	1	1	6	1	
S	Target Address	Wr	A	F	L	Command	A	...
	BMC Slave Address	0	0	First Flag	Last Flag	Receive TCO Command 01 0000b	0	

8	1	8	1		1	8	1	1
Byte Count	A	Data Byte 1	A	...	A	Data Byte N	A	P
N	0		0		0		0	

9.5.7 Receive Pass Through Flow

The integrated 10 GbE LAN controller is used as a channel for receiving packets from the network link and passing them to the external BMC. The BMC configures the integrated 10 GbE LAN controller to pass these specific packets to the BMC. Once a full packet is received from the link and identified as a manageability packet that should be transferred to the BMC, the integrated 10 GbE LAN controller starts the receive TCO flow to the BMC.

The integrated 10 GbE LAN controller uses the SMBus notification method to notify the BMC that it has data to deliver. Since the packet size might be larger than the maximum SMBus fragment size, the packet is divided into fragments, where the integrated 10 GbE LAN controller uses the maximum fragment size allowed in each fragment (configured via the NVM). The last fragment of the packet transfer is always the status of the packet. As a result, the packet is transferred in at least two fragments. The data of the packet is transferred as part of the receive TCO LAN packet transaction.

When SMBus alert is selected as the BMC notification method, the integrated 10 GbE LAN controller notifies the BMC on each fragment of a multi-fragment packet. When asynchronous notify is selected as the BMC notification method, the integrated 10 GbE LAN controller notifies the BMC only on the first fragment of a received packet. It is the BMC's responsibility to read the full packet including all the fragments.

Any timeout on the SMBus notification results in discarding the entire packet. Any NACK by the BMC causes the fragment to be re-transmitted to the BMC on the next Receive Packet command.

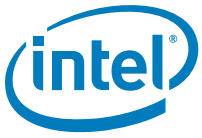
The maximum size of the received packet is limited by the integrated 10 GbE LAN controller to 1536 bytes. Packets larger than 1536 bytes are silently discarded. Any packet smaller than 1536 bytes is processed.

9.5.8 Transmit Pass Through Flow

The integrated 10 GbE LAN controller is used as the channel for transmitting packets from the external BMC to the network link. The network packet is transferred from the BMC over the SMBus and then, when fully received by the integrated 10 GbE LAN controller, is transmitted over the network link.

Each SMBus address is connected to a different LAN port. When a packet is received during a SMBus transaction using SMBus address #0, it is transmitted to the network using LAN port #0; it is transmitted through LAN port #1 if received on SMBus address #1, etc.

The integrated 10 GbE LAN controller supports packets up to an Ethernet packet length of 1536 bytes. Since SMBus transactions can only be up to 240 bytes in length, packets might need to be transferred over the SMBus in more than one fragment. This is achieved using the *F* and *L* bits in the command number of the transmit TCO packet Block Write command. When the *F* bit is set, it is the first fragment of the packet. When the *L* bit is set, it is the last fragment of the packet. When both bits are set, the entire packet is in one fragment. The packet is sent over the network link only after all its fragments are received correctly over the SMBus. The maximum SMBus fragment size is defined within the NVM and cannot be changed by the BMC.



The minimum packet length defined by the 802.3 spec is 64 bytes. The integrated 10 GbE LAN controller pads packets that are less than 64 bytes to meet the specification requirements (there is no need for the external BMC to pad packets less than 64 bytes). If the packet sent by the BMC is larger than 1536 bytes, the integrated 10 GbE LAN controller silently discards the packet.

The integrated 10 GbE LAN controller calculates the L2 CRC on the transmitted packet and adds its four bytes at the end of the packet. Any other packet field (such as XSUM or VLAN) must be calculated and inserted by the BMC (the integrated 10 GbE LAN controller does not change any field in the transmitted packet, other than adding padding and CRC bytes).

If the network link is down when the integrated 10 GbE LAN controller has received the last fragment of the packet from the BMC, it silently discards the packet. Note that any link down event during the transfer of any packet over the SMBus does not stop the operation since the integrated 10 GbE LAN controller waits for the last fragment to end to see whether the network link is up again.

9.5.8.1 Transmit Errors in Sequence Handling

Once a packet is transferred over the SMBus from the BMC to the integrated 10 GbE LAN controller, the *F* and *L* flags should follow specific rules. The *F* flag defines the first fragment of the packet; the *L* flag that the transaction contains the last fragment of the packet. Table 9-9 lists the different flag options in transmit packet transactions.

Table 9-9. Flag Options During Transmit Packet Transactions

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

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

9.5.8.2 TCO Command Aborted Flow

The integrated 10 GbE LAN controller indicates to the BMC an error or an abort condition by setting the *TCO Abort* bit in the general status. The integrated 10 GbE LAN controller might also be configured to send a notification to the BMC (see Section 9.5.11.1.3.3).

Following is a list of possible error and abort conditions:

- Any error in the SMBus protocol (NACK, SMBus timeouts, etc.).
- If the BMC does not respond until the notification timeout (programmed in the EEPROM) expires.
- Any error in compatibility between required protocols to specific functionality (for example, RX Enable command with a byte count not equal to 1/14, as defined in the command specification).

- If the integrated 10 GbE LAN controller does not have space to store the transmitted packet from the BMC (in its internal buffer space) before sending it to the link, the packet is discarded and the external BMC is notified via the *Abort* bit.
- Error in the *F/L* bit sequence during multi-fragment transactions.
- An internal reset to the integrated 10 GbE LAN controller's firmware.

9.5.9 SMBus Link State Control

While in SMBus mode, the default setting of the link is defined by the *Enable All PHYs in D3 N* bit in Common Firmware Parameters 1 NVM word.

When a channel is enabled through NVM setting or through the *RCV_EN* option of the Receive Enable Command, the link is established (if not already required for other purposes).

If the channel is disabled by clearing of the *RCV_EN* option, then the link may move back to the default defined by the Enable All PHYs in D3 N if not needed for other purposes.

Note: When the link is taken down due to *RCV_EN* being cleared the transmit traffic from BMC is also stopped.

Note: Before transitioning to D3 it is the responsibility of the driver to request the PHY to be active for wake-up activities.

9.5.10 SMBus ARP Transactions

All SMBus ARP transactions include the PEC byte.

9.5.10.1 Prepare to ARP

This command clears the *Address Resolved* flag (set to false). It does not affect the status or validity of the dynamic SMBus address and is used to inform all devices that the ARP master is starting the ARP process:

1	7	1	1	8	1	8	1	1
S	Slave Address	Wr	A	Command	A	PEC	A	P
	1100 001	0	0	0000 0001	0	[Data Dependent Value]	0	

9.5.10.2 Reset Device (General)

This command clears the *Address Resolved* flag (set to false). It does not affect the status or validity of the dynamic SMBus address.

1	7	1	1	8	1	8	1	1
S	Slave Address	Wr	A	Command	A	PEC	A	P
	1100 001	0	0	0000 0010	0	[Data Dependent Value]	0	

9.5.10.3 Reset Device (Directed)

The Command field is NACKed if bits 7:1 do not match the current SMBus address. This command clears the *Address Resolved* flag (set to false) and does not affect the status or validity of the dynamic SMBus address.

1	7	1	1	8	1	8	1	1
S	Slave Address	Wr	A	Command	A	PEC	A	P
	1100 001	0	0	Targeted Slave Address | 0	0	[Data Dependent Value]	0	

9.5.10.4 Assign Address

This command assigns SMBus address. The address and command bytes are always acknowledged.

The transaction is aborted (NACKed) immediately if any of the UDID bytes is different from the integrated 10 GbE LAN controller UDID bytes. If successful, the manageability system internally updates the SMBus address. This command also sets the *Address Resolved* flag (set to true).

1	7	1	1	8	1	8	1	
S	Slave Address	Wr	A	Command	A	Byte Count	A	· · ·
	1100 001	0	0	0000 0100	0	0001 0001	0	

8	1	8	1	8	1	8	1	
Data 1	A	Data 2	A	Data 3	A	Data 4	A	· · ·
UDID Byte 15 (MSB)	0	UDID Byte 14	0	UDID Byte 13	0	UDID Byte 12	0	

8	1	8	1	8	1	8	1	
Data 5	A	Data 6	A	Data 7	A	Data 8	A	· · ·
UDID Byte 11	0	UDID Byte 10	0	UDID Byte 9	0	UDID Byte 8	0	

8	1	8	1	8	1	
Data 9	A	Data 10	A	Data 11	A	· · ·
UDID Byte 7	0	UDID Byte 6	0	UDID Byte 5	0	

8	1	8	1	8	1	8	1	
Data 12	A	Data 13	A	Data 14	A	Data 15	A	· · ·
UDID Byte 4	0	UDID Byte 3	0	UDID Byte 2	0	UDID Byte 1	0	

8	1	8	1	8	1	1
Data 16	A	Data 17	A	PEC	A	P
UDID Byte 0 (LSB)	0	Assigned Address	0	[Data Dependent Value]	0	

9.5.10.5 Get UDID (General and Directed)

The general get UDID SMBus transaction supports a constant command value of 0x03 and, if directed, supports a Dynamic command value equal to the dynamic SMBus address.

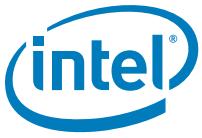
If the SMBus address has been resolved (*Address Resolved* flag set to true), the manageability system does not acknowledge (NACK) this transaction. If it's a General command, the manageability system always acknowledges (ACKs) as a directed transaction.

This command does not affect the status or validity of the dynamic SMBus address or the *Address Resolved* flag.

S	Slave Address	Wr	A	Command	A	S	• • •
	1100 001	0	0	See Below	0		

7	1	1	8	1	
Slave Address	Rd	A	Byte Count	A	· · ·
1100 001	1	0	0001 0001	0	

8	1	8	1	8	1	8	1	
Data 1	A	Data 2	A	Data 3	A	Data 4	A	· · ·



UDID Byte 15 (MSB)	0	UDID Byte 14	0	UDID Byte 13	0	UDID Byte 12	0	

8	**1**	**8**	**1**	**8**	**1**	**8**	**1**	
Data 5	A	Data 6	A	Data 7	A	Data 8	A	· · ·
UDID Byte 11	0	UDID Byte 10	0	UDID Byte 9	0	UDID Byte 8	0	

8	**1**	**8**	**1**	**8**	**1**	
Data 9	A	Data 10	A	Data 11	A	· · ·
UDID Byte 7	0	UDID Byte 6	0	UDID Byte 5	0	

8	**1**	**8**	**1**	**8**	**1**	**8**	**1**	
Data 12	A	Data 13	A	Data 14	A	Data 15	A	· · ·
UDID Byte 4	0	UDID Byte 3	0	UDID Byte 2	0	UDID Byte 1	0	

8	**1**	**8**	**1**	**8**	**1**	**1**
Data 16	A	Data 17	A	PEC	~Ã	P
UDID Byte 0 (LSB)	0	Device Slave Address	0	[Data Dependent Value]	1	

The Get UDID command depends on whether or not this is a Directed or General command.

The General Get UDID SMBus transaction supports a constant command value of 0x03.

The Directed Get UDID SMBus transaction supports a Dynamic command value equal to the dynamic SMBus address with the LSB bit set.

Note: Bit 0 (LSB) of Data byte 17 is always 1b.

9.5.11 SMBus Pass-Through Transactions

This section details commands (both read and write) that the integrated 10 GbE LAN controller SMBus interface supports for pass-through.

9.5.11.1 Write SMBus Transactions

This section details the commands that the BMC can send to the integrated 10 GbE LAN controller over the SMBus interface. The SMBus write transactions table lists the different SMBus write transactions supported by the integrated 10 GbE LAN controller.

TCO Command	Transaction	Command	Fragmentation	Section
Transmit Packet	Block Write	First: 0x84 Middle: 0x04 Last: 0x44	Multiple	9.5.11.1.1
Transmit Packet	Block Write	Single: 0xC4	Single	9.5.11.1.1
Request Status	Block Write	Single: 0xDD	Single	9.5.11.1.2
Receive Enable	Block Write	Single: 0xCA	Single	9.5.11.1.3
Force TCO	Block Write	Single: 0xCF	Single	9.5.11.1.4
Management Control	Block Write	Single: 0xC1	Single	9.5.11.1.5
Update MNG RCV Filter Parameters	Block Write	Single: 0xCC	Single	9.5.11.1.6
Set Common Filters	Block Write	Single: 0xC2	Single	9.5.11.1.7
Clear All Filters	Block Write	Single: 0xC3	Single	9.5.11.1.8

9.5.11.1.1 Transmit Packet Command

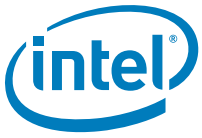
The Transmit Packet command behavior is detailed in Section 9.5.8. The Transmit Packet fragments have the following format.

The payload length is limited to the maximum payload length set in the NVM. If the overall packet length is bigger than 1536 bytes, the packet is silently discarded.

Function	Command	Byte Count	Data 1	...	Data N
Transmit first fragment	0x84	N	Packet data MSB	...	Packet data LSB
Transmit middle fragment	0x04				
Transmit last fragment	0x44				
Transmit single fragment	0xC4				

9.5.11.1.2 Request Status Command

An external BMC can initiate a request to read the integrated 10 GbE LAN controller manageability status by sending a Request Status command. When received, the integrated 10 GbE LAN controller initiates a notification to an external BMC when status is ready. After this, the external controller will be able to read the status, by issuing a read status command (see Section 9.5.11.2.2).



The format is as follows:

Function	Command	Byte Count	Data 1
Request Status	0xDD	1	0

9.5.11.1.3 Receive Enable Command

The Receive Enable command is a single fragment command used to configure the integrated 10 GbE LAN controller. This command has two formats: short, 1-byte legacy format (providing backward compatibility with previous components) and long, 14-byte advanced format (allowing greater configuration capabilities). The Receive Enable command format is as follows:

Function	CMD	Byte Count	Data 1	Data 2	...	Data 7	Data 8	...	Data 11	Data 12	Data 13	Data 14
Legacy Receive Enable	0xCA	1	Receive Control Byte	-	...	-	-	...	-	-	-	-
Advanced Receive Enable		14 (0x0E)		MAC Addr MSB		MAC Addr LSB	IP Addr MSB		IP Addr LSB	BMC SMBus Addr	I/F Data Byte	Alert Value Byte

Field	Bit(s)	Description
RCV_EN	0	Receive TCO Enable. 0b= Disable receive TCO packets. 1b= Enable Receive TCO packets. Setting this bit enables all manageability receive filtering operations. Enabling specific filters is done via the NVM or through special configuration commands. **Note:** When the *RCV_EN* bit is cleared, all receive TCO functionality is disabled, not just the packets that are directed to the BMC (also auto ARP packets).
RCV_ALL	1	Receive All Enable. 0b = Disable receiving all packets. 1b = Enable receiving all packets. Forwards all packets received over the wire that passed L2 filtering to the external BMC. This flag has no effect if bit 0 (Enable TCO packets) is disabled.
EN_STA	2	Enable Status Reporting. 0b = Disable status reporting. 1b = Enable status reporting.

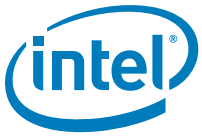
EN_ARP_RES	3	Enable ARP Response. 0b = Disable the integrated 10 GbE LAN controller ARP response. The integrated 10 GbE LAN controller treats ARP packets as any other packet, for example, packet is forwarded to the BMC if it passed other (non-ARP) filtering. 1b = Enable the integrated 10 GbE LAN controller ARP response. The integrated 10 GbE LAN controller automatically responds to all received ARP requests that match the IP address programmed by the BMC. The BMC IP address is provided as part of the Receive Enable message (bytes 8:11). If a short version of the command is used, the integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller uses the IP address configured in the NVM as ARP Response IPv4 Address in the pass-through LAN configuration structure. If the *CBDM* bit is set, the integrated 10 GbE LAN controller uses the BMC dedicated MAC address in ARP response packets. If the CBDM bit is not set, the BMC uses the host MAC address. When the ARP off load feature is activated, the integrated 10 GbE LAN controller uses the following registers to filter the ARP traffic addressed to the BMC. BMC should not modify these registers: • Manageability Decision Filter – MDEF6 (and corresponding bit 6 in Management Only traffic Register – *MNGONLY*). IPv4 address - MIPAF4[3]
NM	5:4	Notification Method. Define the notification method the integrated 10 GbE LAN controller uses. 00b = SMBUS Alert. 01b= Asynchronous notify. 10b = Direct receive. 11b= Not supported. ***Note:*** Changing the notification method in any port will update the notification method of all ports.
Reserved	6	Reserved. Must be set to 1b.
CBDM	7	Configure the BMC Dedicated MAC Address. **Note:** This bit should be 0b when the *RCV_EN* bit (bit 0) is not set. 0b = The integrated 10 GbE LAN controller shares the MAC address for MNG traffic with the host MAC address, which is specified in NVM words 0x0-0x2. 1b= The integrated 10 GbE LAN controller uses the BMC dedicated MAC address as a filter for incoming receive packets. The BMC MAC address is set in bytes 2-7 in this command. If a short version of the command is used, the integrated 10 GbE LAN controller uses the MAC address configured in the most recent long version of the command in which the *CBDM* bit was set. When the dedicated MAC address feature is activated, the integrated 10 GbE LAN controller uses the following registers to filter in all the traffic addressed to the BMC MAC. BMC should not modify these registers: • Manageability Decision Filter – MDEF7 (and corresponding bit 7 in Management Only traffic Register – *MNGONLY*) • Manageability MAC Address Low – *MMAL[3]* • Manageability MAC Address High – *MMAH[3]* ***Note:*** When the dedicated MAC address feature is cleared, these registers are not programmed and the BMC may use other filters to enforce MAC filtering using the Update Management Receive Filter Parameters command.

9.5.11.1.3.1 Management MAC Address (Data Bytes 7:2)

Ignored if the *CBDM* bit is not set. This MAC address is used to configure the dedicated MAC address. In addition, it is used in the ARP response packet when the EN_ARP_RES bit is set. This MAC address is also used when *CBDM* bit is set in subsequent short versions of this command.

9.5.11.1.3.2 Management IP Address (Data Bytes 11:8)

This IP address is used to filter ARP request packets.



9.5.11.1.3.3 Asynchronous Notification SMBus Address (Data Byte 12)

This address is used for the asynchronous notification SMBus transaction and for direct receive. The SMBus address is stored in bit 7:1 of this byte. Bit 0 is always 0.

9.5.11.1.3.4 Interface Data (Data Byte 13)

Interface data byte used in asynchronous notification.

9.5.11.1.3.5 Alert Value Data (Data Byte 14)

Alert Value data byte used in asynchronous notification.

9.5.11.1.4 Force TCO Command

This command causes the integrated 10 GbE LAN controller to perform a TCO reset, TCO isolate, or Firmware Reset

TCO Reset: The force TCO reset clears the data path (Rx/Tx) of the integrated 10 GbE LAN controller to enable the BMC to transmit/receive packets through the integrated 10 GbE LAN controller by assertion of a global reset. Force TCO reset is asserted only to the port related to the SMBus address the command. This command should only be used when the BMC is unable to transmit receive and suspects that the integrated 10 GbE LAN controller is inoperable. The command also causes the LAN device driver to unload. It is recommended to perform a system restart to resume normal operation.

Firmware Reset: This command causes re-initialization of all the embedded controller functions and a re-load of related NVM words. A firmware reset is achieved by setting the *GSCR.SET_FWRST Aux* bit.

The integrated 10 GbE LAN controller considers the Force TCO reset command as an indication that the operating system is hung. The Force TCO command format is as follows:

Function	Command	Byte Count	Data 1
Force TCO Reset	0xCF	1	TCO Mode

Where TCO mode is:

Field	Bit(s)	Description
DO_TCO_RST	0	Perform TCO Reset. 0b= Do nothing. 1b= Perform TCO reset.

Field	Bit(s)	Description
DO_TCO_ISOLATE[1]	1	Do TCO Isolate 0b = Enable PCIe write access to LAN port. 1b = Isolate Host PCIe write operation to the port. Note: Should be used for debug only.
RESET_MGMT	2	Reset manageability; re-load manageability NVM words. 0b = Do nothing. 1b = Issue firmware reset to manageability. Setting this bit generates a one-time firmware reset. Following the reset, management related data from NVM is loaded.
Reserved	7:3	Reserved (set to 0x00).

1. TCO isolate host write operation enabled in NVM.

Note: Only one of the fields should be set in a given command. Setting more than one field might yield unexpected results.

9.5.11.1.5 Management Control

This command is used to set generic manageability parameters. The parameters list is shown in Table 9-10. The command is 0xC1 stating that it is a Management Control command. The first data byte is the parameter number and the data afterwards (length and content) are parameter specific as shown in Management Control Command Parameters/Content.

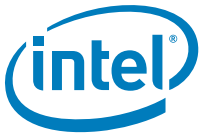
Note: If the parameter that the BMC sets is not supported by the integrated 10 GbE LAN controller. The integrated 10 GbE LAN controller does not NACK the transaction. After the transaction ends, the integrated 10 GbE LAN controller discards the data and asserts a transaction abort status.

The Management Control command format is as follows:

Function	Command	Byte Count	Data 1	Data 2	...	Data N
Management Control	0xC1	N	Parameter Number	Parameter Dependent		

Table 9-10. Management Control Command Parameters/Content

Parameter	#	Parameter Data
Keep PHY Link Up	0x00	A single byte parameter: Data 2: Bit 0= Set to indicate that the PHY link for this port should be kept up throughout system resets. This is useful when the server is reset and the BMC needs to keep connectivity for a manageability session. Bit [7:1] Reserved. 0b= Disabled. 1b= Enabled.



9.5.11.1.6 Update Management Receive Filter Parameters

This command is used to set the manageability receive filters parameters. The command is 0xCC. The first data byte is the parameter number and the data that follows (length and content) are parameter specific as listed in management RCV filter parameters.

If the parameter that the BMC sets is not supported by the integrated 10 GbE LAN controller, then the integrated 10 GbE LAN controller does not NACK the transaction. After the transaction ends, the integrated 10 GbE LAN controller discards the data and asserts a transaction abort status.

The update management RCV receive filter parameters command format is as follows:

Function	Command	Byte Count	Data 1	Data 2	...	Data N
Update Manageability Filter Parameters	0xCC	N	Parameter Number	Parameter Dependent		

Table 9-11 lists the different parameters and their content.

Table 9-11. Management Receive Filter Parameters

Parameter	Number	Parameter Data
Filters Enables	0x1	Defines the generic filters configuration. The structure of this parameter is four bytes as the Manageability Control (MANC) register **Note:** The general filter enable is in the Receive Enable command that enables receive filtering.
MNGONLY configuration	0xF	This parameter defines which of the packets types identified as manageability packets in the receive path will never be directed to the host memory. Data 2:5: MNGONLY register bytes - Data 2 is the MSB.
Flex Filter 0 Enable Mask and Length	0x10	Flex Filter 0 Mask. Data 17:2 = Mask. Bit 0 in data 2 is the first bit of the mask. Data 19:18 = Reserved. Should be set to 00b. Date 20 = Flexible filter length.
Flex Filter 0 Data	0x11	Data 2 — Group of flex filter's bytes: 0x0 = bytes 0-29. 0x1 = bytes 30-59. 0x2 = bytes 60-89. 0x3 = bytes 90-119. 0x4 = bytes 120-127. Data 3:32 = Flex filter data bytes. Data 3 is LSB. Group's length is not a mandatory 30 bytes; it might vary according to filter's length and must NOT be padded by zeros.
Decision Filters	0x61	This command is obsolete and should not be used anymore. Please use 0x68 instead.
VLAN Filters	0x62	Three bytes are required to load the VLAN tag filters. Data 2: VLAN filter number. Data 3: MSB of VLAN filter. Data 4: LSB of VLAN filter.
Flex Port Filters	0x63	Three to four bytes are required to load the manageability flex port filters. Data 2: Flex port filter number. Data 3: MSB of flex port filter. Data 4: LSB of flex port filter.

Table 9-11. Management Receive Filter Parameters (Continued)

IPv4 Filters	0x64	Five bytes are required to load the IPv4 address filter. Data 2: IPv4 address filter number (3:0). Data 3: LSB of IPv4 address filter. … Data 6: MSB of IPv4 address filter.
IPv6 Filters	0x65	17 bytes are required to load the IPv6 address filter. Data 2 — IPv6 address filter number (3:0). Data 3 — LSB of IPv6 address filter. … Data 18 — MSB of IPv6 address filter.
MAC Filters	0x66	Seven bytes are required to load the MAC address filters. Data 2 — MAC address filters pair number (3:0). Data 3 — MSB of MAC address. … Data 8: LSB of MAC address.
EtherType Filters	0x67	5 bytes to load Ethertype Filters (METF). Data 2 — METF filter index (valid values are 0, 1, 2, 3). Data 3 — MSB of METF. ... Data 6 — LSB of METF.
Extended Decision Filter	0x68	9 bytes to load the extended decision filters (MDEF_EXT & MDEF). Data 2 — MDEF filter index (valid values are 0...5). Data 3 — MSB of MDEF_EXT (DecisionFilter1). .... Data 6 — LSB of MDEF_EXT (DecisionFilter1). Data 7 — MSB of MDEF (DecisionFilter0). .... Data 10 — LSB of MDEF (DecisionFilter0). The command must overwrite any previously stored value.

Table 9-12. Filter Enable Parameters

Bit	Name	Description
16:0	Reserved	Reserved.
17	RCV_TCO_EN	Receive TCO Packets Enabled. When this bit is set it enables the receive flow to the manageability block. This bit should be set only if at least one of EN_BMC2OS or EN_BMC2NET bits are set. This bit is usually set using the receive enable command (see Section 9.5.11.1.3).
18	Reserved	Reserved.
22:19	Reserved	Reserved.
23	Enable Xsum Filtering to MNG	When this bit is set, only packets that pass the L3 and L4 checksum are send to the manageability block.
24	EN_IPV4_FILTER	Enable IPv4 address Filters - 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

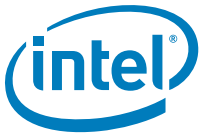


Table 9-12. Filter Enable Parameters (Continued)

Bit	Name	Description
25	FIXED_NET_TYPE	Fixed net type: If set, only packets matching the net type defined by the NET_TYPE field passes to manageability. Otherwise, both tagged and un-tagged packets can be forwarded to the manageability engine.
26	NET_TYPE	NET TYPE: 0b = pass only un-tagged packets. 1b = pass only VLAN tagged packets. Valid only if FIXED_NET_TYPE is set.
31:27	Reserved	Reserved.

9.5.11.1.7 Set Common Filters Command

The Set Common Filters command is a single fragment command capable of configuring the most common filters.

Note: If this command is used, all the other commands that programs forwarding filters should not be used (apart from the Clear All Filters command). When this command is received, an implied Clear All Filters command is done before the application of this command.

The Set Common Filters command has two possible formats:

IPv4 Format:

Function	Command	Byte Count	Data 1	Data 2:4	5:10	Data 11	Data 12	Data 13	Data 14:17
Set Common Filters	0xC2	17	Opcode = 0	Receive Control - see Table 9-13	MAC Address	BMC Alert Address	Interface Data Byte	Alert Value Byte	IPv4 Address

IPv6 Format:

Function	Command	Byte Count	Data 1	Data 2:4	5:10	Data 11	Data 12	Data 13	Data 14:29
Set Common Filters	0xC2	29	Opcode = 0	Receive Control - see Table 9-13	MAC Address	BMC Alert Address	Interface Data Byte	Alert Value Byte	IPv6 Address

Table 9-13. Set Common Filters Receive Control Bytes

Byte	Bit	Field	Description
1	0	RCV_EN	Receive TCO Packets Enabled. When this bit is set it enables the receive flow to the manageability block. This bit should be set only if at least one of EN_BMC2O or EN_BMC2NET bits are set.
	1	EN_STA	Enable Status Reporting. 0b = Disable status reporting. 1b = Enable status reporting.
	2	Auto ARP	Automatically respond to ARP packets. Ignored in IPv6 mode. If this bit is set, broadcast ARP packets will be handled by the integrated 10 GbE LAN controller and ARP requests to the IP address set in the command will be responded. ***Note:*** Mutually exclusive to Configure ARP/ Neighborhood Filter bit. If this bit is set, the IP address must be valid and contain the IP address of the MC. This bit is ignored if RCV_EN is cleared. ***Note:*** When set, ARP requests to the BMC IP defined below (unicast or Broadcast) will be sent to the internal Firmware for processing. ARP response will be dropped.
	3	Enable Xsum Filtering to MNG	When this bit is set, only packets that pass the L3 and L4 checksum are send to the manageability block. This bit is ignored if RCV_EN is cleared.
	5:4	Reserved	Reserved.
	7:6	Notification Method	Notification Method. Define the notification method the integrated 10 GbE LAN controller uses. 00b = SMBUS Alert. 01b = Asynchronous notify. 10b = Direct receive. 11b = Not supported.

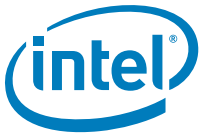


Table 9-13. Set Common Filters Receive Control Bytes (Continued)

Byte	Bit	Field	Description
2	8	CBDM	Configure the BMC Dedicated MAC Address. 0b= The integrated 10 GbE LAN controller shares the MAC address for MNG traffic with the host MAC address, which is specified in RAH/ RAL[0] registers 1b= The integrated 10 GbE LAN controller uses the BMC dedicated MAC address as a filter for incoming receive packets. The BMC MAC address is set in bytes 5-10 in this command. This bit is ignored if RCV_EN is cleared. If this bit is cleared, at least one of bits 9,10 or 11 must be set. If only bit 9 is set, the IP address should be different than the address used by the host.
	9	Configure IP Address Filter	Automatically configure an IP Address Filter. If this bit is set, only packets matching this IP address and the MAC address defined by the CBDM bit is forwarded. This bit is ignored if RCV_EN is cleared
	10	Configure RMCP 26Fh Filter	Automatically configure standard IPMI port 0x26F filters. If this bit is set, only packets matching this port and the MAC address defined by the CBDM bit is forwarded. If the Configure IP address *Filter* bits is set, only packets matching the IP address and this port is forwarded. The other port enable bit (11) might add additional forwarding condition. This bit is ignored if RCV_EN is cleared.
	11	Configure RMCP 298h Filter	Automatically configure standard IPMI port 0x298 filter. If this bit is set, only packets matching this port and the MAC address defined by the CBDM bit is forwarded. If the Configure IP address *Filter* bits is set, only packets matching the IP address and this port is forwarded. The other port enable bit (10) might add additional forwarding condition. This bit is ignored if RCV_EN is cleared.
	12	Configure ARP/ Neighborhood Filter	Automatically configure filters to enable this traffic to the BMC (mutually exclusive to Auto ARP bit). If this bit is set, broadcast or unicast ARP packets are forwarded to the BMC. These packets might also be sent to the host. In IPv4 mode, setting this bit enables forwarding of broadcast or unicast ARP request and response. If IP address is set, only request or response to this address is forwarded. In IPv6 mode, setting this bit enables forwarding of all types of neighbor discovery and MLD ICMPv6 packet types: • 0x86 (134d) - Router advertisement. • 0x87 (135d) - Neighbor solicitation. • 0x88 (136d) - Neighbor advertisement. • 0x89 (137d) - Redirect. • 0x82 (130d) - MLD query. • 0x83 (131d) - MLDv1 report. • 0x84 (132d) - MLD done. • 0x8F (143d) - MLDv2 report. This bit is ignored if RCV_EN is cleared.
	13	Configure DHCP port 44h Filter (DHCP server packets)	Automatically configure DHCP port 44 filter to the BMC. If this bit is set, broadcast packets matching this port is forwarded. Otherwise, this port is not added to the broadcast filtering option. This bit is ignored if RCV_EN is cleared or in IPv6 mode.
	15:14	Reserved	Reserved.
3	16	Disable Host ARP	Configure ARP requests and network neighborhood packets not to go to the host. This bit should be cleared in regular operation. Ignored if both bit 12 and bit 2 are cleared or if RCV_EN is cleared.
	17	Disable Host DHCP	Configure DHCP packets (port 0x44) not to go to the host. This bit should be cleared in regular operation. Ignored if bit 13 is cleared, RCV_EN is cleared, or in IPv6 mode.
	24:18	Reserved	Reserved.

9.5.11.1.8 Clear all Filters Command

The Clear all Filters command is a single fragment command capable of clearing all the receive filters currently programmed for manageability traffic.

Function	Command	Byte Count	Data
Clear all Filters	0xC3	1	0x00

9.5.11.2 Read SMBus Transactions

This section details the pass-through read transactions that the BMC can send to the integrated 10 GbE LAN controller over SMBus.

SMBus read transactions lists the different SMBus read transactions supported by the integrated 10 GbE LAN controller. All the read transactions are compatible with SMBus read block protocol format.

Table 9-14. SMBus Read Transactions

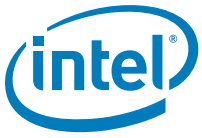
TCO Command	Transaction	Write Command[1]	Read Command	Opcode	Fragments	Section
Receive TCO Packet	Block Read	N/A	0xD0 or 0xC0	First: 0x90 Middle: 0x10 Last[2]: 0x50	Multiple	9.5.11.2.1
Read Status	Block Read	N/A	0xD0 or 0xC0 or 0xDE	Single: 0xDD	Single	9.5.11.2.2
Get System MAC Address	Block Read	N/A	0xD4	Single: 0xD4	Single	9.5.11.2.3
Read Management Parameters	Block Read	0xC1	0xD1	Single: 0xD1	Single	9.5.11.2.4
Read Management RCV Filter Parameters	Block Read	0xCC	0xCD	Single: 0xCD	Single	9.5.11.2.5
Get Controller Information	Block Read	0xD5	0xD5	Single: 0xD5	Single	9.5.11.2.6
Get Common filters	Block Read	0xD3	0xD3	Single: 0xD3	Single	9.5.11.2.7
Read Receive Enable Configuration	Block Read	N/A	0xDA	Single: 0xDA	Single	9.5.11.2.8

1. In some commands, a preliminary Write command is sent to signal firmware to prepare the data for the upcoming Read command. This column describes the OpCode used for the Write command.
2. The last fragment of the receive packet is the packet status.

0xC0 or 0xD0 commands are used for more than one payload. If BMC issues these read commands, and the integrated 10 GbE LAN controller has no pending data to transfer, it always returns as default opcode 0xDD with the integrated 10 GbE LAN controller status and does not NACK the transaction.

If an SMBus Quick Read command is received, it is handled as a the integrated 10 GbE LAN controller Request Status command (See Section 9.5.11.1.2 for details).

9.5.11.2.1 Receive TCO LAN Packet Transaction



The BMC uses this command to read packets received on the LAN and its status. When the integrated 10 GbE LAN controller has a packet to deliver to the BMC, it asserts the SMBus notification for the BMC to read the data (or direct receive). Upon receiving notification of the arrival of a LAN receive packet, the BMC begins issuing a receive TCO packet command using the block read protocol.

A packet can be transmitted to the BMC in at least two fragments (at least one for the packet data and one for the packet status). As a result, BMC should follow the *F* and *L* bit of the OpCode.

The OpCode can have these values:

- 0x90 — First fragment
- 0x10 — Middle fragment
- When the OpCode is 0x50, this indicates the last fragment of the packet, which contains packet status.

If a notification timeout is defined (in the NVM) and the BMC does not finish reading the entire packet within the timeout period, since the packet has arrived, the packet is silently discarded. The time spent in ARA cycle or in reading the packet is not counted by the timeout counter.

Following is the receive TCO packet format and the data format returned from the integrated 10 GbE LAN controller.

Function	Command
Receive TCO Packet	0xC0 or 0xD0

Function	Byte Count	Data 1 (Op-Code)	Data 2	...	Data N
Receive TCO First Fragment	N	0x90	Packet Data Byte	...	Packet Data Byte
Receive TCO Middle Fragment		0x10			
Receive TCO Last Fragment	9 (0x9)	0x50	See Section 9.5.11.2.1.1		

9.5.11.2.1.1 Receive TCO LAN Status Payload Transaction

This transaction is the last transaction that the integrated 10 GbE LAN controller issues when a packet received from the LAN is transferred to the BMC. The transaction contains the status of the received packet.

The format of the status transaction is as follows:

Function	Byte Count	Data 1 (OpCode)	Data 2 – Data 17 (Status Data)
Receive TCO Long Status	9 (0x9)	0x50	See Below

The status is 8 bytes where byte 0 (bits 7:0) is set in Data 2 of the status and byte 7 in Data 9 of the status. Table 9-15 lists the content of the status data.

Table 9-15. TCO LAN Packet Status Data

Name	Bits	Description
Packet Length	13:0	Packet length including CRC, only 14 LSB bits.
Reserved	15:14	Reserved.
Packet status	31:16	See Table 9-16.
VLAN	47:32	The two bytes of the VLAN header tag.
MNG status	63:48	See Table 9-17. This field should be ignored if receive is not enabled,

Table 9-16. Packet Status Info

Field	Bit(s)	Description
Reserved	15:4	Reserved.
LAN#	3:2	Indicates the source port of the packet. 00b = Port 0. 01b = Port 1. 10b - 11b = Reserved.
VP	1	VLAN stripped (indicates if the VLAN is part of the packet, or was removed).
CRC stripped	0	Insertion of CRC is needed.

Table 9-17. MNG Status

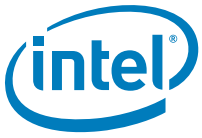
Name	Bits	Description
Reserved	15:9	Reserved.
Decision Filter match	8	Set when there is a match to one of the Decision filters.
Decision Filter index	7:4	Indicates which of the decision filters match the packet. (allows for up to 16 filters - although only 8 are currently supported).
MNG VLAN Address Match	3	Set when the MNG packet matches one of the MNG VLAN filters.
Pass MNG VLAN Filter Index	2:0	Indicates which of the VLAN filters match the packet.

9.5.11.2.2 Read Status Command

The BMC should use this command after receiving a notification from the integrated 10 GbE LAN controller (such as SMBus Alert). The integrated 10 GbE LAN controller also sends a notification to the BMC in either of the following two cases:

- The BMC asserts a request for reading the status.
- The integrated 10 GbE LAN controller detects a change in one of the Status Data 1 bits (and was set to send status to the BMC on status change) in the Receive Enable command.

Note: Commands 0xC0/0xD0 are for backward compatibility and can be used for other payloads. The integrated 10 GbE LAN controller defines these commands in the OpCode as well as which payload this transaction is. When the 0XDE command is set, the integrated 10 GbE LAN controller always returns opcode 0XDD with the integrated 10 GbE LAN controller status. The BMC reads the event causing the notification, using the Read Status command as follows.



The integrated 10 GbE LAN controller response to one of the commands (0xC0 or 0xD0) in a given time as defined in the SMBus Notification Timeout and Flags word in the NVM.

Function	Command
Read Status	0XC0 or 0XD0 or 0XDE

Function	Byte Count	Data 1 (OpCode)	Data 2 (Status Data 1)	Data 3 (Status Data 2)
Receive TCO Partial Status	3	0XDD	See as follows	

This command can also be executed using the I^2C quick read format as follows:

1	7	1	1	8	1	8	1	8	1	1
Start	Slave Address	Rd	Ack	Byte Count	Ack	Status Data 1	Ack	Status Data 2	Ack	Stop
		1	0	0000 0002	0		0		1	

Table 9-18 lists the status data byte 1 parameters.

Table 9-18. Status Data Byte 1

Bit	Name	Description
7	LAN Port	LAN port: defines the port that sent status.
6	TCO Command Aborted	1b = A TCO command abort event occurred since the last read status cycle. 0b = A TCO command abort event did not occur since the last read status cycle.
5	Link Status Indication	0b = LAN link down. 1b = LAN link up.
4	PHY Link Forced Up	Contains the value of the *PHY_Link_Up* bit. When set, indicates that the PHY link is configured to keep the link up.
3	Initialization Indication	0b = An NVM reload event has not occurred since the last Read Status cycle. 1b = An NVM reload event has occurred since the last Read Status cycle[1].
2	Reserved	Reserved.
1:0	Power State	00b = Dr state. 01b = D0u state. 10b = D0 state. 11b = D3 state.

1. This indication is asserted when the integrated 10 GbE LAN controller manageability block reloads the NVM and its internal database is updated to the NVM default values. This is an indication that the external BMC should reconfigure the integrated 10 GbE LAN controller, if other values other than the NVM default should be configured.

Status data byte 2 is used by the BMC to indicate whether the LAN device driver is up and running.

The LAN device driver valid indication is a bit set by the LAN device driver during initialization; the bit is cleared when the LAN device driver enters a Dx state or is cleared by the hardware on a PCI reset.

Bits 2 and 1 indicate that the LAN device driver is stuck. Bit 2 indicates whether the interrupt line of the LAN function is asserted. Bit 1 indicates whether the LAN device driver dealt with the interrupt line before the last Read Status cycle. Table 9-19 lists status data byte 2.

Table 9-19. Status Data Byte 2

Bit	Name	Description
7:4	Reserved	Reserved.
3	Driver Valid Indication	0b = LAN driver is not alive. 1b = LAN driver is alive.
2:1	Reserved	Reserved.
0	Reserved	Reserved.

Table 9-20 lists the possible values of bits 2 and 1 and what the BMC can assume from the bits:

Table 9-20. Status Data Byte 2 (Bits 2 and 1)

Previous	Current	Description
Don't Care	00b	Interrupt is not pending (OK).
00b	01b	New interrupt is asserted (OK).
10b	01b	New interrupt is asserted (OK).
11b	01b	Interrupt is waiting for reading (OK).
01b	01b	Interrupt is waiting for reading by the driver for more than one read cycle (not OK). Possible drive hang state.
Don't Care	11b	Previous interrupt was read and current interrupt is pending (OK).
Don't Care	10b	Interrupt is not pending (OK).

BMC reads should consider the time it takes for the LAN device driver to deal with the interrupt (in µs). Note that excessive reads by the BMC can give false indications.

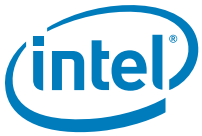
9.5.11.2.3 Get System MAC Address

The Get System MAC Address returns the system MAC address over to the SMBus. This command is a single-fragment Read Block transaction that returns the following the MAC address configured in RAL0, RAH0 registers.

Get system MAC address format:

Function	Command
Get System MAC Address	0xD4

Data returned from the integrated 10 GbE LAN controller:



Function	Byte Count	Data 1 (Op-Code)	Data 2	...	Data 7
Get System MAC Address	7	0xD4	MAC Address MSB	...	MAC Address LSB

9.5.11.2.4 Read Management Parameters

In order to read the management parameters the BMC should execute two SMBus transactions. The first transaction is a block write that sets the parameter that the BMC wants to read. The second transaction is block read that reads the parameter.

Block write transaction:

Function	Command	Byte Count	Data 1
Management Control Request	0xC1	1	Parameter Number

Following the block write the BMC should issue a block read that reads the parameter that was set in the Block Write command:

Function	Command
Read Management Parameter	0xD1

Data returned:

Function	Byte Count	Data 1 (Op-Code)	Data 2	Data 3	...	Data N
Read Management Parameter	N	0xD1	Parameter Number	Parameter Dependent		

The returned data is in the same format of the BMC command.

The returned data is as follows:

Parameter	#	Parameter Data
Keep PHY Link Up	0x00	A single byte parameter: Data 2 — Bit 0 set to indicate that the PHY link for this port should be kept up. Sets the keep_PHY_link_up bit. When cleared, clears the keep_PHY_link_up bit. Bit [7:1] Reserved.
Wrong Parameter Request	0xFE	Returned by the integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller.
Integrated 10 GbE LAN Controller is Not Ready	0xFF	Returned by the integrated 10 GbE LAN controller only, on read parameters command when the data that should have been read is not ready. This parameter has no data. The BMC should retry the read transaction. This value is also returned if the byte count is illegal or if the read command is not preceded by a write command.

The parameter that is returned might not be the parameter requested by the BMC. The BMC should verify the parameter number (default parameter to be returned is 0x1).

If the parameter number is 0xFF, it means that the data that was requested from the integrated 10 GbE LAN controller is not ready yet, or that the adequate Write command was not given.The BMC should retry the read transaction or send the write transaction.

It is responsibility of the BMC to follow the procedure previously defined. When the BMC sends a Block Read command (as previously described) that is not preceded by a Block Write command with bytecount=1, the integrated 10 GbE LAN controller sets the parameter number in the read block transaction to be 0xFF.

9.5.11.2.5 Read Management Receive Filter Parameters

In order to read the management receive filter parameters, the BMC should execute two SMBus transactions. The first transaction is a block write that sets the parameter that the BMC wants to read. The second transaction is block read that read the parameter.

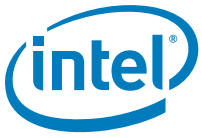
Block write transaction:

Function	Command	Byte Count	Data 1	Data 2
Update MNG RCV Filter Parameters	0xCC	1 or 2	Parameter Number	Parameter Data

The different parameters supported for this command are the same as the parameters supported for update management receive filter parameters.

Following the block write the BMC should issue a block read that reads the parameter that was set in the Block Write command:

Function	Command
Request MNG RCV Filter Parameters	0xCD



Data returned from the integrated 10 GbE LAN controller:

Function	Byte Count	Data 1 (Op-Code)	Data 2	Data 3	...	Data N
Read MNG RCV Filter Parameters	N	0xCD	Parameter Number	Parameter Dependent		

The parameter that is returned might not be the parameter requested by the BMC. The BMC should verify the parameter number (default parameter to be returned is 0x1).

If the parameter number is 0xFF, it means that the data that was requested from the integrated 10 GbE LAN controller is not ready yet or that the adequate Write command was not given. The BMC should retry the read transaction or send the write transaction.

It is BMC responsibility to follow the procedure previously defined. When the BMC sends a Block Read command (as previously described) that is not preceded by a Block Write command with bytecount=1, the integrated 10 GbE LAN controller sets the parameter number in the read block transaction to be 0xFF.

Parameter	#	Parameter Data
Filters Enable	0x01	None.
MNGONLY Configuration	0x0F	None.
Flex Filter Enable Mask and Length	0x10	None.
Flex Filter Data	0x11	Data 2 — Group of Flex Filter's Bytes: 0x0 = bytes 0-29. 0x1 = bytes 30-59. 0x2 = bytes 60-89. 0x3 = bytes 90-119. 0x4 = bytes 120-127.
Decision Filters	0x61	This command is obsolete. Please use 0x68 instead.
VLAN Filters	0x62	One byte to define the accessed VLAN tag filter (MAVTV). Data 2 — VLAN Filter number.
Flex Ports Filters	0x63	One byte to define the accessed manageability flex port filter (MFUTP). Data 2 — Flex Port Filter number.
IPv4 Filter	0x64	One byte to define the accessed IPv4 address filter (MIPAF4). Data 2 — IPv4 address filter number.
IPv6 Filters	0x65	One byte to define the accessed IPv6 address filter (MIPAF6). Data 2 — Pv6 address filter number.
MAC Filters	0x66	One byte to define the accessed MAC address filters pair (MMAL, MMAH). Data 2 — MAC address filters pair number (0-3).
EtherType Filters	0x67	1 byte to define Ethertype filters (METF). Data 2 — METF filter index (valid values are 0 - 3).

Extended Decision Filter	0x68	1 byte to define the extended decisions filters (MDEF_EXT & MDEF). Data 2 — MDEF filter index (valid values are 0 - 5).
Wrong parameter request	0xFE	Returned by the integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller.
Integrated 10 GbE LAN Controller is not ready	0xFF	Returned by the integrated 10 GbE LAN controller only, on read parameters command when the data that should have been read is not ready. This parameter has no data. This value is also returned if the byte count is illegal or if the read command is not preceded by a write command.

9.5.11.2.6 Get Controller Information Command

The BMC uses this command to get the controller identification. Each parameter is returned using a different OpCode.

In order to read the controller information, the BMC should execute two SMBus transactions. The first transaction is a block write that sets the parameter that the BMC wants to read. The second transaction is block read that read the parameter.

Block write transaction:

Function	Command	Byte Count	Data 1 (OpCode)
Get Controller Information	0xD5	1	Parameter Number

Following the block write the BMC should issue a block read that reads the parameter that was set in the Block Write command:

Function	Command
Get Controller Information	0xD5

Data returned from the integrated 10 GbE LAN controller:

Function	Byte Count	Command	Data 2 (Opcode)	Data 3 -n
Get Controller Information	Per Table 9-22	0xD5	Per Table 9-22	See Table 9-22 for the data for each OpCode.

Table 9-22. Get Controller Information Data

Opcode	Byte Count	Description	Notes
0x00	5	Data 4:3: Generic Device ID Data 5: Silicon Revision (RevID)	Device ID: This is the HW default value, not any value programmed via NVM. RevID: Read from setID message result as read from CFG space offset 0x8 using FW_PCI_CIAA/FW_PCI_CIAD registers XORED with the value in PCI_REVID.NVM_REVID.
0x0B	4	Data 4:3 NVM Image version.	

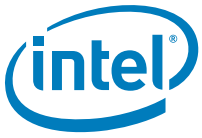


Table 9-22. Get Controller Information Data (Continued)

0x0C	6	Data 6:3: Firmware ROM Internal version.	
0x0D	6	Data 6:3: Firmware Flash Internal version.	
0x0E	4	Data 4:3: PXE Firmware version.	MajorVersion.MinorVersion.Build.SubBuild.
0x0F	4	Data 4:3: iSCSI Firmware version.	
0x10	4	Data 4:3: uEFI Firmware version.	
0x16	4	Reserved.	
0x17	6	Data 6:3: Firmware Mini Loader Internal version.	
0xFE	2	Wrong parameter request.	Returned by the integrated 10 GbE LAN controller 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 integrated 10 GbE LAN controller.
0xFF	2	Integrated 10 GbE LAN Controller is not ready.	Returned by the integrated 10 GbE LAN controller only, on read parameters command when the data that should have been read is not ready. This parameter has no data. The BMC should retry the read transaction. This value is also returned if the byte count is illegal or if the Read command is not preceded by a Write command. It is also returned when opcode 0x00 (device ID/ Rev ID) is requested in Dr state.

9.5.11.2.7 Get Common Filters Command

The BMC uses this command to get the common filters setting. This data can be configured when using Set Common Filters command. The first transaction is a block write that alerts that the BMC wants to read the filters configuration. The second transaction is block read that read the configuration.

Block write transaction:

Function	Command	Byte count	Data
Get Common Filters	0xD3	1	0x00

Following the block write the BMC should issue a block read that reads the filter settings:

Function	Command
Get Common Filters	0xD3

Data returned from the integrated 10 GbE LAN controller:

Function	Byte Count	Command	Data 1	Data 2:4	5:10	Data 11	Data 12	Data 13	Data 14:17
Get Common Filters	18	0xD3	0	Receive Control - see Table 9-13	MAC Address	BMC Alert Address	Interface Data Byte	Alert Value Byte	IPv4 Address

Function	Byte Count	Command	Data 1	Data 2:4	5:10	Data 11	Data 12	Data 13	Data 14:29
Get Common Filters	30	0xD3	0	Receive Control - see Table 9-13	MAC Address	BMC Alert Address	Interface Data Byte	Alert Value Byte	IPv6 Address

If case of error the following answers might be returned:

Function	Command	Byte Count	Data 1
Get Common Filters	0xD3	1	0xFF

This response is by the integrated 10 GbE LAN controller, on read common filter command when the data that should have been read is not ready. This parameter has no data. The BMC should retry the read transaction.

This value is also returned if the byte count is illegal, if the Read command is not preceded by a Write command, or if the filters were not previously configured with a Set Common Filters command (Section 9.5.11.1.7).

9.5.11.2.8 Read Receive Enable Configuration

The BMC uses this command to read the receive configuration data. This data can be configured when using Receive Enable command or through the NVM.

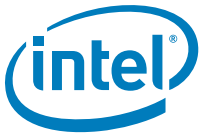
Read Receive Enable Configuration command format (SMBus Read Block) is as follows:

Function	Command
Read Receive Enable	0xDA

Data returned from the integrated 10 GbE LAN controller:

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 9.5.11.1.3.



9.5.12 Example Configuration Steps

This section provides sample configuration settings for common filtering configurations. Three examples are presented. The examples are in pseudo code format, with the name of the SMBus command followed by the parameters for that command and an explanation.

9.5.12.1 Example 1 - Shared MAC and RMCP Only Ports

This example is the most basic configuration. The MAC address filtering is shared with the host operating system and only traffic directed the RMCP ports (0x26Fand 0x298) is filtered. For this example, the BMC must issue gratuitous ARPs because no filter is enabled to pass ARP requests to the BMC.

9.5.12.1.1 Example 1 Pseudo Code

Step 1: Disable existing filtering.

`Receive Enable[00]`

Utilizing the simple form of the Receive Enable command, this prevents any packets from reaching the BMC by disabling filtering:

Receive Enable Control 00h:

- Bit 0 [0] – Disable Receiving of packets

Step 2: Configure MDEF[0].

`Update Manageability Filter Parameters [68, 0, C0000000, 00000000]`

Use the Update Manageability Filter Parameters command to update Decision Filters (MDEF) (parameter 68h). This will update MDEF[0], as indicated by the 2nd parameter (0).

MDEF[0] value of C0000000h:

- Bit 30 [1] – port 0x298
- Bit 31 [1] – port 0x26F

MDEF_EXT[0] value of 0x0000000:

Step 3: Configure MNGONLY.

`Update Manageability Filter Parameters [F, 0, 00000001]`

Use the Update Manageability Filter Parameters command to update Manageability Only (MNGONLY) (parameter 0xF) so that port 0x298 and 0x26F would not be sent to the host.

- Bit [0] - *MDEF[0]* is exclusive to the BMC.

Step 4: - Enable Filtering.

`Receive Enable [05]`

Using the simple form of the Receive Enable command:

Receive Enable Control 0x05:

- Bit 0 [1] – Enable Receiving of packets
- Bit 2 [1] – Enable status reporting (such as link lost)
- Bit 5:4 [00] – Notification method = SMB Alert
- Bit 7 [0] – Use shared MAC

The resulting MDEF filters are as follows:

Table 9-23. Example 1 MDEF Results

		Manageability Decision Filter (MDEF)							
Filter		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
L2 Unicast Address[3:0]	AND								
Broadcast	AND								
Manageability VLAN[7:0]	AND								
IPv6 Address[3:0]	AND								
IPv4 Address[3:0]	AND								
L2 Unicast Address[3:0]	OR								
Broadcast	OR								
Multicast	AND								
ARP Request	OR								
ARP Response	OR								
Neighbor Discovery	OR								
Port 0x298	OR	X							
Port 0x26F	OR	X							
Flex Port 7:0	OR								
Flex TCO	OR								

9.5.12.2 Example 2 - Dedicated MAC, Auto ARP Response and RMCP Port Filtering

This example shows a common configuration; the BMC has a dedicated MAC and IP address. Automatic ARP responses is enabled as well as RMCP port filtering. By enabling Automatic ARP responses the BMC is not required to send the gratuitous ARPs as it did in Example 1.

For demonstration purposes, the dedicated MAC address is calculated by reading the system MAC address and adding 1 do it, assume the System MAC is AABBCCDC. The IP address for this example will be 1.2.3.4. Additionally, the XSUM filtering is enabled.

Note that not all Intel Ethernet Controllers support automatic ARP responses, please refer to product specific documentation.

9.5.12.2.1 Example 2 - Pseudo Code

Step 1: Disable existing filtering.

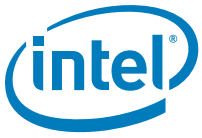
`Receive Enable[00]`

Utilizing the simple form of the Receive Enable command, this prevents any packets from reaching the BMC by disabling filtering:

Receive Enable Control 0x00:

- Bit 0 [0] – Disable Receiving of packets

Step 2: Read system MAC address.



`Get System MAC Address []`

Reads the system MAC address. Assume returned AABBCCDC for this example.

Step 3: Configure XSUM filter.

`Update Manageability Filter Parameters [01, 00800000]`

Use the Update Manageability Filter Parameters command to update filters enable settings (parameter 1). This set the Manageability Control (MANC) register.

MANC Register 0x00800000:

- Bit 23 [1] - XSUM Filter enable

Note that some of the following configuration steps manipulate the MANC register indirectly, this command sets all bits except XSUM to 0b. It is important to either do this step before the others, or to read the value of the MANC and then write it back with only bit 32 changed. Also note that the XSUM enable bit might differ between Ethernet controllers, refer to product specific documentation.

Step 4: Configure MDEF[0].

`Update Manageability Filter Parameters [68, 0, C0000000, 00000000]`

Use the Update Manageability Filter Parameters command to update Decision Filters (MDEF) (parameter 0x68). This updates MDEF[0], as indicated by the 2nd parameter (0).

MDEF value of 0x00000C00:

- Bit 30 [1] – port 0x298
- Bit 31 [1] – port 0x26F

MDEF_EXT[0] value of 0x0000000:

Step 5: Configure MDEF[1].

`Update Manageability Filter Parameters [68, 1, 10000000, 00000000]`

Use the Update Manageability Filter Parameters command to update Decision Filters (MDEF) (parameter 0x61). This updates MDEF[1], as indicated by the 2nd parameter (1).

MDEF value of 10000000:

- Bit 28 [1] – ARP Requests

MDEF_EXT[1] value of 0x0000000:

When enabling automatic ARP responses, the ARP requests still go into the manageability filtering system and as such needs to be designated as also needing to be sent to the host. For this reason a separate MDEF is created with only ARP request filtering enabled.

Refer to the next step for more details.

Step 6: Configure Manageability only.

`Update Manageability Filter Parameters [F, 0, 00000001]`

Use the Update Manageability Filter Parameters command to update Manageability Only (MNGONLY) (parameter 0xF) so that port 0x298 and 0x26F would not be sent to the host.

- Bit [0] - MDEF[0] is exclusive to the BMC.

This enables ARP requests to be passed to both manageability and to the host. Specified separate MDEF filter for ARP requests. If ARP requests had been added to *MDEF[0]* and then *MDEF[0]* specified in management only configuration then not only would RMCP traffic (ports 0x26F and 0x298) be sent only to the BMC, ARP requests would have also been sent to the BMC only.

Step 7: Enable filtering.

`Receive Enable [8D, AABBCCDD, 01020304, 00, 00, 00]`

Using the advanced version Receive Enable command, the first parameter:

Receive Enable Control 0x8D:

- Bit 0 [1] – Enable receiving of packets
- Bit 2 [1] – Enable status reporting (such as link lost)
- Bit 3 [1] – Enable automatic ARP responses
- Bit 5:4 [00] – Notification method = SMB alert
- Bit 7 [1] - Use dedicated MAC

  Second parameter is the MAC address (AABBCCDD).

  Third parameter is the IP address(01020304).

The last three parameters are zero when the notification method is SMB alert.

The resulting MDEF filters are as follows:

Table 9-24. Example 2 MDEF Results

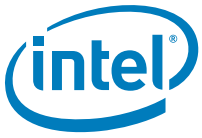
		Manageability Decision Filter (MDEF)							
Filter		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
L2 Unicast Address[3:0]	AND								
Broadcast	AND								
Manageability VLAN[7:0]	AND								
IPv6 Address[3:0]	AND								
IPv4 Address[3:0]	AND								
L2 Unicast Address[3:0]	OR								
Broadcast	OR								
Multicast	AND								
ARP Request	OR		X						
ARP Response	OR								
Neighbor Discovery	OR								
Port 0x298	OR	X							
Port 0x26F	OR	X							
Flex Port 7:0	OR								
Flex TCO	OR								

9.5.12.3 Example 3 - Dedicated MAC and IP Address

This example provides the BMC with a dedicated MAC and IP address and enables it to receive ARP requests. The BMC is then responsible for responding to ARP requests.

For demonstration purposes, the dedicated MAC address is calculated by reading the system MAC address and adding one do it (assume the system MAC is AABBCCDC). The IP address for this example is 1.2.3.4. For this example, the Receive Enable command is used to configure the MAC address filter.

In order for the BMC to be able to receive ARP requests, it needs to specify a filter for this, and that filter needs to be included in the manageability-to-host filtering so that the host operating system might also receive ARP requests.



9.5.12.3.1 Example 3 - Pseudo Code

Step 1: Disable existing filtering.

`Receive Enable[00]`

Utilizing the simple form of the Receive Enable command, this prevents any packets from reaching the BMC by disabling filtering:

Receive enable control 0x00:

- Bit 0 [0] – Disable receiving of packets

Step 2: Read System MAC Address.

`Get System MAC Address []`

Reads the system MAC address. Assume returned AABBCCDC for this example.

Step 3: Configure IP Address Filter

`Update Manageability Filter Parameters [64, 00, 01020304]`

Use the update manageability filter parameters to configure an IPv4 filter.

The 1st parameter (0x64) specifies that we are configuring an IPv4 filter.

The 2nd parameter (0x00) indicates which IPv4 filter is being configured, in this case filter 0.

The 3rd parameter is the IP address – 1.2.3.4.

Step 4: Configure MAC address filter.

`Update Manageability Filter Parameters [66, 00, AABBCCDD]`

Use the update manageability filter parameters to configure a MAC address filter.

The 1st parameter (0x66) specifies that we are configuring a MAC address filter.

The 2nd parameter (0x00) indicates which MAC address filter is being configured, in this case filter 0.

The 3rd parameter is the MAC Address - AABBCCDD

Step 5: Configure MDEF[0] for IP and MAC filtering.

`Update Manageability Filter Parameters [68, 0, 00002001, 00000000]`

Use the Update Manageability Filter Parameters command to update Decision Filters (MDEF) (parameter 0x68). This updates MDEF[0], as indicated by the 2nd parameter (0).

MDEF value of 00002001:

- Bit 0 [1] – MAC[0] address filtering
- Bit 13 [1] – IP[0] address filtering

MDEF_EXT[0] value of 0x0000000:

Step 6: Configure MDEF[1].

`Update Manageability Filter Parameters [68, 1, 10000000]`

Use the Update Manageability Filter Parameters command to update Decision Filters (MDEF) (parameter 0x68). This updates MDEF[1], as indicated by the 2nd parameter (1).

MDEF value of 10000000:

- Bit 28 [1] – ARP requests

MDEF_EXT[1] value of 0x0000000:

Step 7: Configure the management-to-host filter

`Update Manageability Filter Parameters [F, 0, 00000001]`

Use the Update Manageability Filter Parameters command to update Manageability Only (MNGONLY) (parameter 0xF) so that the dedicated MAC/IP traffic would not be sent to the host. Note that given the host does not program this address in it's L2 filtering, this step is not a must, unless the host chooses to work in promiscuous mode.

- Bit [0] - MDEF[0] is exclusive to the BMC.

Step 8: Enable filtering.

Receive Enable [05]

Using the simple form of the Receive Enable command,:

Receive Enable Control 0x05:

- Bit 0 [1] – Enable Receiving of packets
- Bit 2 [1] – Enable status reporting (such as link lost)
- Bit 5:4 [00] – Notification method = SMB alert

The resulting MDEF filters are as follows:

Table 9-25. Example 3 MDEF Results

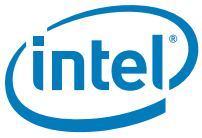
		Manageability Decision Filter (MDEF)							
Filter		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
L2 Unicast Address[3:0]	AND	0001							
Broadcast	AND								
Manageability VLAN[7:0]	AND								
IPv6 Address[3:0]	AND								
IPv4 Address[3:0]	AND	0001							
L2 Unicast Address[3:0]	OR								
Broadcast	OR								
Multicast	AND								
ARP Request	OR		X						
ARP Response	OR								
Neighbor Discovery	OR								
Port 0x298	OR								
Port 0x26F	OR								
Flex Port 7:0	OR								
Flex TCO	OR								

9.5.12.4 Example 4 - Dedicated MAC and VLAN Tag

This example shows an alternate configuration; the BMC has a dedicated MAC and IP address, along with a VLAN tag of 0x32is required for traffic to be sent to the BMC. This means that all traffic with VLAN a matching tag is sent to the BMC.

For demonstration purposes, the dedicated MAC address is calculated by reading the system MAC address and adding 1 to it (assume the system MAC is AABBCCDC). The IP address for this example is 1.2.3.4 and the VLAN tag is 0x0032.

Additionally, the XSUM filtering is enabled.



9.5.12.4.1 Example 4 - Pseudo Code

Step 1: Disable existing filtering.

`Receive Enable[00]`

Utilizing the simple form of the Receive Enable command, this prevents any packets from reaching the BMC by disabling filtering:

Receive enable control 0x00:

- Bit 0 [0] – Disable receiving of packets

Step 2: - Read system MAC address.

`Get System MAC Address []`

Reads the system MAC address. Assume returned AABBCCDC for this example.

Step 3: Configure XSUM filter.

`Update Manageability Filter Parameters [01, 00800000]`

Use the Update Manageability Filter Parameters command to update Filters Enable settings (parameter 1). This sets the Manageability Control (MANC) register.

MANC register 0x00800000:

- Bit 23 [1] – XSUM filter enable

Note that some of the following configuration steps manipulate the MANC register indirectly, this command sets all bits except XSUM to 0b. It is important to either do this step before the others, or to read the value of the MANC and then write it back with only bit 32 changed. Also note that the XSUM enable bit might differ between Ethernet controllers, refer to product specific documentation.

Step 4: Configure VLAN 0 filter.

`Update Manageability Filter Parameters [62, 0, 0032]`

Use the Update Manageability Filter Parameters command to configure VLAN filters. Parameter 0x62 indicates update to VLAN filter, the 2nd parameter indicates which VLAN filter (0 in this case), the last parameter is the VLAN ID (0x0032).

Step 5: Configure MDEF[0].

`Update Manageability Filter Parameters [68, 0, 00000020, 00000000]`

Use the Update Manageability Filter Parameters command to update Decision Filters (MDEF) (parameter 0x68). This updates MDEF[0], as indicated by the 2nd parameter (0).

MDEF value of 00000020:

- Bit 5 [1] – VLAN[0] AND

MDEF_EXT[0] value of 0x0000000:

Step 6: Enable filtering.

`Receive Enable [85, AABBCCDD, 01020304, 00, 00, 00]`

Using the advanced version Receive Enable command, the first parameter:

Receive Enable Control 0x85:

- Bit 0 [1] – Enable receiving of packets
- Bit 2 [1] – Enable status reporting (such as link lost)
- Bit 5:4 [00] – Notification method = SMB alert
- Bit 7 [1] – Use dedicated MAC

Second parameter is the MAC address: AABBCCDD.

Third parameter is the IP address: 01020304.

The last three parameters are zero when the notification method is SMBus alert.

The resulting MDEF filters are as follows:

Table 9-26. Example 4 MDEF Results

		Manageability Decision Filter (MDEF)							
Filter		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
L2 Unicast Address[3:0]	AND								0001
Broadcast	AND								
Manageability VLAN[7:0]	AND	X							
IPv6 Address[3:0]	AND								
IPv4 Address[3:0]	AND								
L2 Unicast Address[3:0]	OR								
Broadcast	OR								
Multicast	AND								
ARP Request	OR								
ARP Response	OR								
Neighbor Discovery	OR								
Port 0x298	OR								
Port 0x26F	OR								
Flex Port 7:0	OR								
Flex TCO	OR								

9.5.13 SMBus Troubleshooting

This section outlines the most common issues found while working with pass-through using the SMBus sideband interface.

9.5.13.1 TCO Alert Line Stays Asserted After a Power Cycle

After the integrated 10 GbE LAN controller resets, all its ports indicates a status change. If the BMC only reads status from one port (slave address), the other ones continue to assert the TCO alert line.

Ideally, the BMC should use the ARA transaction (see Section 9.5.10) to determine which slave asserted the TCO alert. Many customers only wish to use one port for manageability thus using ARA might not be optimal.

An alternate to using ARA is to configure part of the ports to not report status and to set its SMBus timeout period. In this case, the SMBus timeout period determines how long a port asserts the TCO alert line awaiting a status read from a BMC; by default this value is zero (indicates an infinite timeout).

The SMBus configuration section of the NVM has a SMBus notification timeout (ms) field that can be set to a recommended value of 0xFF (for this issue). Note that this timeout value is for all slave addresses. Along with setting the SMBus notification timeout to 0xFF, it is recommended that the other ports be configured in the NVM to disable status alerting. This is accomplished by having the *Enable Status Reporting* bit set to 0b for the desired ports in the LAN configuration section of the NVM.

The third solution for this issue is to have the BMC hard-code the slave addresses to always read from all ports. As with the previous solution, it is recommend that the other ports have status reporting disabled.

9.5.13.2 When SMBus Commands Are Always NACKed

There are several reasons why all commands sent to the integrated 10 GbE LAN controller from a BMC could be NACKed. The following are most common:

- Invalid NVM Image — The image itself might be invalid or it could be a valid image and is not a pass-through image, as such SMBus connectivity is disabled.
- The BMC is not using the correct SMBus address — Many BMC vendors hard-code the SMBus address(es) into their firmware. If the incorrect values are hard-coded, the integrated 10 GbE LAN controller does not respond.
  - The SMBus address(es) can be dynamically set using the SMBus ARP mechanism.
- The BMC is using the incorrect SMBus interface — The NVM might be configured to use one physical SMBus port; however, the BMC is physically connected to a different one.
- Bus Interference — The bus connecting the BMC and the integrated 10 GbE LAN controller might be unstable.

9.5.13.3 SMBus Clock Speed Is 16.6666 KHz

This can happen when the SMBus connecting the BMC and the integrated 10 GbE LAN controller is also tied into another device (such as an ICH) that has a maximum clock speed of 16.6666 KHz. The solution is to not connect the SMBus between the integrated 10 GbE LAN controller and the BMC to this device.

9.5.13.4 A Network Based Host Application Is Not Receiving Any Network Packets

Reports have been received about an application not receiving any network packets. The application in question was NFS under Linux. The problem was that the application was using the RMPC/RMCP+ IANA reserved port 0x26F (623) and the system was also configured for a shared MAC and IP address with the operating system and BMC.

The management control-to-host configuration, in this situation, was setup not to send RMCP traffic to the operating system (this is typically the correct configuration). This means that no traffic send to port 623 was being routed.

The solution in this case is to configure the problematic application NOT to use the reserved port 0x26F.

9.5.13.5 Unable to Transmit Packets from the BMC

If the BMC has been transmitting and receiving data without issue for a period of time and then begins to receive NACKs from the integrated 10 GbE LAN controller when it attempts to write a packet, the problem is most likely due to the fact that the buffers internal to the integrated 10 GbE LAN controller are full of data that has been received from the network but has yet to be read by the BMC.

Being an embedded device, the integrated 10 GbE LAN controller has limited buffers that are shared for receiving and transmitting data. If a BMC does not keep the incoming data read, the integrated 10 GbE LAN controller can be filled up This prevents the BMC form transmitting more data, resulting in NACKs.

If this situation occurs, the recommended solution is to have the BMC issue a Receive Enable command to disable more incoming data, read all the data from the integrated 10 GbE LAN controller, and then use the Receive Enable command to enable incoming data.

9.5.13.6 SMBus Fragment Size

The SMBus specification indicates a maximum SMBus transaction size of 32 bytes. Most of the data passed between the integrated 10 GbE LAN controller and the BMC over the SMBus is RMCP/RMCP+ traffic, which by its very nature (UDP traffic) is significantly larger than 32 bytes in length. Multiple SMBus transactions might therefore be required to move data from the integrated 10 GbE LAN controller to the BMC or to send a data from the BMC to the integrated 10 GbE LAN controller.

Recognizing this bottleneck, the integrated 10 GbE LAN controller handles up to 240 bytes of data in a single transaction. This is a configurable setting in the NVM. The default value in the NVM images is 32, per the SMBus specification. If performance is an issue, increase this size.

During initialization, firmware within the integrated 10 GbE LAN controller allocates buffers based upon the SMBus fragment size setting within the NVM. The integrated 10 GbE LAN controller firmware has a finite amount of RAM for its use: the larger the SMBus fragment size, the fewer buffers it can allocate. Because this is true, BMC implementations must take care to send data over the SMBus efficiently.

For example, the integrated 10 GbE LAN controller firmware has 3 KB of RAM it can use for buffering SMBus fragments. If the SMBus fragment size is 32 bytes then the firmware could allocate 96 buffers of size 32 bytes each. As a result, the BMC could then send a large packet of data (such as KVM) that is 800 bytes in size in 25 fragments of size 32 bytes apiece.

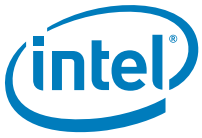
However, this might not be the most efficient way because the BMC must break the 800 bytes of data into 25 fragments and send each one at a time.

If the SMBus fragment size is changed to 240 bytes, the integrated 10 GbE LAN controller firmware can create 12 buffers of 240 bytes each to receive SMBus fragments. The BMC can now send that same 800 bytes of KVM data in only four fragments, which is much more efficient.

The problem of changing the SMBus fragment size in the NVM is if the BMC does not also reflect this change. If a programmer changes the SMBus fragment size in the integrated 10 GbE LAN controller to 240 bytes and then wants to send 800 bytes of KVM data, the BMC can still only send the data in 32 byte fragments. As a result, firmware runs out of memory.

This is because firmware created the 12 buffers of 240 bytes each for fragments; however, the BMC is only sending fragments of size 32 bytes. This results in a memory waste of 208 bytes per fragment. Then when the BMC attempts to send more than 12 fragments in a single transaction, the integrated 10 GbE LAN controller NACKs the SMBus transaction due to not enough memory to store the KVM data.

In summary, if a programmer increases the size of the SMBus fragment size in the NVM (recommended for efficiency purposes) take care to ensure that the BMC implementation reflects this change and uses that fragment size to its fullest when sending SMBus fragments.



9.5.13.7 Losing Link

Normal behavior for the integrated 10 GbE LAN controller when the system powers down or performs a reset is for the link to temporarily go down and then back up again to re-negotiate the link speed. This behavior can have adverse affects on manageability.

For example if there is an active FTP or serial over LAN session to the BMC, this connection might be lost. In order to avoid this possible situation, the BMC can use the Management Control command detailed in Section 9.5.11.1.5 to ensure the link stays active at all times.

This command is available when using the NC-SI sideband interface as well.

Care should be taken with this command, if the software device driver negotiates the maximum link speed, the link speed remains the same when the system powers down or resets. This might have undesirable power consumption consequences. Currently, when using NC-SI, the BMC can re-negotiate the link speed. That functionality is not available when using the SMBus interface.

9.5.13.8 Enable Checksum Filtering

If checksum filtering is enabled, the BMC does not need to perform the task of checking this checksum for incoming packets. Only packets that have a valid checksum is passed to the BMC. All others are silently discarded.

This is a way to offload some work from the BMC.

9.5.13.9 Still Having Problems?

If problems still exist, contact your field representative. Be prepared to provide the following:

- A SMBus trace if possible.
- A dump of the NVM image. This should be taken from the actual Integrated 10 GbE LAN Controller, rather than the NVM image provided by Intel. Parts of the NVM image are changed after writing (such as the physical NVM size).

9.6 NC-SI Pass Through Interface

The Network Controller Sideband Interface (NC-SI) is a DMTF industry standard protocol for the sideband interface. NC-SI uses a modified version of the industry standard RMII interface for the physical layer as well as defining a new logical layer.

The NC-SI specification supported by the integrated 10 GbE LAN controller can be found at:

http://www.dmtf.org/sites/default/files/standards/documents/DSP0222_1.0.1.pdf

9.6.1 Overview

9.6.1.1 Terminology

The terminology in this document is taken from the NC-SI specification.

Table 9-27. NC-SI Terminology

Term	Definition
Frame Versus Packet	Frame is used in reference to Ethernet, whereas packet is used everywhere else.
External Network Interface	The interface of the network controller that provides connectivity to the external network infrastructure (port).
Internal Host Interface	The interface of the network controller that provides connectivity to the host OS running on the platform.
Management Controller (BMC)	An intelligent entity comprising of HW/Firmware/Software, that resides within a platform and is responsible for some or all management functions associated with the platform (BMC, service processor, etc.).
Network Controller (NC)	The component within a system that is responsible for providing connectivity to the external Ethernet network world.
Remote Media	The capability to allow remote media devices to appear as if they were attached locally to the host.
Network Controller Sideband Interface	The interface of the network controller that provides connectivity to a management controller. It can be shorten to sideband interface as appropriate in the context.
Interface	This refers to the entire physical interface, such as both the transmit and receive interface between the management controller and the network controller.
Integrated Controller	The term integrated controller refers to a network controller device that supports two or more channels for NC-SI that share a common NC-SI physical interface. For example, a network controller that has two or more physical network ports and a single NC-SI bus connection.
Multi-Drop	Multi-drop commonly refers to the case where multiple physical communication devices share an electrically common bus and a single device acts as the master of the bus and communicates with multiple slave or target devices. In NC-SI, a management controller serves the role as the master, and the network controllers are the target devices.
Point-to-Point	Point-to-point commonly refers to the case where only two physical communication devices are interconnected via a physical communication medium. The devices might be in a master/slave relationship, or could be peers. In NC-SI, point-to-point operation refers to the situation where only a single management controller and single network controller package are used on the bus in a master/slave relationship where the management controller is the master.
Channel	The control logic and data paths supporting NC-SI pass-through operation on a single network interface (port). A network controller that has multiple network interface ports can support an equivalent number of NC-SI channels.

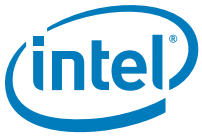


Table 9-27. NC-SI Terminology

Package	One or more NC-SI channels in a network controller that share a common set of electrical buffers and common buffer control for the NC-SI bus. Typically, there will be a single, logical NC-SI package for a single physical network controller package (chip or module). However, the specification allows a single physical chip or module to hold multiple NC-SI logical packages.
Control Traffic/Messages/Packets	Command, response and notification packets transmitted between BMC and the integrated 10 GbE LAN controller for the purpose of managing NC-SI.
Pass-Through Traffic/Messages/ Packets	Non-control packets passed between the external network and the BMC through the integrated 10 GbE LAN controller.
Channel Arbitration	Refer to operations where more than one of the network controller channels can be enabled to transmit pass-through packets to the BMC at the same time, where arbitration of access to the RXD, CRS_DV, and RX_ER signal lines is accomplished either by software of hardware means.
Logically Enabled/Disabled NC	Refers to the state of the network controller wherein pass-through traffic is able/unable to flow through the sideband interface to and from the management controller, as a result of issuing Enable/Disable Channel command.
NC RX	Defined as the direction of ingress traffic on the external network controller interface
NC TX	Defined as the direction of egress traffic on the external network controller interface
NC-SI RX	Defined as the direction of ingress traffic on the sideband enhanced NC-SI Interface with respect to the network controller.
NC-SI TX	Defined as the direction of egress traffic on the sideband enhanced NC-SI Interface with respect to the network controller.

9.6.1.2 System Topology

In NC-SI each physical endpoint (NC package) can have several logical slaves (NC channels).

NC-SI defines that one management controller and up to four network controller packages can be connected to the same NC-SI link.

Figure 9-6 shows an example topology for a single BMC and a single NC package. In this example, the NC package has two NC channels.

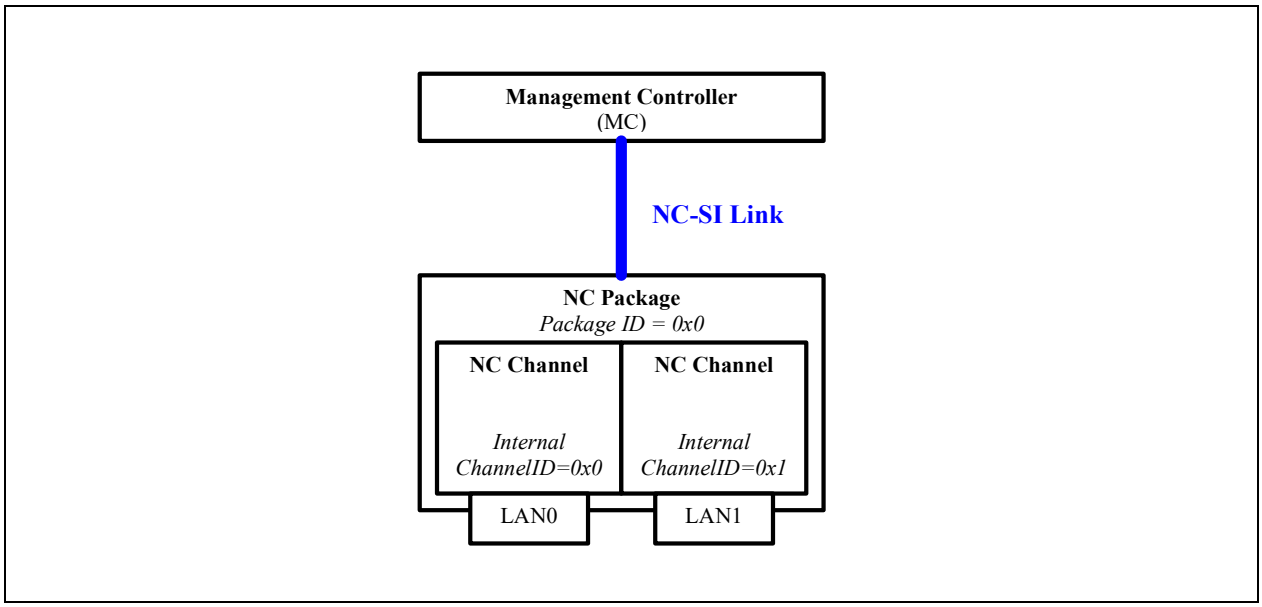


Figure 9-6. Single NC Package, Two NC Channels

Figure 9-7 shows an example topology for a single BMC and two NC packages. In this example, one NC package has two NC channels and the other has only one NC channel. Scenarios in which the NC-SI lines are shared by multiple NCs (Figure 9-7) mandate an arbitration mechanism. The arbitration mechanism is described in Section 9.6.7.1.

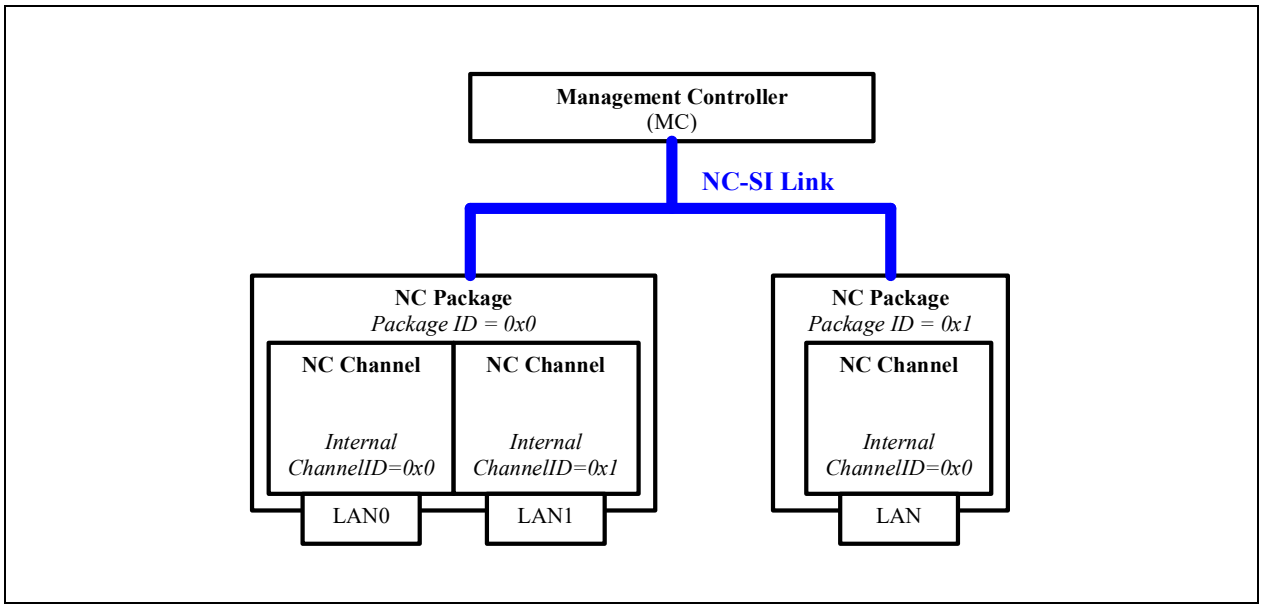
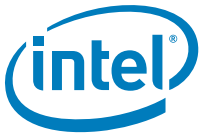


Figure 9-7. Two NC Packages (Left, with Two NC Channels and Right, with One NC Channel)

Note: Channel numbers should match PCI function numbers. If more than one function is defined on a port, the function with the lowest value associated with this port is used (it is assumed



that the first function numbers are assigned to different ports). The association of functions to ports is reflected in the PFGEN_PORTNUM registers.

9.6.1.3 Data Transport

Since NC-SI is based upon the RMII transport layer, data is transferred in the form of Ethernet frames.

NC-SI defines two types of transmitted frames:

1. Control frames:
   a. Configures and control the interface.
   b. Identified by a unique EtherType in their L2 header.
2. Pass-through frames:
   a. Actual LAN pass-through frames transferred from/to the BMC.
   b. Identified as not being a control frame.
   c. Attributed to a specific NC channel by their source MAC address (as configured in the NC by the BMC).

9.6.1.3.1 Control Frames

NC-SI control frames are identified by a unique NC-SI EtherType (0x88F8).

Control frames are used in a single-threaded operation, meaning commands are generated only by the BMC and can only be sent one at a time. Each command from the BMC is followed by a single response from the NC (command-response flow), after which the BMC is allowed to send a new command.

The only exception to the command-response flow is the Asynchronous Event Notification (AEN). These control frames are sent unsolicited from the NC to the BMC.

AEN functionality by the NC must be disabled by default, until activated by the BMC using the Enable AEN commands.

In order to be considered a valid command, a control frame must:

1. Comply with the NC-SI header format.
2. Be targeted to a valid channel in the package via the Package ID and Channel ID fields. For example, to target a NC channel with package ID of 0x2 and internal channel ID of 0x5, the BMC must set the channel ID inside the control frame to 0x45. The channel ID is composed of three bits of package ID and five bits of internal channel ID.
3. Contain a correct payload checksum (if used).
4. Meet any other condition defined by NC-SI.

There are also commands (such as select package) targeted to the package as a whole. These commands must use an internal channel ID of 0x1F.

For details, refer to the NC-SI specification.

9.6.1.3.2 NC-SI Frames Receive Flow

Figure 9-8 shows the flow for frames received on the NC from the BMC.

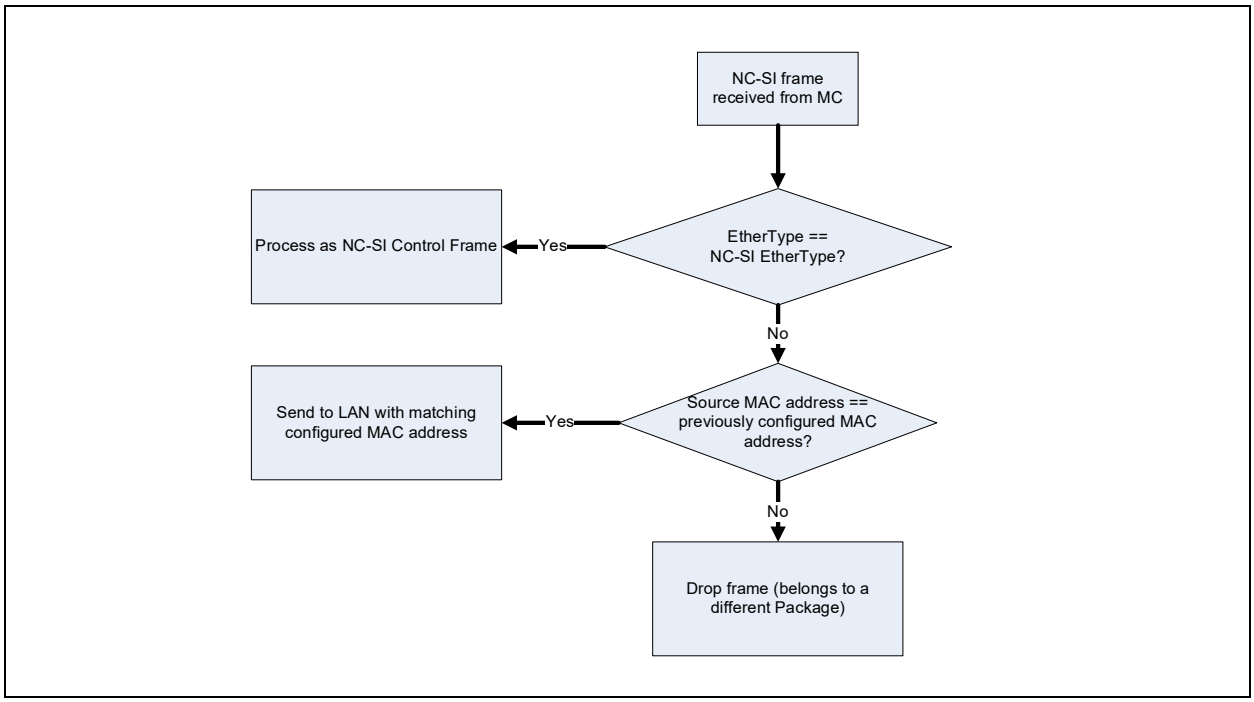


Figure 9-8. NC-SI Frames Receive Flow for the NC

9.6.2 NC-SI Standard Support

9.6.2.1 Supported Features

The integrated 10 GbE LAN controller supports all the mandatory features of the NC-SI specification (rev 1.0.1).

Table 9-28 lists the supported commands.

Table 9-29 lists optional features supported.

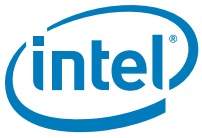


Table 9-28. Supported NC-SI Commands

Command	Supported Over RMII	Supported Over MCTP With Pass Through	Supported Over MCTP Without Pass Through
Clear Initial State	Yes	Yes	Yes
Get Version ID	Yes	Yes	Yes
Get Parameters	Yes	Yes	Yes
Get Controller Packet Statistics	Yes, partially	Yes, partially	Yes, partially
Get Link Status	Yes	Yes	Yes
Enable Channel	Yes	Yes	Yes
Disable Channel	Yes	Yes	Yes
Reset Channel	Yes	Yes	Yes
Enable VLAN	Yes[1,2]	Yes[1]	No[3]
Disable VLAN	Yes	Yes	No[3]
Enable Broadcast Filter	Yes	Yes	No[3]
Disable Broadcast Filter	Yes	Yes	No[3]
Set MAC Address[4]	Yes	Yes	No[3]
Get NC-SI Statistics	Yes, partially	Yes, partially	Yes, partially
Set NC-SI Flow Control	Yes	No	No[3]
Set Link Command	Yes	Yes	Yes
Enable Global Multicast Filter	Yes	Yes	No[3]
Disable Global Multicast Filter	Yes	Yes	No[3]
Get Capabilities	Yes	Yes	Yes
Set VLAN Filters	Yes	Yes	No[3]
AEN Enable	Yes	Yes	Yes
Get NC-SI Pass-Through Statistics	Yes, partially	Yes, partially	No[3]
Select Package	Yes	Yes	Yes
Deselect Package	Yes	Yes	Yes
Enable Channel Network Tx[5]	Yes	Yes	No
Disable Channel Network Tx	Yes	Yes	No
OEM Command[6]	Yes	Yes	Yes

1. In cases that one of the LAN devices is assigned for the sole use of the manageability and its LAN PCIe function is disabled, using the NC-SI Set Link command while advertising multiple speeds and enabling auto-negotiation, results in the lowest possible speed chosen. To enable link of higher a speed, the BMC should not advertise speeds that are below the desired link speed. When doing it, changing the power state of the LAN device has not effect and the link speed is not re-negotiated.
2. The integrated 10 GbE LAN controller does not support filtering of user priority/DEI bits of VLAN.
3. In MCTP without pass through mode, only control commands are supported and not pass through traffic.As a result, many of the regular NC-SI commands are not supported or are supported in a limited manner, only to enable control and status reporting for the integrated 10 GbE LAN controller.
4. Set MAC address command fails with a 0x002 (Parameter Is Invalid, Unsupported, or Out-of-Range) reason code if received on a Tx enabled port with a unicast MAC address already configured on another Tx enabled port.
5. Enable Channel Network TX command fails with a 0x002 (Parameter Is Invalid, Unsupported, or Out-of-Range) reason code if received on a port configured with an unicast MAC address equal to the MAC address of another Tx enabled port.
6. See Section 9.6.3 for details.

Table 9-29. Optional NC-SI Features Support

Feature	Implement	Details
AENs	Yes	The software device driver state AEN might be emitted up to one minute after actual software device driver change if the software device driver was taken down unexpectedly. A re-configuration required AEN is sent before a firmware reset initiated due to a firmware code update
Get Controller Packet Statistics command	Yes, partially	Supports the following counters[1]: 2-8,13-16[2] The statistics are cleared between reads.
Get NC-SI statistics	Yes	Supports all counters.
Get NC-SI Pass-Through Statistics	Yes, partially	Support the following counters: 1, 2, 6[3], 7.
VLAN Modes	Yes, partially	Support only modes 1, 3.
Buffering Capabilities	Yes	8 Kb
MAC Address Filters	Yes	Supports two MAC addresses per port.
Channel Count	Yes	Supports four channels.
VLAN Filters	Yes	Supports eight VLAN filters per port. Filtering is ignoring the *DEI* bit and the 802.1P priority bits.
Broadcast Filters	Yes	Support the following filters: ARP. DHCP. Net BIOS.
Multicast Filters	Yes	Supports the following filters: IPv6 neighbor advertisement. IPv6 router advertisement. DHCPv6 relay and server multicast.
Hardware Arbitration	Yes	Supports NC-SI hardware arbitration.

1. *TCTL.EN* should be set to 1b to activate Tx related counters and *RCTL.RXEN*, *MANC.RCV_EN* or *GRC.APME* should be set to enable RX related counters.
2. As described in the get controller packet statistics counter numbers table in NC-SI specification.
3. The Total Pass-through RX Packets Received On the LAN Interface counter includes also OS2BMC traffic.

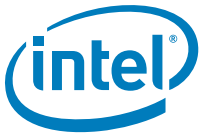
9.6.2.2 Allow Link Down (ALD) Support

NC-SI PHY power down conditions:

In NC-SI mode, the integrated 10 GbE LAN controller might dynamically change the PHY power mode according to the NC-SI channel state assuming no other functionality requires the PHY to be active (host or wake up).

The following algorithm is used to define if PHY activity is required:

- At initialization time, if the manageability mode is NC-SI, a 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 and the PHY is disabled.
- If the channel is disabled via a Reset Channel command, the PHY power state is set back to the initialization value as defined by the *All PHYs in D3 N* bit.



Note: Before transitioning to D3 it is the responsibility of the software device driver to request the PHY to be active for walk up activities using WUC register .

9.6.2.3 AEN Handling

Asynchronous events might occur when the integrated 10 GbE LAN controller is not allowed to send them. The following rules defines the behavior of the integrated 10 GbE LAN controller in these cases:

1. While the integrated 10 GbE LAN controller is disabled, for each type of AEN only the last event is kept
2. Outstanding AENs that occurred while package was deselected is transmitted when package is selected.
3. On a transition from channel disabled to channel enabled, all outstanding events are erased to prevent stale events notifications.

9.6.3 NC-SI Mode — Intel Specific Commands

In addition to regular NC-SI commands, the following Intel vendor specific commands are supported. The purpose of these commands is to provide a means for the BMC to access some of the Intel-specific features present in the integrated 10 GbE LAN controller.

9.6.3.1 Overview

The following features are available via the NC-SI OEM specific commands:

- Receive filters:
- Packet addition decision filters 0x0…0x4
- Packet reduction decision filters 0x5…0x7
- MNGONLY register (controls the forwarding of manageability packets to the host)
- Flex 128 filters
- Flex TCP/UDP port filters 0x0...0x2
- IPv4/IPv6 filters
- Get System MAC Address — This command enables the BMC to retrieve the system MAC address used by the NC. This MAC address can be used for a shared MAC address mode.
- Keep PHY Link Up (*Veto* bit) Enable/Disable — This feature enables the BMC to block PHY reset, which might cause session loss.
- TCO Reset — Enables the BMC to reset the integrated 10 GbE LAN controller.
- Checksum offloading — Offloads IP/UDP/TCP checksum checking from the BMC.
- OS2BMC Control commands.
- Firmware Version commands.

These commands are designed to be compliant with their corresponding SMBus commands (if existing). All of the commands are based on a single DMTF defined NC-SI command, known as OEM command. This command is as follows.

9.6.3.2 OEM Command (0x50)

The OEM command can be used by the BMC to request the sideband interface to provide vendor-specific information. The Vendor Enterprise Number (VEN) is the unique MIB/SNMP private enterprise number assigned by IANA per organization. Vendors are free to define their own internal data structures in the vendor data fields.

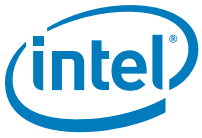
	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...	Intel Command Number	Optional Data		
...	...			
...	Optional Data		Padding to 32 bits (0x00)	
...	Checksum			

9.6.3.2.1 OEM Response (0xD0)

	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		
...	...			
...	Optional Return Data		Padding to 32 bits (0x00)	
...	Checksum			

Note: Responses have no command-specific reason code, unless otherwise specified within the command.

Note: The commands/responses described in the sections that follow includes only the part up to the data. The padding and checksum are implied.



9.6.3.3 OEM Commands Summary

Table 9-30. OEM Specific Command Response Reason Codes

Response Code		Reason Code	
Value	**Description**	**Value**	**Description**
0x1	Command Failed	0x5081	Invalid Intel Command Number.
		0x5082	Invalid Intel Command Parameter Number.
		0x5085	Internal Network Controller Error.
		0x5086	Invalid Vendor Enterprise Code.

Table 9-31. OEM Commands Summary

Intel Command	Parameter	Command Name	Supported in MCTP Without Pass Through	Description
0x00	0x00	Set IP Filters Control	No	9.6.3.5
0x01	0x00	Get IP Filters Control	No	9.6.3.6
0x02	0x0F	Set Manageability Only	No	9.6.3.7.2
	0x10	Set Flexible 128 Filter Mask and Length		9.6.3.7.3
	0x11	Set Flexible 128 Filter Data		9.6.3.7.4
	0x63	Set Flex TCP/UDP Port Filters		9.6.3.7.5
	0x64	Set Flex IPv4 Address Filters		9.6.3.7.6
	0x65	Set Flex IPv6 Address Filters		9.6.3.7.7
	0x67	Set EtherType Filter		9.6.3.7.8
	0x68	Set Packet Addition Extended Filter		9.6.3.7.9
0x03	0x0F	Get Manageability Only	No	9.6.3.8.2
	0x10	Get Flexible 128 Filter Mask and Length		9.6.3.8.3
	0x11	Get Flexible 128 Filter Data		9.6.3.8.4
	0x63	Get Flex TCP/UDP Port Filters		9.6.3.8.5
	0x64	Get Flex IPv4 Address Filters		9.6.3.8.6
	0x65	Get Flex IPv6 Address Filters		9.6.3.8.7
	0x67	Get EtherType Filter		9.6.3.8.8
	0x68	Get Packet Addition Extended Filter		9.6.3.8.9
0x04	0x10	Set Extended Unicast Packet Reduction	No	9.6.3.9.1
	0x11	Set Extended Multicast Packet Reduction		9.6.3.9.2
	0x12	Set Extended Broadcast Packet Reduction		9.6.3.9.4
0x05	0x10	Get Extended Unicast Packet Reduction	No	9.6.3.10.1
	0x11	Get Extended Multicast Packet Reduction		9.6.3.10.2
	0x12	Get Extended Broadcast Packet Reduction		9.6.3.10.3
0x06	N/A	Get System MAC Address	Yes	9.6.3.11
0x20	N/A	Set Intel Management Control	No	9.6.3.12
0x21	N/A	Get Intel Management Control	No	9.6.3.13

Table 9-31. OEM Commands Summary (Continued)

Intel Command	Parameter	Command Name	Supported in MCTP Without Pass Through	Description
0x22	N/A	Perform TCO Reset	Yes	9.6.3.14
0x23	N/A	Enable IP/UDP/TCP Checksum Offloading	No	9.6.3.15.1
0x24	N/A	Disable IP/UDP/TCP Checksum Offloading	No	9.6.3.15.3
0x40	0x01	Enable OS2BMC Flow	No	9.6.3.16.1
	0x02	Enable Network-to-BMC Flow		9.6.3.16.2
	0x03	Enable Both Network-to-BMC and Host-to-BMC Flow		9.6.3.16.3
	0x04	Set BMC IP Address	Yes	9.6.3.16.4
0x41	N/A	Get OS2BMC Parameters	No	9.6.3.16.5
0x48	0x1	Get Controller Information	Yes	9.6.3.17.1

Note: All the commands are supported both over RMII NC-SI and over MCTP.

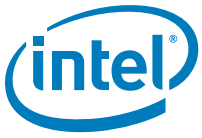
9.6.3.4 Proprietary Commands Format

9.6.3.4.1 Set Intel 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	Filter Control Index		

9.6.3.4.2 Set Intel Filters Control Response Format (Intel Command 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	Filter Control Index		



9.6.3.5 Set Intel Filters Control — IP Filters Control Command (Intel Command 0x00, Filter Control Index 0x00)

This command controls different aspects of the Intel filters.

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x00	0x00	IP Filters control (3-2)	
24...27	IP Filters Control (1-0)			

Where IP Filters Control has the following format:

Bit #	Name	Description	Default Value
0	IPv4/IPv6 Mode	IPv6 (0b) = There are zero IPv4 filters and four IPv6 filters. IPv4 (1b) = There are four IPv4 filters and four IPv6 filters.	1b
1...31	Reserved	Reserved.	

9.6.3.5.1 Set Intel Filters Control — IP Filters Control Response (Intel Command 0x00, Filter Control Index 0x00)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x00	0x00		

9.6.3.6 Get Intel Filters Control Commands (Intel Command 0x01)

9.6.3.6.1 Get Intel Filters Control — IP Filters Control Command (Intel Command 0x01, Filter Control Index 0x00)

This command controls different aspects of the Intel filters:

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x01	0x00		

9.6.3.6.1.1 Get Intel Filters Control — IP Filters Control Response (Intel Command 0x01, Filter Control Index 0x00)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**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)			

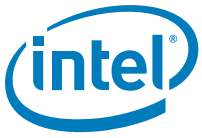
9.6.3.7 Set Intel Filters Formats

9.6.3.7.1 Set Intel Filters Command (Intel Command 0x02)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x02	Parameter Number	Filters Data (optional)	

9.6.3.7.1.1 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	



	Bits		
20...23	Manufacturer ID (Intel 0x157)		
24...	0x02	Filter Control Index	Return Data (Optional)

9.6.3.7.2 Set Intel Filters — Manageability Only Command (Intel Command 0x02, Filter Parameter 0x0F)

This command sets the MNGONLY register. The MNGONLY register controls whether pass-through packets destined to the BMC are not forwarded to the host operating system. The MNGONLY register is listed in Table 9-4.

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x0F	Manageability Only (3-2)	
24...25	Manageability Only (1-0)			

9.6.3.7.2.1 Set Intel Filters — Manageability Only Response (Intel Command 0x02, Filter Parameter 0x0F)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x0F		

9.6.3.7.3 Set Intel Filters — Flex Filter Enable Mask and Length Command (Intel Command 0x02, Filter Parameter 0x10)

The following command sets the Intel flex filters mask and length:

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			

20...23	0x02	0x10	Mask Byte 1	Mask Byte 2
24...27	...	...	...	...
28...31	...	...	...	...
32...35	...	...	...	...
36...37	Mask Byte 15	Mask Byte 16	Reserved	Reserved
38	Length			

9.6.3.7.3.1 Set Intel Filters — Flex Filter Enable Mask and Length Response (Intel Command 0x02, Filter Parameter 0x10)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x10		

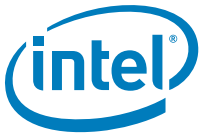
9.6.3.7.4 Set Intel Filters — Flex Filter Data Command (Intel Command 0x02, Filter Parameter 0x11)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...	0x02	0x11	Filter Data Group	Filter Data 1
	...	Filter Data N		

The Filter Data Group parameter defines which bytes of the flex filter are set by this command:

Table 9-32. Filter Data Group

Code	Bytes Programmed	Filter Data Length
0x0	bytes 0-29	1 - 30
0x1	bytes 30-59	1 - 30
0x2	bytes 60-89	1 - 30
0x3	bytes 90-119	1 - 30
0x4	bytes 120-127	1 - 8



Note: Using this command to configure the filters data must be done after the flex filter mask command is issued and the mask is set.

9.6.3.7.4.1 Set Intel Filters — Flex Filter Data Response (Intel Command 0x02, Filter Parameter 0x11)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x11		

9.6.3.7.5 Set Intel Filters — Flex TCP/UDP Port Filter Command (Intel Command 0x02, Filter Parameter 0x63)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**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 Invalid Intel Parameter Number reason (0x5082).

9.6.3.7.5.1 Set Intel Filters — Flex TCP/UDP Port Filter Response (Intel Command 0x02, Filter Parameter 0x63)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x02	0x63		

9.6.3.7.6 Set Intel Filters — IPv4 Filter Command (Intel Command 0x02, Filter Parameter 0x64)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x64	IP filter index	IPv4 Address (3)
24...26	IPv4 Address (2-0)			

Note: The filters index range can vary according to the IPv4/IPv6 mode setting in the Filters Control command.

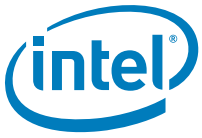
IPv4 Mode: Filter index range: 0x0...0x3.

IPv6 Mode: This command should not be used in IPv6 mode.

9.6.3.7.6.1 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 IPv4 address equals all zero, a command failed Response Code is returned, with Invalid Intel Parameter Number reason (0x5082).



9.6.3.7.7 Set Intel Filters — IPv6 Filter Command (Intel Command 0x02, Filter Parameter 0x65)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x65	IP filter index	...IPv6 Address (MSB, byte 15)
24...27	...	...	...	...
28...31	...	...	...	...
32...35	...	...	...	...
36...37	...		IPv6 Address (LSB, byte 0)	

Note: The filters index range can vary according to the IPv4/IPv6 mode setting in the Filters Control command.

IPv4 Mode: Filter index range: 0x1...0x3.

IPv6 Mode: Filter index range: 0x0...0x3.

9.6.3.7.7.1 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 does not match the previous ranges, a command failed Response Code is returned, with Invalid Intel Parameter Number reason (0x5082).

If the IPv6 address equals all zero, a command failed Response Code is returned, with Invalid Intel Parameter Number reason (0x5082).

9.6.3.7.8 Set Intel Filters - EtherType Filter Command (Intel Command 0x02, Filter parameter 0x67)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x02	0x67	EtherType Filter Index	EtherType Filter MSB
24...27	...	...	EtherType Filter LSB	

Where the EtherType filter has the format as described in Section 8.2.2.20.4.

9.6.3.7.8.1 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		

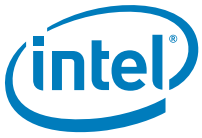
If the Ethertype filter Index is larger than three, a command failed Response Code is returned with Invalid Intel Parameter Number reason (0x5082).

9.6.3.7.9 Set Intel Filters - Packet Addition Extended Decision Filter Command (Intel Command 0x02, Filter parameter 0x68)

See Figure 9-2 for description of the decision filters structure.

The command must overwrite any previously stored value. The value set is not checked.

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			



	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
20...23	0x02	0x68	Extended Decision filter Index	Extended Decision filter 1 MSB
24...27	...	...	Extended Decision filter 1 LSB	Extended Decision filter 0 MSB
28...30	...	...	Extended Decision filter 0 LSB	

Extended Decision filter Index Range: 0...4

Filter 0: See Table 9-33.

Filter 1: See Table 9-34.

Table 9-33. Filter Values

Bit #	Name	Description
1:0	Unicast (AND)	If set, packets must match unicast filter 0 to 1, respectively.
3:2	Reserved	Reserved.
4	Broadcast (AND)	If set, packets must match the broadcast filter.
12:5	VLAN (AND)	If set, packets must match VLAN filter 0 to 7, respectively.
16:13	IPv4 Address (AND)	If set, packets must match IPv4 filter 0 to 3, respectively
20:17	IPv6 Address (AND)	If set, packets must match IPv4 filter 0 to 3, respectively
22:21	Unicast (OR)	If set, packets can pass if match unicast filter 0 to 1, respectively or a different OR filter.
24:23	Reserved	Reserved.
25	Broadcast (OR)	If set, packets can pass if match the broadcast filter or a different OR filter.
26	Multicast (AND)	If set, packets must match the multicast filter.
27	ARP Request (OR)	If set, packets can pass if match the ARP request filter or a different OR filter.
28	ARP Response (OR)	If set, packets can pass if match the ARP response filter or a different OR filter.
29	Neighbor Discovery - 134 (OR)	If set, packets can pass if match the neighbor discovery filter (type134 - router advertisement) or a different OR filter.
30	Port 0x298 (OR)	If set, packets can pass if match a fixed TCP/UDP port 0x298 filter or a different OR filter.
31	Port 0x26F (OR)	If set, packets can pass if match a fixed TCP/UDP port 0x26F filter or a different OR filter.

Table 9-34. Extended Filter 1 Values

Bit #	Name	Description
3:0	Ethertype 0 -3 (AND)	If set, packets must match the Ethertype filter 0 to 3, respectively.
7:4	Ethertype 0 -3 (OR)	If set, packets must match the Ethertype filter 0 to 3, respectively or a different OR filter.
18:8	Flex port 10:0 (OR)	If set, packets can pass if match the TCP/UDP port filter 10:0.
19	DHCPv6 (OR)	If set, packets can pass if match the DHCPv6 port (0x0223).

Table 9-34. Extended Filter 1 Values (Continued)

Bit #	Name	Description
20	DHCP Client (OR)	If set, packets can pass if match the DHCP server port (0x0043).
21	DHCP Server (OR)	If set, packets can pass if match the DHCP client port (0x0044).
22	NetBIOS Name Service (OR)	If set, packets can pass if match the NetBIOS name service port (0x0089).
23	NetBIOS Datagram Service (OR)	If set, packets can pass if match the NetBIOS datagram service port (0x008A).
24	Flex TCO (OR)	If set, packets can pass if match the flex 128 TCO filter.
25	Neighbor Discovery - 135 (OR)	If set, packets must also match the neighbor discovery filter (type135 - neighbor solicitation. or a different OR filter.
26	Neighbor Discovery - 136 (OR)	If set, packets must also match the neighbor discovery filter (type136 - neighbor advertisement) or a different OR filter.
27	Neighbor Discovery - 137 (OR)	If set, packets must also match the neighbor discovery filter (type137 - redirect) or a different OR filter.
28	Reserved	Reserved.
29	MLD (OR)	If set, packets must also match one of the MLD ICMPv6 types or a different OR filter.
31:30	Reserved	Reserved.

9.6.3.7.9.1 Set Intel Filters – Packet Addition Extended Decision Filter Response (Intel Command 0x02, Filter parameter 0x68)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**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 Invalid Intel Parameter Number reason (0x5082).

9.6.3.8 Get Intel Filters Formats

9.6.3.8.1 Get Intel Filters Command (Intel Command 0x03)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x03	Parameter Number		

9.6.3.8.1.1 Get Intel Filters Response (Intel Command 0x03)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x03	Parameter Number	Optional Return Data	

9.6.3.8.2 Get Intel Filters — Manageability Only Command (Intel Command 0x03, Filter Parameter 0x0F)

This command retrieves the MNGONLY register. The MNGONLY register controls whether pass-through packets destined to the BMC are also be forwarded to the host operating system.

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x03	0x0F		

9.6.3.8.2.1 Get Intel Filters — Manageability Only Response (Intel Command 0x03, Filter Parameter 0x0F)

The MNGONLY register structure is listed in Table 9-4.

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x0F	Manageability to Host (3-2)	
28...29	Manageability to Host (1-0)			

9.6.3.8.3 Get Intel Filters — Flex Filter 0 Enable Mask and Length Command (Intel Command 0x03, Filter Parameter 0x10)

The following command retrieves the Intel flex filters mask and length. See Section 9.3.3.5 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		

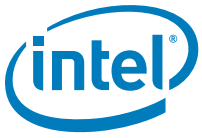
9.6.3.8.3.1 Get Intel Filters — Flex Filter 0 Enable Mask and Length Response (Intel Command 0x03, Filter Parameter 0x10)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x10	Mask Byte 1	Mask Byte 2
28...31	...	...	...	...
32...35	...	...	...	...
36...39	...	...	...	...
40...43	...	Mask Byte 16	Reserved	Reserved
44	Flexible Filter Length			

9.6.3.8.4 Get Intel Filters — Flex Filter 0 Data Command (Intel Command 0x03, Filter Parameter 0x11)

The following command retrieves the Intel flex filters data.

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x03	0x11	Filter Data Group 0...4	



The Filter Data Group parameter defines which bytes of the Flex filter are returned by this command:

Table 9-35. 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

9.6.3.8.4.1 Get Intel Filters — Flex Filter 0 Data Response (Intel Command 0x03, Filter Parameter 0x11)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...	0x03	0x11	Filter Group Number	Filter Data 1
	...	Filter Data N		

9.6.3.8.5 Get Intel Filters — Flex TCP/UDP Port Filter Command (Intel Command 0x03, Filter Parameter 0x63)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**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.

9.6.3.8.5.1 Get Intel Filters — Flex TCP/UDP Port Filter Response (Intel Command 0x03, Filter Parameter 0x63)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**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.

9.6.3.8.6 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	

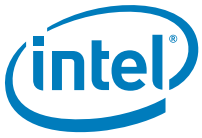
Note: The filters index range can vary according to the IPv4/IPv6 mode setting in the Filters Control command.

IPv4 Mode: Filter index range: 0x0...0x3.

IPv6 Mode: This command should not be used in IPv6 mode.

9.6.3.8.6.1 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)			



9.6.3.8.7 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.

9.6.3.8.7.1 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)	

9.6.3.8.8 Get Intel Filters - EtherType Filter Command (Intel Command 0x03, Filter parameter 0x67)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x03	0x67	EtherType Filter Index	

Valid indices: 0...3

9.6.3.8.8.1 Get Intel Filters - EtherType Filter Response (Intel Command 0x03, Filter parameter 0x67)

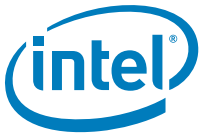
	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x67	EtherType Filter Index	EtherType Filter MSB
28...30	..	..	EtherType Filter LSB	

If the Ethertype filter index is larger than three, a command failed Response Code is returned with Invalid Intel Parameter Number reason (0x5082).

9.6.3.8.9 Get Intel Filters – Packet Addition Extended Decision Filter Command (Intel Command 0x03, Filter parameter 0x68)

This command enables 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	



9.6.3.8.9.1 Get Intel Filters – Packet Addition Extended Decision Filter Response (Intel Command 0x03, Filter parameter 0x68)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x03	0x68	Decision Filter Index	Decision Filter 1 MSB
28...31	..	..	Decision Filter 1 LSB	Decision Filter 0 MSB
32...34	..	..	Decision Filter 0 LSB	

Where Decision Filter 0 and Decision Filter 1 have the structure as detailed in the respective Set commands.

If the Extended Decision Filter Index is bigger than four, a command failed Response Code is returned with Invalid Intel Parameter Number reason (0x5082).

9.6.3.9 Set Intel Packet Reduction Filters Formats

The non-extended commands are obsolete. The extended commands (Section 9.6.3.9.2 to Section 9.6.3.9.4.1) should be used instead.

9.6.3.9.1 Set Intel Packet Reduction Filters Command (Intel Command 0x04)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x04	Packet Reduction Index	Packet Reduction Data...	

Note: Intel recommends that the BMC only use the Extended Packet Reduction commands.

The *Packet Reduction Data* field has the following structure:

Table 9-36. Packet Reduction Field Description

Bit #	Name	Description
12:0	Reserved	Reserved.
16:13	IPv4 Address (AND)	If set, packets must match IPv4 filter 0 to 3, respectively.
20:17	IPv6 Address (AND)	If set, packets must match IPv4 filter 0 to 3, respectively.
27:21	Reserved	Reserved.
28	ARP Response (OR)	If set, packets can pass if match the ARP response filter or a different OR filter.
29	Reserved	Reserved.
30	Port 0x298	If set, packets can pass if match a fixed TCP/UDP port 0x298 filter.
31	Port 0x26F	If set, packets can pass if match a fixed TCP/UDP port 0x26F filter.

Table 9-37. Extended Packet Reduction Field Description

Bit #	Name	Description
3:0	Ethertype 0 -3 (AND)	If set, packets must match the Ethertype filter 0 to 3, respectively.
7:4	Ethertype 0-3 (OR)	If set, packets can pass if match the Ethertype filter 0 to 3,respectively.
15:12	Reserved	Reserved.
8:18	Flex port 10:0 (OR)	If set, packets can pass if match the TCP/UDP port filter 10:0.
23:19	Reserved	Reserved.
24	Flex TCO (OR)	If set, packets can pass if match the flex 128 TCO filter.
28:25	Reserved	Reserved.
29	MLD (OR)	If set, packets must also match one of the MLD ICMPv6 types or a different OR filter.
31:30	Reserved	Reserved.

The filtering is divided into two decisions:

- Bit 20:13 in Table 9-36 and bits 3:2 in Table 9-37 works in an AND manner; it must be true in order for a packet to pass (if was set).

Bits 28 in Table 9-36 and bits 24:10 in Table 9-37 work in an OR manner; at least one of them must be true for a packet to pass (if any were set).

9.6.3.9.1.1 Set Intel Packet Reduction Filters Response (Intel Command 0x04)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...	0x04	Packet Reduction Index		

9.6.3.9.2 Set Extended Unicast Packet Reduction Command (Intel Command 0x04, Reduction Filter Index 0x10)

The command must 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	Extended Unicast Reduction Filter MSB	..
24..27	..	Extended Unicast Reduction Filter LSB	Unicast Reduction Filter MSB	..
28..29	..	Unicast Reduction Filter LSB		

The command must overwrite any previously stored value.

Note: See Table 9-36 and Table 9-37 for description of the Unicast Extended Packet Reduction format.

9.6.3.9.2.1 Set Extended Unicast Packet Reduction Response (Intel Command 0x04, Reduction Filter Index 0x10)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x04	0x10		

9.6.3.9.3 Set Extended Multicast Packet Reduction Command (Intel Command 0x04, Reduction Filter Index 0x11)

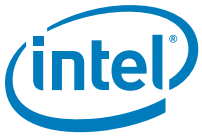
	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x04	0x11	Extended Multicast Reduction Filter MSB	..
24..27	..	Extended Multicast Reduction Filter LSB	Multicast Reduction Filter MSB	..
28..29	..	Multicast Reduction Filter LSB		

Note: See Table 9-36 and Table 9-37 for description of the Multicast Extended Packet Reduction format.

The command must overwrite any previously stored value.

9.6.3.9.3.1 Set Extended Multicast Packet Reduction Response (Intel Command 0x04, Reduction Filter Index 0x11)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x04	0x11		



9.6.3.9.4 Set Extended Broadcast Packet Reduction Command (Intel Command 0x04, Reduction Filter Index 0x12)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00..15	NC-SI Header			
16..19	Manufacturer ID (Intel 0x157)			
20..23	0x04	0x12	Extended Broadcast Reduction Filter MSB	..
24..27	..	Extended Broadcast Reduction Filter LSB	Broadcast Reduction Filter MSB	..
28..29	..	Broadcast Reduction Filter LSB		

Note: See Table 9-36 and Table 9-37 for description of the Broadcast Extended Packet Reduction format.

The command must overwrite any previously stored value.

9.6.3.9.4.1 Set Extended Broadcast Packet Reduction Response (Intel Command 0x04, Reduction Filter Index 0x12)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00..15	NC-SI Header			
16..19	Response Code		Reason Code	
20..23	Manufacturer ID (Intel 0x157)			
24..25	0x04	0x12		

9.6.3.10 Get Intel Packet Reduction Filters Formats

Note: The non-extended commands are not supported anymore. Use the extended commands (Section 9.6.3.10.1 to Section 9.6.3.10.3.1) instead.

9.6.3.10.1 Get Extended Unicast Packet Reduction Command (Intel Command 0x05, Reduction Filter Index 0x10)

\

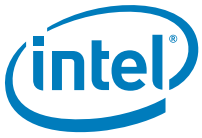
	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x05	0x10		

9.6.3.10.1.1 Get Extended Unicast Packet Reduction Response (Intel Command 0x05, Reduction Filter Index 0x10)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x05	0x00	Extended Unicast Packet Reduction (3-2)	
28...29	Extended Unicast Packet Reduction (1-0)		Unicast Packet Reduction (3-2)	
30...31	Unicast Packet Reduction (1-0)			

9.6.3.10.2 Get Extended Multicast Packet Reduction Command (Intel Command 0x05, Reduction Filter Index 0x11)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x05	0x11		



9.6.3.10.2.1 Get Extended Multicast Packet Reduction Response (Intel Command 0x05, Reduction Filter Index 0x11)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x05	0x11	Extended Multicast Packet Reduction (3-2)	
28...29	Extended Multicast Packet Reduction (1-0)		Multicast Packet Reduction (3-2)	
30...31	Multicast Packet Reduction (1-0)			

9.6.3.10.3 Get Extended Broadcast Packet Reduction Command (Intel Command 0x05, Reduction Filter Index 0x12)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x05	0x12		

9.6.3.10.3.1 Get Extended Broadcast Packet Reduction Response (Intel Command 0x05, Reduction Filter Index 0x12)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x05	0x12	Extended Broadcast Packet Reduction (3-2)	
28...29	Extended Broadcast Packet Reduction (1-0)		Broadcast Packet Reduction (3-2)	
30...31	Broadcast Packet Reduction (1-0)			

9.6.3.11 System MAC Address

9.6.3.11.1 Get System MAC Address Command (Intel Command 0x06)

In order to support a system configuration that requires the integrated 10 GbE LAN controller to hold the MAC address for the BMC (such as shared MAC address mode), the following command is provided to enable the BMC to query the integrated 10 GbE LAN controller for a valid MAC address.

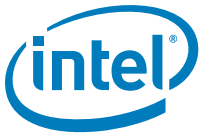
The NC must return the system MAC addresses. The BMC should use the returned MAC addressing as a shared MAC address by setting it using the Set MAC Address command as defined in NC-SI 1.0.

It is also recommended that the BMC use packet reduction and Manageability-to-host command to set the proper filtering method.

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20	0x06			

9.6.3.11.2 Get System MAC Address Response (Intel Command 0x06)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x06	MAC Address		
28...30	MAC Address			



9.6.3.12 Set Intel Management Control Formats

9.6.3.12.1 Set Intel Management Control Command (Intel Command 0x20)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...22	0x20	0x00	Intel Management Control 1	

Where Intel Management Control 1 is as follows:

Bit #	Default value	Description
0	0b	Enable Critical Session Mode (Keep PHY Link Up and Veto Bit) 0b = Disabled. 1b = Enabled. When critical session mode is enabled, the following behaviors are disabled: • The PHY is not reset on PE_RST# and PCIe resets (in-band and link drop). Other reset events are not affected — Internal_Power_On_Reset, device disable, Force TCO, and PHY reset by software. • The PHY does not change its power state. As a result, link speed does not change. • The integrated 10 GbE LAN controller does not initiate configuration of the PHY to avoid losing link.
2:1	00b	Reserved
4:3	00b	NC-SI Auto configuration: • 00b: No change • 01b: Capture current configuration and enable auto configuration from NVM • 10b: Disable auto configuration from NVM ***Note:*** Setting this field to 01b will lead to a non compliant NC-SI behavior. When received as part of a Get Intel Management response the meaning of this field is as follow: • 00b: Reserved • 01b: NC-SI is auto configured at reset (Enable NVM configuration bit in Common Manageability Parameters 2[9] is set). • 10b: NC-SI is NOT auto configured at reset (Enable NVM configuration bit in Common Manageability Parameters 2[9] is cleared). • 11b: Reset.
7:5	0x0	Reserved.

When the command is received with the NC-SI Auto configuration field set to 01b, Firmware reads the current setting of the manageability filters of all the ports and stores them in the right “Pass Through Control Words” structures and sets the new Enable NVM configuration bit in Common Manageability Parameters 2[9].

Note: The flex filter data should also be saved. Since flex filter data is optional, the flex filter information will be saved only if the section exists in NVM.

Note: This command returns a response only after the NVM is updated, thus it may violate the 50 ms limit on NC-SI commands response.

When a command is received with the NC-SI Auto configuration field set to 10b, Firmware clears the Enable NVM configuration bit in Common Manageability Parameters 2[9].

9.6.3.12.2 Set Intel Management Control Response (Intel Command 0x20)

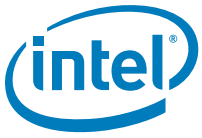
	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x20	0x00		

9.6.3.13 Get Intel Management Control Formats

9.6.3.13.1 Get Intel Management Control Command (Intel Command 0x21)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x21	0x00		

Where Intel Management Control 1 is as described in Section 9.6.3.12.2.



9.6.3.13.2 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	

9.6.3.14 TCO Reset

Depending on the bit set in the TCO mode field this command will cause the integrated 10 GbE LAN controller to perform either:

1. **TCO Reset**

- If Force TCO reset is enabled in the NVM, The Force TCO reset clears the data-path (Rx/Tx) of the integrated 10 GbE LAN controller to enable the BMC to transmit/receive packets through the integrated 10 GbE LAN controller.
- If the BMC has detected that the operating system is hung and has blocked the Rx/Tx path The Force TCO reset clears the data-path (Rx/Tx) of the integrated 10 GbE LAN controller to enable the BMC to transmit/receive packets through the integrated 10 GbE LAN 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 integrated 10 GbE LAN controller considers Force TCO command as an indication that the operating system is hung and clears the DRV_LOAD flag (disable the driver). If TCO reset is disabled in NVM, the integrated 10 GbE LAN controller clears the *CTRL_EXT.DRV_LOAD* bit but does not reset the data-path and notifies BMC on successful completion.

2. **TCO Isolate**.

- If TCO isolate is enabled in the NVM. The TCO Isolate command disables PCIe write operations to the LAN port.
- If TCO Isolate is disabled in NVM, the integrated 10 GbE LAN controller does not execute the command but sends a response to the BMC with successful completion.
- Following TCO Isolate, management isolates the function related to the port on which the command was received.

3. **Firmware Reset.**

- This command causes re-initialization of all the manageability functions and re-load of manageability related NVM words.
- When the BMC has loaded new management related NVM image the Firmware Reset command loads management related NVM information without need to power down the system.
- This command is issued to the package and affects all channels. After the Firmware Reset the Firmware Semaphore register (FWSM) is re-initialized.

Note: TCO Isolate affects only the channel (port) that the command was issued to. Force TCO resets the entire the integrated 10 GbE LAN controller (all channels in the package).

Following firmware reset, BMC needs to re-initialize all ports. Only one of the fields should be set in a given command. Setting more than one field might yield unexpected results.

9.6.3.14.1 Perform Intel TCO Reset Command (Intel Command 0x22)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20	0x22	TCO Mode		

Where TCO mode is:

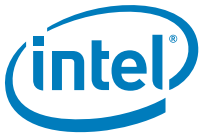
Field	Bit(s)	Description
DO_TCO_RST	0	Perform TCO Reset. 0b= Do nothing. 1b= Perform TCO reset.
DO_TCO_ISOLATE [1]	1	Do TCO Isolate. 0b = Enable PCIe write access to LAN port. 1b = Isolate host PCIe write operation to the port. **Note:** Should be used for debug only. **Note:** The TCO isolate do not impact MCTP traffic. **Note:** When Isolate is set, the OS2BMC flow is disabled also.
RESET_MGMT	2	Reset manageability; re-load manageability NVM words. 0b = Do nothing. 1b = Issue firmware reset to manageability. Setting this bit generates a one-time firmware reset. Following the reset, management related data from NVM is loaded.
Reserved	7:3	Reserved (set to 0x00).

Note: For compatibility, the TCO Reset command without the TCO mode parameter is accepted (TCO reset is performed).

1. TCO Isolate Host Write operation enabled in NVM.

9.6.3.14.2 Perform Intel TCO Reset Response (Intel Command 0x22)

Note: When a Firmware Reset is requested (TCO Mode = RESET_MGMT), there is no response, as the integrated 10 GbE LAN controller goes to Initial State as part of the command execution.



	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			

9.6.3.15 Checksum Offloading

This command enables the checksum offloading filters in the integrated 10 GbE LAN controller.

When enabled, these filters block any packets that did not pass IP, UDP or TCP checksum from being forwarded to the BMC.

9.6.3.15.1 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			

9.6.3.15.2 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			

9.6.3.15.3 Disable Checksum Offloading Command (Intel Command 0x24)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20	0x24			

9.6.3.15.4 Disable Checksum Offloading Response (Intel Command 0x24)

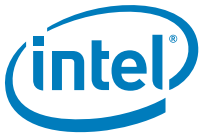
	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...26	0x24			

9.6.3.16 OS 2 BMC configuration

These commands control enabling of the OS-to-BMC flow.

9.6.3.16.1 EnableOS2BMC Flow Command (Intel Command 0x40, Index 0x1)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x40	0x01		



9.6.3.16.1.1 EnableOS2BMC Flow Response (Intel Command 0x40, Index 0x1)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x40	0x01		

9.6.3.16.2 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		

9.6.3.16.2.1 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		

9.6.3.16.3 Enable both Host and Network to BMC flows Command (Intel Command 0x40, Index 0x3)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...21	0x40	0x03		

9.6.3.16.3.1 Enable both Host and Network to BMC flows Response (Intel Command 0x40, Index 0x3)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...25	0x40	0x03		

9.6.3.16.4 Set BMC IP Address Command (Intel Command 0x40, Index 0x4)

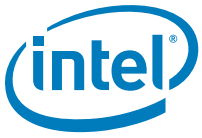
This command is used to expose the BMC IP address to the host.

The IP type entry indicate whether the IP address is an IPv4 or an IPv6 address:

0 = Ipv4

1 = IPv6

2 = No IP address, then the command should not include an IP address.



	Bits			
Bytes	**31..24**	**23..16**	**15..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	

9.6.3.16.4.1 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		

9.6.3.16.5 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			

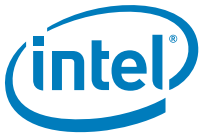
9.6.3.16.5.1 Get OS2BMC Parameters Response (Intel Command 0x41)

	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..41	..	IPv6 Address (LSB, byte 0)/Reserved		

Where the status byte partition is as follow:

Table 9-38. Status byte description

Bits	Content
0	0b = IPv4. 1b = IPv6. Relevant only if the IP address valid bit is set.
1	IP address valid.
2	Network to BMC status. 0b = network 2 BMC flow is disabled. 1b = network 2 BMC flow is enabled.
3	OS2BMC status. 0b = OS 2 BMC flow is disabled. 1b = OS 2 BMC flow is enabled.
7:4	Reserved.



9.6.3.17 Get Controller information Command (Intel Command 0x48, Index 0x1)

This command gather the controller identification information and return it back to the BMC.

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Manufacturer ID (Intel 0x157)			
20...23	0x48	0x1		

9.6.3.17.1 Get Controller Information response (Intel Command 0x48, Index 0x1)

	Bits			
Bytes	**31:24**	**23:16**	**15:08**	**07:00**
00...15	NC-SI Header			
16...19	Response Code		Reason Code	
20...23	Manufacturer ID (Intel 0x157)			
24...27	0x48	0x01	Reserved	Number of Inventory entries
28...31	Controller Info Item 1 ID	Controller Info Item 1 length	Controller Info Item 1 Data	
...	....			
...	Controller Info Item 2 ID	Controller Info Item 2 length	Controller Info Item 2 Data	
...	....			
...	Controller Info Item n ID	Controller Info Item n length	Controller Info Item n Data	
...	....			

Where the possible inventory items are described as follows. Note that not all the inventory items would be present in all the implementations of this command.

Table 9-39. Controller information Items

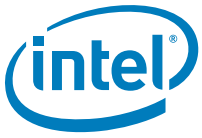
ID	Length (in bytes)	Data	Notes
0x00	3	Device ID (2 bytes) + RevID	Device ID: This is the hardware default value, not any value programmed via NVM. RevID: Read from setID message result as read from CFG space offset 0x8 using FW_PCI_CIAA/FW_PCI_CIAD registers XORED with the value in PCI_REVID.NVM_REVID. ***Note:*** This TLV is not returned in Dr state.
0x0B	2	NVM Image Version	
0x0C	4	EMP ROM Internal Version	
0x0D	4	Flash Firmware Image Internal version	
0x0E	2	PXE Firmware version	MajorVersion.MinorVersion.Build.
0x0F	2	iSCSI Firmware version	
0x10	2	uEFI Firmware version	
0x16	2	Reserved	
0x17	4	Firmware Mini Loader Internal version	

If an address outside of the space or with a wrong alignment or an unknown address selector is received, the command fails with an Invalid Intel Command Parameter Number reason (0x5082).

Address Selector	Space	Allowed Address	Alignment
0x0	CSR	0:0x3FFFC	Dword
0x1	Aux	0:0x2003	Byte
0x2	Memory	0x10000:0x93FFC	Dword

If an address outside of the space or with a wrong alignment or an unknown address selector is received, the command fails with an Invalid Intel Command Parameter Number reason (0x5082).

Address Selector	Space	Allowed Address	Alignment
0x0	CSR	0:0x3FFFC	Dword
0x1	Aux	0:0x2003	Byte
0x2	Memory	0x10000:0x93FFC	Dword



9.6.4 Asynchronous Event Notifications

The asynchronous event notifications are unsolicited messages sent from the NC to the BMC to report status changes (such as link change, operating system state change, etc.).

Recommendations:

- The BMC firmware designer should use AENs. To do so, the designer must take into account the possibility that a NC-SI response frame (such as a frame with the NC-SI EtherType), arrives out-of-context (not immediately after a command, but rather after an out-of-context AEN).
- To enable AENs, the BMC should first query which AENs are supported, using the Get Capabilities command, then enable desired AEN(s) using the Enable AEN command, and only then enable the channel using the Enable Channel command.

9.6.5 Querying Active Parameters

The BMC can use the Get Parameters command to query the current status of the operational parameters.

9.6.6 Resets

In NC-SI there are two types of resets defined:

1. Synchronous entry into the initial state.
2. Asynchronous entry into the initial state.

Recommendations:

- It is very important that the BMC firmware designer keep in mind that following any type of reset, all configurations are considered as lost and thus the BMC must re-configure everything.
- As an asynchronous entry into the initial state might not be reported and/or explicitly noticed, the BMC should periodically poll the NC with NC-SI commands (such as Get Version ID, Get Parameters, etc.) to verify that the channel is not in the initial state. Should the NC channel respond to the command with a Clear Initial State Command Expected reason code, the BMC should consider the channel (and most probably the entire NC package) as if it underwent a (possibly unexpected) reset event. Thus, the BMC should re-configure the NC. See the NC-SI specification section on Detecting Pass-through Traffic Interruption.
- The Intel recommended polling interval is 2-3 seconds.

For exact details on the resets, refer to NC-SI specification.

9.6.7 Advanced Workflows

9.6.7.1 Multi-NC Arbitration

As described in Section 9.6.1.2, in a multi-NC environment, there is a need to arbitrate the NC-SI lines.

Figure 9-9 shows the system topology of such an environment.

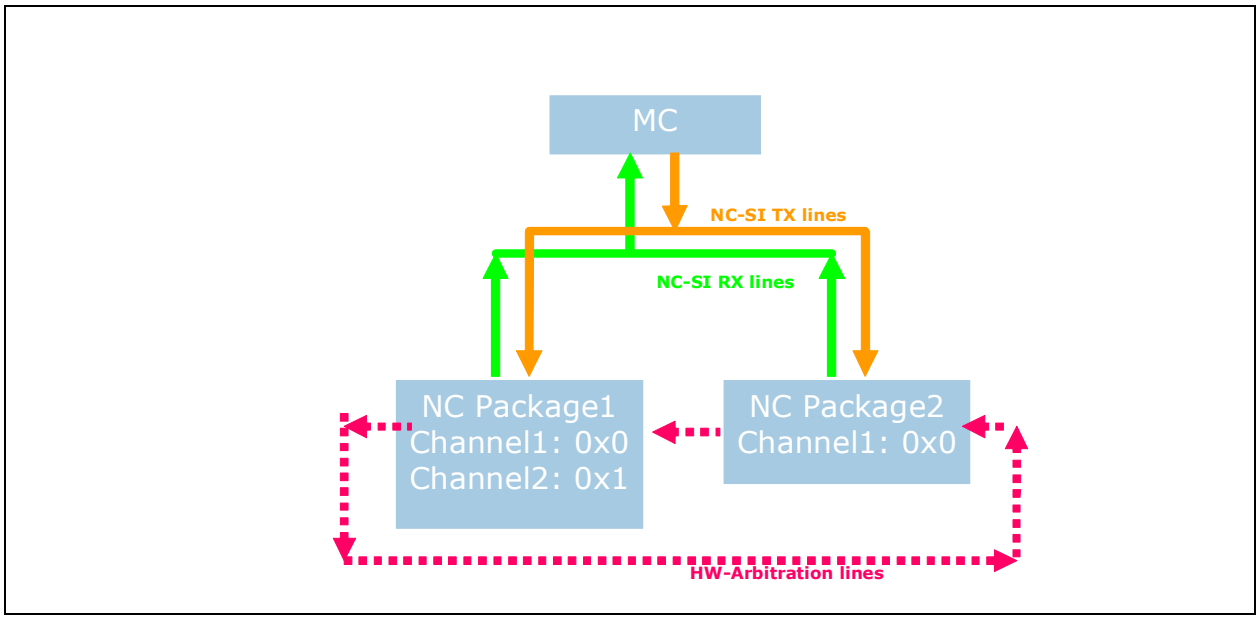


Figure 9-9. Multi-NC Environment

See Figure 9-9. The NC-SI Rx lines are shared between the NCs. To enable sharing of the NC-SI Rx lines, NC-SI has defined an arbitration scheme.

The arbitration scheme mandates that only one NC package can use the NC-SI Rx lines at any given time. The NC package that is allowed to use these lines is defined as selected. All the other NC packages are de-selected.

NC-SI has defined two mechanisms for the arbitration scheme:

1. Package selection by the BMC. In this mechanism, the BMC is responsible for arbitrating between the packages by issuing NC-SI commands (Select/De-Select Package). The BMC is responsible for having only one package selected at any given time.
2. Hardware arbitration. In this mechanism, two additional pins on each NC package are used to synchronize the NC package. Each NC package has an ARB_IN and ARB_OUT line and these lines are used to transfer Tokens. A NC package that has a token is considered selected.

Note: Hardware arbitration is enabled by the NC-SI HW Arbitration Enable configuration bit in the NC-SI Configuration 1 NVM word.

For details, refer to the NC-SI specification.

9.6.7.2 Package Selection Sequence Example

Following is an example work flow for a BMC and occurs after the discovery, initialization, and configuration.

Assuming the BMC needs to share the NC-SI bus between packages, the BMC should:

1. Define a time-slot for each device.
2. Discover, initialize, and configure all the NC packages and channels.
3. Issue a De-Select Package command to all the channels.

4. Set active_package to 0x0 (or the lowest existing package ID).
5. At the beginning of each time slot the BMC should:
   a. Issue a De-Select Package to the active_package. The BMC must then wait for a response and then an additional timeout for the package to become de-selected (200 µs). See the NC-SI specification table 10 — parameter NC Deselect to Hi-Z Interval.
   b. Find the next available package (typically active_package = active_package + 1).
   c. Issue a Select Package command to active_package.

9.6.7.3 Multiple Channels (Fail-Over)

In order to support a fail-over scenario, it is required from the BMC to operate two or more channels. These channels might or might not be in the same package.

The key element of a fault-tolerance fail-over scenario is having two (or more) channels identifying to the switch with the same MAC address, but only one of them being active at any given time (such as switching the MAC address between channels). To accomplish this, NC-SI provides the following commands:

1. Enable Network Tx command — This command enables shutting off the network transmit path of a specific channel. This enables the BMC to configure all the participating channels with the same MAC address but only enable one of them.
2. Link Status Change AEN or Get Link Status command.

9.6.7.3.1 Fail-Over Algorithm Example

The following is a sample workflow for a fail-over scenario for the integrated 10 GbE LAN controller quad-port 10 GbE controller (one package and four channels):

1. BMC initializes and configures all channels after power-up. However, the BMC uses the same MAC address for all of the channels.
2. The BMC queries the link status of all the participating channels. The BMC should continuously monitor the link status of these channels. This can be accomplished by listening to AENs (if used) and/or periodically polling using the Get Link Status command.
3. The BMC then only enables channel 0 for network transmission.
4. The BMC then issues a gratuitous ARP (or any other packet with its source MAC address) to the network. This packet informs the switch that this specific MAC address is registered to channel 0's specific LAN port.
5. The BMC begins normal workflow.
6. Should the BMC receive an indication (AEN or polling) that the link status for the active channel (channel 0) has changed, the BMC should:
   a. Disable channel 0 for network transmission.
   b. Check if a different channel is available (link is up).
   c. If found:
      - Enable network Tx for that specific channel.
      - Issue a gratuitous ARP (or any other packet with its source MAC address) to the network. This packet informs the switch that this specific MAC address is registered to channel 0's specific LAN port.
      - Resume normal workflow.
      - If not found, report the error and continue polling until a valid channel is found.

The previous algorithm can be generalized such that the start-up and normal workflow are the same. In addition, the BMC might need to use a specific channel (such as channel 0). In this case, the BMC should switch the network transmit to that specific channel as soon as that channel becomes valid (link is up).

Recommendations:

- Wait for a link-down-tolerance timeout before a channel is considered invalid. For example, a link re-negotiation might take a few seconds (normally 2 to 3 or might be up to 9). Thus, the link must be re-established after a short time.
- Typically, this timeout is recommended to be three seconds.
- Even when enabling and using AENs, periodically poll the link status, as dropped AENs might not be detected.

9.6.7.4 Statistics

The BMC might use the statistics commands as defined in NC-SI. These counters are meant mostly for debug purposes and are not all supported.

The statistics are divided into three commands:

1. Controller statistics — These are statistics on the network interface (to the host operating system and the pass through traffic). See the NC-SI specification for details.
2. NC-SI statistics — These are statistics on the NC-SI control frames (such as commands, responses, AENs, etc.). See the NC-SI specification for details.

NC-SI pass-through statistics — These are statistics on the NC-SI pass-through frames. See the NC-SI specification for details.

9.6.8 External Link Control via NC-SI

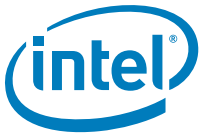
The BMC can use the NC-SI Set Link command to control the external interface link settings. This command enables the BMC to set the auto-negotiation, link speed, duplex, and other parameters.

This command is only available when the host operating system is not present. Indicating the host operating system status can be obtained via the Get Link Status command and/or Host OS Status Change AEN command.

Recommendation:

- Unless explicitly needed, it is not recommended to use this feature. The NC-SI Set Link command does not expose all the possible link settings and/or features. This might cause issues under different scenarios. Even if you decided to use this feature, use it only if the link is down (trust the integrated 10 GbE LAN controller until proven otherwise).
- It is recommended that the BMC first query the link status using the Get Link Status command. The BMC should then use this data as a basis and change only the needed parameters when issuing the Set Link command.

For details, refer to the NC-SI specification.



9.6.8.1 Set Link While LAN PCIe Functionality is Disabled

In cases where the integrated 10 GbE LAN controller is used solely for manageability and its LAN PCIe function is disabled, using the NC-SI Set Link command while advertising multiple speeds and enabling auto-negotiation results in the lowest possible speed chosen.

To enable link of higher a speed, the BMC should not advertise speeds that are below the desired link speed, as the lowest advertised link speed is chosen.

When the integrated 10 GbE LAN controller is only used for manageability and the link speed advertisement is configured by the BMC, changes in the power state of the LAN device is not effected and the link speed is not re-negotiated by the LAN device.

9.6.8.2 Set Link Error codes

The following rules are used to define the error code returned for Set Link command in case an invalid configuration is requested:

1. Host Driver Check: If host device driver is present, return a Command Specific Response (0x9) with a Set Link Host OS/Driver Conflict Reason (0x1).
2. Speed Present Check: If no speed is selected, return a General Reason Code for a failed command (0x1) with Parameter Is Invalid, Unsupported, or Out-of-Range Reason (0x2).
3. Parameter Validity:
   a. Auto-Negotiation Parameter Validation: If auto-negotiation is requested and none of the selected parameters are valid for the device, return a General Reason Code for a failed command (0x1) with a Parameter Is Invalid, Unsupported, or Out-of-Range Reason (0x2).

Note: This means that, for example, a command requesting 10 GbE on a 1 GbE device succeeds provided that the command requests at least one other supported speed.

The same goes for an unsupported duplex setting (a device with no HD support accepts a command with both FD and HD set), and also for HD being requested with speeds of 1 GbE and higher as long as a speed below 1 GbE is also requested (and is supported in HD). The integrated 10 GbE LAN controller ignore the unsupported parameters.

   b. Force Mode Parameter Validation:

- If more than one link speed is being forced, then return a General Reason Code for a failed command (0x1) and a Command Specific Reason with a Set Link Speed Conflict Error (0x0905).
- If more than one duplex setting is being forced, then return a General Reason Code for a failed command (0x1) with Parameter Is Invalid, Unsupported, or Out-of-Range Reason (0x2).
- If 1 GbE and above is requested with HD, then return a General Reason Code for a failed command (0x1) and a Command Specific Reason with Set Link Parameter Conflict Reason (0x0903).

4. Media Type Compatibility Check: If current media type is not compatible for the requested link parameters, return a General Reason Code for a failed command (0x1) and a Command Specific Reason with Set Link Media Conflict Error (0x0902).
5. Power State Compatibility Check: If current power state does not allow for the requested link parameters, return a General Reason Code for a failed command (0x1) and a Command Specific Reason with Set Link Power Mode Conflict Reason (0x0904).
6. If for some reason the hardware cannot perform the flow required for the command, return a General Reason Code for a failed command (0x1) and a Command Specific Response (0x9) with Link Command Failed-Hardware Access Error (0x6).

9.7 Management Component Transport Protocol (MCTP)

9.7.1 MCTP Overview

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, a specification exists for MCTP 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.

9.7.1.1 NC-SI Over MCTP

MCTP is a transport layer protocol that do not include the functionality required to control the pass through traffic required for BMC connection to the network. This functionality is provided by encapsulating NC-SI traffic as defined in DMTF's DSP0222 document.

The details of NC_SI over MCTP protocol are defined in the NC-SI Over MCTP Specification.

The NC-SI over MCTP specification defines two types of MCTP message types: NC-SI (0x2) and Ethernet (0x3). The integrated 10 GbE LAN controller supports both messages. When used only for control, then only the NC-SI (0x2) message type is supported.

In addition to the above message types supported by the integrated 10 GbE LAN controller, the PCIe based VDM message type is also supported over PCIe to support ACL commands.

Details of the NC-SI over MCTP can be found in Section 9.7.4.

9.7.1.2 MCTP Usage Model

The integrated 10 GbE LAN controller supports NC-SI over MCTP protocol over the SMBus bus. The integrated 10 GbE LAN controller can connect through MCTP to a BMC or the ME engine in the chipset as shown in Figure 9-10.

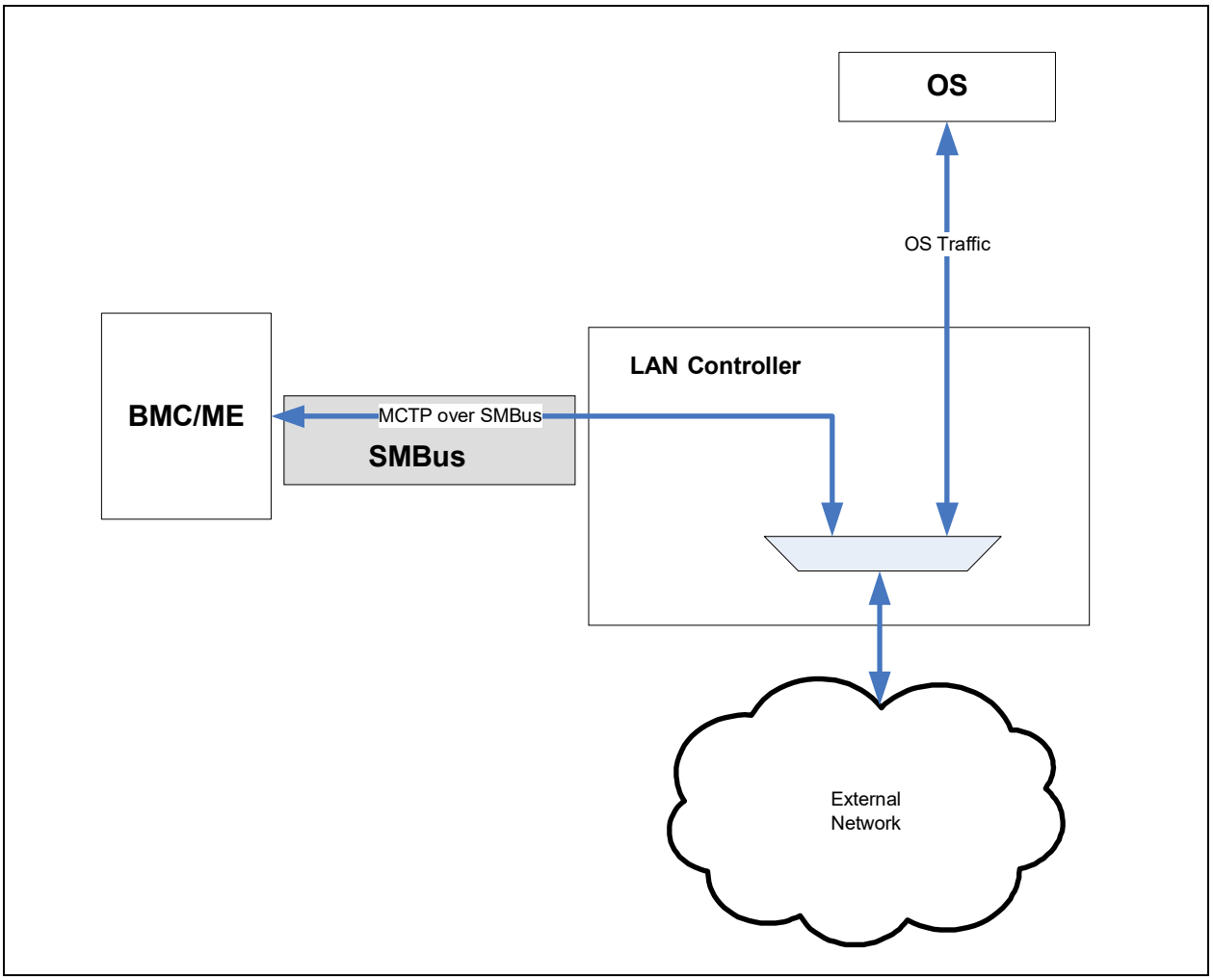


Figure 9-10. MCTP Connections of the Integrated 10 GbE LAN Controller

9.7.2 NC-SI to MCTP Mapping

The two network ports of the integrated 10 GbE LAN controller (mapped to two NC-SI channels) are mapped to a single MCTP endpoint.

Section 9.7.2.1 describes the transition between the bus.

For SMBus, the integrated 10 GbE LAN controller should expect MCTP commands from two sources: the bus owner and the BMC. In addition, it should expect pass-through traffic. Thus, it should be able to process up to three interleaved commands/data:

- An MCTP control/OEM command from the SMBus bus owner (single packet message).
- An MCTP control/OEM command from the BMC over SMBus (single packet message).
- An NC-SI command or Ethernet packet from the BMC over the active channel.

A single source should not interleave packets it sends.

The topology used for MCTP connection is shown in Figure 9-10.

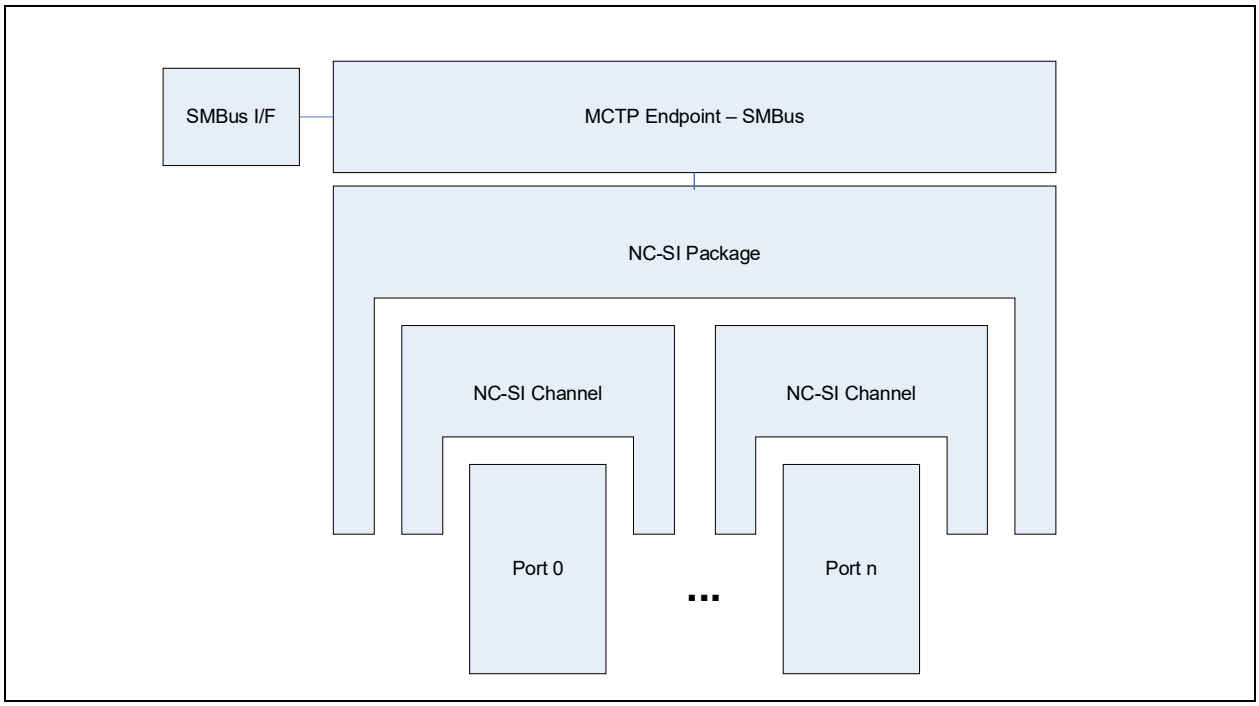


Figure 9-11. MCTP Endpoint Topology

9.7.2.1 Detection of BMC EID and Physical Address

In order to enable transactions between the BMC and the integrated 10 GbE LAN controller, the bus physical address (SMBus) and the EID of the partner needs to be discovered. The integrated 10 GbE LAN controller does not try to discover the BMC and assume the BMC initiates the connection. If the integrated 10 GbE LAN controller is in NC-SI initial state, then the EID and the physical address of the BMC are extracted from the *Clear Initial State* command parameters or any other NC-SI command received later with a channel ID of the integrated 10 GbE LAN controller. Subsequent pass through traffic is received from or sent to this address only.

If the EID or the physical address of the integrated 10 GbE LAN controller changes, it indicates the changes to bus owner so that the routing tables can be updated. There is no attempt to directly send an indication to the BMC about the change.

9.7.2.2 Bus Transition

The following section defines the transition flow between the SMBus bus on which MCTP flows. Figure 9-10 describes the flow to transition between the SMBus. The following parameters are used to define the flow:

- Integrated 10 GbE LAN Controller EID on SMBus

- Bus Owner EID on SMBus
- Bus Owner SMBus address
- BMC EID on SMBus
- BMC SMBus address
- Integrated 10 GbE LAN Controller SMBus address

All these variables are initialized to zero at power on apart from the SMBus address of the endpoint (the integrated 10 GbE LAN controller) which might be initialized from a NVM value.

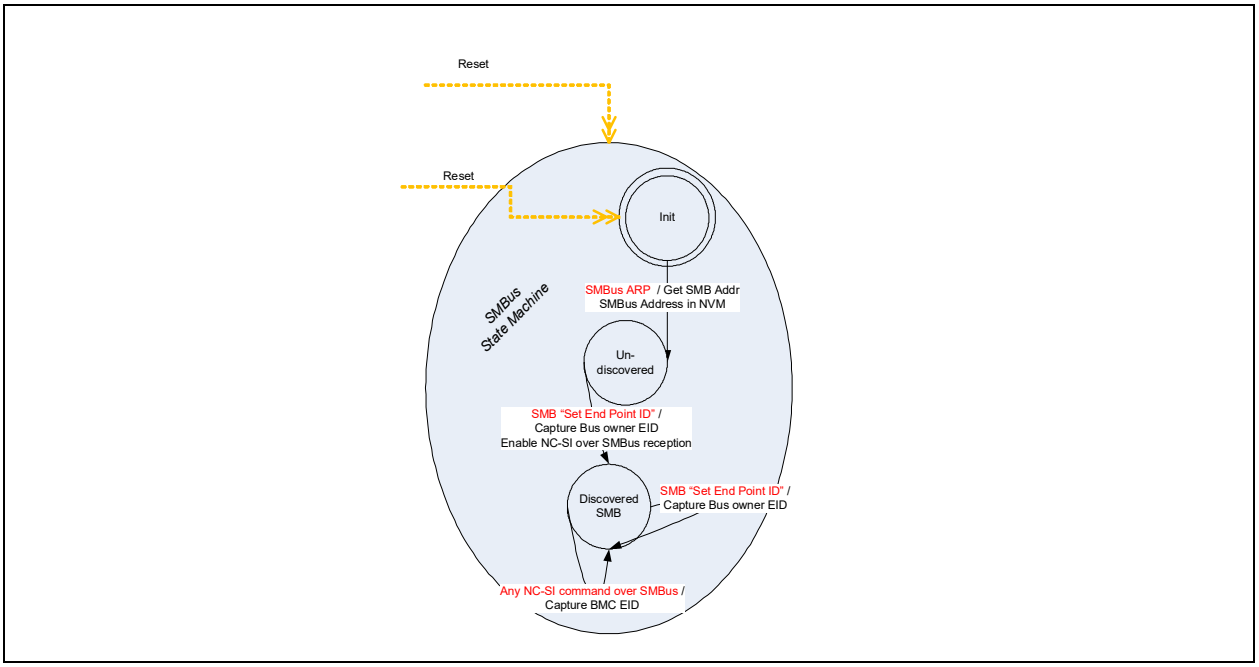


Figure 9-12. MCTP Bus Transition State Machine

9.7.2.2.1 Initial Assignment Flow

- At power on, the integrated 10 GbE LAN controller or BMC MCTP channel is connected to the SMBus, is not assigned an EID and is in undiscovered state.
- The bus owner might preform an SMBus ARP cycle to assign an SMBus address to the integrated 10 GbE LAN controller or to the BMC. Otherwise, a fixed address may be used. It is assumed that the SMBus address does not change after initialization time.
- The bus owner performs an EID assignment using a *Set Endpoint ID* MCTP command. The integrated 10 GbE LAN controller or the BMC captures the SMBus address of the bus owner from the *SMBus Source Slave address* field, the bus owner EID from the source endpoint ID field and the integrated 10 GbE LAN controller/BMC EID from the destination endpoint ID field in the MCTP header as described in section 10.3 of DSP0236. The integrated 10 GbE LAN controller/BMC is now in discovered state.
- The BMC might detect the integrated 10 GbE LAN controller EID using one of the two following modes:
  - Static configuration of the integrated 10 GbE LAN controller SMBus address in the BMC database and Get Routing Table Entries command to find the EID matching the SMBus address.

— Get all endpoints through a Get Routing Table Entries command and find endpoints supporting NC-SI using the Get Message Type Support command for each endpoint.

- Once the integrated 10 GbE LAN controller is found, the BMC might send a Clear Initial State command to the integrated 10 GbE LAN controller to start the NC-SI configuration. The the integrated 10 GbE LAN controller captures the BMC SMBus address and BMC EID from any NC-SI command received.
- After the NC-SI channels are enabled, traffic might be sent using the BMC and the integrated 10 GbE LAN controller addresses previously discovered.
- The BMC might send a Get UUID command to get a unique identifier of the the integrated 10 GbE LAN controller that might be used later for re-connection upon topology changes.
- If a Firmware Reset occurs, a Discovery Notify MCTP message should be sent by the integrated 10 GbE LAN controller to restart the flow.

9.7.3 MCTP Over SMBus

The message format used for NC-SI over MCTP over SMBus is as follows:

+0								+1								+2								+3							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Destination Slave Address							0	Command Code = MCTP = 0Fh								Byte count								Source Slave Address							1
MCTP Reserved				Header version = 1				Destination endpoint ID								Source endpoint ID								SOM		EOM		SEQ#		TO	Tag
IC[1]	Message Type = 0x02							NC-SI Command/Pass Through data																							
.....																															
NC-SI Command/Pass Through data																															
PEC																															

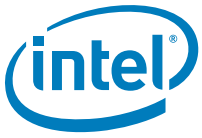
1. IC = 0

SMBus header/PEC
MCTP header
NC-SI header and payload

9.7.3.1 SMBus Discovery Process

The integrated 10 GbE LAN controller follows the discovery process described in section 6.5 of the MCTP SMBus/I2C Transport Binding Specification (DSP0237). It indicates support for ASF in the SMBus getUID command (see Section 9.5.5.4). It responds to any SMBus command using the MCTP command code - so that the bus owner knows the integrated 10 GbE LAN controller supports MCTP.

Note: MCTP commands over SMBus are received from any master address and are answered to the sender. There is no capturing of the bus owner address from any specific command.



9.7.3.2 MCTP over SMBus Special Features

The integrated 10 GbE LAN controller supports the following optional feature of MCTP when running over SMBus:

1. Simplified MCTP mode.
2. Fairness arbitration.

9.7.3.2.1 Simplified MCTP Mode

For some point-to-point implementations of MCTP the assembly process is simplified. In this mode, the Destination EID, Source EID, Packet sequence number, Tag Owner (TO) bit and Message tag are ignored and the assembly is based only on the SOM and EOM bits. This bit is set according to the *Simplified MCTP* bit in the MCTP configuration word in the NVM.

This mode is relevant only for MCTP over SMBus traffic and when the Redirection Sideband Interface is set to 10b (MCTP over SMBus only - no pass through).

9.7.3.2.2 Fairness Arbitration

When sending MCTP messages over SMBus, the integrated 10 GbE LAN controller should respect the fairness arbitration as defined in section 6.13 of DSP0237 when sending MCTP messages.

9.7.4 NC-SI over MCTP

The integrated 10 GbE LAN controller 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 expose the NC-SI over MCTP capabilities.
2. The format of the packets is modified to account for the new transport layer described in the sections that follow.

9.7.4.1 NC-SI Packets Format

NC-SI over MCTP defines two different message type for pass through and for control packets.

Packets with a message type equal to the *Control packets message type* field (default = 0x02) in the NVM are NC-SI control packets (commands, responses and AENs) and packets with a message type equal to the *Pass through packets message type* field (default = 0x03) in the NVM are NC-SI pass through packets

9.7.4.1.1 Control Packets

The format used for control packets (commands, responses and AENs) is as follows:

+0								+1								+2								+3							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SMBus or PCIe header																															
MCTP Reserved								Header version = 1								Destination endpoint ID								Source endpoint ID							
																								SOM				EOM		SEQ#	
TO = 1				Tag																											
IC = 0		Message Type = Control Packets Message type (0x02)						MC ID = 0x00								Header revision								Reserved							
IID								Command								Channel ID[1]								Reserved							
																								Payload Length[11:8]							
Payload Length[7:0]								Reserved																							
Reserved																															
Reserved								Command Data																							
....																															
Command Data																Checksum															
Checksum																															

1. The channel ID is defined as described in Section 9.2.2.2.

Note that the MAC header and MAC FCS present when operating over NC-SI are not part of the packet in MCTP mode.

9.7.4.1.2 Pass-Through Packets

The format used for pass-through packets is as follows. This format is the same for either packets received from the network or packets received from the host.

The CRC is never included in the packet. In receive, the CRC is checked and removed by the integrated 10 GbE LAN controller in transmit, the CRC is added by the integrated 10 GbE LAN controller.

+0								+1								+2								+3							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
SMBus or PCIe header																															
MCTP Reserved								Header version = 1								Destination endpoint ID								Source endpoint ID							
																								SOM				EOM		SEQ#	
TO = 1								Tag																							
IC = 0								Message Type = Pass Through Packets Control Type								DA															
DA																								SA							
SA																															
SA																Ether type								Ethernet Packet							
Ethernet packet																															
....																															
....																															

9.7.5 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 9.7.5.1.
2. A subset of the NC-SI commands used in the regular NC-SI interface, including all the OEM commands as described in Section 9.6.2 (NC-SI programming I/F). The specific commands supported are listed in Table 9-28 and Table 9-31.

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.

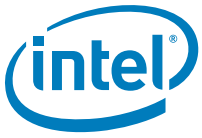
9.7.5.1 MCTP Commands Support

Table 9-40 lists the MCTP commands supported by the integrated 10 GbE LAN controller.

Table 9-40. MCTP commands support

Command Code	Command Name	General Description	Integrated 10 GbE LAN Controller Support as Initiator	Integrated 10 GbE LAN Controller 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 9.7.5.1.1 for details.	No	Yes
0x03	Get Endpoint UUID	Retrieves a per-device unique UUID associated with the endpoint. See Section 9.7.5.1.2 for details.	No	Yes
0x04	Get MCTP Version Support	Lists which versions of the MCTP control protocol are supported on an endpoint. See Section 9.7.5.1.3 for details.	No	Yes
0x05	Get Message Type Support	Lists the message types that an endpoint supports. See Section 9.7.5.1.3.1 for details.	No	Yes
0x06	Get Vendor Defined Message Support	Used to discover an MCTP endpoint's vendor specific MCTP extensions and capabilities. See Section 9.7.5.1.4 for details.	No	Yes[1]
0x07	Resolve Endpoint ID	Used to get the physical address associated with a given EID.	No	N/A
0x08	Allocate Endpoint IDs	Used by the bus owner to allocate a pool of EIDs to an MCTP bridge.	N/A	N/A
0x09	Routing Information Update	Used by the bus owner to extend or update the routing information that is maintained by an MCTP bridge.	N/A	N/A
0x0A	Get Routing Table Entries	Used to request an MCTP bridge to return data corresponding to its present routing table entries.	No	N/A
0x0B	Prepare for Endpoint Discovery	Used to direct endpoints to clear their discovered flags to enable them to respond to the Endpoint Discovery command.	N/A	Yes[1]
0x0C	Endpoint Discovery	Used to discover MCTP-capable devices on a bus, provided that another discovery mechanism is not defined for the particular physical medium.	No	Yes[1]
0x0D	Discovery Notify	Used to notify the bus owner that an MCTP device has become available on the bus.	Yes[1]	N/A
0x0E	Get Network ID	Used to get the MCTP network ID.	No	No
0x0F	Query Hop	Used to discover what bridges, if any, are in the path to a given target endpoint and what transmission unit sizes the bridges 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.



9.7.5.1.1 Get Endpoint ID

The Get Endpoint ID response of the integrated 10 GbE LAN controller 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.

9.7.5.1.2 Get Endpoint UUID

The UUID returned is calculated according to the following function:

Time Low = Read from MCTP UUID - Time Low LSB/MSB NVM words of Sideband Configuration Structure.

Time mid = Read from MCTP UUID - Time Mid NVM word of Sideband Configuration Structure

Time High and version = Read from MCTP UUID - Time High and version NVM word of Sideband Configuration Structure

Clock Sec and Reserved = Read from MCTP UUID - Clock Seq NVM word of Sideband Configuration Structure

Node = MAC address as taken from the *PCI_SERL* and *PCI_SERH* registers.

9.7.5.1.3 Get MCTP Version Support

The following table lists the returned value according to the requested message type.

Byte	Description	Message Type					
		0xFF(Base)	0x00 (Control Protocol Message)	0x02 (NC-SI Over MCTP)	0x03 (Ethernet)	0x7E (PCIe Based VDM Messages)	All Other or Unsupported Messages
1	Completion Code	0x0					0x80
2	Version Number entry count	3	3	2	2	2	0
6:3	Version number entry	0xF1F0FF00 (1.0)	0xF1F0FF00 (1.0)	0xF1F0F000 (1.0.0)	0xF1F0F000 (1.0)	0xF1F0FF00 (1.0)	0
9:7	Version number entry 2	0xF1F1F000 (1.1.0)	0xF1F1F000 (1.1.0)	0xF1F1F000 (1.1.0)	0xF1F1F000 (1.1.0)	0xF1F1F000 (1.1.0)	
13:10	Version number entry 3	0xF1F2F000 (1.2.0)	0xF1F2F000 (1.2.0)				

9.7.5.1.3.1 Get Message Type Support Command

The Get Message type support response of the integrated 10 GbE LAN controller is listed in the following table:

Byte	Description	Value
1	Completion Code	0x00.
2	MCTP Message Type Count	0x01/0x02/0x03 - The integrated 10 GbE LAN controller supports up to three additional message types, depending on the mode of operation and the bus used.
3:5	List of Message Type numbers	0x02 (NC-SI over MCTP).
		0x03 (Ethernet). If pass through is supported.
		0x7E (PCIe based VDM messages) - over PCIe only.

9.7.5.1.4 Get Vendor Defined Message Support Command

This command is supported only for MCTP over PCIe.

The Get Vendor Defined Message type support response of the integrated 10 GbE LAN controller is listed in the following table if the Vendor ID Set Selector equal 0x00:

Byte	Description	Value
1	Completion Code	0x00.
2	Vendor ID Set Selector	0xFF = No more capability sets.
2:4	Vendor ID	0x008086 (PCI ID indicator + Intel vendor ID).
5:6	Version	0x0100 (version 1.0).

9.7.5.1.5 Set Endpoint ID Command

The integrated 10 GbE LAN controller supports the Set EID and Force EID operations defined in the Set Endpoint ID command. When operating over PCIe, the Set Discovered Flag operation is also supported. As endpoints in the integrated 10 GbE LAN controller can be set only through their own interface, Set EID and Force EID are equivalent. The Reset EID operation is not supported by the integrated 10 GbE LAN controller.

The Set Endpoint ID response of the integrated 10 GbE LAN controller is listed 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.
3	EID Setting	If the EID setting was accepted, this value will match the EID passed in the request. Otherwise, this value returns the present EID setting.
4	EID Pool Size	Always return a zero.

9.8 Manageability Host Interface

This section details host interaction with the manageability portion of the integrated 10 GbE LAN controller. The information within this section is only available to the host driver; the BMC does not have access.

9.8.1 HOST CSR Interface (Function 1/0)

The software device driver of all functions communicates with the manageability block through CSR access. The manageability is mapped to address space 0x15800 to 0x15FFF on the slave bus of each function.

Note: Writing to address 0x15800 from any function is targeted to the same address in the RAM.

9.8.2 Host Slave Command Interface to Manageability

This interface is used by the software device driver for several of the commands and for delivering various types of data in both directions (Manageability-to-Host and Host-to-Manageability).

The address space is separated into two areas:

- Direct access to the internal data RAM: The internal shared (between Firmware and Software) RAM is mapped to address space 0x15800 to 0x15EFF. Writing/reading to this address space goes directly to the RAM.
- Control register located at address 0x15F00.

9.8.2.1 Host Slave Command Interface Low Level Flow

This interface is used for the external host software to access the manageability subsystem. Host software writes a command block or read data structure directly from the data RAM. Host software controls these transactions through a slave access to the HICR control register (see Section 8.2.2.21.7).

The following flow shows the process of initiating a command to the manageability block:

1. Software clears the *FWSTS.FWRI* flag (clear by write one) to clear any previous firmware reset indications.
2. Software device driver takes ownership of the management host interface using the flow described in Section 3.6.
3. The software device driver reads the HOST Interface Control register (See Section 8.2.2.21.7) and checks that the *Enable* (*HICR.En*) bit is set.
4. The software device driver writes the relevant command block into the RAM area that is mapped to addresses 0x15800-0x15EFF.
5. The software device driver sets the *Command* (*HICR.C*) bit in the HOST Interface Control register (See Section 8.2.2.21.7). Setting this bit causes an interrupt to the ARC (can be masked).

6. The software checks the *FWSTS.FWRI* flag to make sure a firmware reset didn't occur during the command processing. If this bit is set, the command might have failed.
7. The software device driver polls the HOST Interface Control register for the *Command* (*HICR.C*) bit to be cleared by firmware. The command should complete within half a second.
8. When firmware finishes with the command, it clears the *Command* (*HICR.C*) bit (if firmware replies with data, it should clear the bit only after the data is placed in the shared RAM area where the software device driver can read it).

If the software device driver reads the HOST Interface Control register and the *HICR.SV* bit is set to 1b, then there is a valid status of the last command in the shared RAM. If the *HICR.SV* bit is not set, then the command has failed with no status in the RAM.

On completion of access to the shared RAM, the software device driver should release ownership of the shared RAM using the flow described in Section 3.6.

9.8.2.2 Host Interface Structure

9.8.2.2.1 Host Interface Command Structure

Table 9-41 lists the structure used by the software device driver to send a command to firmware using the Host A Slave command interface (shared RAM mapped to addresses 0x15800-0x15EFF).

Table 9-41. Host Driver Command Structure

#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	Reserved	7:0		Reserved
3	Checksum	7:0	Defined Below	Checksum signature. If the value is 0xFF, the checksum is not checked by the firmware.
255:4	Data Buffer	7:0	Command Dependent	Command Specific Data. Minimum buffer size: 0. Maximum buffer size: 252.

9.8.2.2.2 Host Interface Status Structure

Table 9-42 lists the structure used by firmware to return a status to the software device driver via the Host Slave command interface. A status is returned after a command has been executed.

Table 9-42. Status Structure Returned to Host Driver

#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.

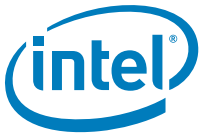


Table 9-42. Status Structure Returned to Host Driver (Continued)

2	Return Status	7:0	Depends on Command Executing Results.	0x1 = Status OK. 0x2 = Illegal command ID. 0x3 = Unsupported command. 0x4 = Illegal payload length. 0x5 = Checksum failed. 0x6 = Data error. 0x7 = Invalid parameter. 0x8 -0x7F Reserved. 0x80 - 0xFF Command Specific Errors. Might be used by individual commands for command specific errors.
3	Checksum	7:0	Defined in the sections that follow.	Checksum signature.
255:4	Data Buffer		Status Dependent.	Status configuration parameters. Minimum Buffer Size: 0. Maximal Buffer Size: 252. ***Note:*** If return status is not Status OK the data buffer is empty.

9.8.2.2.3 Checksum Calculation Algorithm

The host command/status structure is summed with this field cleared to 0b. The calculation is done using 8-bit unsigned math with no carry. The inverse of this sum is stored in this field (0b minus the result). Result: The current sum of this buffer (8-bit unsigned math) is 0b.

9.8.3 Host Interface Commands

9.8.3.1 Driver Info Host Command

This command is used to provide the driver information in NC-SI mode.

Table 9-43. Driver Info Host Command

Byte	Name	Bit	Value	Description
0	Command	7:0	0xDD	Driver info command.
1	Buffer Length	7:0	0x5	Port Number + 4 bytes of the driver information.
2	Reserved	7:0	0x0	Reserved.
3	Checksum	7:0		Checksum signature of the Host command.
4	Function Number	7:0	Function Number	Indicates the function currently reporting its driver 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:

Table 9-44. Driver Info Host Status

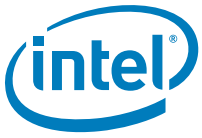
Byte	Name	Bit	Value	Description
0	Command	7:0	0xDD	Driver Info command.
1	Buffer Length	7:0	0x0	No data in return status.
2	Return Status	7:0	0x1	See Table 9-42
3	Checksum	7:0		Checksum signature.

9.8.3.2 Disable RXEN Command

This command is used to allow the driver to request a safe disable of Receive Enable.

Table 9-45. Disable RXEN Command

Byte	Name	Bit	Value	Description
0	Command	7:0	0xDE	Driver info command.
1	Buffer Length	7:0	0x1	1 data byte attached to this command (the port number)
2	Reserved	7:0	0x0	Reserved
3	Checksum	7:0		Checksum signature of the Host command.
4	Port Number	7:0	Port Number	Indicates the port for which the operation is requested



Following is the status returned on this command:

Table 9-46. OS2BMC Control Status

Byte	Name	Bit	Value	Description
0	Command	7:0	0xDE	Driver Info command
1	Buffer Length	7:0	0x0	No data in return status
2	Return Status	7:0	0x1	See Table 9-42
3	Checksum	7:0		Checksum signature

The Firmware should execute the following flow when receiving this command:

- Store OS2BMC enable bit.
- Clear OS2BMC enable bit.
- Clear RXEN
- Restore OS2BMC enable bit to original value.
- Return from command.

9.8.3.3 Flash I/F interface

These commands enable buffers of up to 1 KB of data. In order to do this, the *buffer length* field is expanded to 2 bytes.

Note: This field in the command is in little endian order - i.e. byte 1 contains the MSB part of the buffer length and byte 2 contains the LSB part of the buffer length.

9.8.3.3.1 Flash Read

This command is used to request a read from the Flash. This command enables access to the region of the Flash owned by the device (in case of non-shared SPI, it is the entire Flash).

This command always returns the value from the Flash, even if read within the shadow RAM boundaries.

Table 9-47. Flash Read Command

Byte	Name	Bit	Value	Description
0	Command	7:0	0x30	Flash read.
2:1	Buffer Length	15:0	0x6	
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to read from the Flash.
9:8	Length to Read	15:0		How many bytes to read. Can be up to 1024.

Table 9-48. Flash Read Response

Byte	Name	Bit	Value	Description
0	Command	7:0	0x30	Flash read.
1	Buffer Length (LS Byte)	7:0		Buffer length[7:0].
2	Buffer Length (MS Byte)	7:5		Buffer length[10:8].
2	Return Status	4:0		See Table 9-42.
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to read from the Flash.
9:8	Length Read	15:0		How many bytes where actually read.
11:10	Reserved			Reserved.
12: Length Read+12	Read Data			The requested data read from Flash.

9.8.3.3.2 Shadow RAM Read

This command is used to request a read from the shadow RAM. Requesting addresses above shadow RAM boundaries causes an error.

Table 9-49. Shadow RAM Read Command

Byte	Name	Bit	Value	Description
0	Command	7:0	0x31	Shadow RAM read.
2:1	Buffer Length	15:0	0x6	
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to read from the Shadow Ram.
9:8	Length to Read	15:0		How many bytes to read. Can be up to 1024.

Table 9-50. Shadow RAM Read Response

Byte	Name	Bit	Value	Description
0	Command	7:0	0x31	Shadow RAM read.
1	Buffer Length (LS Byte)	7:0		Buffer length[7:0].
2	Buffer Length (MS bits)	7:5		Buffer length[10:8].
2	Return Status	4:0		See Table 9-42.
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to read from the Shadow RAM

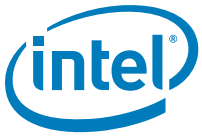


Table 9-50. Shadow RAM Read Response

9:8	Length Read	15:0		How many bytes where actually read.
11:10	Reserved			Reserved.
12: Length Read+12	Read Data			The requested data read from Flash.

9.8.3.3.3 Flash Write

This command is used to update the Flash sections out of the shadow RAM. This command enables access to the writable region of the Flash owned by the device that is not part of the shadow RAM. If not all the area to erase is writable, the command returns an error.

Table 9-51. Flash Write Command

Byte	Name	Bit	Value	Description
0	Command	7:0	0x32	Flash write.
2:1	Buffer Length	15:0	0x8 + Length to Write	
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to write to the Flash.
9:8	Length to Write	15:0		How many bytes to write. Can be up to 1024.
11:10	Reserved			Reserved.
12: Length to Write +12	Write Data			The data to write to the flash.

Table 9-52. Flash Write Response

Byte	Name	Bit	Value	Description
0	Command	7:0	0x32	Flash write.
1	Buffer Length	7:0	0x6	
2	Return Status	7:0		See Table 9-42.
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to write to the Flash.
9:8	Data Written	15:0		How many bytes where actually written.

9.8.3.3.4 Shadow RAM Write

This command is used to update the shadow RAM. It enables write to the writable parts of the shadow RAM. If not all the area to write is writable, or not all the area is in the shadow RAM range, the command returns an error.

Table 1-1. Shadow RAM Write Command

Byte	Name	Bit	Value	Description
0	Command	7:0	0x33	Shadow RAM write.
2:1	Buffer Length	15:0	0x8 + Length to Write	
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to write to the Shadow Ram.
9:8	Length to Write	15:0		How many bytes to write. Can be up to 1024.
11:10	Reserved			Reserved.
12: Length to Write+12	Write Data			The data to write to the shadow RAM.

Table 9-53. Shadow RAM Write Response

Byte	Name	Bit	Value	Description
0	Command	7:0	0x33	Shadow RAM write.
1	Buffer Length	7:0	0x6	
2	Return Status	7:0		See Table 9-42.
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address to write to the Shadow Ram.
9:8	Data Written	15:0		How many bytes where actually written.

9.8.3.3.5 Flash Module Update

This command is used to update a secured module. After a successful response to this command, for a firmware code update, an Apply Update command (Section 9.8.3.3.6) must be sent to activate the new firmware.

Table 9-54. Flash Module Update Command

Byte	Name	Bit	Value	Description
0	Command	7:0	0x34	Flash module update.
2:1	Buffer Length	15:0	0x1	
3	Checksum	7:0		Checksum signature
4	Module ID	7:0		Which module to update: firmware code 0x1. PHY firmware image 0x5. Option ROM 0xFE.

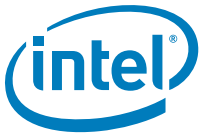


Table 9-55. Flash Module Update Response

Byte	Name	Bit	Value	Description
0	Command	7:0	0x34	Flash module update.
1	Buffer Length	7:0	0x0	
2	Return Status	7:0		See Table 9-42. If the authentication failed, an authentication error (0x80) is returned.
3	Checksum	7:0		Checksum signature.

9.8.3.3.6 Apply Update

This command is used to request the firmware to switch to the new uploaded code. This command involves a firmware reset, so no response should be expected after this command is given. The software device driver might read the *FWRESETCNT* register before and after the command is given to check if the reset took place.

Note: An Apply Update command sent not after a successful Flash Module Update Command is ignored.

If after 100 ms the FWRESETCNT register has not increased, software should assume the command failed.

Table 9-56. Apply Update Command

Byte	Name	Bit	Value	Description
0	Command	7:0	0x38	Apply update.
2:1	Buffer Length	15:0	0x0	
3	Checksum	7:0		Checksum signature.

9.8.3.3.7 Flash Block Erase

This command is used to erase some of the Flash sections. This command enables access to the writable region of the Flash owned by the device that is not part of the shadow RAM. If not all the area to erase is writable, the command returns an error.

Table 9-57. Flash Block Erase Command

Byte	Name	Bit	Value	Description
0	Command	7:0	0x35	Flash block erase.
2:1	Buffer Length	15:0	0x5	

Table 9-57. Flash Block Erase Command

3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address of block to erase – must be 4 KB aligned.
8	Number of Sectors to Erase	7:0		How many 4 KB sectors to erase. In order to avoid a too long command, this number should be no larger than three.

Table 9-58. Flash Block Erase Response

Byte	Name	Bit	Value	Description
0	Command	7:0	0x35	Flash block erase.
1	Buffer Length	7:0	0x5	
2	Return Status	7:0		See Table 9-42.
3	Checksum	7:0		Checksum signature.
7:4	Address	31:0		Address of block to erase.
8	Erased Blocks	7:0		How many blocks where actually erased.

9.8.3.3.8 Shadow RAM Dump

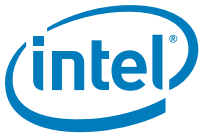
This command is used to trigger a shadow RAM dump. It should be used after updating a module in Shadow RAM.

Table 9-59. Shadow RAM Dump Command

Byte	Name	Bit	Value	Description
0	Command	7:0	0x36	Shadow RAM dump.
2:1	Buffer Length	15:0	0x0	
3	Checksum	7:0		Checksum signature.

Table 9-60. Shadow RAM Dump Response

Byte	Name	Bit	Value	Description
0	Command	7:0	0x36	Shadow RAM dump.
1	Buffer Length	7:0	0x0	
2	Return Status	7:0		See Table 9-42.
3	Checksum	7:0		Checksum signature.



9.8.3.3.9 Flash Info

This command is used to provide information about the Flash

Table 9-61. Flash Info Command

Byte	Name	Bit	Value	Description
0	Command	7:0	0x37	Flash information.
2:1	Buffer Length	15:0	0x0	
3	Checksum	7:0		Checksum signature.

Table 9-62. Flash Info Response

Byte	Name	Bit	Value	Description
0	Command	7:0	0x37	Flash information.
1	Buffer Length	7:0	0x8	
2	Return Status	7:0		See Table 9-42.
3	Checksum	7:0		Checksum signature.
7:4	Flash Size	31:0		Size of Flash available to this device. When the Flash is shared this is the size of the region allocated to the integrated 10 GbE LAN controller.
11:8	Reserved			Reserved

9.8.3.4 PHY Token Request Command

This command is used to manage the access to the SOC_GEN_CTRL_REG register.

For details on the flow see Section 2.7.3.1.

If the software device driver fails to send a token release command after 1 second, firmware considers the software device driver to be unavailable and release the relevant shared resource request in the SOC_GEN_CTRL_REG register.

Table 9-63. PHY Token Request

Byte	Name	Bit	Value	Description
0	Command	7:0	0xA	PHY Token Request.
1	Buffer length	7:0	0x2	
2	Reserved	7:0		
3	Checksum	7:0		Checksum signature.
4	Port Number	7:0	Port Number	Indicates the physical port number.
5	Command Type	7:0		0b = Token request. 1 = Token release.

Table 9-64. PHY Token Response

Byte	Name	Bit	Value	Description
0	Command	7:0	0xA	PHY Token Response.
1	Buffer length	7:0	0x0	
2	Status	7:0		0x1 = Status OK. 0x80 = Retry.
3	Checksum	7:0		Checksum signature.

9.8.4 Software and Firmware Synchronization

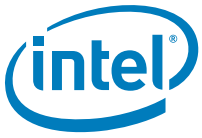
Software and firmware synchronize accesses to shared resources in the integrated 10 GbE LAN controller through a semaphore mechanism and a shared configuration register between the host interface of the two ports and the firmware. This semaphore enables synchronized accesses to the following shared resources:

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

The *SW_FW_SYNC.REGSMP* bit is used as a semaphore mechanism between software and firmware. Once software or firmware takes control over this semaphore flag, it can access the SW_FW_SYNC register and claim ownership over specific resources. The SW_FW_SYNC includes pairs of bits (one owned by software and the other by firmware), where each pair of bits controls a different resource. A resource is owned by software or firmware when its respective bit is set. It is illegal to have both pair bits set at the same time. Following are the required sequences for gaining and releasing control over shared resources:

9.8.4.1 Gaining Control of Shared Resource by Software

- The software device driver checks that the software device driver of the other LAN function does not use the software/firmware semaphore
  - The software device driver polls the *SWSM.SMBI* bit until it is read as 0b or time expires (recommended expiration is ~10 ms + expiration time used for the *SW_FW_SYNC.REGSMP*).
  - If the *SWSM.SMBI* is found at 0b, the semaphore is taken. Note that following this read cycle the hardware auto sets the bit to 1b.
  - If time expired, it is assumed that the software device driver of the other function malfunctioned. The software proceeds to the next step.
- The software device driver checks that the firmware does not use the software/firmware semaphore and then takes its control.
  - Software polls the SW_FW_SYNC.REGSMP bit until it is read as 0b or time expires (recommended expiration is ~50 ms). If time has expired, the software assumes that the firmware malfunctioned and proceeds to the next step, while ignoring the firmware bits in the SW_FW_SYNC register.
- Software takes control of the requested resource(s).



— The software device driver reads the firmware and software bit(s) of the requested resource(s) in the SW_FW_SYNC register. If the bit(s) is cleared, the resource(s) is accessible [such as no other entity owns the resource(s)]. In this case the software device driver sets the software bit(s) of the requested resource(s) in the SW_FW_SYNC register. Software then clears the *SW_FW_SYNC.REGSMP* and *SWSM.SMBI* bits (releasing the software/firmware semaphore register), and can use the specific resource(s).

— Otherwise (either firmware or software of the other LAN function owns the resource), software clears the *SW_FW_SYNC.REGSMP* and *SWSM.SMBI* bits and then repeats the entire process after some delay (recommended 5-10 ms).

- If the resources are not released by the software device driver of the other LAN function in a timely manner (recommended expiration time is ~1 second), the software device driver can assume that the other software device driver malfunctioned. In that case, the software device driver should clear all software flags that it does not own (including *SW_FW_SYNC.REGSMP* bit) and then repeat the entire process once again.
- If the resource is not released by the firmware (recommended expiration time for firmware is ~50 ms) software can assume that the firmware malfunctions. In that case, the software device driver should set the software bit(s) of the requested resource(s) while ignoring the corresponding firmware bits in the SW_FW_SYNC register.

Note that the firmware initializes its semaphore flags as part of its initialization flow. The software semaphores are not reset.

9.8.4.2 Releasing a Shared Resource by Software

- The software device driver takes control over the software/firmware semaphore as previously described for gaining shared resources.
- The software device driver clears the bit(s) of the released resource(s) in the SW_FW_SYNC register.
- The software device driver releases the software/firmware semaphore by clearing the *SW_FW_SYNC.REGSMP* and *SWSM.SMBI* bits.
- Software should delay (recommended 5-10 ms) before trying to gain the semaphore immediately following its release.

9.8.4.3 Gaining Control of Shared Resource by Firmware

- The firmware takes control over the software/firmware semaphore (SW_FW_SYNC register)
  — The firmware polls the *SW_FW_SYNC.REGSMP* bit until it is read as ‘0b or timeout expires (recommended expiration time is ~10 ms).
  — If timeout has expired, the firmware clears the *SW_FW_SYNC.REGSMP* bit (it is assumed the software device driver is not functional).
- The firmware takes ownership of the requested resources.
  — The firmware reads the software bit(s) corresponding to the requested firmware resource(s) in the SW_FW_SYNC register.
  — If the *Software* bit is cleared (such as the software device driver does not own the resource), firmware sets the firmware bit(s) of the requested resource(s). Firmware then clears the *SW_FW_SYNC.REGSMP* bit (releasing the software/firmware semaphore) and can use the specific resource(s).

— Otherwise (software owns the resource), the firmware clears the *SW_FW_SYNC.REGSMP* bit and then repeats the previous process after some delay (recommended delay of 5-10 ms).

- If the resources owned by software are not released in a timely manner (~1 second), the firmware forces its ownership over the requested resources. Firmware clears the software flags of the requested resources in the SW_FW_SYNC register (assuming the software that set those flags is not functional).

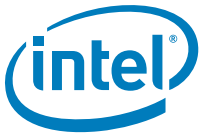
9.8.4.4 Releasing a Shared Resource by Firmware

- Firmware takes control over the software/firmware semaphore as previously described for gaining shared resources.
- Firmware clears the bit(s) of the selected resource(s) in the SW_FW_SYNC register.
- Firmware releases the software/firmware semaphore by clearing the *SW_FW_SYNC.REGSMP* bit.
- Firmware should delay before trying to gain the selected resource semaphore immediately following its release (recommended 5-10 ms).

9.9 Host Isolate Support

If a BMC decides that a malicious software prevents its usage of the LAN, it might decide to isolate the integrated 10 GbE LAN controller from its software device driver. This is done using the TCO reset command (Section 9.6.3.14).

If TCO isolate is enabled in the shared SPI Flash, The TCO Isolate command disables PCIe write operations to the LAN port. As the software device driver needs to access the CSR space in order to provide descriptors to the integrated 10 GbE LAN controller, this operation also stops the network traffic including OS-to-BMC and BMC-to-OS traffic as soon as the existing transmit and receive descriptor queues are exhausted.



Appendix A Packet Formats

A.1 Legacy Packet Formats

A.1.1 ARP Packet Formats

A.1.1.1 ARP Request Packet

Offset	# Of Bytes	Field	Value (In Hex)	Action
0	6	Destination Address		Compare
6	6	Source Address		Stored
12	E=(0/4/6/8)	Outer Tag (Outer VLANor E-tag)	0x8100 **** 0x893F **********	Ignore
12 + E	S=(0/4)	Possible VLAN Tag		Stored
12 + E + S	2	Type	0x0806	Compare
14 + E + S	2	HW Type	0001	Compare
16 + E + S	2	Protocol Type	0x0800	Compare
18 + E + S	1	Hardware Size	06	Compare
19 + E + S	1	Protocol Address Length	04	Compare
20 + E + S	2	Operation	0001	Compare
22 + E + S	6	Sender HW Address	-	Stored
28 + E + S	4	Sender IP Address	-	Stored
32 + E + S	6	Target HW Address	-	Ignore
38 + E + S	4	Target IP Address	ARP IP address	Compare

A.1.1.2 ARP Response Packet

Offset	# of bytes	Field	Value
0	6	Destination Address	ARP Request Source Address.
6	6	Source Address	Programmed from shared SPI Flash or BMC.
12	E=(0/4/6/8)	Outer Tag (Outer VLAN or E-tag)	
12 + E	S=(0/4)	Possible VLAN Tag	From ARP Request.
12 + E + S	2	Type	0x0806

Offset (Continued)	# of bytes	Field	Value
14 + E + S	2	HW Type	0x0001
16 + E + S	2	Protocol Type	0x0800
18 + E + S	1	Hardware Size	0x06
19 + E + S	1	Protocol Address Length	0x04
20 + E + S	2	Operation	0x0002
22 + E + S	6	Sender HW Address	Programmed from shared SPI Flash or BMC.
28 + E + S	4	Sender IP Address	Programmed from shared SPI Flash or BMC.
32 + E + S	6	Target HW Address	ARP Request Sender HW Address.
38 + E + S	4	Target IP Address	ARP Request Sender IP Address.

A.1.1.3 Gratuitous ARP Packet

Offset	# of bytes	Field	Value
0	6	Destination Address	Broadcast address.
6	6	Source Address	
12	E=(0/4/6/8)	Outer Tag (Outer VLAN or E-tag)	
12 + E	S=(0/4)	Possible VLAN Tag	
12 + E + S	D=(0/8)	Possible Length + LLC/SNAP Header	
12 + E + S + D	2	Type	0x0806
14 + E + S + D	2	Hardware Type	0x0001
16 + E + S + D	2	Protocol Type	0x0800
18 + E + S + D	1	Hardware Size	0x06
19 + E + S + D	1	Protocol Address Length	0x04
20 + E + S + D	2	Operation	0x0001
22 + E + S + D	6	Sender Hardware Address	
28 + E + S + D	4	Sender IP Address	
32 + E + S + D	6	Target Hardware Address	
38 + E + S + D	4	Target IP Address	

A.1.2 IP and TCP/UDP Headers for TSO

This section outlines the format and content for the IP, TCP and UDP headers. The integrated 10 GbE LAN controller requires baseline information from the device driver in order to construct the appropriate header information during the segmentation process.

Header fields that are modified by the integrated 10 GbE LAN controller 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 is fetched from memory in Little Endian format.

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
Version / IP Hdr Length	TYPE of service	Total length (**IP header + payload** length)	
Identification		Flags	Fragment Offset
Time to Live	Layer 4 Protocol ID	Header Checksum	
Source Address			
Destination Address			
Options			

Figure A.1. IPv4 Header (Traditional Representation - most left byte first on the wire)

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 A.2. IPv4 Header (Little Endian Order - Byte 0 first on the wire)

Identification is increased on each packet.

Flags Field Definitions:

The Flags field is defined as follows. Note that hardware does not evaluate or change these bits.

- MF - More Fragments
- NF - No Fragments
- Reserved

The integrated 10 GbE LAN controller does TCP segmentation, not IP Fragmentation. IP Fragmentation might occur in transit through a network's infrastructure.

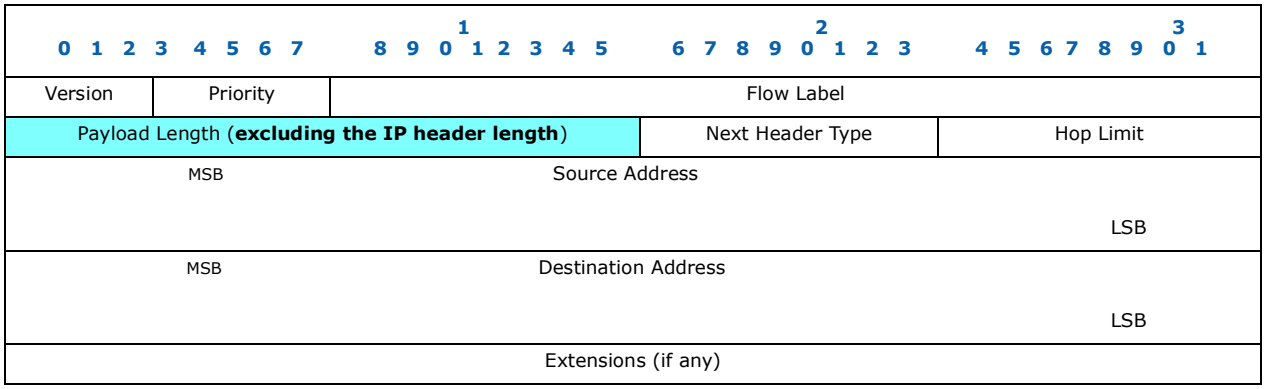


Figure A.3. IPv6 Header (Traditional Representation - most left byte first on the wire)

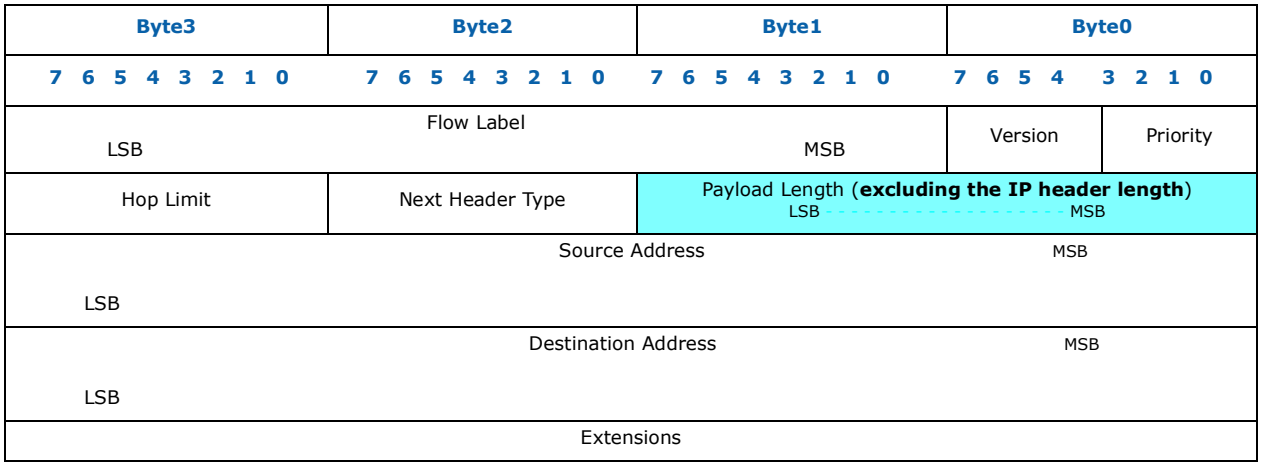


Figure A.4. IPv6 Header (Little Endian Order - byte 0 first on the wire)

A TCP or UDP frame uses a 16 bit wide one's complement checksum. The checksum word is computed on the outgoing TCP or UDP header and payload, and on the pseudo header. Details on checksum computations are provided in Section 6.2.4.6.

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.

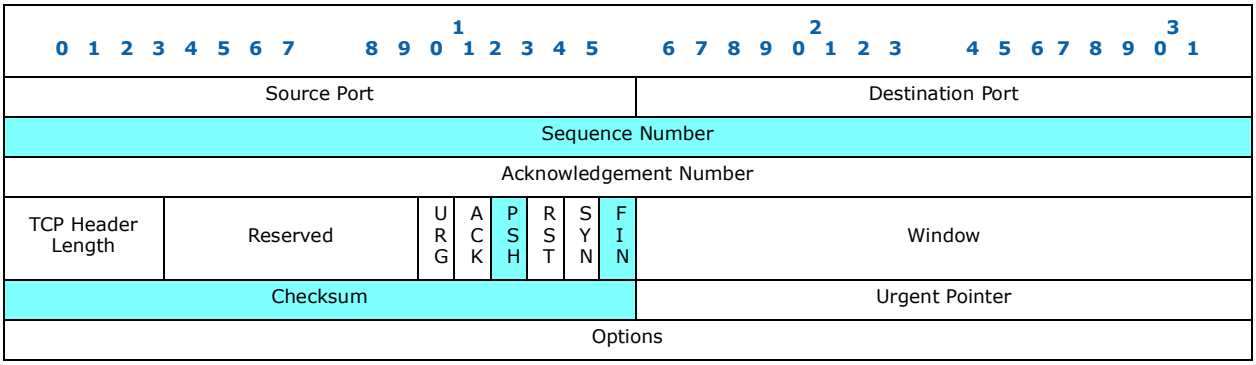


Figure A.5. TCP Header (Traditional Representation)

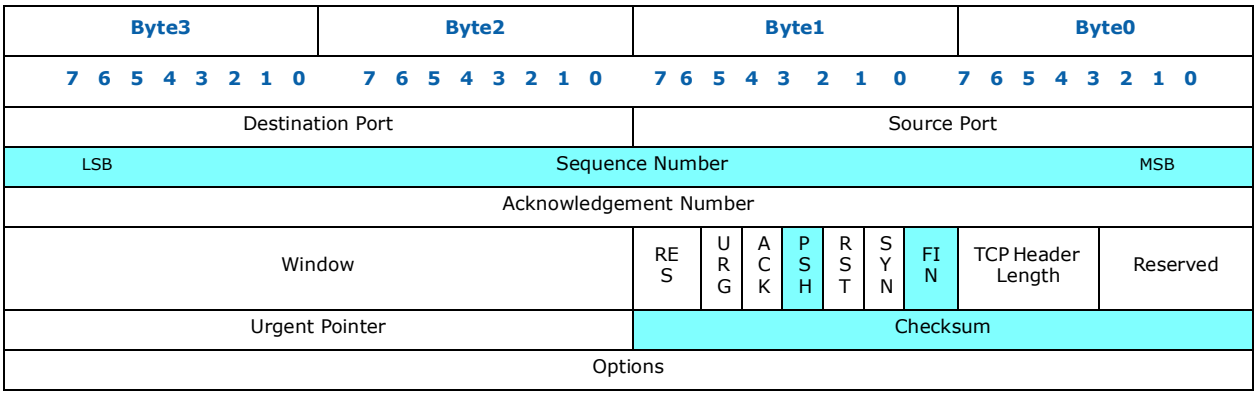


Figure A.6. TCP Header (Little Endian)

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 A.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 tricky. One common question is whether the *Protocol ID* field is added to the lower or upper byte of the 16- bit sum. The *Protocol ID* field should be added to the Least Significant Byte (LSB) of the 16-bit pseudo header sum, where the Most Significant Byte (MSB) of the 16-bit sum is the byte that corresponds to the first checksum byte out on the wire.

The *TCP 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 6.2.4.7.

Payload is normally MSS except for the last packet where it represents the remainder of the payload.

<table>
<tr><td colspan="3">IPv4 Source Address</td></tr>
<tr><td colspan="3">IPv4 Destination Address</td></tr>
<tr><td>Zero</td><td>Layer 4 Protocol ID</td><td>TCP/UDP Length</td></tr>
</table>

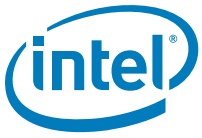
Figure A.7. TCP/UDP Pseudo Header Content for IPv4 (Traditional Representation)

<table>
<tr><td colspan="2">IPv6 Source Address</td></tr>
<tr><td colspan="2">IPv6 Final Destination Address</td></tr>
<tr><td colspan="2">TCP/UDP Packet Length</td></tr>
<tr><td>Zero</td><td>Next Header</td></tr>
</table>

Figure A.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 (such as TCP header plus TCP data). Some upper-layer protocols carry their own length information (like the *Length* field in the UDP header); for such protocols, that is the length used in the pseudo- header. Other protocols (such as TCP) do not carry their own length information, in which case the length used in the pseudo-header is the payload length from the IPv6 header, minus the length of any extension headers present between the IPv6 header and the upper-layer header.



- Unlike IPv4, when UDP packets are originated by an IPv6 node, the UDP checksum is not optional. That is, whenever originating a UDP packet, an IPv6 node must compute a UDP checksum over the packet and the pseudo-header, and, if that computation yields a result of zero, it must be changed to hex FFFF for placement in the UDP header. IPv6 receivers must discard UDP packets containing a zero checksum, and should log the error.

A type 0 routing header has the following format:

Next Header	Hdr Ext Len	Routing Type "0"	Segments Left "n"
Reserved			
Address[1]			
Address[2]			
...			
Final Destination Address [n]			

Figure A.9. IPv6 Routing Header (Traditional Representation)

- Next Header - 8-bit selector. Identifies the type of header immediately following the routing header. Uses the same values as the *IPv4 Protocol* field [RFC-1700 et seq.].
- Hdr Ext Len - 8-bit unsigned integer. Length of the routing header in 8-octet units, not including the first 8 octets. For the type 0 routing header, *Hdr Ext Len* is equal to two times the number of addresses in the header.
- Routing Type - 0.
- Segments Left - 8-bit unsigned integer. Number of route segments remaining. 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
Source Port		Destination Port	
Length		Checksum	

Figure A.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 A.11. UDP Header (Little Endian Order)

UDP pseudo header has the same format as the TCP pseudo header. The pseudo header prefixed to the UDP header contains the IPv4 source address, the IPv4 destination address, the IPv4 protocol field, and the UDP length (same as the TCP length 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.

A.1.3 Magic Packet

A Magic Packet is a broadcast frame, but could also be a multicast or unicast Ethernet MAC addresses. The integrated 10 GbE LAN controller accepts this packet if it matches any of its pre-programmed Ethernet MAC addresses. 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 integrated 10 GbE LAN controller expects the Ethernet MAC Address stored in the RAL[0] and RAH[0] registers.

The integrated 10 GbE LAN controller looks for the synchronization pattern and the sequence of 16 Ethernet MAC addresses. It does not check the packet content and the length of the header that precedes the magic pattern nor any data that follows it.

A.2 Packet Types for Packet Split Filtering

The following packet types are supported by the packet split feature in the integrated 10 GbE LAN controller. This section describes the packets from the split-header point of view. This means that when describing the different fields that are checked and compared, it emphasizes only the fields that are needed to calculate the header length. This document describes the checks that are done after the decision to pass the packet to the host memory was done.

Terminology:

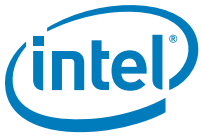
- Compare - The field values are compared and must be exactly equal to the value specified in this document.
- Checked - The field values are checked for calculation (header length …).
- Ignore - The field values are ignored but the field is counted as part of the header.

A.2.1 Type 1.1: Ethernet (VLAN/SNAP) IP Packets

A.2.1.1 Type 1.1: Ethernet, IP, Data

This type contains only Ethernet header and Ipv4 header while the payload header of the IP is not IPv6/ TCP/UDP.

Offset	# of bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – processed by main address filter, or broadcast.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN or E-tag)	0x8100 **** 0x893F **********	Ignore	



12+D	S=(0/4)	Possible VLAN Tag	0x8100 ****	Compare	
12+D+S	D=(0/8)	Possible Length + LLC/ SNAP Header		Compare	
12+D+S	2	Type	0800h	Compare	IP
IPv4 Header					
14+D+S	1	Version/ HDR Length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	>0 or MF bit is set	Check	Check that the packet is fragmented.
22+D+S	1	Time to Live	-	Ignore	
23+D+S	1	Protocol		Ignore	Has no meaning if the packet is fragmented.
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	

A.2.1.2 Type 1.2: Ethernet (SNAP/VLAN), IPv4, UDP

This type contains only Ethernet header, IPv4 header, and UDP header.

Offset	# of bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/ 8)	Outer Tag (Outer VLAN or E-tag)	0x8100 **** 0x893F **********	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100 ****	Check	
12+D+S	2	Type	0x0800	Compare	IP
IP Header					
14+D+S	1	Version/ HDR Length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	(xx00) 0x000	Compare	
22+D+S	1	Time to Live	-	Ignore	
23+D+S	1	Protocol	0x11	Compare	UDP header.

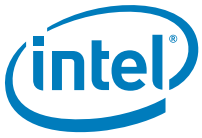
Offset	# of bytes	Field	Value	Action	Comment
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
UDP Header					
34+D+S+N	2	Source Port	Not (0x801)	Check	Not NFS packet.
36+D+S+N	2	Destination Port	Not (0x801)	Check	Not NFS packet.
38+D+S+N	2	Length	-	Ignore	
40+D+S+N	2	Checksum	-	Ignore	

In this case, the packet is split after (42+D+S+N) bytes.

A.2.1.3 Type 1.3: Ethernet (VLAN/SNAP) IPv4 TCP

This type contains only Ethernet header, Ipv4 header, and TCP header.

Offset	# of bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN or E-tag)	0x8100 **** 0x893F **********	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100 ****	Check	
12+D+S	2	Type	0x0800	Compare	IP
IPv4 Header					
14+D+S	1	Version/ HDR Length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to Live	-	Ignore	
23+D+S	1	Protocol	0x06	Compare	TCP header.
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
TCP Header					



34+D+S+N	2	Source Port	Not (0x801)	Check	Not NFS packet.
36+D+S+N	2	Destination Port	Not (0x801)	Check	Not NFS packet.
38+D+S+N	4	Sequence Number	-	Ignore	
42+D+S+N	4	Acknowledge Number	-	Ignore	
46+D+S+N	1/2	Header Length		Check	
46.5+D+S+N	1.5	Different Bits	-	Ignore	
48+D+S+N	2	Window Size	-	Ignore	
50+D+S+N	2	TCP Checksum	-	Ignore	
52+D+S+N	2	Urgent Pointer	-	Ignore	
54+D+S+N	F	TCP Options	-	Ignore	

In this case, the packet is split after (54+D+S+N+F) bytes.

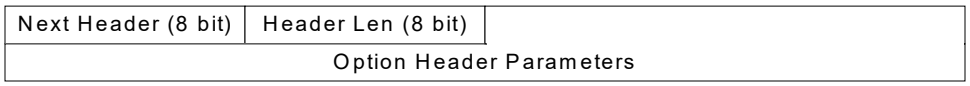
- N = (IP HDR length -5) * 4.
- F = (TCP header length - 5) * 4.

A.2.1.4 Type 1.4: Ethernet IPv4 IPv6

A.2.1.4.1 IPv6 Header Options Processing

This type of processing looks at the next-header field and header length 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).

A.2.1.4.2 IPv6 Next Header Values

When parsing an IPv6 header, the integrated 10 GbE LAN controller does not parse all possible extension headers and if there is an extension header that is not supported by the integrated 10 GbE LAN controller 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.

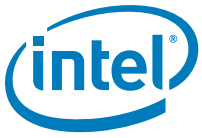
A.2.1.4.3 Type 1.4.1: Ethernet IPv4 IPv6 Data

This type contains only Ethernet header, IPv4 header, and IPv6 header.

Offset	# of bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN or E-tag)	0x8100 **** 0x893F **********	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100	Check	
12+D+S	2	Type	0x0800	Compare	IP
IP4 Header					
14+D+S	1	Version/ HDR Length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to Live	-	Ignore	
23+D+S	1	Protocol	0x29	Compare	IPv6
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
IPv6 Header					
34+D+S+N	1	Version/ Traffic Class	0x6X	Compare	Check IPv6
35+D+S+N	3	Traffic Class/Flow Label	-	Ignore	
38+D+S+N	2	Payload Length	-	Ignore	
40+D+S+N	1	Next Header	IPv6 extension headers	Check	
41+D+S+N	1	Hop Limit	-	Ignore	
42+D+S+N	16	Source Address	-	Ignore	
48+D+S+N	16	Destination Address		Ignore	
74+D+S+N	B	Possible IPv6 Next Headers	-	Ignore	

In this case the packet is split after (74+D+S+N+B) bytes.

- N = (IP HDR length - 5) * 4.



- One of the extension headers of the IPv6 packets must be a “fragment header” in order for the packet to be parsed.

A.2.1.4.4 Type 1.4.2: Ethernet (VLAN/SNAP) Ipv4 Ipv6 UDP

This type contains only Ethernet header, Ipv4 header, IPv6 header and UDP header.

Offset	# of bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN or E-tag)	0x8100 **** 0x893F **********	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100	Check	
12+D+S	2	Type	0x0800	Compare	IP
IPv4 Header					
14+D+S	1	Version/ HDR length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to Live	-	Ignore	
23+D+S	1	Protocol	0x29	Compare	IPv6
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
IPv6 Header					
34+D+S+N	1	Version/ Traffic Class	0x6X	Compare	Check IPv6
35+D+S+N	3	Traffic Class/Flow Label	-	Ignore	
38+D+S+N	2	Payload Length	-	Ignore	
40+D+S+N	1	Next Header	IPv6 extension header or 0x11	Check	IPv6 extension headers.
41+D+S+N	1	Hop Limit	-	Ignore	
42+D+S+N	16	Source Address	-	Ignore	
58+D+S+N	16	Destination Address		Ignore	
74+D+S+N	B	Possible IPv6 Next Headers	-	Ignore	
UDP Header					

Offset	# of bytes	Field	Value	Action	Comment
74+D+S+N+B	2	Source Port	Not (0x801)	Check	Not NFS packet.
76+D+S+N+B	2	Destination Port	Not (0x801)	Check	Not NFS packet.
78+D+S+N+B	2	Length	-	Ignore	
80+D+S+N+B	2	Checksum	-	Ignore	

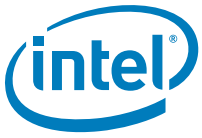
In this case the packet is split after (82+D+S+N+B) bytes.

N = (IP HDR length - 5) * 4.

A.2.1.4.5 Type 1.4.3: Ethernet (VLAN/SNAP) Ipv4 Ipv6 TCP

This type contain only Ethernet header, IPv4 header, IPv6 header and TCP header.

Offset	# of bytes	Field	Value	Action	Comment
0	6	Destination Address		Ignore	MAC Header – processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN or E-tag)	0x8100 **** 0x893F **********	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100	Check	
12+D+S	2	Type	0x0800	Compare	IP
IPv4 Header					
14+D+S	1	Version/ HDR Length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to live	-	Ignore	
23+D+S	1	Protocol	0x2A	Compare	IPv6
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
IPv6 Header					
34+D+S+N	1	Version/ Traffic Class	0x6X	Compare	Check IPv6.
35+D+S+N	3	Traffic Class/Flow Label	-	Ignore	



Offset	# of bytes	Field	Value	Action	Comment
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	
90+T	2	TCP Checksum	-	Ignore	
92+T	2	Urgent Pointer	-	Ignore	
94+T	F	TCP Options	-	Ignore	

In this case the packet is split after (94+D+S+N+B+F) bytes.

- T = D+S+N+B
- N = (IP HDR length - 5) * 4.
- F = (TCP HDR length - 5)*4

A.2.2 Type 2: Ethernet, IPv6

A.2.2.1 Type 2.1: Ethernet, IPv6 data

This type contains only an Ethernet header and an IPv6 header while the packet should be a fragmented packet. If the packet is not fragmented and the next header is not supported then the header is not split. The supported packet types for header split are programmed in PSRTYPE register (per VF).

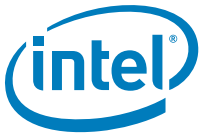
Offset	# of bytes	Field	Value (hex)	Action	Comment
0	6	Destination Address		Ignore	MAC Header – processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN or E-tag)	0x8100 **** 0x893F ******** ****	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100	Check	
IPv6 Header					
12+D+S	2	Type	0x86DD	Compare	IP
14+D+S	1	Version/ Traffic Class	0x6X	Compare	Check IPv6.
15+D+S	3	Traffic Class/Flow Label	-	Ignore	
18+D+S	2	Payload Length	-	Ignore	
20+D+S	1	Next Header	IPv6 next header types.	Check	The last header must be fragmented header in order for the header to be split.
21+D+S	1	Hop Limit	-	Ignore	
22+D+S	16	Source Address	-	Ignore	
38+D+S	16	Destination Address		Ignore	
54+D+S	N	Possible IPv6 Next Headers	-	Ignore	

In this case the packet is split after (54+D+S+N) bytes.

- The last next header field of the IP section field should not be0x11/0x06 (TCP/UDP).

A.2.2.1.1 Type 2.2: Ethernet (VLAN/SNAP) IPv6 UDP

This type contains only Ethernet header, IPv6 header, and UDP header.



Offset	# of bytes	Field	Value (hex)	Action	Comment
0	6	Destination Address		Ignore	MAC Header – processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN or E-tag)	0x8100 **** 0x893F ********* ***	Ignore	
12+D	S=(0/4)	Possible VLAN Tag		Check	
IPv6 Header					
12+D+S	2	Type	0x86DD	Compare	IP
14+D+S	1	Version/ Traffic Class	0x6X	Compare	Check IPv6.
15+D+S	3	Traffic Class/Flow Label	-	Ignore	
18+D+S	2	Payload Length	-	Ignore	
20+D+S	1	Next Header	IPv6 next header types Or 0x11	Check	
21+D+S	1	Hop Limit	-	Ignore	
22+D+S	16	Source Address	-	Ignore	
38+D+S	16	Destination Address		Ignore	
54+D+S	N	Possible IPv6 Next Headers	-	Ignore	
UDP Header					
54+D+S+N	2	Source Port	Not (0x801)	Check	Not NFS packet.
56+D+S+N	2	Destination Port	Not (0x801)	Check	Not NFS packet.
58+D+S+N	2	Length	-	Ignore	
60+D+S+N	2	Checksum	-	Ignore	

In this case the packet is split after (62+D+S+N) bytes.

- The last *next-header* field of the last header of the IP section must be 0x06.

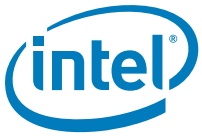
A.2.2.2 Type 2.3: Ethernet (VLAN/SNAP) IPv6 TCP

This type contains only Ethernet header, IPv6 header, and UDP header.

Offset	# of bytes	Field	Value (hex)	Action	Comment
0	6	Destination Address		Ignore	MAC Header – processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN or E-tag)	0x8100 **** 0x893F ********* ***	Ignore	
12+D	S=(0/4)	Possible VLAN Tag		Check	
IPv6 Header					
12+D+S	2	Type	0x86DD	Compare	IP
14+D+S	1	Version/ Traffic Class	0x6X	Compare	Check IPv6.
15+D+S	3	Traffic Class/Flow Label	-	Ignore	
18+D+S	2	Payload Length	-	Ignore	
20+D+S	1	Next Header	IPv6 next header types Or TCP	Check	
21+D+S	1	Hop Limit	-	Ignore	
22+D+S	16	Source Address	-	Ignore	
38+D+S	16	Destination Address		Ignore	
54+D+S	N	Possible IPv6 Next Headers	-	Ignore	
TCP Header					
54+D+S+N	2	Source Port	Not (0x801)	Check	Not NFS packet.
56+D+S+N	2	Destination Port	Not (0x801)	Check	Not NFS packet.
58+D+S+N	4	Sequence number	-	Ignore	
62+D+S+N	4	Acknowledge Number	-	Ignore	
66+D+S+N	1/2	Header Length		Check	
66.5+D+S+N	1.5	Different Bits	-	Ignore	
68+D+S+N	2	Window Size	-	Ignore	
70+D+S+N	2	TCP Checksum	-	Ignore	
72+D+S+N	2	Urgent Pointer	-	Ignore	
74+D+S+N	F	TCP Options	-	Ignore	

In this case the packet is split after (54+D+S+N+F) bytes.

- F = (TCP header length - 5) * 4.
- The last 'next-header' field of the last header of the IP section must be 0x11.



A.2.3 Type 3: Reserved

Type 3 used to be iSCSI packets (header split is not supported for iSCSI packets in the integrated 10 GbE LAN controller).

A.2.4 Type 4: Reserved

A.2.5 Type 5: Cloud Packets

A.2.5.1 Type 5.1: Ethernet, IPv4, NVGRE, IPv4/6, TCP/UDP

This type contains an Ethernet header an IPv4 header, a GRE header an IPv4/6 header and a TCP or UDP header. The supported packet types for header split are programmed in PSRTYPE register (per VF).

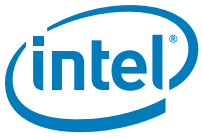
Offset	# of bytes	Field	Value (hex)	Action	Comment
0	6	Destination Address		Ignore	MAC Header – processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN or E-tag)	0x8100 **** 0x893F ******** ****	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100	Check	
12+D+S	2	Type	0x0800	Compare	IP Split on outer L2 is done after this field.
IPv4 Header					
14+D+S	1	Version/ HDR length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to Live	-	Ignore	
23+D+S	1	Protocol	0x2F	Compare	GRE
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
GRE Header					
34 + D +S +N	2	Flags/Version	0x2000	Compare	
36 + D +S +N	2	Protocol Type	0x6558	Compare	
38+ D +S +N	3	TNI		Store	Used for flow director.

Offset	# of bytes	Field	Value (hex)	Action	Comment
41 + D +S + N	1	Reserved	0x0	Ignore	Split on cloud header splits after this field.
Inner MAC Header					
42 + D +S + N	6	Inner Destination Address		Ignore	Used for flow director.
48 + D +S + N	6	Inner Source Address		Ignore	
54 + D + S + N	I=(0/4)	Possible Inner VLAN Tag	0x8100	Store	Used for flow director.
54 + D + S + N + I	2	Type	0x0800/ 0x86DD	Compare	Split on L2 is done after this field.
IPv4/6 Header - as previously described.					Split on L3 is done after these fields.
UDP/TCP - as previously described.					Split on L4 is done after these fields.

A.2.5.2 Type 5.2: Ethernet, IPv4, VXLAN, IPv4/6, TCP/UDP

This type contains an Ethernet header an IPv4 header, a UDP header with a specific UDP destination port a VXLAN header an IPv4/6 header and a TCP or UDP header. The supported packet types for header split are programmed in PSRTYPE register (per VF).

Offset	# of bytes	Field	Value (hex)	Action	Comment
0	6	Destination Address		Ignore	MAC Header – processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	
12	D=(0/4/6/8)	Outer Tag (Outer VLAN or E-tag)	0x8100 **** 0x893F ******** ****	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100	Check	
12+D+S	2	Type	0x0800	Compare	IP Split on outer L2 is done after this field.
IPv4 Header					
14+D+S	1	Version/ HDR Length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to Live	-	Ignore	
23+D+S	1	Protocol	0x11	Compare	UDP



Offset	# of bytes	Field	Value (hex)	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	
UDP Header					
34+D+S+N	2	Source Port		Ignore	
36+D+S+N	2	Destination Port		Compare	Compare to VXLANCN-TRL.UDPPORT.
38+D+S+N	2	Length	-	Ignore	
40+D+S+N	2	Checksum	-	Ignore	
VXLAN Header					
42 + D +S + N	1	Flags	0x08	Compare	VNI exists.
43 + D + S + N	3	Reserved	0x000000	Compare	
46 + D + S + N	3	VNI		Store	Used for flow director.
49 + D + S + N	1	Reserved	0x00	Compare	Split on cloud header splits after this field.
Inner MAC Header					
50 + D +S + N	6	Inner Destination Address		Ignore	Used for Flow director
56 + D +S + N	6	Inner Source Address		Ignore	
62 + D + S + N	I=(0/4)	Possible Inner VLAN Tag	0x8100	Store	Used for Flow director
62 + D + S + N + I	2	Type	0x0800/ 0x86DD	Compare	Split on L2 is done after this field
IPv4/6 Header - as previously described.					Split on L3 is done after these fields
UDP/TCP - as previously described.					Split on L4 is done after these fields

A.2.5.3 Type 5.2: Ethernet, IPv4, GENEVE, IPv4/6, TCP/UDP

This type contains an Ethernet header an IPv4 header, a UDP header with a specific UDP destination port a VXLAN header an IPv4/6 header and a TCP or UDP header. The supported packet types for header split are programmed in PSRTYPE register (per VF).

Offset	# of bytes	Field	Value (hex)	Action	Comment
0	6	Destination Address		Ignore	MAC Header – processed by main address filter, or broadcast filter.
6	6	Source Address		Ignore	

12	D=(0/4/6/8)	Outer Tag (Outer VLAN and E-tag)	0x8100 **** 0x893F ******** **** 0x8926 ********	Ignore	
12+D	S=(0/4)	Possible VLAN Tag	0x8100	Check	
12+D+S	2	Type	0x0800	Compare	IP. Split on outer L2 is done after this field.
Ipv4 header					
14+D+S	1	Version/ HDR length	0x4X	Compare	Check IPv4 and header length.
15+D+S	1	Type of Service	-	Ignore	
16+D+S	2	Packet Length	-	Ignore	
18+D+S	2	Identification	-	Ignore	
20+D+S	2	Fragment Info	0x00	Compare	
22+D+S	1	Time to Live	-	Ignore	
23+D+S	1	Protocol	0x11	Compare	UDP.
24+D+S	2	Header Checksum	-	Ignore	
26+D+S	4	Source IP Address	-	Ignore	
30+D+S	4	Destination IP Address	-	Ignore	
34+D+S	N	Possible IP Options		Ignore	
UDP header					
34+D+S+N	2	Source Port		Ignore	
36+D+S+N	2	Destination Port		Compare	Compare to VXLANCNTRL.GENEVE_UDP_PORT.
38+D+S+N	2	Length	-	Ignore	
40+D+S+N	2	Checksum	-	Ignore	
Geneve header					
42 + D +S + N	2	Ver (2 bits) Option Length (6 bits)		Compare	Ver must be zero.
43 + D +S + N	2	O (OAM) C Reserved		Compare	
44 + D + S + N	2	Next Protocol		Compare	
46 + D + S + N	3	VNI		Store	Used for flow director
49 + D + S + N	1	Reserved	0x00	Compare	Split on cloud header splits after this field.
Inner MAC Header					
50 + D +S + N	6	Inner Destination Address		Ignore	Used for flow director.
56 + D +S + N	6	Inner Source Address		Ignore	
62 + D + S + N	I=(0/4)	Possible Inner VLAN Tag	0x8100	Store	Used for flow director.

62 + D + S + N + I	2	Type	0x0800/ 0x86DD	Compare	Split on L2 is done after this field.
Ipv4/6 header - as described above					Split on L3 is done after these fields.
UDP/TCP - as described above					Split on L4 is done after these fields.

A.2.5.4 Ethernet MAC Addresses

L2 destination and source Ethernet MAC addresses (each of them is six bytes long). The Ethernet MAC address of the target is assumed to be assigned by the network. The mechanism that is used for Ethernet MAC address assignment and Ethernet MAC address detection is outside of the scope of this document.

A.2.6 FC Frame Format

Note: This section is provided as a background on FC and is not required for hardware implementation. For a complete description of the FC fields please refer to FC-FS-2 specification.

The FC frame as defined in FC-FS-2 specification is shown in Figure A.12 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 A.12. FC Frame Format

A.2.6.1 FC SOF and EOF

FC SOF delimiter and EOF delimiter.

A.2.6.2 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.The FC CRC offload.

A.2.6.3 FC Optional Headers

Note: Most of the following section is provided as a background on FC and is not required for hardware implementation. The reader can skip the detailed explanation of the optional headers provided and concentrate in the tables and figures that follow the text.

The following table and figures describe the FC frame structure with optional headers and lists the optional headers. The optional headers (that are present) are always ordered as shown in Figure A.13 and Figure A.14. Their presence is indicated in the Data Field Control (DF_CTL) field in the FC header as indicated in Table A.5.

Maximum FC frame size: The sum of the length in bytes of the FC Payload, the number of fill bytes, and the lengths in bytes of all optional headers shall not exceed 2112.

Table A.5 FC Optional Headers

DF_CTL	Optional Header	Length
bit 6	ESP Header / ESP Trailer	Variable
bit 5	Network Header	16 bytes
bit 4	Association Header	32 byte
bits 1:0	Device Header	0, 16, 32, or 64 bytes

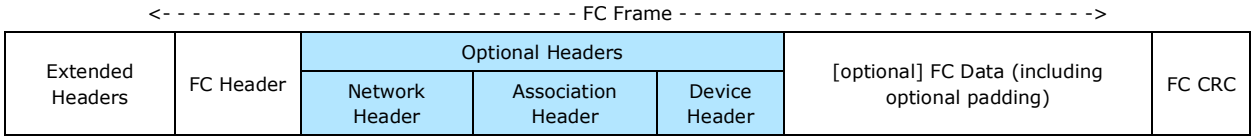


Figure A.13. FC Frame format with Optional Headers (without ESP Header)

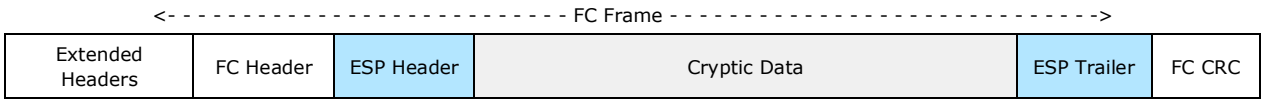


Figure A.14. FC Frame format with Optional Headers (with ESP Header)

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 large send offload when the 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 FC networks of different fabric address spaces, or FC and non-FC networks. The network header contains name identifiers for the network destination address and network source address.

Association Header

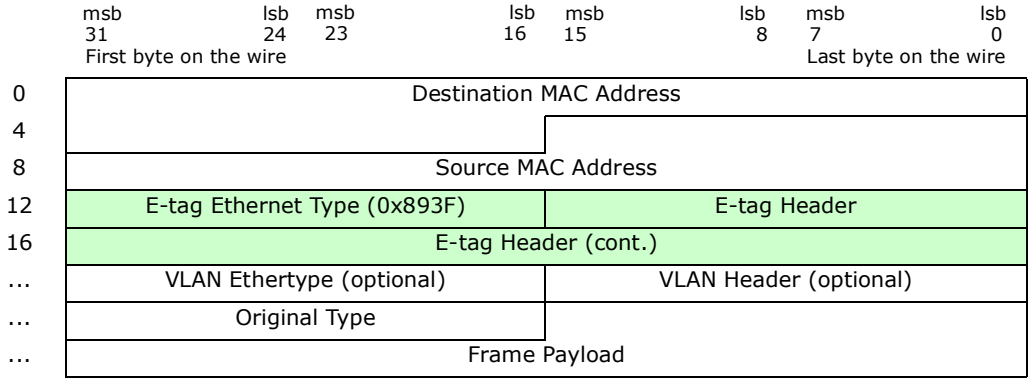
The association header, if used, must be present only in the first data frame of a sequence. The association header is an optional header 32 bytes long within the data field content. Its presence is indicated by bit 4 in the DF_CTL field being set to 1b. The association header might be used to identify 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 integrated 10 GbE LAN controller does not use the association for any filtering purposes but rather uses the OX_ID.

Device Header

The device header, if present, must be present either in the first data frame or in all data frames of a sequence. If large send off load is used 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 Layer Protocol (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.

A.3 E-tag formats

Packets with E-tag has the following format:



The E-tag format is the format defined in the IEEE 802.1BR specification described as follows:

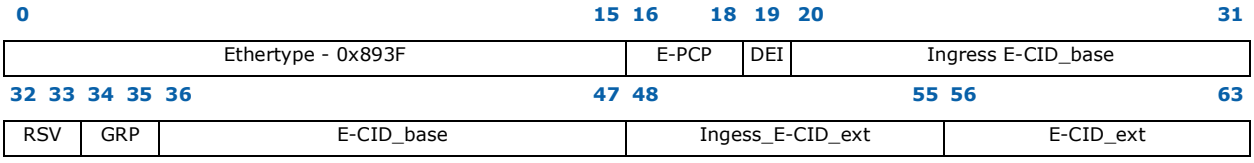
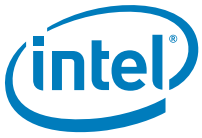


Figure A.16. E-tag Format

The Ingess_E-CID_ext and E-CID_ext are always zero for endpoints and are effectively reserved.

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



Appendix B Integrated PHY Support

B.1 Integrated 10 GbE Interface

The integrated 10 GbE LAN controller provides complete functionality to support the 10 Gb/s ports. The integrated 10 GbE LAN controller performs all functions required for transmission and reception defined in the various standards.

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

The integrated 10 GbE LAN controller enables the following connectivity modes:

- IEEE802.3 clause 72 10GBASE-KR
- iXFI - Intel XFI
- SFI
- 2.5 GbE - A single lane configuration based on IEEE802.3 Clause 70 1000BASE-X running at higher frequency.
- IEEE Std 802.3 Clause 70 1000BASE-X
- SGMII 1 GbE, 100 Mb/s and 10 Mb/s

B.1.1 10GBASE-KR Operating Mode

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

The MAUI interface is configured as a KR interface while auto-negotiation to a KR link partner is detected. KR operation can also be forced by shared SPI Flash or software by setting the relevant bits in the control register and disabling auto-negotiation (see Section B.5).

B.1.1.1 KR Overview

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

KR is a full-duplex interface that uses a single self-clocked serial differential link in each direction to achieve 10 Gb/s data throughput. The serial link transfers scrambled data at 10.3125 Gbaud to accommodate both data and the overhead associated with 64B/66B coding. Refer to the *Intel® Atom™ Processor C3000 Product Family Platform Design Guide (PDG)* for more detail.

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

Figure B.1 shows the architectural positioning of 10GBASE-KR.

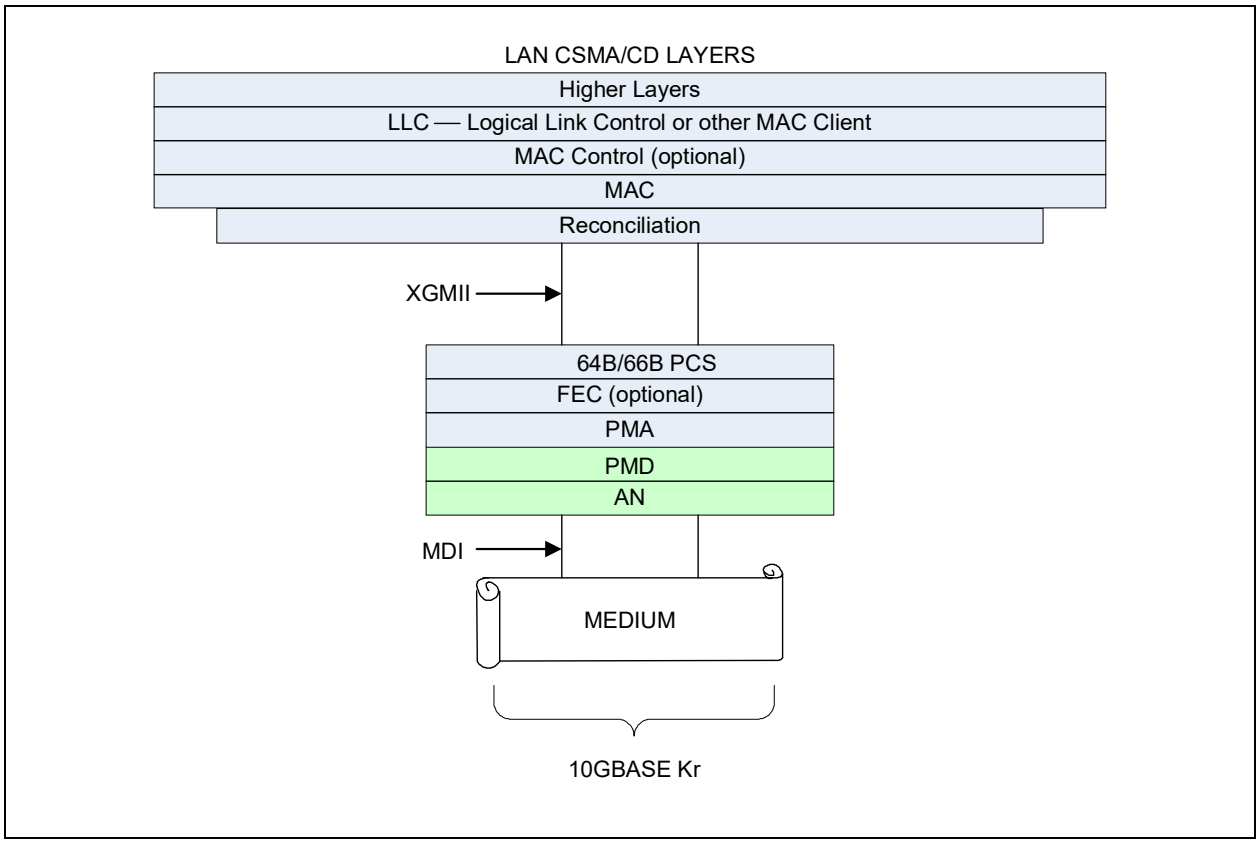
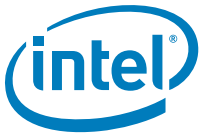


Figure B.1. Architectural Positioning of 10GBASE-KR

B.1.1.2 KR Electrical Characteristics

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



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

The 10GBASE-KR link requires a nominal 100 Ω differential source and load terminations with AC coupling on the receive side. Refer to the *Intel® Atom™ Processor C3000 Product Family Platform Design Guide (PDG)* for more detail.

B.1.1.3 KR Reverse Polarity

The KR PHY supports reverse polarity of the KR transmit and receive lanes via the shared SPI Flash. Contact your Intel representative for more details.

B.1.2 iXFI - Intel XFI

iXFI is a static version of the 10GBASE-KR Operating Mode. When configured to iXFI, the PHY operates electrically at KR without the Clause 73 auto-negotiation and electrical training.

B.2 2.5 GbE Interface

The 2500BASEX link mode is a special high frequency mode of the 1000BASE-X link. The 2.5Gb/s rate is achieved by over clocking a 1 Gb/s link to 3.125 GHz.

Electrical parameters of this mode are aligned with the 1000BASE-X specification.

B.3 1 GbE Interface

The integrated 10 GbE LAN controllerprovides complete support for up to two 1 Gb/s port implementations. The device performs all functions required for transmission and reception defined by the different standards.

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

SoC enables 1 GbE operation compliant with IEEE802.3 Clause 70 1000BASE-KX.

B.3.1 1000BASE-KX Operating Mode

The MAUI interface, when operating as a KX Interface, supports data rates of 1 Gb/s over copper traces on improved FR4 PCBs. Data is transferred over a single differential path in each direction for a total of two pairs with each path operating at 1.25 Gbaud to support overhead of 8B/10B coding. The interface is used to connect the integrated 10 GbE LAN controller to a KX compliant switch port over the backplane or to KX compliant 1 GbE PHY device. In the event of auto-negotiation defined in IEEE802.3 Clause 73 ending with 1 Gb/s as the HCD, the MAUI interface is configured as a KX interface. KX operating mode can also be forced by software see Section B.5.

B.3.1.1 KX Overview

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

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

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

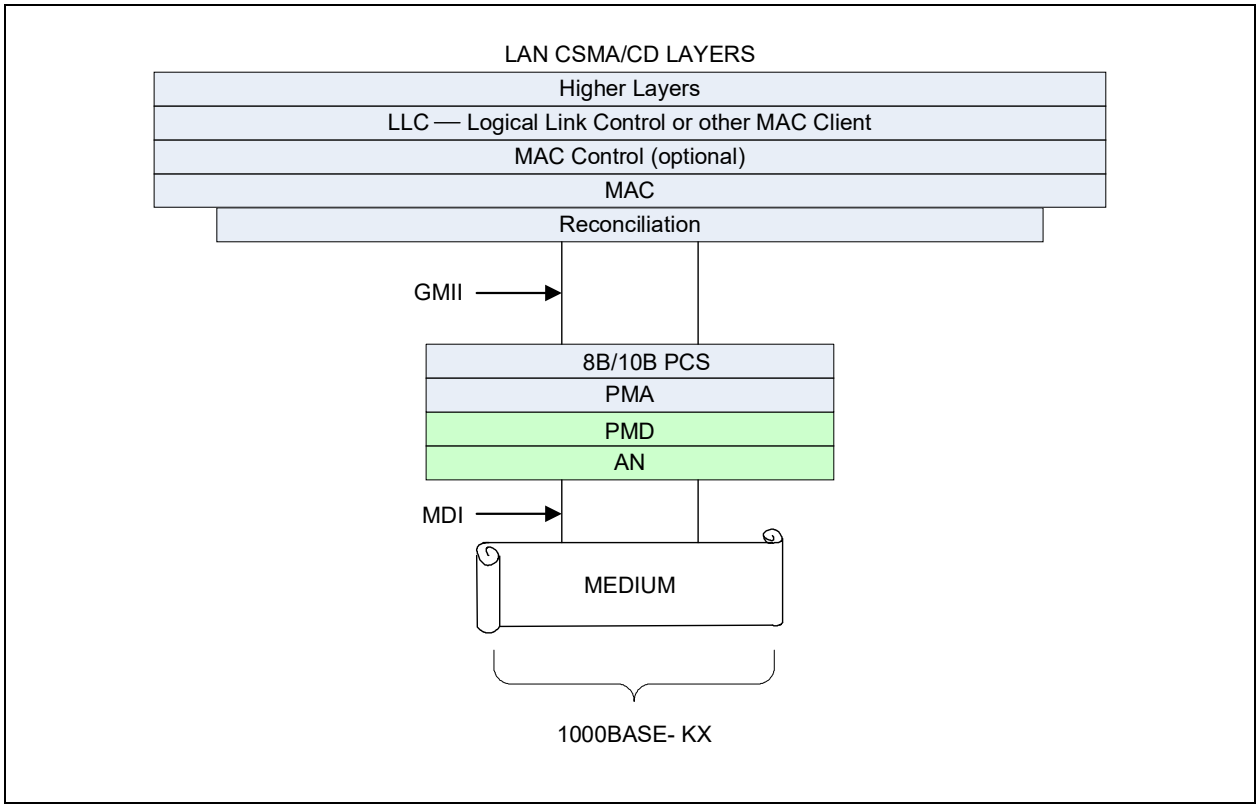


Figure B.3. Architectural Positioning of 1000BASE-KX

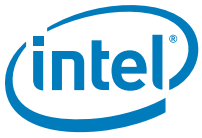
B.3.1.2 KX Electrical Characteristics

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

The KX signal paths are point-to-point connections. Each path corresponds to a KX lane and is comprised of two complementary signals making a balanced differential pair. There is one differential path in each direction for a total of two pairs, or four connections. Refer to the *Intel® Atom™ Processor C3000 Product Family Platform Design Guide (PDG)* for more detail.

B.3.2 SGMII Support

The integrated 10 GbE LAN controller supports 1 Gb/s, 100 Mb/s and 10 Mb/s operation using the SGMII protocol over the KX electrical interface (AC coupling, no source synchronous Tx clock, etc.).



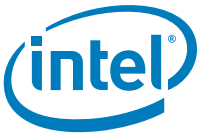
B.3.2.1 SGMII Overview

SGMII interface supported by the integrated 10 GbE LAN controller enables operation at 1 Gb/s over two differential, controlled impedance pairs of traces (one pair for transmit, one pair for receive). When operating in SGMII, the MAUI interface uses the 1000BASE-X PCS and PMA as defined in IEEE802.3 Clause 36 and the 1000BASE-KX PMD as defined in IEEE802.3ap Clause 70 or the 1000BASE-BX as defined in the PCIMG 3.1 standard. In SGMII operating mode, the MAUI interface can support data rates of 1 Gb/s, 100 Mb/s and 10 Mb/s.

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

SGMII control information, as listed in the following table is transferred from the PHY to the MAC to signal change of link speed. This is achieved by using the auto-negotiation functionality defined in Clause 37 of the IEEE 802.3 Specification. Instead of the ability advertisement, the PHY sends the control information via its tx_config_reg[15:0] as listed in the following table each time the link speed information changes. Upon receiving control information, the MAC acknowledges the update of the control information by asserting bit 14 of its tx_config_reg[15:0] as listed in the following table. Compared to the definition in IEEE802.3 clause 37.

Bit Number	TX_CONFIG_REG[15:0]	Comments
15	Link: 1b = Link up 0b = Link down.	
14	Auto-negotiation acknowledge as specified in 802.3.	
13	0b: Reserved for future use.	
12	Duplex mode: 1b = full duplex, 0b = half duplex.	Only full duplex is supported in the integrated 10 GbE LAN controller.
11:10	Speed: 11b = Reserved. 10b = 1000 Mb/s. 01b = 100 Mb/s. 00b = 10 Mb/s.	
9	EEE: 1b = EEE is supported. 0b = EEE is not supported.	The integrated 10 GbE LAN controller always communicates 0b since EEE is not supported in SGMII.
8:1	0x0 = Reserved for future use.	
0	1b.	



B.4 Auto Negotiation For Backplane Ethernet and Link Setup Features

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

Auto-negotiation for backplane Ethernet is defined in IEEE802.3 Clause 73 and is based on IEEE802.3 clause 28 definition of auto-negotiation for twisted-pair link segments.

Auto-negotiation for backplane Ethernet uses an extended base page and next page format and modifies the timers to allow rapid convergence. Furthermore, auto-negotiation does not use Fast Link Pulses (FLPs) for link code word signaling and instead uses Differential Manchester Encoding (DME) signaling, which is more suitable for electrical backplanes. Since DME provides a DC balanced signal. Auto-negotiation for backplane Ethernet also includes support for parallel detection of 1000BASE-KX links in addition to transmission and reception of extended base page and next page auto-negotiation frames.

B.4.1 MAC Link Setup and Auto Negotiation

The MAC block in the integrated 10 GbE LAN controller supports both 10 GbE and 1 GbE link modes and the appropriate functionality specified in the standards for these link modes.

Each of these link modes can use different PMD sub-layer and base band medium types.

In 10 GbE operating mode, the integrated 10 GbE LAN controller supports 10GBASE-KR, while in 1 GbE operating mode, the integrated 10 GbE LAN controller supports 1000BASE-KX protocol . The different protocols supported in 10 GbE operating mode and 1 GbE operating mode affect only the configuration of the MAUI AFE and MAUI PHY logic blocks (PCS, FEC, etc.) while the MAC supports rates of either 1 Gb/s or 10 Gb/s, without need to know the electrical medium actually being interfaced.

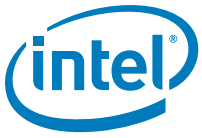
Link speed and link characteristics can be determined through static configuration, parallel detect and auto-negotiation or forced operation for diagnostic purposes. The auto-negotiation processes defined in IEEE802.3 Clause 73 enables selection between KR(10 GbE) and KX (1 GbE) compliant link partners and defining link characteristics and link speed.

Link setting is done by configuring the speed configuration, defining the appropriate physical interface and restarting auto-negotiation see Link Configuration Flows - Section B.5.

B.4.2 Hardware Detection of Legacy Link Partner (Parallel Detection)

The integrated 10 GbE LAN controller’s companion PHYs supports the IEEE802.3 clause 73 parallel detection process to enable a connection to legacy link partners that do not support auto-negotiation. Parallel detection enables detecting the link partner operating mode (KX as defined in IEEE802.3 clause 73) by activating KX and attempting to achieve link synchronization by the related PCS block.

Parallel detection is enabled as part of clause 73 backplane auto-negotiation process.



B.5 Link Configuration Flows

Note: All register access to the HIP is done using the operation described in Section 2.8.1. The flow to control the link is through the following register:

Lane Configuration Lane 0: KRM_PMD_RX_FLEX_PORT0/FLX_MASK_ST20 Lane 1: KRM_PMD_RX_FLEX_PORT1/FLX_MASK_ST20			
	Reset	**Init.**	**Description**
31		0b	Firmware Lane Reset. Set this bit to indicate to the lane firmware that auto-negotiation reset/restart (KRM_KR_PCS_PORT/LINK_CNTL_1/teth_an_restart) has been asserted. Firmware clears this bit once it has completed the reset sequence and the lane firmware has entered the LANE_INIT FW state.
30:28		111b	Lane Speed. 000b = Negotiated. 001b = 10 Mb/s. 010b = 100 Mb/s. 011b = Reserved. 100b = 1 GbE. 101b = 2.5 GbE. 110b = Reserved. 111b = 10 GbE.
27	0	0b	Auto Negotiation Enable (including Parallel Detect). 0b = Force mode. 1b = Auto-negotiation. ***Note:*** If Auto Negotiation is enabled, Lane Speed is ignored.
26	0	0b	AN37 Mode Enable. 0b= AN37 disabled. 1b = AN37 enabled.
25		0b	SGMII Enable. 0b = Disabled. 1b = Enabled.
24		0b	KR Training Enable. 0b = Disabled. 1b = KR training enabled (without DME AN pages).
23:22		0b	Reserved.
21:20		00b	SFI-10 GbE Mode. 00 Direct Attach. 01 Optical-SR. 10 Optical-LR. 11 Reserved. Only valid when using Force Mode (AN disabled) at 10G without KR Training
19		0b	Reserved.
18		0b	Force Link Down. When set and followed by a firmware lane reset, returns the lane firmware to LANE_INIT and stops firmware execution and releases the firmware execution scheduler.
17		0b	Disable Watch Dog Timer. Disable firmware lane 5 watchdog timer.

Lane Configuration Lane 0: KRM_PMD_RX_FLEX_PORT0/FLX_MASK_ST20 Lane 1: KRM_PMD_RX_FLEX_PORT1/FLX_MASK_ST20			
	Reset	**Init.**	**Description**
16		0b	Halt at LANE_INIT. Enable lane firmware halt at LANE_INIT (debug mode only). Does not set *Force Link Down* flag.
15:1		0b	Reserved.
0		0b	Lane Configuration Updated. Set to 1b when the integrated 10 GbE LAN controller has configured this register. Used to enable lane firmware to verify that the configuration is updated.

The configuration for each type of link is as follows:

Note: All fields should be zeroed unless mentioned as follows.

- Native SFI-SFP+ (LANE SPEED=7)
- KR-Inphi-SFP+ (AN Enabled + Capabilities)
- KR-CPVL-10GBT (AN Enabled + Capabilities)
- SGMII-BP-BCM 89501 (LANE SPEED=4, SGMII enable, AN37)
- MRVL 1512+1514 - 1GbT (SGMII enable, AN37)
- KR/KX BP (AN Enabled + Capabilities)
- 2.5G-X-BP (LANE SPEED=5)
- MRVL-1543-1GBT (SGMII enable, AN37)

B.5.1 Low Power Link Up (LPLU)

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.

B.5.2 Behavior in Non-D0 State

If the *LPLU* bit is set in the shared SPI Flash, 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. If link establishment fails, the PHY tries to negotiate at additional speeds. For example, the PHY advertises 1 Gb/s only and the partner supports 10 Gb/s only. After the first try fails, the PHY enables 1 Gb/s and 10 Gb/s and tries again.