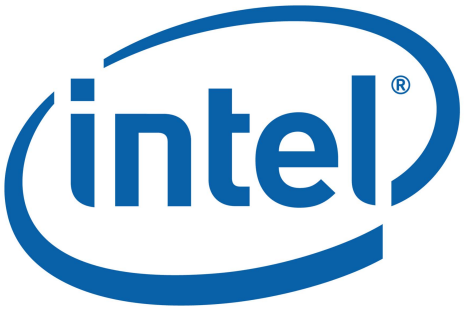


# Intel® 82599 10 GbE Controller Datasheet

## Networking Division (ND)

**PRODUCT FEATURES**

### General

- Dual port 10 GbE device or Single Port device (82599EN)
- Serial Flash Interface
- 4-wire SPI EEPROM Interface
- Configurable LED operation for software or OEM customization of LED displays
- Protected EEPROM space for private configuration
- Device disable capability
- Package Size - 25 mm x 25 mm Networking
- Complies with the 10 Gb/s and 1 Gb/s Ethernet/802.3ap (KX/KX4/KR) specification
- Complies with the 10 Gb/s Ethernet/802.3ae (XAUI) specification
- Complies with the 1000BASE-BX specification
- Complies with the IEEE 802.3x 100BASE-TX specification
- Support for jumbo frames of up to 15.5 KB
- Auto negotiation Clause 73 for supported mode
- CX4 per 802.3ak
- Flow control support: send/receive pause frames and receive FIFO thresholds
- Statistics for management and RMON
- 802.1q VLAN support
- TCP segmentation offload: up to 256 KB
- IPv6 support for IP/TCP and IP/UDP receive checksum offload
- Fragmented UDP checksum offload for packet reassembly
- Message Signaled Interrupts (MSI)
- Message Signaled Interrupts (MSI-X)
- Interrupt throttling control to limit maximum interrupt rate and improve CPU usage
- Receive packet split header
- Multiple receive queues (Flow Director) 16 x 8 and 32 x 4
- 128 transmit queues
- Receive header replication
- Dynamic interrupt moderation
- DCA support
- TCP timer interrupts
- Relaxed ordering
- Support for 64 virtual machines per port (64 VMs x 2 queues)
- Support for Data Center Bridging (DCB)(802.1Qaz, 802.1Qbb, 802.1p)

### Host Interface

- PCIe Base Specification 2.0 (2.5GT/s) or (5GT/s)
- Bus width — x1, x2, x4, x8
- 64-bit address support for systems using more than 4 GB of physical memory

### MAC Functions

- Descriptor ring management hardware for transmit and receive
- ACPI register set and power down functionality supporting D0 and D3 states
- A mechanism for delaying/reducing transmit interrupts
- Software-controlled global reset bit (resets everything except the configuration registers)
- Eight Software-Definable Pins (SDP) per port
- Four of the SDP pins can be configured as general-purpose interrupts
- Wake up
- Ipv6 wake-up filters
- Configurable flexible filter (through EEPROM)
- LAN function disable capability
- Programmable memory transmit buffers (160 KB/port)
- Default configuration by EEPROM for all LEDs for pre-driver functionality
- Support for SR-IOV

### Manageability

- Eight VLAN L2 filters
- 16 flex L3 port filters
- Four Flexible TCO filters
- Four L3 address filters (IPv4)
- Advanced pass through-compatible management packet transmit/receive support
- SMBus interface to an external manageability controller
- NC-SI interface to an external manageability controller
- Four L3 address filters (IPv6)
- Four L2 address filters

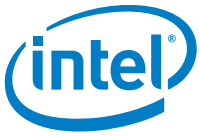
March 2016
Revision 3.3
331520-004

# Revision History

| Rev | Date | Comments |
|---|---|---|
| 0.5 | May 2008 | Initial release (Intel Confidential).<br>This release contains advanced information. |
| 0.6 | October 2008 | Updated to reflect developments, corrections. |
| 0.75 | February 2009 | Major update (all sections) — Reflects latest device developments and corrections. |
| 0.76 | March 2009 | Updated the following sections: Programming Interface, Manageability, NVM, Initialization, Power Management, and Interconnects. |
| 1.0 | March 2009 | Major update (all sections) — Reflects latest device developments and corrections. |
| 1.5 | May 2009 | Major update (all sections) — Reflects latest device developments and corrections. |
| 1.9 | June 2009 | Minor update (all sections) — Reflects latest device developments and corrections. |
| 2.0 | July 2009 | Initial release (Intel Public). |
| 2.01 | July 2009 | Added x8 lane note to Section 1.2.1. |
| 2.1 | October 2009 | • Changed jumbo frame size from KB to bytes (all occurrences).<br>• Changed "XTAL_25_MODE" to "RSVDAC6_VCC".<br>• Updated section 2.1.4 (changed type from T/s to O).<br>• Added F20 and H7 to the table in section 2.1.12.<br>• Changed "OSC_FREQ_SEL" to "RSVDAC6_VCC".<br>• Corrected PCIe versions to "PCIe V2.0 (2.5GT/s or 5GT/s)".<br>• Updated the table in section 3.2.7.2.1 (added text to the vendor ID column).<br>• Updated the jumbo frame calculations in sections 3.7.7.3.3, 3.7.7.3.4, and 3.7.7.3.5.<br>• Added section 4.6.13 "Alternate MAC Address Support".<br>• Updated section 5.2.2 "Auxiliary Power Usage".<br>• Added text to section 6.3.6 "Alternate Ethernet MAC Address - Word Address 0x37".<br>• Updated Table 6.1 (added /1 to row 4).<br>• Updated section 6.4.5.8.<br>• Added L34TIMIR register name to the Queue Enable bit in section 8.2.3.7.19.<br>• Corrected the D10GMP and LMS bit descriptions in section 8.2.3.22.19.<br>• Corrected the LP AN page D low bit description in section 8.2.3.22.23.<br>• Updated the PRDC bit description is section 8.2.3.23.75.<br>• Changed the bit length (31 to 8 to 31 to 0) to the table heading in section 8.2.3.25.12.<br>• Updated the Restart_AN bit description in section 8.2.3.23.22.<br>• Corrected the bit 8 description in section 9.3.7.1.4.<br>• Updated section 10.2.2.2.4 (bits RAGEN and TFOENODX; read/write value).<br>• Added text "Jumbo packets above 2 KB . . . to Filtering exceptions in section 10.3.1.<br>• Correct the Buffer Length (byte 1) description in section 10.5.3.8.2.<br>• Changed the title of table 11.6, 11.7, and 11.8.<br>• Changed Watts to mW in the Power row of table 11.6.<br>• Updated the power values in table 11.7 and 11.8.<br>• Updated the mechanical package drawing in section 11.5.4. |

| Rev | Date | Comments |
|---|---|---|
| 2.1 (cont.) | | • Added power summary table (table 11.6).<br>• Updated section 1.2.1, 3.1.4.5.3, 5.2.5.3.2 (note), and 6.4.5.2.2 (bit descriptions).<br>• Updated bit descriptions for MRQE, RRM, TDRM, and PRDC.<br>• Updated tables in sections 10.3.1, 10.5.1.13.1, and 10.5.2.1.5.<br>• Added Single Port Power table (table 11.8)<br>• Added SFI optics references.<br>• Changed the bit name in section 5.3.1 from APM Wake Up (APM) to APM Enable (APME). |
| 2.2 | January 2010 | • Updated BX4 spec reference (changed 1000BASE-BX4 to 10GBASE-BX4).<br>• Added jumbo frame KB value to note after Table 1.2.<br>• Added new section 1.6.2 "Byte Count".<br>• Added BX4 and CX4 references.<br>• Updated the note in section 2.1.8.<br>• Updated pin name (SDP0_6) in section 2.1.10.<br>• Updated section 3.1.4.5.3 (Relaxed Ordering); last paragraph.<br>• Added BX4 info to section 3.7.<br>• Added new BX4 section (3.7.1.5).<br>• Updated section 3.7.4.4 (link speed).<br>• Updated section 3.7.7.3.3 and 3.7.7.3.5 (jumbo frame values).<br>• Added note after table 3.27 (IPG pacing feature).<br>• Added VFLR note after table 4.6.<br>• Added BX4 reference to section 4.6.4.2.<br>• Added IPG pacing feature note at the end of section 4.6.11.4.<br>• Added jumbo frame value to section 4.6.11.4 and table 4.9 (KB value).<br>• Changed the bit name in section 5.3.1 from APM Wake Up (APM) to APM Enable (APME).<br>• Updated the note in section 5.2.5.3.2 (DMA completions).<br>• Changed GIO Master Disable to PCIe Master Disable (throughout entire EAS).<br>• Changed GIO Master Enable Status to PCIe Master Enable Status (throughout entire EAS).<br>• Updated bullet list in section 5.3.1 and added WKEN bit note at the end of section 5.3.1.<br>• Swapped fields "Possible Len/LLC/SNAP Header" and "Possible VLAN Tag" in sections 5.3.3.1.4. through 5.3.3.1.7 and sections 5.3.3.2.1 and 5.3.3.2.2.<br>• Updated section 6.3.5.4 (changed GIO to PCIe; bit 3 description).<br>• Changed the default setting for CDQMH in section 6.3.6.5 to 0x1404.<br>• Updated section 6.3.5.22 (MSIX and CDO bit definitions).<br>• Removed old 6.3.6.7 section title (Spare 0/1 - Offset 0x05).<br>• Added 5-tuple note to section 7.1.2.5.<br>• Removed sub-bullet under 4-bit RSS Type field in section 7.1.2.8.<br>• Removed TcpIPv6Ex, IPv6Ex and UdpIPV6E info from section 7.1.2.8.1. Updated TCP segment bullet and IPv4 packet sub-bullet in section 7.1.2.8.1.<br>• Updated table 7.10 (Destination Address/Port and Source Address/Port; first row).<br>• Changed RXCTL to DCA_RXCTL[n] under table 7.15 (Packet Buffer Address (64) paragraph).<br>• Changed descriptors per queue value from 64 to 40 in section 7.2.3.3.<br>• Updated figure 7.39 (changed BCN to transmit rate scheduler).<br>• Updated SecTag bullet description in section 7.8.4.<br>• Updated the APME bit description in section 8.2.3.2.9.<br>• Updated the MRQE bit description in section 8.2.3.7.12.<br>• Added a note to the Queue Enable bit description in section 8.2.3.7.19.<br>• Removed the note from section 8.2.3.8.5. |

| Rev | Date | Comments |
|---|---|---|
| 2.2 (cont.) | | • Changed GIO to PCIe in section 8.2.2.1.1 (bit 2 description).<br>• Updated the RRM bit description in section 8.2.2.11.1.<br>• Updated SECTX_OFF_DIS and ECC_TXERR bit descriptions in section 8.2.2.13.2.<br>• Updated SECRX_OFF_DIS and ECC_RXERR bit descriptions in section 8.2.2.13.7.<br>• Added a note to the KX_support bit description in section 8.2.2.23.22.<br>• Updated the PRDC bit description in section 8.2.2.24.75.<br>• Updated bit 4 description (WKEN) in section 8.2.2.25.1.<br>• Added a VF Mailbox note to section 8.3.5.1.5.<br>• Changed RW to RO in section 9.3.10.13 title.<br>• Updated the Filters table in section 10.3.1.<br>• Added note to section 10.5.1.13.1 (TCO Mode reference).<br>• Updated the TCO Mode table in section 10.5.2.1.4.<br>• Updated section 11.3.1.1 (rise time relationships).<br>• Added Single Port Power table (Table 11.8)<br>• Changed all SFI Optics references to unconditional text (now exposed to external customers).<br>• Added single port power numbers (table 11.8).<br>• Added BX4 to section 11.4.4.<br>• Changed crystal load capacitance to 27 pF. |
| 2.3 | April 2010 | • Updated section 3.7.7.1.4 (changed TXOFF to TC_XON).<br>• Changed VMBMEM to VFMBMEM.<br>• Updated section 5.3.2 (last paragraph).<br>• Added a note after the table in section 6.4.2.3.<br>• Updated section 8.2.3.5.13 - changed VT31 to VT32.<br>• Changed all occurrences of SPD to SDP in section 8.2.3.1.4.<br>• Updated the TC_XON field description.<br>• Updated Table 9.6 - Address Space (low register for 64-bit memory BARs) description.<br>• Added recommended and minimum EEPROM sizes to section 12.6.2. |
| 2.4 | September 2010 | The following was updated and or changed for this release:<br>• Section 4.6.11.3.1 (changed MRQC.VT_Ena to MTQC.VT_Ena).<br>• Section 4.6.11.3.3 (changed “via setting RTTDQSEL first for the lowest indexed queue of a pool” to “via setting RTTDQSEL first for the pool index”).<br>• Section 4.6.11.6.1 (updated first step under “Refill Credits”).<br>• Section 4.6.12 (updated Security Offload description).<br>• Section 6.3.2.3 (APM Enable Port 1/0 bit descriptions).<br>• Section 6.3.3 (PBA Number Module — Word Address 0x15-0x16).<br>• Section 6.3.8 (Checksum Word Calculation (Word 0x3F)).<br>• Section 6.4.5.5 (PCIe Control 1 —Offset 0x04).<br>• Section 6.4.5.8 (PCIe Control 3 — Offset 0x07).<br>• Section 7.1.2.3 (L2 Ethertype Filters, step 9).<br>• Section 7.1.2.5 (L3/L4 5-tuple Filters, removed “If the packet is mirrored or replicated . . .”.<br>• Section 7.1.2.7 (Flow Director Filters, removed “In case of mirroring or replication. . .”.<br>• Section 7.2.5.3 (added a note to Tx SCTP CRC Offload).<br>• Section 8.2.3.11.4 (TXDQ_IDX bit description).<br>• Section 8.2.3.11.5 (register RTTDT1C description).<br>• Section 8.2.3.10.3 (VT bit description).<br>• Section 8.2.3.8.2 (VET bit description).<br>• Section 8.2.3.11.9 (DCB Transmit Descriptor Plane T2 Config bit descriptions). |



| Rev | Date | Comments |
|---|---|---|
| 2.4 (cont.) | | • Section 8.2.3.13 (updated Security Offload description).<br>• Section 8.2.3.13.5 (updated MINSECIFG and SECTXDCB bit descriptions).<br>• Section 8.2.3.21.22 (updated Rx Queue Index bit description). |
| 2.5 | November 2010 | The following was updated and or changed for this release:<br>• Section 2.1.8 (changed pull-up to pull-down in the note following the table).<br>• Section 6.4.2 (updated bit 15 bit description).<br>• Section 7.1.2.2 (updated RSS queues reference).<br>• Section 7.1.11 (updated IPv6 filter description).<br>• Section 7.7.2.2 (added a note about using advanced transmit descriptors in DCB mode).<br>• Section 8.2.3.6.1 (added notation about the EICR register).<br>• Section 8.2.3.8.4 (updated the RQPL bit description).<br>• Section 8.2.3.25.3 (updated the WUS register description).<br>• Section 9.3.10.7 (updated bit description for bits 9:4).<br>• Section 11.4.5.1 (changed load capacitance value to 20 pF).<br>• Added new Table 12-1 (Microstrip Trace Dimensions for SFI Using Different Dielectric Materials).<br>• Section 12.12.1.1 (updated the part numbers for recommended crystals).<br>• Updated Figures 12-20 and 12-21 (changed 10 KΩ to 100 Ω).<br>• Section 13.11.4 (updated the maximum static normal load value). |
| 2.6 | December 2010 | • Updated section 3.4.7 EEPROM Recovery (changed Data Byte value from 0xD8 to 0xB6).<br>• Added reference clock specifications note to section 11.4.3.<br>• Updated table 11.25 (changed duty cycle values and added p-noise for non-high serial speed parameter.<br>• Added new figure 11.16 (refclk phase noise as a function of frequency). |
| 2.7 | April 2011 | • Updated Table 1.5 (Flow Director Filters).<br>• Revised section 2.1.13 (LAN1_DIS_N and LAN1_DIS_N name and function description).<br>• Revised section 3.1.4.6.1 (changed “two credits” to “four credits” under “Rules for FC updates”).<br>• Revised table 4.4 (LAN Disable Strapping Pins row; removed “X” from PCIe PERST# and In-band PCIe Reset<br>• columns).<br>• Added SECTXMINIFG.SECTXDCB field reference to sections 4.6.11.3.1 4.6.11.3.2.<br>• Revised section 6.4.5.11 (PCIe Dummy Device ID — Offset 0x0A; changed default value to 0x10A6).<br>• Revised section 7.1.2.7 (Flow Director Filters).<br>• Revised table 7.5 (Flow Director Filters).<br>• Revised section 7.1.2 (added cross reference to last bullet).<br>• Revised section 7.1.2.1 (Queuing in a Non-virtualized Environment).<br>• Revised section 7.1.2.2 (Queuing in a Virtualized Environment).<br>• Revised table 7.19 (Receive Errors (RDESC.ERRORS) Layout).<br>• Revised section 7.1.7.1 (Fetch On Demand; removed Figure 12 reference).<br>• Revised section 8.2.3.21.1 (Flow Director Filters Control Register; bits 1:0 description).<br>• Added a FTFT register note to section 8.2.3.25.12.<br>• Revised the tables at the end of section 8.2.3.24.9 (Flexible Host Filter Table Registers — FHFT). |

| Rev | Date | Comments |
|---|---|---|
| 2.7 (cont.) | | • Revised section 8.2.3.22.8 (MAC Core Control 0 Register; changed MDCSPD default value to 1b).<br>• Revised section 8.2.3.8.6 (Receive Descriptor Control — RXDCTL[n]; corrected bit assignments). |
| 2.71 | September 22, 2011 | • Section 3.1.3.1.2. Case 1 table updated. Tag ID 30 information added.<br>• Section 6.3.5.8, PCIe Control 3 — Offset 0x07. PREFBAR, Bit 14 exposed.<br>• Section 6.2.10, Software Reserved Word 16 — Alternate SAN MAC Block Pointer — Word Address 0x27 and Section 6.2.11, Software Reserved Word 17 — Active SAN MAC Block Pointer — Word Address 0x28 updated.<br>• Table 6-7, Usable Flash_Size. Exposed in Datasheet.<br>• Section 7.1.2.7.4; note at the end of section updated. See phrase "RXPBSIZE[0...3] to 0x18000 (96 K) and RXPBSIZE[4...7] to zero."<br>• Section 8.2.3.7.4, Packet Split Receive Type Register — PSRTYPE[n] (0x0EA00 + 4*n, n=0...63 / 0x05480 + 4*n, n=0…15; RW), Additional bullet added to note. See: "PSR_type4 should be set to enable RSC, regardless header split mode."<br>• Table 9-4, Bit 3 description field updated. New text: "This bit should be set only on systems that do not generate prefetchable cycles."<br>• Table 10-1, "Clear Ethernet MAC Address" command removed from list of supported commands. This command no longer exists in specification. Also, "Set Ethernet MAC Address" command corrected; now called "Set MAC Address". |
| 2.72 | October 18, 2011 | • Section 6.2.7, Alternate Ethernet MAC Address — Word Address 0x37. In table, word "port" changed to "function".<br>• Section 6.2.11.1, Active SAN MAC Address Block. Added this section. Was missing from Datasheet.<br>• Section 6.4.4.7, NC-SI Configuration Offset 0x06. Updated. Description added to bits 4:0. |
| 2.73 | December 7, 2011 | • Section 2.1.16. Pin name assignments corrected for SDP0[7:0] and SDP0[7:0].<br>• Section 5.2.5.3.2. Sentence changed in section. See the new wording: "The driver then reads the change made to the PCIe Master Disable bit and then polls the PCIe Master Enable Status bit."<br>• Section 5.2.4.2. Sentence changed. See the revised sentence: "When a XAUI interface is in low-power state, the 82599 asserts the respective SDP pin to enable an external PHY device to power down as well."<br>• Section 5.2.5.4.1. Sentence updated. See the revised sentence: "Note that the state of the **SDP pins** is undefined once power is removed from the device."<br>• Section 6.2.9, Software Reserved Word 15 — Ext. Thermal Sensor Configuration Block Pointer — Word Address 0x26. Section added: Pointer to External Thermal Sensor Configuration block.<br>• Figure 7-37 updated to correct rendering problem in figure.<br>• Section 7.2.1.2. Under discussion of PAYLEN field in FCoE; revised sentence. New text is: "In FCoE TSO offload, the PAYLEN field defines the FC payload size."<br>• Table 11-25, SerDes Crystal Specifications. In table, Shunt Capacitance recommendation changed from 6[pf] maximum to 7[pf] maximum.<br>• Section 8.2.3.1.4. Table footnote (#2) added.<br>• Fix needed in flow control Statistic Counters: |
| 2.74 | February 03, 2012 | • Section 1.2, Product Overview. Note explaining single and dual port information context added.<br>• Section 7.1.5, Legacy Receive Descriptor Format; see VP (VLAN Packet subsection. This text has been updated. New text is: "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 active VP field also means that the VLAN has been stripped from the packet to the receive descriptor. See further description of 802.1q VLANs in Section 7.4." |

| Rev | Date | Comments |
|---|---|---|
| 2.74 (cont.) | | • Section 8.2.3.23.13, Priority XON Transmitted Count — PXONTXC[n] (0x03F00 + 4*n, n=0...7; RC) - New description for XONTXC field - "Number of XON packets transmitted per TC. Sticks to 0xFFFF".<br>• Section 8.2.3.23.14, Priority XON Received Count — PXONRXCNT[n] (0x04140 + 4*n, n=0...7; RC) - New description for XONRXC field - "Number of XON packets received per UP. Sticks to 0xFFFF".<br>• Section 8.2.3.23.15, Priority XOFF Transmitted Count — PXOFFTXCNT[n] (0x03F20 + 4*n, n=0...7; RC) - New description for XOFFTXC field - "Number of XOFF packets transmitted per TC. Sticks to 0xFFFF".<br>• Section 8.2.3.23.16, Priority XOFF Received Count — PXOFFRXCNT[n] (0x04160 + 4*n, n=0...7; RC) - New description for XOFFRXC field - "Number of XOFF packets received per UP. Sticks to 0xFFFF".<br>• Section 8.2.3.27.7, PF VF Receive Enable — PFVFRE[n] (0x051E0 + 4*n, n=0...1; RW) and Section 8.2.3.27.8, PF VF Transmit Enable — PFVFTE[n] (0x08110 + 4*n, n=0...1; RW). Text changed in both descriptions. New text is: "Respective bits per VF are reset on VFLR, BME bit clear or on VF software reset."<br>• Tables modified: Table 11-6, Power Summary for Dual Port Devices (82599ES, 82599EB) and Table 11-7, Power Summary for Single Port Device (82599EN). These tables now provide clear power summaries for single port and dual port devices. |
| 2.75 | April 24, 2012 | • Section 2.1.8, NC-SI. Note corrected; now specifies correct pull-ups/downs used when NC-SI is disconnected.<br>• Section 4.2.3, Reset Effects. Note #11. PSRTYPE removed from the list.<br>• Table 4-3, Power-Up Timing Guarantees. The $t_{opll}$ and $t_{pcipll}$ descriptions were updated. The $t_{pgres}$ MAX value was added.<br>• Section 5.2.5.3.2, Master Disable. There is new material in the section. The new text begins: "In the above situation, the data path must be flushed before the software resets the 82599. The recommended method to flush the transmit data path is...."; the discussion continues with a methodology presentation.<br>• Table 5-3, Start-up and Power State Transition Timing Parameters. Footnote with link to Table 4.4 added. $t_{pres}$ MIN value entered at 100 ms.<br>• Section 8.2.3.5.1, Extended Interrupt Cause Register- EICR (0x00800; RW1C). Bit 31 exposed (Other Cause Interrupt bit).<br>• Section 7.10.2.2.7, Serial ID. Section updated. New text is "The serial ID capability is not supported in VFs."<br>• Section 8.2.3.1.3, Extended Device Control Register — CTRL_EXT (0x00018; RW). Note text has been added to internal version of the bit 16 discussion (which has also been exposed for external use). The new text is: "The bit must be set during Rx flow initialization for proper device operation."<br>• Section 8.2.3.4.12, PCIe Control Extended Register — GCR_EXT (0x11050; RW). Exposed Buffers Clear Function (bit 30) in the Datasheet. Was RESERVED.<br>• Section 8.2.3.22.8, MAC Core Control 0 Register — HLREG0 (0x04240; RW). Exposed RXCRCSTRP bit in the Datasheet. Was RESERVED.<br>• Section 8.2.3.22.19, Auto Negotiation Control Register — AUTOC (0x042A0; RW). Description of bits 11 &10 updated (DI0GMP, RATD).<br>• Section 12.2.5, Trace Geometries. The inadequate data provided in this section has been replaced by a reference to a document that contains complete information. |
| 2.76 | September 6, 2012 | • Section 3.7.7.3.1, Priority Flow Control. The first sentence in the section was updated for clarity.<br>• Figure 7-6 and Figure 7-7. These have been updated for clarity.<br>• Section 7.1.2.3, L2 Ethertype Filters:<br>— Text has been updated in items 8 & 9.<br>— Check the mirroring rules bullet in the same section; both second-level bullets after this bullet have been updated.<br>• Section 7.1.2.7.11, Query Filter Flow. The table in this section was updated. Note the N/A entries. |

| Rev | Date | Comments |
|---|---|---|
| 2.76 (cont.) | | • Section 7.1.2.7, Flow Director Filters. A note has been added. See: "Note: IPv6 extended headers are parsed by the 82599, 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.<br>• Section 7.1.2.8.1, RSS Hash Function. A note has been added. See "Note: IPv6 extended headers are parsed by the 82599, 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.<br>• Section 7.1.5, Legacy Receive Descriptor Format. A paragraph has been updated for clarity. Search for "The VP field indicates whether the incoming packet's type is a VLAN (802.1q). It is set if the packet type matches VLNCTRL.VET. Furthermore, if the RXDCTL.VME bit is set then active VP bit also indicates that VLAN has been stripped in the 802.1q packet..."<br>• Section 7.2.1.2, Transmit Path in the 82599. A sentence in one of the subsections was rephrased. Search for "Each on-die descriptor queue contains up to 40 descriptors..."<br>• Section 7.2.3.2.4, Advanced Transmit Data Descriptor. A sentence has been updated. Search for "optional VLAN tagging, the FCoE trailer containing the FC CRC and EOF (for FCoE packets), Ethernet CRC or Ethernet padding."<br>• Section 7.1.10, Header Splitting. A note (indicating a restriction) has been added to this section. See "Note: Header Splitting mode might cause unpredictable behavior and should not be used with the 82599. For more information, see the product specification update errata on this subject."<br>• Section 7.13.3.3.6, DDP Context. A sentence in a subsection was updated. Search for "Hardware uses the SEQ_CNT for checking in order reception."<br>• Section 8.2.3.21.20, Flow Director Filters VLAN and FLEX Bytes — FDIRVLAN (0x0EE24; RW). Find the VLAN Field; the description for this field has been updated.<br>• Section 8.2.3.22.23, Auto Negotiation Link Partner Link Control Word 1 Register — ANLP1 (0x042B0; RO). The description for the ANAS field has been updated.<br>• Section 8.2.3.21.22, Flow Director Filters Command Register — FDIRCMD (0x0EE2C; RW). The description has been updated for the FDIRCMD,Drop bit.<br>• Section 9.3.10.7, Link Capabilities Register (0xAC; RO). A value for bits 14:12 has been updated. The new value is "111b = More than 64 ms."<br>• Section 12.11.1, LAN Disable. A paragraph was deleted because it referred to an obsolete function. The current second paragraph is also new.<br>• Section 12.2.1, MAUI Channels Lane Connections. A sentence in the second paragraph was deleted because it did not apply.<br>• Section 15.0, Glossary and Acronyms. The list was updated.Obsolete entries were removed. |
| 2.8 | June 21, 2013 | • Section 1.2, Product Overview - Modified version information.<br>• Section 3.2.5.1, Transmit Errors in Sequence Handling - Fixed typos in note at end of section.<br>• Section 3.7.4.2, MAC Link Setup and Auto Negotiation - Added content from Specification Update as note to auto-negotiation discussion.<br>• Section 4.6.9, FCoE Initialization Flow - Exposed TSOFF, TEOFF, RSOFF and REOFF registers to external documentation.<br>• Section 6.2.4, Software Reserved Word — PXE VLAN Configuration Pointer — Word Address 0x20 - Add PXE VLAN NVM words to NVM Maps.<br>• Added Section 6.2.6.1, PXE Setup Options PCI Function 0 — Word Address 0x30.<br>• Section 6.3.5.6, PCIe Control 2 — Offset 0x05 - Changed default value of Bit 2 (Dummy Function Enable) from 1b to 0b.<br>• Section 7.1.2.7.2, Flow Director Filters Status Reporting - Replaced paragraph regarding packets that do not match a flow director filter. |

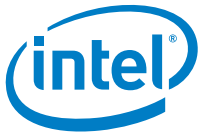
| Rev | Date | Comments |
|---|---|---|
| 2.8 (cont.) | | • Section 7.1.2.7.7, Update Filter Flow - Add content from Specification update regarding internal memory space requirements.<br>• Section 7.1.6.2, Advanced Receive Descriptors — Write-Back Format - Corrected typos in Table 7-16 and related paragraph.<br>• Section 7.2.3.2.3, Advanced Transmit Context Descriptor - Added titles to tables 7-35, 7-36 and 7-37, and added respective cross references in related body text.<br>• Section 7.2.3.2.3, Advanced Transmit Context Descriptor - Changed text in FCoEF description from "EOFF to "TEOFF."<br>• Section 7.13.2.7.4, Dynamic End Of Frame Fields - Replaced Table 7-93, EOF Codes in TSO.<br>• Section 7.7.2.4.1, Definition and Description of Parameters - Modified text in first paragraph.<br>• Section 7.3.4.3.2, MSI-X Vectors Used by Virtual Functions (VFs) - Corrected typos in Figure 7-25 through Figure 7-27 .<br>• Section 8.2.3.1.6, LED Control — LEDCTL (0x00200; RW) - Modified text for LINK/ ACTIVITY description.<br>• Section 8.2.3.5.1, Extended Interrupt Cause Register- EICR (0x00800; RW1C), Removed Step 3 from Flow Director description.<br>• Section 8.2.3.20, FCoE Registers - Exposed TSOFF, TEOFF, RSOFF and REOFF registers to external documentation.<br>• Section 8.2.3.21.10, Flow Director Filters Free — FDIRFREE (0x0EE38; RW) - Changed bits 30:16 to Reserved.<br>• Section 8.2.3.21.11, Flow Director Filters Length — FDIRLEN (0x0EE4C; RC) - Changed bits 30:16 to Reserved.<br>• Section 8.2.3.21.13, Flow Director Filters Failed Usage Statistics — FDIRFSTAT (0x0EE54; RW/RC) - Changed description for bits 7:0.<br>• Section 8.2.3.22.19, Auto Negotiation Control Register — AUTOC (0x042A0; RW) - Added note preceding register description.<br>• Section 8.2.3.22.22, Auto Negotiation Control 2 Register — AUTOC2 (0x042A8; RW) - Exposed bit 28. Bits 27:19 and bit 29 Reserved.<br>• Section 11.6.2, EEPROM - Based on minimum and recommended EEPROM sizes presented in Section 11.6.2.1 and Section 11.6.2.2, removed 8, 16, 32 and 64 Kb devices from Table 11-29<br>• Section 12.3.1, Supported EEPROM Devices - Based on minimum and recommended EEPROM sizes presented in Section 11.6.2.1 and Section 11.6.2.2, removed 8, 16, 32 and 64 Kb devices and accompanying note from Table 12-1<br>• Section 11.3.1.1, Power On/Off Sequence - Added rows to Table 11-5 for Tlpgw, Tlpg-per and Tlpg. |
| 2.9 | January 8, 2014 | • Section 1.2, Product Overview — Updated product version information.<br>• Section 3.4.2, EEPROM Device — Updated EEPROM compatibility information, and added reference to table of support EEPROM devices.<br>• Section 4.6.11.4, Transmit Rate Scheduler — Fixed typo.<br>• Section 6.3.5.13, IOV Control Word 1 — Offset 0x0C — Updated default value for Max VFs filed, and added note to field description.<br>• Section 7.2.3.1, Introduction — Modified legacy descriptors information in Transmit Descriptors "Introduction" section.<br>• Section 7.2.3.2.2, Legacy Transmit Descriptor Format — Corrected typo from "Rx" to "Tx" in Report Status (RS) description.<br>• Section 7.2.3.2.4, Advanced Transmit Data Descriptor — Updated list in "Check Context bit" description. |
| 3.0 | November 5, 2014 | • Section 4.6.3.2, Global Reset and General Configuration — Updated text related to FCRTH[n].RTH fields.<br>• Section 4.6.9, FCoE Initialization Flow — Text updates.<br>• Section 7.1.2.3, L2 Ethertype Filters — Text updates.<br>• Section 8.2.3.23.3, Error Byte Packet Count — ERRBC (0x04008; RC) — Changed long register name from "Error Byte Count". |

| Rev | Date | Comments |
|---|---|---|
| 3.1 | February 1, 2015 | • Section 4.6.6, Interrupt Initialization — Provided additional information on "Operating with Legacy or MSI Interrupts".<br>• Section 6.4, Firmware Module — Made changes associated with the addition of Appendix B.<br>• Section 6.4.4.7, NC-SI Configuration Offset 0x06 — Exposed Enable Channel Swap field (Bit 13).<br>• Section 8.2.3.1.2, Device Status Register — STATUS (0x00008; RO) — Added text describing the *LinkUp* bit as Read/Write.<br>• Section 11.5.1, Mechanical — Corrected typo (FCGBA -> FCBGA)<br>• Section 12.2.1, MAUI Channels Lane Connections — Added note regarding unused pins and design with 82599EN single port SKU.<br>• Added Appendix A, "Packets and Frames".<br>• Added Appendix B, "LESM - Link Establishment State Machine for the 82599". |
| 3.2 | October 19, 2015 | • Section 6.2.3, iSCSI Boot Configuration — Word Address 0x17 - Updated section. |
| 3.3 | March 11, 2016 | • Section 4.2.1.5.3, Virtual Function FLR (VFLR) — Added note related to VFMBMEM.<br>• Removed VFMBMEM from Note #11 related to Table 4-6.<br>• Section 14.1, Link Loopback Operations — Removed a textual reference to a non-existent register. |

# Contents

**1.0 Introduction** ..... 21
1.1 Scope ..... 21
1.2 Product Overview ..... 21
1.2.1 82599 Silicon/Software Features ..... 22
1.2.2 System Configurations ..... 23
1.2.3 External Interfaces ..... 24
1.2.4 PCI-Express* (PCIe*) Interface ..... 24
1.2.5 Network Interfaces ..... 24
1.2.6 EEPROM Interface ..... 25
1.2.7 Serial Flash Interface ..... 26
1.2.8 SMBus Interface ..... 26
1.2.9 NC-SI Interface ..... 26
1.2.10 MDIO Interfaces ..... 26
1.2.11 I2C Interfaces ..... 27
1.2.12 Software-Definable Pins (SDP) Interface (General-Purpose I/O) ..... 27
1.2.13 LED Interface ..... 28
1.3 Features Summary ..... 28
1.4 Overview of New Capabilities Beyond the 82598 ..... 33
1.4.1 Security ..... 33
1.4.2 Transmit Rate Limiting ..... 33
1.4.3 Fibre Channel over Ethernet (FCoE) ..... 33
1.4.4 Performance ..... 34
1.4.5 Rx/Tx Queues and Rx Filtering ..... 35
1.4.6 Interrupts ..... 35
1.4.7 Virtualization ..... 35
1.4.8 VPD ..... 36
1.4.9 Double VLAN ..... 36
1.4.10 Time Sync — IEEE 1588 — Precision Time Protocol (PTP) ..... 37
1.5 Conventions ..... 37
1.5.1 Terminology and Acronyms ..... 37
1.5.2 Byte Count ..... 37
1.5.3 Byte Ordering ..... 37
1.6 Register/Bit Notations ..... 38
1.7 References ..... 38
1.8 Architecture and Basic Operation ..... 41
1.8.1 Transmit (Tx) Data Flow ..... 41
1.8.2 Receive (Rx) Data Flow ..... 42
**2.0 Pin Interface** ..... 43
2.1 Pin Assignment ..... 43
2.1.1 Signal Type Definition ..... 43
2.1.2 PCIe Symbols and Pin Names ..... 44
2.1.3 MAUI ..... 45
2.1.4 EEPROM ..... 47
2.1.5 Serial Flash ..... 47
2.1.6 SMBus ..... 48
2.1.7 $I^2C$ ..... 48
2.1.8 NC-SI ..... 49

2.1.9 MDIO ..... 49
2.1.10 Software Defined Pins (SDPs) ..... 50
2.1.11 LEDs ..... 50
2.1.12 RSVD and No Connect Pins ..... 51
2.1.13 Miscellaneous ..... 53
2.1.14 JTAG ..... 54
2.1.15 Power Supplies ..... 54
2.1.16 Pull-Ups ..... 55
2.2 Ball Out — Top Level ..... 58
**3.0 Interconnects ..... 61**
3.1 PCI-Express* (PCIe*) ..... 61
3.1.1 Overview ..... 61
3.1.2 General Functionality ..... 64
3.1.3 Host Interface ..... 64
3.1.4 Transaction Layer ..... 68
3.1.5 Link Layer ..... 75
3.1.6 Physical Layer ..... 76
3.1.7 Error Events and Error Reporting ..... 80
3.1.8 Performance Monitoring ..... 86
3.2 SMBus ..... 87
3.2.1 Channel Behavior ..... 87
3.2.2 SMBus Addressing ..... 87
3.2.3 SMBus Notification Methods ..... 88
3.2.4 Receive TCO Flow ..... 90
3.2.5 Transmit TCO Flow ..... 91
3.2.6 Concurrent SMBus Transactions ..... 93
3.2.7 SMBus ARP Functionality ..... 93
3.2.8 LAN Fail-Over Through SMBus ..... 97
3.3 Network Controller — Sideband Interface (NC-SI) ..... 97
3.3.1 Electrical Characteristics ..... 97
3.3.2 NC-SI Transactions ..... 98
3.4 EEPROM ..... 98
3.4.1 General Overview ..... 98
3.4.2 EEPROM Device ..... 98
3.4.3 EEPROM Vital Content ..... 98
3.4.4 Software Accesses ..... 99
3.4.5 Signature Field ..... 99
3.4.6 Protected EEPROM Space ..... 100
3.4.7 EEPROM Recovery ..... 101
3.4.8 EEPROM Deadlock Avoidance ..... 102
3.4.9 VPD Support ..... 103
3.5 Flash ..... 104
3.5.1 Flash Interface Operation ..... 104
3.5.2 Flash Write Control ..... 105
3.5.3 Flash Erase Control ..... 105
3.5.4 Flash Access Contention ..... 105
3.6 Configurable I/O Pins — Software-Definable Pins (SDP) ..... 106
3.7 Network Interface (MAUI Interface) ..... 109
3.7.1 10 GbE Interface ..... 110
3.7.2 GbE Interface ..... 121




3.7.3 SGMII Support ..... 123
3.7.4 Auto Negotiation For Backplane Ethernet and Link Setup Features ..... 125
3.7.5 Transceiver Module Support ..... 129
3.7.6 Management Data Input/Output (MDIO) Interface ..... 130
3.7.7 Ethernet Flow Control (FC) ..... 136
3.7.8 Inter Packet Gap (IPG) Control and Pacing ..... 146
3.7.9 MAC Speed Change at Different Power Modes ..... 147

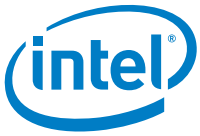
**4.0 Initialization ..... 151**

4.1 Power Up ..... 151
4.1.1 Power-Up Sequence ..... 151
4.1.2 Power-Up Timing Diagram ..... 152
4.2 Reset Operation ..... 155
4.2.1 Reset Sources ..... 155
4.2.2 Reset in PCI-IOV Environment ..... 158
4.2.3 Reset Effects ..... 159
4.3 Queue Disable ..... 162
4.4 Function Disable ..... 163
4.4.1 General ..... 163
4.4.2 Overview ..... 163
4.4.3 Control Options ..... 165
4.4.4 Event Flow for Enable/Disable Functions ..... 165
4.5 Device Disable ..... 166
4.5.1 Overview ..... 166
4.5.2 BIOS Disable of the Device at Boot Time by Using the Strapping Option ..... 167
4.6 Software Initialization and Diagnostics ..... 167
4.6.1 Introduction ..... 167
4.6.2 Power-Up State ..... 167
4.6.3 Initialization Sequence ..... 168
4.6.4 100 Mb/s, 1 GbE, and 10 GbE Link Initialization ..... 169
4.6.5 Initialization of Statistics ..... 170
4.6.6 Interrupt Initialization ..... 170
4.6.7 Receive Initialization ..... 171
4.6.8 Transmit Initialization ..... 175
4.6.9 FCoE Initialization Flow ..... 176
4.6.10 Virtualization Initialization Flow ..... 177
4.6.11 DCB Configuration ..... 180
4.6.12 Security Initialization ..... 191
4.6.13 Alternate MAC Address Support ..... 193

**5.0 Power Management and Delivery ..... 195**

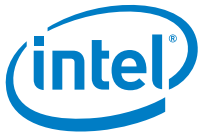
5.1 Power Targets and Power Delivery ..... 195
5.2 Power Management ..... 195
5.2.1 Introduction to the 82599 Power States ..... 195
5.2.2 Auxiliary Power Usage ..... 196
5.2.3 Power Limits by Certain Form Factors ..... 196
5.2.4 Interconnects Power Management ..... 197
5.2.5 Power States ..... 199
5.2.6 Timing of Power-State Transitions ..... 204
5.3 Wake Up ..... 208
5.3.1 Advanced Power Management Wake Up ..... 208
5.3.2 ACPI Power Management Wake Up ..... 208

5.3.3 Wake-Up Packets ..... 209
5.3.4 Wake Up and Virtualization ..... 215
**6.0 Non-Volatile Memory Map ..... 217**
6.1 EEPROM General Map ..... 217
6.2 EEPROM Software ..... 219
6.2.1 SW Compatibility Module — Word Address 0x10-0x14 ..... 219
6.2.2 PBA Number Module — Word Address 0x15-0x16 ..... 219
6.2.3 iSCSI Boot Configuration — Word Address 0x17 ..... 220
6.2.4 Software Reserved Word — PXE VLAN Configuration Pointer — Word Address 0x20 ..... 223
6.2.5 VPD Module Pointer — Word Address 0x2F ..... 224
6.2.6 EEPROM PXE Module — Word Address 0x30-0x36 ..... 224
6.2.7 Alternate Ethernet MAC Address — Word Address 0x37 ..... 227
6.2.8 Checksum Word Calculation (Word 0x3F) ..... 227
6.2.9 Software Reserved Word 15 — Ext. Thermal Sensor Configuration Block Pointer — Word Address 0x26 ..... 229
6.2.10 Software Reserved Word 16 — Alternate SAN MAC Block Pointer — Word Address 0x27 ..... 229
6.2.11 Software Reserved Word 17 — Active SAN MAC Block Pointer — Word Address 0x28 ..... 230
6.3 EEPROM Hardware Sections ..... 231
6.3.1 EEPROM Hardware Section — Auto-Load Sequence ..... 231
6.3.2 EEPROM Init Module ..... 231
6.3.3 PCIe Analog Configuration Module ..... 233
6.3.4 Core 0/1 Analog Configuration Modules ..... 234
6.3.5 PCIe General Configuration Module ..... 235
6.3.6 PCIe Configuration Space 0/1 Modules ..... 244
6.3.7 LAN Core 0/1 Modules ..... 246
6.3.8 MAC 0/1 Modules ..... 249
6.3.9 CSR 0/1 Auto Configuration Modules ..... 255
6.4 Firmware Module ..... 257
6.4.1 Test Configuration Module ..... 257
6.4.2 Common Firmware Parameters — (Global MNG Offset 0x3) ..... 258
6.4.3 Pass Through LAN 0/1 Configuration Modules ..... 259
6.4.4 Sideband Configuration Module ..... 268
6.4.5 Flexible TCO Filter Configuration Module ..... 270
6.4.6 NC-SI Microcode Download Module ..... 272
6.4.7 NC-SI Configuration Module ..... 272
**7.0 Inline Functions ..... 277**
7.1 Receive Functionality ..... 277
7.1.1 Packet Filtering ..... 278
7.1.2 Rx Queues Assignment ..... 282
7.1.3 MAC Layer Offloads ..... 310
7.1.4 Receive Data Storage in System Memory ..... 310
7.1.5 Legacy Receive Descriptor Format ..... 310
7.1.6 Advanced Receive Descriptors ..... 313
7.1.7 Receive Descriptor Fetching ..... 323
7.1.8 Receive Descriptor Write-Back ..... 323
7.1.9 Receive Descriptor Queue Structure ..... 324
7.1.10 Header Splitting ..... 327
7.1.11 Receive Checksum Offloading ..... 330
7.1.12 SCTP Receive Offload ..... 333
7.1.13 Receive UDP Fragmentation Checksum ..... 334



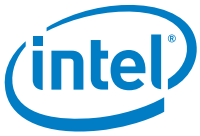

7.2 Transmit Functionality ..... 335
7.2.1 Packet Transmission ..... 335
7.2.2 Transmit Contexts ..... 344
7.2.3 Transmit Descriptors ..... 345
7.2.4 TCP and UDP Segmentation ..... 361
7.2.5 Transmit Checksum Offloading in Non-segmentation Mode ..... 369
7.3 Interrupts ..... 373
7.3.1 Interrupt Registers ..... 373
7.3.2 Interrupt Moderation ..... 377
7.3.3 TCP Timer Interrupt ..... 381
7.3.4 Mapping of Interrupt Causes ..... 381
7.4 802.1q VLAN Support ..... 388
7.4.1 802.1q VLAN Packet Format ..... 388
7.4.2 802.1q Tagged Frames ..... 388
7.4.3 Transmitting and Receiving 802.1q Packets ..... 389
7.4.4 802.1q VLAN Packet Filtering ..... 390
7.4.5 Double VLAN and Single VLAN Support ..... 390
7.5 Direct Cache Access (DCA) ..... 394
7.5.1 PCIe TLP Format for DCA ..... 395
7.6 LEDs ..... 396
7.7 Data Center Bridging (DCB) ..... 397
7.7.1 Overview ..... 397
7.7.2 Transmit-side Capabilities ..... 400
7.7.3 Receive-Side Capabilities ..... 413
7.8 LinkSec ..... 417
7.8.1 Packet Format ..... 418
7.8.2 LinkSec Header (SecTag) Format ..... 418
7.8.3 LinkSec Management – KaY (Key Agreement Entity) ..... 420
7.8.4 Receive Flow ..... 421
7.8.5 Transmit Data Path ..... 424
7.8.6 LinkSec and Manageability ..... 425
7.8.7 Key and Tamper Protection ..... 425
7.8.8 LinkSec Statistics ..... 426
7.9 Time SYNC (IEEE1588 and 802.1AS) ..... 428
7.9.1 Overview ..... 428
7.9.2 Flow and Hardware/Software Responsibilities ..... 428
7.9.3 Hardware Time Sync Elements ..... 430
7.9.4 Time Sync Related Auxiliary Elements ..... 433
7.9.5 PTP Packet Structure ..... 434
7.10 Virtualization ..... 437
7.10.1 Overview ..... 437
7.10.2 PCI-SIG SR-IOV Support ..... 441
7.10.3 Packet Switching ..... 452
7.10.4 Virtualization of Hardware ..... 463
7.11 Receive Side Coalescing (RSC) ..... 464
7.11.1 Packet Viability for RSC Functionality ..... 466
7.11.2 Flow Identification and RSC Context Matching ..... 468
7.11.3 Processing New RSC ..... 470
7.11.4 Processing Active RSC ..... 470
7.11.5 Packet DMA and Descriptor Write Back ..... 472

7.11.6 RSC Completion and Aging ..... 474
7.12 IPsec Support ..... 476
7.12.1 Overview ..... 476
7.12.2 Hardware Features List ..... 476
7.12.3 Software/Hardware Demarcation ..... 479
7.12.4 IPsec Formats Exchanged Between Hardware and Software ..... 480
7.12.5 TX SA Table ..... 484
7.12.6 TX Hardware Flow ..... 485
7.12.7 AES-128 Operation in Tx ..... 487
7.12.8 RX Descriptors ..... 489
7.12.9 Rx SA Tables ..... 489
7.12.10 RX Hardware Flow without TCP/UDP Checksum Offload ..... 492
7.12.11 RX Hardware Flow with TCP/UDP Checksum Offload ..... 493
7.12.12 AES-128 Operation in Rx ..... 493
7.13 Fibre Channel over Ethernet (FCoE) ..... 495
7.13.1 Introduction ..... 495
7.13.2 FCoE Transmit Operation ..... 496
7.13.3 FCoE Receive Operation ..... 502
7.14 Reliability ..... 518
7.14.1 Memory Integrity Protection ..... 518
7.14.2 PCIe Error Handling ..... 518
**8.0 Programming Interface ..... 519**
8.1 Address Regions ..... 519
8.1.1 Memory-Mapped Access ..... 519
8.1.2 I/O-Mapped Access ..... 520
8.1.3 Registers Terminology ..... 522
8.2 Device Registers — PF ..... 523
8.2.1 MSI-X BAR Register Summary PF ..... 523
8.2.2 Registers Summary PF — BAR 0 ..... 523
8.2.3 Detailed Register Descriptions — PF ..... 543
8.3 Device Registers — VF ..... 734
8.3.1 Registers Allocated Per Queue ..... 734
8.3.2 Non-Queue Registers ..... 734
8.3.3 MSI—X Register Summary VF — BAR 3 ..... 735
8.3.4 Registers Summary VF — BAR 0 ..... 737
8.3.5 Detailed Register Descriptions —VF ..... 739
**9.0 PCIe Programming Interface ..... 749**
9.1 PCI Compatibility ..... 749
9.2 Configuration Sharing Among PCI Functions ..... 750
9.3 PCIe Register Map ..... 752
9.3.1 Register Attributes ..... 752
9.3.2 PCIe Configuration Space Summary ..... 752
9.3.3 Mandatory PCI Configuration Registers — Except BARs ..... 754
9.3.4 Subsystem ID Register (0x2E; RO) ..... 757
9.3.5 Cap_Ptr Register (0x34; RO) ..... 757
9.3.6 Mandatory PCI Configuration Registers — BARs ..... 758
9.3.7 PCIe Capabilities ..... 759
9.3.8 MSI-X Capability ..... 765
9.3.9 VPD Registers ..... 770
9.3.10 PCIe Configuration Registers ..... 771



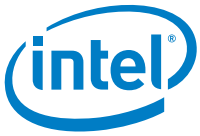

9.4 PCIe Extended Configuration Space ..... 782
9.4.1 Advanced Error Reporting Capability (AER) ..... 783
9.4.2 Serial Number ..... 788
9.4.3 Alternate Routing ID Interpretation (ARI) Capability Structure ..... 790
9.4.4 IOV Capability Structure ..... 791
9.5 Virtual Functions Configuration Space ..... 798
9.5.1 Mandatory Configuration Space ..... 800
9.5.2 PCI Capabilities ..... 802
**10.0 Manageability ..... 805**
10.1 Platform Configurations ..... 805
10.1.1 On-Board BMC Configurations ..... 805
10.1.2 82599 NIC ..... 806
10.2 Pass Through (PT) Functionality ..... 806
10.2.1 DMTF NC-SI Mode ..... 807
10.2.2 SMBus Pass Through (PT) Functionality ..... 809
10.3 Manageability Receive Filtering ..... 813
10.3.1 Overview and General Structure ..... 813
10.3.2 L2 EtherType Filters ..... 815
10.3.3 VLAN Filters - Single and Double VLAN Cases ..... 815
10.3.4 L3 and L4 Filters ..... 816
10.3.5 Manageability Decision Filters ..... 818
10.3.6 Possible Configurations ..... 820
10.4 LinkSec and Manageability ..... 822
10.4.1 Handover of LinkSec Responsibility Between BMC and Host ..... 823
10.5 Manageability Programming Interfaces ..... 825
10.5.1 NC-SI Programming ..... 825
10.5.2 SMBus Programming ..... 874
10.5.3 Manageability Host Interface ..... 911
10.5.4 Software and Firmware Synchronization ..... 915
**11.0 Electrical / Mechanical Specification ..... 919**
11.1 Introduction ..... 919
11.2 Operating Conditions ..... 919
11.2.1 Absolute Maximum Ratings ..... 919
11.2.2 Recommended Operating Conditions ..... 920
11.3 Power Delivery ..... 920
11.3.1 Power Supply Specifications ..... 920
11.3.2 In-Rush Current ..... 922
11.4 DC/AC Specification ..... 922
11.4.1 DC Specifications ..... 922
11.4.2 Digital I/F AC Specifications ..... 927
11.4.3 PCIe Interface AC/DC Specification ..... 938
11.4.4 Network (MAUI) Interface AC/DC Specification ..... 938
11.4.5 SerDes Crystal/Reference Clock Specification ..... 940
11.5 Package ..... 946
11.5.1 Mechanical ..... 946
11.5.2 Thermal ..... 946
11.5.3 Electrical ..... 946
11.5.4 Mechanical Package ..... 947
11.6 Devices Supported ..... 947
11.6.1 Flash ..... 947

11.6.2 EEPROM ..... 948
**12.0 Design Considerations and Guidelines ..... 949**
12.1 Connecting the PCIe Interface ..... 949
12.1.1 Link Width Configuration ..... 949
12.1.2 Polarity Inversion and Lane Reversal ..... 950
12.1.3 PCIe Reference Clock ..... 950
12.1.4 PCIe Analog Bias Resistor ..... 950
12.1.5 Miscellaneous PCIe Signals ..... 950
12.1.6 PCIe Layout Recommendations ..... 950
12.2 Connecting the MAUI Interfaces ..... 951
12.2.1 MAUI Channels Lane Connections ..... 951
12.2.2 MAUI Bias Resistor ..... 951
12.2.3 XAUI, KX/KR, BX4, CX4, BX and SFI+ Layout Recommendations ..... 951
12.2.4 Board Stack-Up Example ..... 952
12.2.5 Trace Geometries ..... 952
12.2.6 Other High-Speed Signal Routing Practices ..... 953
12.2.7 Reference Planes ..... 956
12.2.8 Dielectric Weave Compensation ..... 958
12.2.9 Impedance Discontinuities ..... 959
12.2.10 Reducing Circuit Inductance ..... 959
12.2.11 Signal Isolation ..... 960
12.2.12 Power and Ground Planes ..... 960
12.2.13 KR and SFI+ Recommended Simulations ..... 966
12.2.14 Additional Differential Trace Layout Guidelines for SFI+ Boards ..... 967
12.3 Connecting the Serial EEPROM ..... 969
12.3.1 Supported EEPROM Devices ..... 969
12.4 Connecting the Flash ..... 969
12.4.1 Supported Flash Devices ..... 970
12.5 SMBus and NC-SI ..... 970
12.6 NC-SI ..... 972
12.6.1 NC-SI Design Requirements ..... 972
12.6.2 NC-SI Layout Requirements ..... 974
12.7 Resets ..... 978
12.8 Connecting the MDIO Interfaces ..... 979
12.9 Connecting the Software-Definable Pins (SDPs) ..... 979
12.10 Connecting the Light Emitting Diodes (LEDs) ..... 980
12.11 Connecting Miscellaneous Signals ..... 980
12.11.1 LAN Disable ..... 980
12.11.2 BIOS Handling of Device Disable ..... 981
12.12 Oscillator Design Considerations ..... 982
12.12.1 Oscillator Types ..... 982
12.12.2 Oscillator Solution ..... 983
12.12.3 Oscillator Layout Recommendations ..... 983
12.12.4 Reference Clock Measurement Recommendations ..... 983
12.13 Power Supplies ..... 983
12.13.1 Power Supply Sequencing ..... 984
12.13.2 Power Supply Filtering ..... 984
12.13.3 Support for Power Management and Wake Up ..... 985
12.14 Connecting the JTAG Port ..... 985



**13.0 Thermal Design Recommendations** ..... 987
13.1 Thermal Considerations ..... 987
13.2 Importance of Thermal Management ..... 988
13.3 Packaging Terminology ..... 988
13.4 Thermal Specifications ..... 989
13.5 Case Temperature ..... 990
13.6 Thermal Attributes ..... 990
13.6.1 Designing for Thermal Performance ..... 990
13.6.2 Model System Definition ..... 990
13.6.3 Package Thermal Characteristics ..... 991
13.7 Thermal Enhancements ..... 992
13.8 Clearances ..... 992
13.9 Default Enhanced Thermal Solution ..... 994
13.10 Extruded Heatsinks ..... 995
13.11 Attaching the Extruded Heatsink ..... 996
13.11.1 Clips ..... 996
13.11.2 Thermal Interface (PCM45 Series) ..... 996
13.11.3 Avoid Damaging Die-Side Capacitors with Heat Sink Attached ..... 996
13.11.4 Maximum Static Normal Load ..... 997
13.12 Reliability ..... 998
13.12.1 Thermal Interface Management for Heat-Sink Solutions ..... 998
13.13 Measurements for Thermal Specifications ..... 999
13.13.1 Case Temperature Measurements ..... 999
13.13.2 Attaching the Thermocouple (No Heatsink) ..... 1000
13.13.3 Attaching the Thermocouple (Heatsink) ..... 1000
13.14 Heatsink and Attach Suppliers ..... 1001
13.15 PCB Guidelines ..... 1002
**14.0 Diagnostics** ..... 1003
14.1 Link Loopback Operations ..... 1003
**15.0 Glossary and Acronyms** ..... 1005
15.1 Register Attributes ..... 1016
**Appendix A Packets and Frames** ..... 1017
A.1 Legacy Packet Formats ..... 1017
A.1.1 ARP Packet Formats ..... 1017
A.1.2 IP and TCP/UDP Headers for TSO ..... 1019
A.1.3 Magic Packet ..... 1025
A.1.4 SNAP Packet Format ..... 1025
A.2 Packet Types for Packet Split Filtering ..... 1025
A.2.1 Type 1.1: Ethernet (VLAN/SNAP) IP Packets ..... 1026
A.2.2 Type 2: Ethernet, Ipv6 ..... 1034
A.2.3 Type 3: Reserved ..... 1037
A.2.4 Type 4: NFS Packets ..... 1037
A.3 IPsec Formats Run Over the Wire ..... 1042
A.3.1 AH Formats ..... 1042
A.3.2 ESP Formats ..... 1046
A.4 BCN Frame Format ..... 1051
A.5 FCoE Framing ..... 1052
A.5.1 FCoE Frame Format ..... 1052
A.5.2 FC Frame Format ..... 1055
**Appendix B LESM - Link Establishment State Machine for the 82599** ..... 1063
B.1 Background ..... 1063
B.2 Location in the NVM ..... 1064

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



# 1.0 Introduction

## 1.1 Scope

This document describes the external architecture (including device operation, pin descriptions, register definitions, etc.) for the 82599, a dual 10 Gigabit Ethernet (GbE) Network Interface Controller.

This document is intended as a reference for logical design group, architecture validation, firmware development, software device driver developers, board designers, test engineers, or anyone else who may need specific technical or programming information about the 82599.

## 1.2 Product Overview

The 82599 is a derivative of previous generations of Intel 1 GbE and 10 GbE Network Interface Card (NIC) designs. Many features of its predecessors remain intact; however, some have been removed or modified as well as new features introduced.

**Note:** This device is available in dual port and single port variants. The single port device is limited to port 0 and to the SFI interface. If you are using the single port device, information in this document pertaining to port 1 and other interfaces does not apply.

Three versions are available:

- 82599EB — PCI Express* (PCIe*) 2.0, dual-port 10 Gigabit Ethernet controller for XAUI, KX, KX4, BX, BX4 and CX4 interfaces.
- 82599ES — Dual-port serial 10 GbE backplane interface for blade implementations (includes the 82599EB SKU functionality plus KR and SFI interfaces).
- 82599EN — Single-port serial 10 GbE SFI interface for blade implementations.

**Note:** For addition information, see the introductory material in the *Intel® 82599 10 Gigabit Ethernet Controller Specification Update.*

### 1.2.1 82599 Silicon/Software Features

The base software device driver supports the following interfaces:

- XAUI (BX4)
- SFI
- KX/KX4
- KR

Linux software features include:

- LLI — Low Latency Interrupts
- DCA — Direct Cache Access
- RSC — Receive Side Coalescing
- All sleep states (S0 through S5); however, for sleep states S3 through S5, there are power and airflow conditions that need to be met. Refer to Section 5.0 and Section 11.0 for more details.
- Header Split — This feature consists of splitting a packet header to a different memory space and help the host to fetch headers only for processing.
- Flow Director (SW ATR only) — A large number of flow affinity filters that direct receive packets by their flows to queues for classification, load balancing, and matching between flows and CPU cores.

Windows software features include:

- LLI
- DCA
- Wake on LAN (WoL) support:
  - WoL from S3 and S4 are not currently supported for the 82599. Also, there are no plans to support it in the future.
  - WoL from S5 is supported for connections capable of KR to KX transitions only. Implementation of this feature has special requirements, contact your Intel representative for more details.
- Header Split — This feature consists of splitting a packet header to a different memory space and help the host to fetch headers only for processing.

**Note:** Some PCIe x8 slots are actually configured as x4 slots. These slots have insufficient bandwidth for full 10 GbE line rate with dual port 10 GbE devices. If a solution suffers bandwidth issues when both 10 GbE ports are active, it is recommended to verify that the PCIe slot is indeed a true PCIe x8.

## 1.2.2 System Configurations

The 82599 is targeted for system configurations such as rack mounted or pedestal servers, where it can be used as an add-on NIC or LAN on Motherboard (LOM). Another system configuration is for blade servers, where the 82599 can be used in a LOM or mezzanine card.

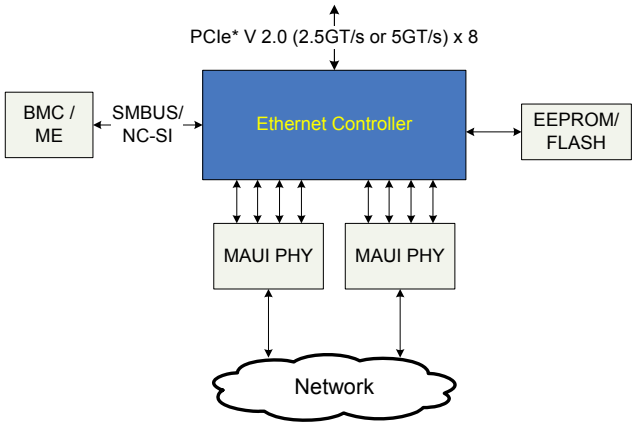



**Figure 1-1 Typical Rack / Pedestal System Configuration**

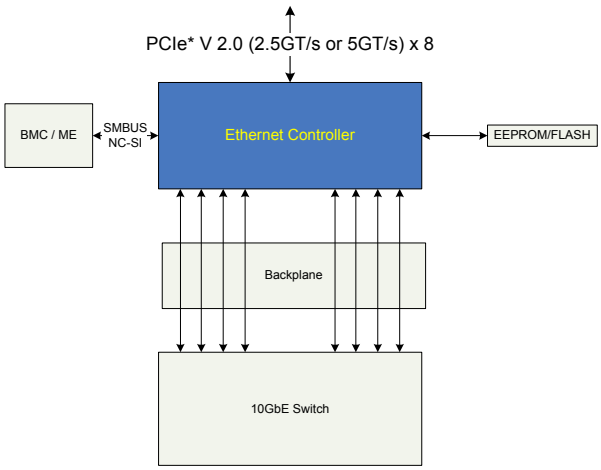



**Figure 1-2 Typical Blade System Configuration**

## 1.2.3 External Interfaces

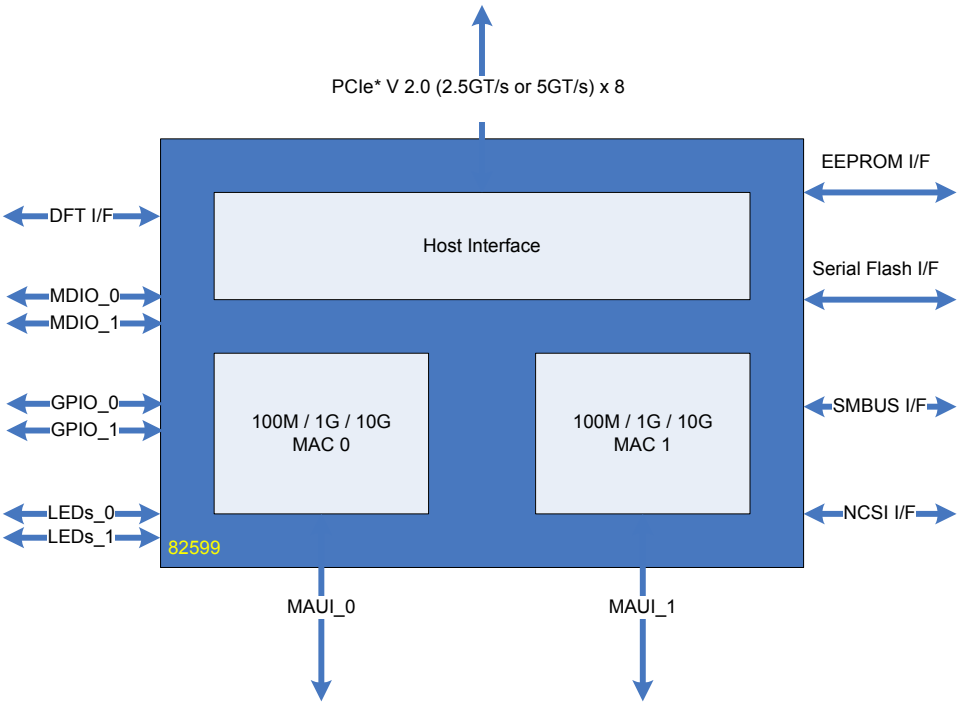



**Figure 1-3 82599 External Interfaces Diagram**

## 1.2.4 PCI-Express* (PCIe*) Interface

The 82599 supports PCIe V2.0 (2.5GT/s or 5GT/s). See Section 2.1.2 for full pin description and Section 11.4.3 for interface timing characteristics.

## 1.2.5 Network Interfaces

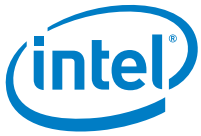
The 82599 interfaces the network through the Multi-Speed Attachment Unit Interface also referred to as the MAUI interface.

Two independent MAUI interfaces are used to connect two the 82599 ports to external devices. Each MAUI interface can be configured to interface using the following high speed links:

a. XAUI for connection to another XAUI compliant PHY device or optical module.

b. SGMII for connection to another SGMII compliant PHY using 1000BASE-BX or 1000BASE-KX electrical signaling.

c. 1000BASE-KX for connection over a backplane to another 1000BASE-KX compliant device.



d. 10GBASE-KX4 for connection over a backplane to another 10GBASE-KX4 device.

e. 1000BASE-BX for connection over a backplane to another 1000BASE-BX compliant device.

f. 10GBASE-CX4 for connection over a CX4 compliant cable to another 10GBASE-CX4 compliant device.

g. SFI for connection to another SFI compliant PHY or optical module.

h. 10GBASE-KR for connection over a backplane to another 10GBASE-KR compliant device.

i. 10GBASE-BX4 for connection over a backplane to another 10GBASE-BX4 device.

The 82599 also supports:

- IEEE 802.3ae (10 Gb/s) implementations. It performs all of the functions required for transmission and reception handling called out in the standards for a XAUI Media interface.
- IEEE 802.3ak, IEEE 802.3ap Backplane Ethernet (KX, KX4, or KR), and PICMG3.1 (BX only) implementations including an Auto-Negotiation layer and PCS layer synchronization.
- SFP+ MSA (SFI) implementations.

These interfaces can be configured to operate in 100 Mb/s mode (SGMII), 1 Gb/s mode (SGMII, BX and KX) and 10 Gb/s mode (XAUI, CX4, KX4, KR and SFI). In 100 Mb/s mode, 1 Gb/s mode and in KR and SFI 10 Gb/s modes, only one of the four MAUI lanes (lane 0) is used and the remaining lanes (lanes 1 to 3) are powered down. For more information on how to configure the 82599 for 100 Mb/s,
1 Gb/s or 10 Gb/s operating modes, refer to Section 3.7.

Refer to Section 2.1.3 for full-pin descriptions and to the respective specifications (IEEE802.3, optical module MSAs...). For the timing characteristics of those interfaces see the relevant external specifications as listed in Section 11.4.4 for interface timing characteristics.

## 1.2.6 EEPROM Interface

The 82599 uses an EEPROM device for storing product configuration information. Several words of the EEPROM are accessed automatically by the 82599 after reset in order to provide pre-boot configuration data that must be available to it before it is accessed by host software. The remainder of the stored information is accessed by various software modules used to report product configuration, serial number, etc.

The 82599 uses a SPI (4-wire) serial EEPROM devices. Refer to Section 2.1.4 for the I/O pin descriptions; Section 11.4.2.4 for timing characteristics of this interface and Section 11.6.2 for a list of supported EEPROM devices.

### 1.2.7 Serial Flash Interface

The 82599 provides an external SPI serial interface to a Flash (or boot ROM) device. The 82599 supports serial Flash devices with up to 64 Mb (8 MB) of memory. The size of the Flash used by the 82599 can be configured by the EEPROM. See Section 2.1.5 for full pin description and Section 11.4.2.3 for interface timing characteristics.

**Note:** Though the 82599 supports devices with up to 8 MB of memory, bigger devices can also be used. Accesses to memory beyond the Flash device size results in access wrapping as only the lower address bits are used by the Flash control unit.

### 1.2.8 SMBus Interface

SMBus is an optional interface for pass-through and/or configuration traffic between an external MC and the 82599.

The 82599's SMBus interface supports standard SMBus, up to a frequency of 400 KHz. Refer to Section 2.1.6 for full-pin descriptions and Section 11.4.2.2 for timing characteristics of this interface.

### 1.2.9 NC-SI Interface

NC-SI is an optional interface for pass-through traffic to and from a MC. The 82599 meets the NC-SI version 1.0.0a specification.

Refer to Section 3.3 for an additional description of the NC-SI interface, Section 2.1.8 for the pin descriptions, Section 10.5.1 for NC-SI programming and Section 11.4.1.4 for the timing characteristics.

### 1.2.10 MDIO Interfaces

The 82599 implements two serial management interfaces known as the Management Data Input/Output (MDIO) Interface that controls and manages PHY devices (master side). This interface provides the Media Access Controller (MAC) and software with the ability to monitor and control the state of the PHY. The 82599 supports the MDIO frame formats specified in both IEEE802.3 clause 22 and IEEE802.3 clause 45 using the electrical specification defined in IEEE802.3 clause 22 (LVTTL signaling). The MDIO interface can be controlled by software via a MDI single command and address register — MSCA (see Section 8.2.3.22.11 for more details).

Each MDIO interface should be connected to the relevant PHY as shown in the following example (each MDIO interface is driven by the appropriate MAC function).

Refer to Section 3.7.6 for complete description of the MDIO interface, Section 2.1.9 for the pin descriptions, the MSCA register in Section 8.2.3.22.11, and Section 11.4.2.7 for the timing characteristics.

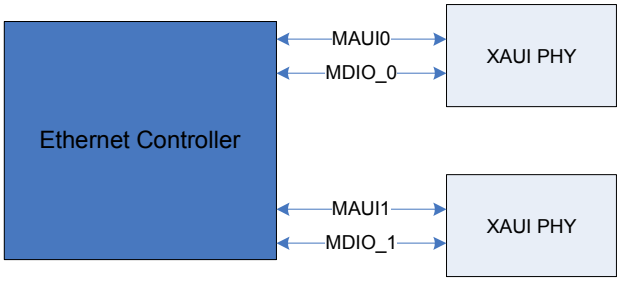



**Figure 1-4 MDIO Connection Example**

## 1.2.11 I²C Interfaces

The 82599 implements two serial management interfaces known as I²C Management Interfaces for the control and management of external optical modules (XFP and SFP+). This interface provides the MAC and software with the ability to monitor and control the state of the optical module. The use, direction, and values of the I²C pins are controlled and accessed using fields in the I2C Control (I2CCTL) register.

Each I²C interface should be connected to the relevant PHY as shown in the following example (each I²C interface is driven by the appropriate MAC function).

Refer to Section 2.1.7 for the pin descriptions, I2CCTL register information in Section 8.2.3.1.5 for I²C programming, and Section 11.4.2.2 for timing characteristics.

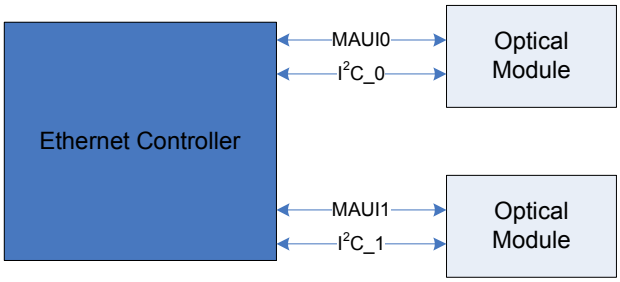



**Figure 1-5 I²C Connection Example**

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

The 82599 has eight SDP pins per port that can be used for miscellaneous hardware or software-controllable purposes. These pins can each be individually configured to act as either input or output pins. Via the SDP pins, the 82599 can support IEEE1588 auxiliary device connections, control of the low speed optical module interface, and other functionality. For more details on the SDPs see Section 3.6 and the ESDP register information in Section 8.2.3.1.4.

### 1.2.13 LED Interface

The 82599 implements four output drivers intended for driving external LED circuits per port. Each of the four LED outputs can be individually configured to select the particular event, state, or activity, which is indicated on that output. In addition, each LED can be individually configured for output polarity as well as for blinking versus non-blinking (steady-state) indications.

The configuration for LED outputs is specified via the LEDCTL register (see Section 8.2.3.1.6). In addition, the hardware-default configuration for all LED outputs can be specified via an EEPROM field (see Section 6.3.7.3), thereby supporting LED displays configured to a particular OEM preference.

See Section 2.1.11 for a full pin description.

## 1.3 Features Summary

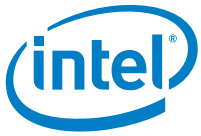
Table 1-1 to Table 1-7 list the 82599's features in comparison to previous dual-port 1 Gb/s and 10 Gb/s Ethernet controllers.

**Table 1-1 General Features**

| Feature | 82599 | 82598 | Reserved |
|---|---|---|---|
| Serial Flash Interface | Y | Y | |
| 4-wire SPI EEPROM Interface | Y | Y | |
| Configurable LED Operation for Software or OEM Customization of LED Displays | Y | Y | |
| Protected EEPROM Space for Private Configuration | Y | Y | |
| Device Disable Capability | Y | Y | |
| Package Size | 25 x 25 mm | 31 x 31 mm | |
| Watchdog Timer | Y | N | |
| Time Sync (IEEE 1588) | Y | N | |

**Table 1-2 Network Features**

| Feature | 82599 | 82598 | Reserved |
|---|---|---|---|
| Compliant with the 10 Gb/s and 1 Gb/s Ethernet/802.3ap (KX/KX4) Specification | Y | Y | |
| Compliant with the 10 Gb/s 802.3ap (KR) specification | Y | N | |
| Support of 10GBASE-KR FEC | Y | N | |



## Table 1-2 Network Features (Continued)

| Feature | 82599 | 82598 | Reserved |
|---|---|---|---|
| Compliant with the 10 Gb/s Ethernet/802.3ae (XAUI) Specification | Y | Y | |
| Compliant with SFI interface | Y | N | |
| Support for EDC | N | N | |
| Compliant with the 1000BASE-BX Specification | Y | Y | |
| Half-duplex at 10/100 Mb/s Operation and Full-Duplex Operation at all Supported Speeds | Y (100 Mb FDX) | NA | |
| 10/100/1000 Copper PHY Integrated On-chip | N | N | |
| Support Jumbo Frames of up to 15.5 KB (15872 bytes) | Y [1] | Y | |
| Auto-Negotiation Clause 73 for Supported Modes | Y | Y | |
| Flow Control Support: Send/Receive Pause Frames and Receive FIFO Thresholds | Y | Y | |
| Statistics for Management and RMON | Y | Y | |
| 802.1q VLAN Support | Y | Y | |
| SerDes Interface for External PHY Connection or System Interconnect | Y | Y | |
| SGMII Interface | Y (100 M/1G only) | N | |
| SerDes Support of non Auto-Negotiation Partner | Y | Y | |
| Double VLAN | Y | N | |

1. The 82599 supports full-size 15.5 KB (15872-byte) jumbo packets while in a basic mode of operation. When DCB mode is enabled, or security engines enabled or virtualization is enabled, the 82599 supports 9.5 KB (9728-byte) jumbo packets.

## Table 1-3 Host Interface Features

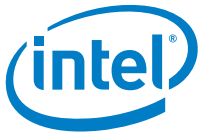
| Feature | 82599 | 82598 | Reserved |
|---|---|---|---|
| PCIe* Host Interface | PCIe V2.0 (2.5GT/s or 5GT/s) | PCIe v2.0 (2.5GT/s) | |
| Number of Lanes | x1, x2, x4, x8 | x1, x2, x4, x8 | |
| 64-bit Address Support for Systems Using More Than 4 GB of Physical Memory | Y | Y | |
| Outstanding Requests for Tx Data Buffers | 16 | 16 | |
| Outstanding Requests for Tx Descriptors | 8 | 8 | |
| Outstanding Requests for Rx Descriptors | 8 | 4 | |
| Credits for P-H/P-D/NP-H/NP-D (shared for the 2 ports) | 16/16/4/4 | 8/16/4/4 | |

## Table 1-3 Host Interface Features (Continued)

| Feature | 82599 | 82598 | Reserved |
|---|---|---|---|
| Max Payload Size Supported | 512 Bytes | 256 Bytes | |
| Max Request Size Supported | 2 KB | 256 Bytes | |
| Link Layer Retry Buffer Size (shared for the 2 ports) | 3.4 KB | 2 KB | |
| Vital Product Data (VPD) | Y | N | |
| End to End CRC (ECRC) | Y | N | |

## Table 1-4 LAN Functions Features

| Feature | 82599 | 82598 | Reserved |
|---|---|---|---|
| Programmable Host Memory Receive Buffers | Y | Y | |
| Descriptor Ring Management Hardware for Transmit and Receive | Y | Y | |
| ACPI Register Set and Power Down Functionality Supporting D0 & D3 States | Y | Y | |
| Integrated LinkSec security engines: AES-GCM 128-bit; Encryption + Authentication; One SC x 2 SA per port. Replay Protection with Zero Window | Y | N | |
| Integrated IPsec security engines: AES-GCM 128bit; AH or ESP encapsulation; IPv4 and IPv6 (no option or extended headers) | 1024 SA / port | N | |
| Software-Controlled Global Reset Bit (Resets Everything Except the Configuration Registers) | Y | Y | |
| Software-Definable Pins (SDP); (per port) | 8 | 8 | |
| Four SDP Pins can be Configured as General Purpose Interrupts | Y | Y | |
| Wake-on-LAN (WoL) | Y | Y | |
| IPv6 Wake-up Filters | Y | Y | |
| Configurable (through EEPROM) Wake-up Flexible Filters | Y | Y | |
| Default Configuration by EEPROM for all LEDs for Pre-Driver Functionality | Y | Y | |
| LAN Function Disable Capability | Y | Y | |
| Programmable Memory Transmit Buffers | 160 KB / port | 320 KB / port | |
| Programmable Memory Receive Buffers | 512 KB / port | 512 KB / port | |



## Table 1-5 LAN Performance Features[1]

| Feature | 82599 | 82598 | Reserved |
|---|---|---|---|
| TCP/UDP Segmentation Offload | 256 KB in all modes | 256 KB in legacy mode, 32 KB in DCB | |
| TSO Interleaving for Reduced Latency | Y | N | |
| TCP Receive Side Coalescing (RSC) | 32 flows / port | N | |
| Data Center Bridging (DCB), IEEE Compliance to:<br>Priority Groups (up to 8) and Bandwidth Allocation (ETS) IEEE802.1Qaz<br>Priority-based Flow Control (PFC) IEEE802.1Qbb | <br>Y<br>Y | <br>Y<br>Y | |
| Transmit Rate Scheduler | Y | N | |
| IPv6 Support for IP/TCP and IP/UDP Receive Checksum Offload | Y | Y | |
| Fragmented UDP Checksum Offload for Packet Reassembly | Y | Y | |
| FCoE Tx / Rx CRC Offload | Y | N | |
| FCoE Transmit Segmentation | 256 KB | N | |
| FCoE Coalescing and Direct Data Placement | 512 outstanding Read — Write requests / port | N | |
| Message Signaled Interrupts (MSI) | Y | Y | |
| Message Signaled Interrupts (MSI-X) | Y | Y | |
| Interrupt Throttling Control to Limit Maximum Interrupt Rate and Improve CPU Use | Y | Y | |
| Rx Packet Split Header | Y | Y | |
| Multiple Rx Queues (RSS) | Y (multiple modes) | 8x8<br>16x4 | |
| Flow Director Filters: up to 32 K - 2 Flows by Hash Filters or up to 8 K- 2 Perfect Match Filters | Y | N | |
| Number of Rx Queues (per port) | 128 | 64 | |
| Number of Tx Queues (per port) | 128 | 32 | |
| Low Latency Interrupts<br>DCA Support<br>TCP Timer Interrupts<br>Relax Ordering | Yes to all | Yes to all | |
| Rate Control of Low Latency Interrupts | Y | N | |

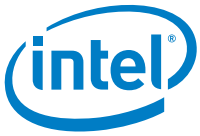
1. The 82599 10 GbE operation is focused on performance improvement; note that the 82599's 1GbE mode is optimized for power saving.

### Table 1-6 Virtualization Features

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

### Table 1-7 Manageability Features

| Feature | 82599 | 82598 | Reserved |
|---|---|---|---|
| Advanced Pass Through-Compatible Management Packet Transmit/Receive Support | Y | Y | |
| SMBus Interface to an External MC | Y | Y | |
| NC-SI Interface to an External MC | Y | Y | |
| New Management Protocol Standards Support (NC-SI) | Y | Y | |
| L2 Address Filters | 4 | 4 | |
| VLAN L2 Filters | 8 | 8 | |
| Flex L3 Port Filters | 16 | 16 | |
| Flexible TCO Filters | 4 | 4 | |
| L3 Address Filters (IPv4) | 4 | 4 | |
| L3 Address Filters (IPv6) | 4 | 4 | |



# 1.4 Overview of New Capabilities Beyond the 82598

## 1.4.1 Security

The 82599 supports the IEEE P802.1AE LinkSec specification. It incorporates an inline packet crypto unit to support both privacy and integrity checks on a packet by packet basis. The transmit data path includes both encryption and signing engines. On the receive data path, the 82599 includes both decryption and integrity checkers. The crypto engines use the AES GCM algorithm, which is designed to support the 802.1AE protocol. Note that both host traffic and Manageability Controller (MC) management traffic might be subject to authentication and/or encryption.

The 82599 supports IPsec offload for a given number of flows. It is the operating system's responsibility to submit (to hardware) the most loaded flows in order to take maximum benefits of the IPsec offload in terms of CPU utilization savings. Main features are:

- Offload IPsec for up to 1024 Security Associations (SA) for each of Tx and Rx
- AH and ESP protocols for authentication and encryption
- AES-128-GMAC and AES-128-GCM crypto engines
- Transport mode encapsulation
- IPv4 and IPv6 versions (no options or extension headers)

## 1.4.2 Transmit Rate Limiting

The 82599 supports Transmit Rate Scheduler (TRS) in addition to the Data Center Bridging (DCB) functionality provided in the 82598. TRS is enabled for each transmit queue. The following modes of TRS are used:

- Frame Overhead — IPG is extended by a fixed value for all transmit queues.
- Payload Rate — IPG, stretched relative to frame size, provides pre-determined data (bytes) rates for each transmit queue.

## 1.4.3 Fibre Channel over Ethernet (FCoE)

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

Existing FC Host Bus Adapters (HBAs) used to connect between FC initiator and FC targets provide full offload of the FC protocol to the initiator that enables maximizing storage performance. The 82599 offloads the main data path of I/O Read and Write commands to the storage target.

### 1.4.4 Performance

The 82599 improves on previous 10 GbE products in the following performance vectors:

- Throughput — The 82599 aims to provide wire speed dual-port 10 Gb/s throughput. This is accomplished using the PCIe physical layer (PCIe V2.0 (5GT/s), by tuning the internal pipeline to 10 Gb/s operation, and by enhancing the PCIe concurrency capabilities.
- Latency — The 82599 reduces end-to-end latency for high priority traffic in presence of other traffic. Specifically, the 82599 reduces the delay caused by preceding TCP Segmentation Offload (TSO) packets. Unlike previous products, a TSO packet might be interleaved with other packets going to the wire. Interleaving is done at the Ethernet packet boundary, therefore reducing the maximum delay due to a TSO from a TSO-worth of data to an MTU-worth of data.
- CPU utilization — The 82599 supports reduction in CPU utilization, mainly by supporting Receive Side Coalescing (RSC)
- Flow affinity filters

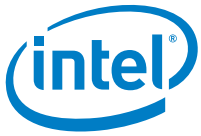
#### 1.4.4.1 Receive Side Coalescing (RSC)

RSC coalesces incoming TCP/IP packets into larger receive segments. It is the inverse operation to TSO on the transmit side. It has the same motivation, reducing CPU utilization by executing the TCP/IP stack only once for a set of received Ethernet packets. The 82599 can handle up to 32 flows per port at any given time. See Section 7.11 for more details on RSC.

#### 1.4.4.2 PCIe V2.0 (5GT/s)

Several changes are defined in the size of PCIe transactions to improve the performance in virtualization environments. Larger request sizes decrease the number of independent transactions on PCIe and therefore decreases trashing of the IOTLB cache. Changes include:

- Increase in the number of outstanding requests (data, descriptors) to a total of 32 requests
- Increase in the number of credits for posted transaction (such as for tail updates) to 16
- Increase in the maximum payload size supported from 256 bytes to 512 bytes
- Increase in the supported maximum read request size from 256 bytes to 2 KB. Note that the amount of outstanding request data does not change. That is, if the 82599 supports N outstanding requests of 256 bytes, then it would support N/2 requests of 512 bytes, etc.
- Retry buffer size — The link layer retry buffer size increases to 3.4 KB to meet the higher speed of PCIe V2.0 (5GT/s).



### 1.4.5 Rx/Tx Queues and Rx Filtering

The 82599 Tx and Rx queues have increased in size to 128 Tx queues and 128 Rx queues. Additional filtering capabilities are provided based on:

- L2 Ethertype
- 5-tuples
- SYN identification
- Flow Director — a large number of flow affinity filters that direct receive packets by their flows to queues for classification, load balancing, and matching between flows and CPU cores.

See Section 7.0 for a complete description.

### 1.4.6 Interrupts

Several changes in the interrupt scheme are available in the 82599:

- Control over the rate of Low Latency Interrupts (LLI)
- Extensions to the filters that invoke LLIs
- Additional MSI-X vectors for the five-tuple filters and for IOV virtualization

See Section 7.3 for more details.

### 1.4.7 Virtualization

See Section 7.10 for more details.

#### 1.4.7.1 PCI- IOV

The 82599 supports the PCI-SIG single-root I/O Virtualization initiative (SR-IOV), including the following functionality:

- Replication of PCI configuration space
- Allocation of BAR space per virtual function
- Allocation of requester ID per virtual function
- Virtualization of interrupts

The 82599 provides the infrastructure for direct assignment architectures through a mailbox mechanism. Virtual Functions (VFs) might communicate with the Physical Function (PF) through the mailbox and the PF can allocate shared resources through the mailbox channel.

#### 1.4.7.2 Packet Filtering and Replication

The 82599 adds extensive coverage for packet filtering for virtualization by supporting the following filtering modes:

- Filtering by unicast Ethernet MAC address
- Filtering by VLAN tag
- Filtering of multicast Ethernet MAC address
- Filtering of broadcast packets

For each of the above categories, the 82599 can replicate packets to multiple Virtual Machines (VMs). Various mirroring modes are supported, including mirroring a VM, a Virtual LAN (VLAN), or all traffic into a specific VM.

#### 1.4.7.3 Packet Switching

The 82599 forwards transmit packets from a transmit queue to an Rx software queue to support VM-VM communication. Transmit packets are filtered to an Rx queue based on the same criteria as packets received from the wire.

#### 1.4.7.4 Traffic Shaping

Transmit bandwidth is allocated among the virtual interfaces to avoid unfair use of bandwidth by a single VM. Allocation is done separately per DCB traffic class so that bandwidth assignment to each traffic class is partitioned among the different VMs.

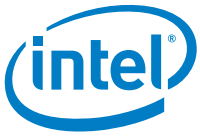
### 1.4.8 VPD

The 82599 supports VPD capability defined in the PCI Specification, version 3.0. See Section 3.4.9 for more details.

### 1.4.9 Double VLAN

The 82599 supports a mode where all received and sent packets have at least one VLAN tag in addition to the regular tagging that can optionally be added. This mode is used for systems where the switches add an additional tag containing switching information.

When a port is configured to double VLAN, the 82599 assumes that all packets received or sent to this port have at least one VLAN. The only exception to this rule is flow control packets, which don't have a VLAN tag. See Section 7.4.5 for more details.



### 1.4.10 Time Sync — IEEE 1588 — Precision Time Protocol (PTP)

The IEEE 1588 International Standard lets networked Ethernet equipment synchronize internal clocks according to a network master clock. The protocol is implemented mostly in software, with the 82599 providing accurate time measurements of special Tx and Rx packets close to the Ethernet link. These packets measure the latency between the master clock and an end-point clock in both link directions. The endpoint can then acquire an accurate estimate of the master time by compensating for link latency. See Section 7.9 for more details.

The 82599 provides the following support for the IEEE 1588 protocol:

- Detecting specific PTP Rx packets and capturing the time of arrival of such packets in dedicated CSRs
- Detecting specific PTP Tx packets and capturing the time of transmission of such packets in dedicated CSRs
- A software-visible reference clock for the above time captures

## 1.5 Conventions

### 1.5.1 Terminology and Acronyms

See Section 15.0 for a list of terminology and acronyms used throughout this document.

### 1.5.2 Byte Count

When referencing jumbo packet size, 1 KB equals 1024 bytes.

For example:

- 9.5 KB equals 9.5 x 1024 = 9728 bytes
- 15.5 KB equals 15.5 x 1024 = 15872 bytes

### 1.5.3 Byte Ordering

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

- Any register defined in big endian notation can be transferred as is to/from Tx and Rx buffers in the host memory. Big endian notation is also referred to as being in network order or ordering.

- Any register defined in little endian notation must be swapped before it is transferred to/from Tx and Rx buffers in the host memory. Registers in little endian order are referred to being in host order or ordering.

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

**Note:** Registers not transferred on the wire are defined in little endian notation. Registers transferred on the wire are defined in big endian notation, unless specified differently.

## 1.6 Register/Bit Notations

This document refers to device register names with all capital letters. To refer to a specific bit in a register the convention REGISTER.BIT is used. For example CTRL.GIO Master Disable refers to the GIO Master Disable bit in the Device Control (CTRL) register.

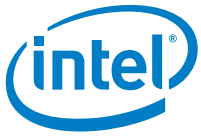
This document also refers to bit names as initial capital letters in an italic font. For example, *GIO Master Disable*.

## 1.7 References

The 82599 implements features from the following specifications:

IEEE Specifications

- IEEE standard 802.3-2005 (Ethernet). Incorporates various IEEE Standards previously published separately. Institute of Electrical and Electronic Engineers (IEEE).
- 10GBASE-X — An IEEE 802.3 physical coding sublayer for 10 Gb/s operation over XAUI and four lane PMDs as per IEEE 802.3 Clause 48.
- 1000BASE-CX — 1000BASE-CX over specially shielded 150 Ω balanced copper jumper cable assemblies as specified in IEEE 802.3 Clause 39.
- 10GBASE-LX4 — IEEE 802.3 Physical Layer specification for 10 Gb/s using 10GBASE-X encoding over four WWDM lanes over multimode fiber as specified in IEEE 802.3 Clause 54.
- 10GBASE-CX4 — IEEE 802.3 Physical Layer specification for 10 Gb/s using 10GBASE-X encoding over four lanes of 100 Ω shielded balanced copper cabling as specified in IEEE 802.3 Clause 54.
- 1000BASE-KX — IEEE 802.3ap Physical Layer specification for 1 Gb/s using 1000BASE-X encoding over an electrical backplane as specified in IEEE 802.3 Clause 70.
- 10GBASE-KX4 — IEEE 802.3ap Physical Layer specification for 10 Gb/s using 10GBASE-X encoding over an electrical backplane as specified in IEEE 802.3 Clause 71.



- 10GBASE-KR — IEEE 802.3ap Physical Layer specification for 10 Gb/s using 10GBASE-R encoding over an electrical backplane as specified in IEEE 802.3 Clause 72.
- 1000BASE-BX — 1000BASE-BX is the PICMG 3.1 electrical specification for transmission of 1 Gb/s Ethernet or 1 Gb/s Fibre Channel encoded data over the backplane.
- 10GBASE-BX4 — 10GBASE-BX4 is the PICMG 3.1 electrical specification for transmission of 10 Gb/s Ethernet or 10 Gb/s Fibre Channel encoded data over the backplane.
- IEEE standard 802.3ap, draft D3.2.
- IEEE standard 1149.1, 2001 Edition (JTAG). Institute of Electrical and Electronics Engineers (IEEE).
- IEEE standard 802.1Q for VLAN.
- IEEE 1588 International Standard, Precision clock synchronization protocol for networked measurement and control systems, 2004-09.
- IEEE P802.1AE/D5.1, Media Access Control (MAC) Security, January 19, 2006.

PCI-SIG Specifications

- PCI Express 2.0 Base specification, 12/20/2006.
- PCI Express™ 2.0 Card Electromechanical Specification, Revision 0.9, January 19, 2007.
- PCI Bus Power Management Interface Specification, Rev. 1.2, March 2004.
- PICMG3.1 Ethernet/Fibre Channel Over PICMG 3.0 Draft Specification January 14, 2003 Version D1.0.
- Single Root I/O Virtualization and Sharing, Revision 0.7, 1/11/2007.

IETF Specifications

- IPv4 specification (RFC 791)
- IPv6 specification (RFC 2460)
- TCP specification (RFC 793)
- UDP specification (RFC 768)
- ARP specification (RFC 826)
- RFC4106 — The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP).
- RFC4302 — IP Authentication Header (AH)
- RFC4303 — IP Encapsulating Security Payload (ESP)
- RFC4543 — The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH.
- IETF Internet Draft, Marker PDU Aligned Framing for TCP Specification.
- IETF Internet Draft, Direct Data Placement over Reliable Transports.

**Other**

- Serial-GMII Specification, Cisco Systems document ENG-46158, Revision 1.7.
- Advanced Configuration and Power Interface Specification, Rev 2.0b, October 2002
- Network Controller Sideband Interface (NC-SI) Specification, Version cPubs-0.1, 2/18/2007.
- System Management Bus (SMBus) Specification, SBS Implementers Forum, Ver. 2.0, August 2000.
- EUI-64 specification, http://standards.ieee.org/regauth/oui/tutorials/EUI64.html.
- Backward Congestion Notification Functional Specification, 11/28/2006.
- Definition for new PAUSE function, Rev. 1.2, 12/26/2006.
- GCM spec — McGrew, D. and J. Viega, "The Galois/Counter Mode of Operation (GCM)", Submission to NIST. http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec.pdf, January 2004.
- FRAMING AND SIGNALING-2 (FC-FS-2) Rev 1.00
- Fibre Channel over Ethernet Draft Presented at the T11 on May 2007
- Per Priority Flow Control (by Cisco* Systems) — Definition for new PAUSE function, Rev 1.2, EDCS-472530

In addition, the following document provides application information:

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

# 1.8 Architecture and Basic Operation

## 1.8.1 Transmit (Tx) Data Flow

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

**Table 1-8 Tx Data Flow**

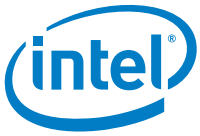
| Step | Description |
|---|---|
| 1 | The host creates a descriptor ring and configures one of the 82599's transmit queues with the address location, length, head, and tail pointers of the ring (one of 128 available Tx queues). |
| 2 | The host is requested by the TCP/IP stack to transmit a packet, it gets the packet data within one or more data buffers. |
| 3 | The host initializes the descriptor(s) that point to the data buffer(s) and have additional control parameters that describes the needed hardware functionality. The host places that descriptor in the correct location at the appropriate Tx ring. |
| 4 | The host updates the appropriate Queue Tail Pointer (TDT) |
| 5 | The 82599's DMA senses a change of a specific TDT and as a result sends a PCIe request to fetch the descriptor(s) from host memory. |
| 6 | The descriptor(s) content is received in a PCIe read completion and is written to the appropriate location in the descriptor queue. |
| 7 | The DMA fetches the next descriptor and processes its content. As a result, the DMA sends PCIe requests to fetch the packet data from system memory. |
| 8 | The packet data is being received from PCIe completions and passes through the transmit DMA that performs all programmed data manipulations (various CPU offloading tasks as checksum offload, TSO offload, etc.) on the packet data on the fly. |
| 9 | While the packet is passing through the DMA, it is stored into the transmit FIFO.<br>After the entire packet is stored in the transmit FIFO, it is then forwarded to transmit switch module. |
| 10 | The transmit switch arbitrates between host and management packets and eventually forwards the packet to the MAC. |
| 11 | The MAC appends the L2 CRC to the packet and sends the packet over the wire using a pre-configured interface. |
| 12 | When all the PCIe completions for a given packet are complete, the DMA updates the appropriate descriptor(s). |
| 13 | The descriptors are written back to host memory using PCIe posted writes. The head pointer is updated in host memory as well. |
| 14 | An interrupt is generated to notify the host driver that the specific packet has been read to the 82599 and the driver can then release the buffer(s). |

## 1.8.2 Receive (Rx) Data Flow

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

**Table 1-9 Rx Data Flow**

| Step | Description |
|---|---|
| 1 | The host creates a descriptor ring and configures one of the 82599's receive queues with the address location, length, head, and tail pointers of the ring (one of 128 available Rx queues) |
| 2 | The host initializes descriptor(s) that point to empty data buffer(s). The host places these descriptor(s) in the correct location at the appropriate Rx ring. |
| 3 | The host updates the appropriate Queue Tail Pointer (RDT). |
| 6 | A packet enters the Rx MAC. |
| 7 | The MAC forwards the packet to the Rx filter. |
| 8 | If the packet matches the pre-programmed criteria of the Rx filtering, it is forwarded to an Rx FIFO. |
| 9 | The receive DMA fetches the next descriptor from the appropriate host memory ring to be used for the next received packet. |
| 10 | After the entire packet is placed into an Rx FIFO, the receive DMA posts the packet data to the location indicated by the descriptor through the PCIe interface.<br>If the packet size is greater than the buffer size, more descriptors are fetched and their buffers are used for the received packet. |
| 11 | When the packet is placed into host memory, the receive DMA updates all the descriptor(s) that were used by the packet data. |
| 12 | The receive DMA writes back the descriptor content along with status bits that indicate the packet information including what offloads were done on that packet. |
| 13 | The 82599 initiates an interrupt to the host to indicate that a new received packet is ready in host memory. |
| 14 | The host reads the packet data and sends it to the TCP/IP stack for further processing. The host releases the associated buffer(s) and descriptor(s) once they are no longer in use. |



# 2.0 Pin Interface

## 2.1 Pin Assignment

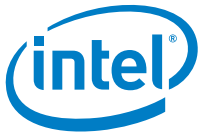
### 2.1.1 Signal Type Definition

| Signal | Definition | DC Specification |
|---|---|---|
| In | Input is a standard input-only signal. | Section 11.4.1.2 |
| Out (O) | Totem Pole Output (TPO) is a standard active driver. | Section 11.4.1.2 |
| T/s | Tri-state is a bi-directional, tri-state input/output pin. | Section 11.4.1.2 |
| O/d | Open drain enables multiple devices to share as a wire-OR. | Section 11.4.1.3 |
| A-in | Analog input signals. | Section 11.4.3 and Section 11.4.4 |
| A-out | Analog output signals. | Section 11.4.3 and Section 11.4.4 |
| B | Input BIAS. | - |
| CML-in | CML input signal. | Section 11.4.5 |
| NCSI-in | NC-SI input signal. | Section 11.4.1.4 |
| NCSI-out | NC-SI output signal. | Section 11.4.1.4 |
| Pu | Internal pull-up. | - |
| Pd | Internal pull-down. | - |

## 2.1.2 PCIe Symbols and Pin Names

See AC/DC specifications in Section 11.4.3.

| Reserved | Pin Name | Ball # | Type | Name and Function |
|---|---|---|---|---|
| | PE_CLK_p<br>PE_CLK_n | AB23<br>AB24 | A-in | PCIe Differential Reference Clock In. A 100 MHz differential clock input. This clock is used as the reference clock for the PCIe Tx/Rx circuitry and by the PCIe core PLL to generate clocks for the PCIe core logic. |
| | PET_0_p<br>PET_0_n | Y23<br>Y24 | A-out | PCIe Serial Data Output. A serial differential output pair running at 5 Gb/s or 2.5 Gb/s. This output carries both data and an embedded 5 GHz or 2.5 GHz clock that is recovered along with data at the receiving end. |
| | PET_1_p<br>PET_1_n | V23<br>V24 | A-out | PCIe Serial Data Output. A serial differential output pair running at 5 Gb/s or 2.5 Gb/s. This output carries both data and an embedded 5 GHz or 2.5 GHz clock that is recovered along with data at the receiving end. |
| | PET_2_p<br>PET_2_n | T23<br>T24 | A-out | PCIe Serial Data Output. A serial differential output pair running at 5 Gb/s or 2.5 Gb/s. This output carries both data and an embedded 5 GHz or 2.5 GHz clock that is recovered along with data at the receiving end. |
| | PET_3_p<br>PET_3_n | P23<br>P24 | A-out | PCIe Serial Data Output. A serial differential output pair running at 5 Gb/s or 2.5 Gb/s. This output carries both data and an embedded 5 GHz or 2.5 GHz clock that is recovered along with data at the receiving end. |
| | PET_4_p<br>PET_4_n | J23<br>J24 | A-out | PCIe Serial Data Output. A serial differential output pair running at 5 Gb/s or 2.5 Gb/s. This output carries both data and an embedded 5 GHz or 2.5 GHz clock that is recovered along with data at the receiving end. |
| | PET_5_p<br>PET_5_n | G23<br>G24 | A-out | PCIe Serial Data Output. A serial differential output pair running at 5 Gb/s or 2.5 Gb/s. This output carries both data and an embedded 5 GHz or 2.5 GHz clock that is recovered along with data at the receiving end. |
| | PET_6_p<br>PET_6_n | E23<br>E24 | A-out | PCIe Serial Data Output. A serial differential output pair running at 5 Gb/s or 2.5 Gb/s. This output carries both data and an embedded 5 GHz or 2.5 GHz clock that is recovered along with data at the receiving end. |
| | PET_7_p<br>PET_7_n | C23<br>C24 | A-out | PCIe Serial Data Output. A serial differential output pair running at 5 Gb/s or 2.5 Gb/s. This output carries both data and an embedded 5 GHz or 2.5 GHz clock that is recovered along with data at the receiving end. |
| | PER_0_p<br>PER_0_n | AC20<br>AC21 | A-in | PCIe Serial Data Input. A serial differential input pair running at 5 Gb/s or 2.5 Gb/s. An embedded clock present in this input is recovered along with the data. |
| | PER_1_p<br>PER_1_n | AA20<br>AA21 | A-in | PCIe Serial Data Input. A serial differential input pair running at 5 Gb/s or 2.5 Gb/s. An embedded clock present in this input is recovered along with the data. |
| | PER_2_p<br>PER_2_n | U20<br>U21 | A-in | PCIe Serial Data Input. A serial differential input pair running at 5 Gb/s or 2.5 Gb/s. An embedded clock present in this input is recovered along with the data. |



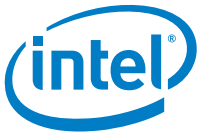
| Reserved | Pin Name | Ball # | Type | Name and Function |
|---|---|---|---|---|
| | PER_3_p<br>PER_3_n | R20<br>R21 | A-in | PCIe Serial Data Input. A serial differential input pair running at 5 Gb/s or 2.5 Gb/s. An embedded clock present in this input is recovered along with the data. |
| | PER_4_p<br>PER_4_n | K20<br>K21 | A-in | PCIe Serial Data Input. A serial differential input pair running at 5 Gb/s or 2.5 Gb/s. An embedded clock present in this input is recovered along with the data. |
| | PER_5_p<br>PER_5_n | H20<br>H21 | A-in | PCIe Serial Data Input. A serial differential input pair running at 5 Gb/s or 2.5 Gb/s. An embedded clock present in this input is recovered along with the data. |
| | PER_6_p<br>PER_6_n | D20<br>D21 | A-in | PCIe Serial Data Input. A serial differential input pair running at 5 Gb/s or 2.5 Gb/s. An embedded clock present in this input is recovered along with the data. |
| | PER_7_p<br>PER_7_n | B20<br>B21 | A-in | PCIe Serial Data Input. A serial differential input pair running at 5 Gb/s or 2.5 Gb/s. An embedded clock present in this input is recovered along with the data. |
| | PE_WAKE_N | AA18 | O/d | Wake. Pulled to 0b to indicate that a Power Management Event (PME) is pending and the PCIe link should be restored. Defined in the PCIe specifications. |
| | PE_RST_N | AD18 | In | Power and Clock Good Indication. Indicates that power and PCIe reference clock are within specified values. Defined in the PCIe specifications; also called: PCIe Reset and PERST. |
| | PE_RBIAS<br>PE_RSENSE | M24<br>N24 | B | PCIe BIAS.<br>A 24.9 Ω ±0.5%, 50 ppm resistor should be connected from PE_RBIAS to the chip's 1.2V Analog PCIe supply rail (VCC1P2_PE). Connection should be as close as possible to the chip. Resistor is used for internal impedance compensation and BIAS current generation circuitry.<br>PE_RSENSE is used as sensing node and should be shorted on board to PE_RBIAS as close as possible to the external resistor's pad. |

## 2.1.3 MAUI

See AC/DC specifications in Section 11.4.4 and Section 11.4.5.

| Reserved | Pin Name | Ball # | Type | Name and Function |
|---|---|---|---|---|
| | XA_RBIAS_p<br>XA_RBIAS_n | L2<br>L1 | B | MAUI BIAS.<br>A 1 KΩ ±0.5%, 50 ppm resistor should be connected between XA_RBIAS_p and XA_RBIAS_n and located close to the chip.<br>Resistor generates internal BIAS currents used for impedance compensation. XA_RBIAS_n is internally connected to ground. |
| | REFCLKIN_p<br>REFCLKIN_n | P2<br>P1 | CML-in | External Reference Clock Input/Crystal Oscillator Input. If an external clock is applied, it must be 25 MHz ±0.01%. |
| | RX0_L3_p<br>RX0_L3_n | B4<br>A4 | A-in | XAUI Serial Data Input for Port 0. A serial differential input pair running at up to 3.125 Gb/s. An embedded clock present in this input is recovered along with the data. |

| Reserved | Pin Name | Ball # | Type | Name and Function |
|---|---|---|---|---|
| | RX0_L2_p<br>RX0_L2_n | D4<br>D5 | A-in | XAUI Serial Data Input for Port 0. A serial differential input pair running at up to 3.125 Gb/s. An embedded clock present in this input is recovered along with the data. |
| | RX0_L1_p<br>RX0_L1_n | F4<br>F5 | A-in | XAUI Serial Data Input for Port 0. A serial differential input pair running at up to 3.125 Gb/s. An embedded clock present in this input is recovered along with the data. |
| | RX0_L0_p<br>RX0_L0_n | H4<br>H5 | A-in | XAUI Serial Data Input for Port 0. A serial differential input pair running at up to 3.125 Gb/s. An embedded clock present in this input is recovered along with the data.<br>This lane is also used in BX, BX4, CX4, KX, KR, and SFI modes. |
| | TX0_L3_p<br>TX0_L3_n | C1<br>C2 | A-out | XAUI Serial Data Output for Port 0. A serial differential output pair running at up to 3.125 Gb/s. This output carries both data and an embedded clock that is recovered along with data at the receiving end. |
| | TX0_L2_p<br>TX0_L2_n | E1<br>E2 | A-out | XAUI Serial Data Output for Port 0. A serial differential output pair running at up to 3.125 Gb/s. This output carries both data and an embedded clock that is recovered along with data at the receiving end. |
| | TX0_L1_p<br>TX0_L1_n | G1<br>G2 | A-out | XAUI Serial Data Output for Port 0. A serial differential output pair running at up to 3.125 Gb/s. This output carries both data and an embedded clock that is recovered along with data at the receiving end. |
| | TX0_L0_p<br>TX0_L0_n | J1<br>J2 | A-out | XAUI Serial Data Output for Port 0. A serial differential output pair running at up to 3.125 Gb/s. This output carries both data and an embedded clock that is recovered along with data at the receiving end.<br>This lane is also used in BX, BX4, CX4, KX, KR, and SFI modes. |
| | RX1_L3_p<br>RX1_L3_n | U4<br>U5 | A-in | XAUI Serial Data Input for Port 1. A serial differential input pair running at up to 3.125 Gb/s. An embedded clock present in this input is recovered along with the data. |
| | RX1_L2_p<br>RX1_L2_n | W4<br>W5 | A-in | XAUI Serial Data Input for Port 1. A serial differential input pair running at up to 3.125 Gb/s. An embedded clock present in this input is recovered along with the data. |
| | RX1_L1_p<br>RX1_L1_n | AA4<br>AA5 | A-in | XAUI Serial Data Input for Port 1. A serial differential input pair running at up to 3.125 Gb/s. An embedded clock present in this input is recovered along with the data. |
| | RX1_L0_p<br>RX1_L0_n | AC4<br>AD4 | A-in | XAUI Serial Data Input for Port 1. A serial differential input pair running at up to 3.125 Gb/s. An embedded clock present in this input is recovered along with the data.<br>This lane is also used in BX, BX4, CX4, KX, KR, and SFI modes. |
| | TX1_L3_p<br>TX1_L3_n | T1<br>T2 | A-out | XAUI Serial Data Output for Port 1. A serial differential output pair running at up to 3.125 Gb/s. This output carries both data and an embedded clock that is recovered along with data at the receiving end. |
| | TX1_L2_p<br>TX1_L2_n | V1<br>V2 | A-out | XAUI Serial Data Output for Port 1. A serial differential output pair running at up to 3.125 Gb/s. This output carries both data and an embedded clock that is recovered along with data at the receiving end. |



| Reserved | Pin Name | Ball # | Type | Name and Function |
|---|---|---|---|---|
| | TX1_L1_p<br>TX1_L1_n | Y1<br>Y2 | A-out | XAUI Serial Data Output for Port 1. A serial differential output pair running at up to 3.125 Gb/s. This output carries both data and an embedded clock that is recovered along with data at the receiving end. |
| | TX1_L0_p<br>TX1_L0_n | AB1<br>AB2 | A-out | XAUI Serial Data Output for Port 1. A serial differential output pair running at up to 3.125 Gb/s. This output carries both data and an embedded clock that is recovered along with data at the receiving end.<br>This lane is also used in BX, BX4, CX4, KX, KR, and SFI modes. |

## 2.1.4 EEPROM

See AC specifications in Section 11.4.2.4.

| Reserved | Pin Name | Ball # | Type | Name and Function |
|---|---|---|---|---|
| | EE_DI | B18 | O | Data output to EEPROM. |
| | EE_DO | A18 | In<br>Pu | Data input from EEPROM. |
| | EE_SK | B19 | O | EEPROM serial clock operates at maximum of 2 MHz. |
| | EE_CS_N | C19 | O | EEPROM chip select output. |

## 2.1.5 Serial Flash

See AC specifications in Section 11.4.2.3.

| Reserved | Pin Name | Ball # | Type | Name and Function |
|---|---|---|---|---|
| | FLSH_SI | B6 | T/s | Serial data output to the Flash. |
| | FLSH_SO | A7 | In<br>Pu | Serial data input from the Flash. |
| | FLSH_SCK | A8 | T/s | Flash serial clock operates at 12.5 MHz. |
| | FLSH_CE_N | B7 | T/s | Flash chip select output. |

## 2.1.6 SMBus

See the AC specifications in Section 11.4.2.2.

| Reserved | Pin Name | Ball # | Type | Name and Function |
|---|---|---|---|---|
| | SMBCLK | AC19 | o/d | SMBus Clock. One clock pulse is generated for each data bit transferred. |
| | SMBD | AB19 | o/d | SMBus Data. Stable during the high period of the clock (unless it is a start or stop condition). |
| | SMBALRT_N | AA19 | o/d | SMBus Alert. Acts as an interrupt pin of a slave device on the SMBus. |

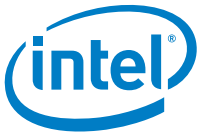
**Note:** If the SMBus is disconnected, an external pull-up should be used for the SMBCLK, SMBD pins.

## 2.1.7 $I^2C$

See the $I^2C$ specification and Section 11.4.2.2 for AC specifications.

| Reserved | Pin Name | Ball # | Type | Name and Function |
|---|---|---|---|---|
| | SCL0 | AB12 | o/d | $I^2C$ Clock. One clock pulse is generated for each data bit transferred. |
| | SDA0 | AA12 | o/d | $I^2C$ Data. Stable during the high period of the clock (unless it is a start or stop condition). |
| | SCL1 | AD17 | o/d | $I^2C$ Clock. One clock pulse is generated for each data bit transferred. |
| | SDA1 | AC18 | o/d | $I^2C$ Data. Stable during the high period of the clock (unless it is a start or stop condition). |

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



## 2.1.8 NC-SI

See AC specifications in Section 11.4.2.5.

| Reserved | Pin Name | Ball # | Type | Name and Function |
|---|---|---|---|---|
| | NCSI_CLK_IN | AC11 | NCSI-In | NC-SI Reference Clock Input. Synchronous clock reference for receive, transmit, and control interface. It is a 50 MHz clock ± 50 ppm. |
| | NCSI_CRS_DV | AB11 | NCSI-Out | Carrier Sense/Receive Data Valid (CRS/DV). |
| | NCSI_RXD_0<br>NCSI_RXD_1 | AA11<br>AC10 | NCSI-Out | Receive Data. Data signals to the BMC. |
| | NCSI_TX_EN | AB10 | NCSI-In | Transmit Enable. |
| | NCSI_TXD_0<br>NCSI_TXD_1 | AA10<br>AD11 | NCSI-In | Transmit Data. Data signals from the BMC. |

**Note:** If NC-SI is disconnected: an external pull-down should be used for the NCSI_CLK_IN and NCSI_TX_EN pins; a pull-up (10 kΩ) should be used for NCSI_TXD[1:0].

## 2.1.9 MDIO

See AC specifications in Section 11.4.2.7.

| Reserved | Pin Name | Ball # | Type | Name and Function |
|---|---|---|---|---|
| | MDIO0 | AD12 | T/s | Management Data. Bi-directional signal for serial data transfers between the 82599 and the PHY management registers for port 0.<br>*Note:* Requires an external pull-up device. |
| | MDC0 | AC12 | O | Management Clock. Clock output for accessing the PHY management registers for port 0. MDC clock frequency is Proportional to link speed. At 10 Gb/s Link speed MDC frequency can be set to 2.4 MHz (default) or 24 MHz. |
| | MDIO1 | AC17 | T/s | Management Data. Bi-directional signal for serial data transfers between the 82599 and the PHY management registers for port 1.<br>*Note:* Requires an external pull-up device. |
| | MDC1 | AB18 | O | Management Clock. Clock output for accessing the PHY management registers for port 1. MDC clock frequency is Proportional to link speed. At 10 Gb/s Link speed MDC frequency can be set to 2.4 MHz (default) or 24 MHz. |

## 2.1.10 Software Defined Pins (SDPs)

See AC specifications in Section 11.4.2.1.

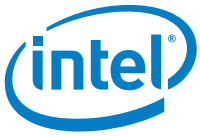
See Section 3.6 for more details on configurable SDPs.

| Reserved | Pin Name | Ball # | Type | Name and Function |
|---|---|---|---|---|
| | SDP0_0<br>SDP0_1<br>SDP0_2<br>SDP0_3<br>SDP0_4<br>SDP0_5<br>SDP0_6<br>SDP0_7 | AD8<br>AC8<br>AB8<br>AA8<br>AD7<br>AC7<br>AB7<br>AA7 | T/s<br>Pu | General Purpose SDPs. 3.3V I/Os for function 0. Can be used to support IEEE1588 Auxiliary devices, Low speed optical module interface<br>SDP0_4 is dedicated input pin for Security enablement. Security offload on both ports is enabled if the Security Enablement flags in the SKU Fuses register are set to 1b and SDP0_4 input pin is driven high.<br>See Section 3.6 for possible usages of the pins. |
| | SDP1_0<br>SDP1_1<br>SDP1_2<br>SDP1_3<br>SDP1_4<br>SDP1_5<br>SDP1_6<br>SDP1_7 | AC16<br>AB16<br>AB17<br>AA17<br>AA16<br>AC15<br>AB15<br>AA15 | T/s<br>Pu | General purpose SDPs. 3.3V I/Os for function 1. Can be used to support IEEE1588 auxiliary devices, low speed optical module interface<br>See Section 3.6 for possible usages of the pins. |

## 2.1.11 LEDs

See AC specifications in Section 11.4.2.1.

| Reserved | Pin Name | Ball # | Type | Name and Function |
|---|---|---|---|---|
| | LED0_0 | AD14 | O | Port 0 LED0. Programmable LED that indicates Link-Up (default). |
| | LED0_1 | AC14 | O | Port 0 LED1. Programmable LED that indicates 10 Gb/s Link (default). |
| | LED0_2 | AB14 | O | Port 0 LED2. Programmable LED that indicates a Link/Activity indication (default). |
| | LED0_3 | AA14 | O | Port 0 LED3. Programmable LED that indicates a 1 Gb/s Link (default). |
| | LED1_0 | AD13 | O | Port 1 LED0. Programmable LED that indicates Link-Up (default). |
| | LED1_1 | AC13 | O | Port 1 LED1. Programmable LED that indicates 10 Gb/s Link (default). |
| | LED1_2 | AB13 | O | Port 1 LED2. Programmable LED that indicates a Link/Activity indication (default). |
| | LED1_3 | AA13 | O | Port 1 LED3. Programmable LED that indicates a 1 Gb/s Link (default). |



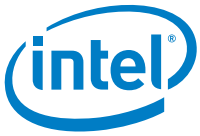
## 2.1.12 RSVD and No Connect Pins

Connecting RSVD pins based on naming convention:

- NC – pin is not connected in the package
- RSVD_NC – reserved pin. Should be left unconnected.
- RSVD_VSS – reserved pin. Should be connected to GND.

| Reserved | Pin Name | Ball # | Name and Function |
|---|---|---|---|
| | RSVDA11_NC<br>RSVDA12_NC<br>RSVDA17_NC<br>RSVDA20_NC<br>RSVDA21_NC<br>RSVDB10_NC<br>RSVDB11_NC<br>RSVDB12_NC<br>RSVDB17_NC | A11<br>A12<br>A17<br>A20<br>A21<br>B10<br>B11<br>B12<br>B17 | RSVD* pins. |
| | RSVDB8_NC<br>RSVDB9_NC<br>RSVDC10_NC<br>RSVDC11_NC<br>RSVDC12_NC<br>RSVDC13_NC<br>RSVDC14_NC<br>RSVDC15_NC<br>RSVDC16_NC | B8<br>B9<br>C10<br>C11<br>C12<br>C13<br>C14<br>C15<br>C16 | RSVD* pins. |
| | RSVDC17_NC<br>RSVDC18_NC<br>RSVDC7_NC<br>RSVDC8_NC<br>RSVDC9_NC<br>RSVDD10_NC<br>RSVDD11_NC<br>RSVDD12_NC<br>RSVDD13_NC | C17<br>C18<br>C7<br>C8<br>C9<br>D10<br>D11<br>D12<br>D13 | RSVD* pins. |
| | RSVDD14_NC<br>RSVDD15_NC<br>RSVDD16_NC<br>RSVDD17_NC<br>RSVDD18_NC<br>RSVDD7_NC<br>RSVDD8_NC<br>RSVDD9_NC<br>RSVDE11_NC | D14<br>D15<br>D16<br>D17<br>D18<br>D7<br>D8<br>D9<br>E11 | RSVD* pins. |

| Reserved | Pin Name | Ball # | Name and Function |
|---|---|---|---|
| | RSVDE13_NC<br>RSVDE15_NC<br>RSVDE9_NC<br>RSVDJ6_NC<br>RSVDJ7_NC<br>RSVDL23_NC<br>RSVDL24_NC | E13<br>E15<br>E9<br>J6<br>J7<br>L23<br>L24 | RSVD* pins. |
| | RSVDM1_NC<br>RSVDM2_NC<br>RSVDM20_NC<br>RSVDM21_NC<br>RSVDN1_NC<br>RSVDN2_NC<br>RSVDN20_NC<br>RSVDN21_NC<br>RSVDN4_NC | M1<br>M2<br>M20<br>M21<br>N1<br>N2<br>N20<br>N21<br>N4 | RSVD* pins. |
| | RSVDN5_NC<br>RSVDT6_NC<br>RSVDT7_NC<br>RSVDW20_NC<br>RSVDW21_NC | N5<br>T6<br>T7<br>W20<br>W21 | RSVD* pins. |
| | RSVDY11_NC<br>RSVDY13_NC<br>RSVDY15_NC<br>RSVDY17_NC<br>RSVDY18_NC | Y11<br>Y13<br>Y15<br>Y17<br>Y18 | RSVD* pins. |
| | NCY16<br>NCY14<br>NCY12<br>NCY10<br>NCY8<br>NCU7<br>NCE18<br>NCE16<br>NCE14<br>NCE12<br>NCE10<br>NCE8<br>NCP4<br>NCL4<br>NCF20<br>NCH7 | Y16<br>Y14<br>Y12<br>Y10<br>Y8<br>U7<br>E18<br>E16<br>E14<br>E12<br>E10<br>E8<br>P4<br>L4<br>F20<br>H7 | NC pins. |
| | RSVDY9_VSS<br>RSVDV16_VSS<br>RSVDW16_VSS<br>RSVDF21_VSS<br>RSVDE17_VSS | Y9<br>V16<br>W16<br>F21<br>E17 | RSVD* pins. |



See AC specifications in Section 11.4.2.1.

## 2.1.13 Miscellaneous

| Reserved | Pin Name | Ball # | Type | Name and Function |
|---|---|---|---|---|
| | LAN_PWR_GOOD | A14 | In<br>Pu | LAN Power Good. A 3.3V input signal. A transition from low to high initializes the 82599 into operation. If not used (POR_BYPASS = 0b), an internal Power-on-Reset (POR) circuit triggers the 82599 power-up. |
| | POR_BYPASS | D19 | In<br>Pu | Bypass indication as to whether or not to use the internal POR or the LAN_PWR_GOOD pin. When set to 1b, the 82599 disables the internal POR circuit and uses the LAN_PWR_GOOD pin as a POR indication. |
| | RSVDAC6_VCC | AC6 | | This input requires an external pull up. See the product's Schematic Checklist for detail. |
| | OSC_SEL | AA9 | T/s<br>Pu | Defines the input clock connected to the REFCLKIN_p/ REFCLKIN_n pins:<br>0b XTAL Clock (valid only for 25 MHz)<br>1b OSC Clock<br>This pin is a strapping option latched at LAN_PWR_GOOD. |
| | AUX_PWR | AB9 | T/s | Auxiliary Power Available. When set, indicates that auxiliary power is available and the 82599 should support $D3_{COLD}$ power state if enabled to do so. This pin is latched at the rising edge of LAN_PWR_GOOD. |
| | MAIN_PWR_OK | AC9 | In | Main Power OK. Indicates that platform main power is up. Must be connected externally. |
| | LAN1_DIS_N | AD20 | T/s<br>Pu | This pin is a strapping pin latched at the rising edge of LAN_PWR_GOOD. If this pin is not connected or driven high during initialization, LAN 1 is enabled. If this pin is driven low during initialization, LAN 1 port is disabled. |
| | LAN0_DIS_N | AD21 | T/s<br>Pu | This pin is a strapping option pin latched at the rising edge of LAN_PWR_GOOD. If this pin is not connected or driven high during initialization, LAN 0 is enabled. If this pin is driven low during initialization, LAN 0 port is disabled.<br>When LAN 0 port is disabled MNG is not functional and it must not be enabled in the EEPROM Control Word 1. |

## 2.1.14 JTAG

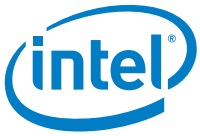
See AC specifications in Section 11.4.2.6.

| Reserved | Pin Name | Ball # | Type | Name and Function |
|---|---|---|---|---|
| | JTCK | B16 | In | JTAG Clock Input. |
| | JTDI | A13 | In<br>Pu | JTAG Data Input. |
| | JTDO | B15 | O/d | JTAG Data Output. |
| | JTMS | B13 | In<br>Pu | JTAG TMS Input. |
| | JRST_N | B14 | In Pu | JTAG Reset Input. Active low reset for the JTAG port. |

## 2.1.15 Power Supplies

See AC specifications in Section 11.3.1.

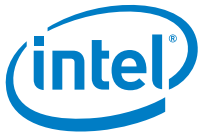
| Reserved | Pin Name | Ball # | Type | Name and Function |
|---|---|---|---|---|
| | VCC1P2 | | 1.2V | Power supply. |
| | | W14, W11, W9, V14, V11, V9, U16, U14, U11, U9, T16, T14, T11, R16, R14, R13, R12, R11, P14, P11, N14, N11, M14, M11, L14, L11, K16, K14, K13, K12, K11, J16, J14, J11, H16, H14, H11, H9, G16, G14, G11, G9, F16, F14, F11, F9, U18, T18, R18, P18, P16, N18, N16, M18, M16, L18, L16, K18, J18, H18, K9, K7, J9, T9, R9, R7, M9, M7, L9, L7, P9, P7, N9, N7 | | |
| | VCC3P3 | | 3.3V | Power supply. |
| | | AD19, AD15, AD10, AD6, A19, A15, A10, A6, E7, Y7, L5, P5 | | |
| | VSS | | 0V | Ground |
| | | AD16, AD9, W18, W17, W15, W13, W12, W10, W8, W7, V17, V15, V13, V12, V10, V8, U15, U13, U12, U10, T15, T13, T12, T10, R15, R10, P15, P13, P12, P10, N15, N13, N12, N10, M15, M13, M12, M10, L15, L13, L12, L10, K15, K10, J15, J13, J12, J10, H15, H13, H12, H10, G17, G15, G13, G12, G10, G8, F18, F17, F15, F13, F12, F10, F8, F7, A16, A9, K8, K6, J8, J5, J4, H8, H6, G7, G6, G5, G4, F6, E6, E5, E4, D6, C6, C5, C4, B5, B3, A5, A3, AD5, AD3, AC5, AC3, AB6, AB5, AB4, AA6, Y6, Y5, Y4, W6, V7, V6, V5, V4, U8, U6, T8, T5, T4, R8, R6, M8, M6, M5, M4, M3, L8, L6, L3, K5, K4, K3, K2, K1, J3, H3, H2, H1, G3, F3, F2, F1, E3, D3, D2, D1, C3, B2, B1, A2, A1, AD2, AD1, AC2, AC1, AB3, AA3, AA2, AA1, Y3, W3, W2, W1, V3, U3, U2, U1, T3, R5, R4, R3, R2, R1, P8, P6, P3, N8, N6, N3, AD24, AD23, AD22, AC24, AC23, AC22, AB22, AB21, AB20, AA24, AA23, AA22, Y22, Y21, Y20, Y19, W24, W23, W22, W19, V22, V21, V20, V19, V18, U24, U23, U22, U19, U17, T22, T21, T20, T19, T17, R24, R23, R22, R19, R17, P22, P21, P20, P19, P17, N23, N22, N19, N17, M23, M22, M19, M17, L22, L21, L20, L19, L17, K24, K23, K22, K19, K17, J22, J21, J20, J19, J17, H24, H23, H22, H19, H17, G22, G21, G20, G19, G18, F24, F23, F22, F19, E22, E21, E20, E19, D24, D23, D22, C22, C21, C20, B24, B23, B22, A24, A23, A22 | | |



## 2.1.16 Pull-Ups

| Reserved | Pin Name | Reserved | Internal Pull Up at Power Up | | Internal Pull Up at Nominal Active State | |
|---|---|---|---|---|---|---|
| | | | PUP | Comment | PUP | Comment |
| | EE_DI | | N | | N | |
| | EE_DO | | Y | | Y | |
| | EE_SK | | N | | N | |
| | EE_CS_N | | N | | N | |
| | FLSH_SI | | Y | | N | |
| | FLSH_SO | | Y | | Y | |
| | FLSH_SCK | | Y | | N | |
| | FLSH_CE_N | | Y | | N | |
| | SMBCLK | | N | | N | |
| | SMBD | | N | | N | |
| | SMBALRT_N | | N | | N | |
| | SCL0/SCL1 | | N | | N | |
| | SDA0/SDA1 | | N | | N | |
| | NCSI_CLK_IN | | N | | N | |
| | NCSI_CRS_DV | | N | | N | |
| | NCSI_RXD_0 | | N | | N | |
| | NCSI_RXD_1 | | N | | N | |
| | NCSI_TX_EN | | N | | N | |
| | NCSI_TXD_0 | | N | | N | |
| | NCSI_TXD_1 | | N | | N | |
| | MDIO0 | | N | | N | |
| | MDC0 | | N | | N | |
| | MDIO1 | | N | | N | |
| | MDC1 | | N | | N | |

| Reserved | Pin Name | Reserved | Internal Pull Up at Power Up | | Internal Pull Up at Nominal Active State | |
|---|---|---|---|---|---|---|
| | | | PUP | Comment | PUP | Comment |
| | SDP0_0<br>SDP0_1<br>SDP0_2<br>SDP0_3<br>SDP0_4<br>SDP0_5<br>SDP0_6<br>SDP0_7 | | Y | | Y | |
| | SDP1_0<br>SDP1_1<br>SDP1_2<br>SDP1_3<br>SDP1_4<br>SDP1_5<br>SDP1_6<br>SDP1_7 | | Y | | Y | |
| | LED0_0<br>LED0_1<br>LED0_2<br>LED0_3 | | N | | N | |
| | LED1_0<br>LED1_1<br>LED1_2<br>LED1_3 | | N | | N | |
| | LAN_PWR_GOOD | | Y | | Y | |
| | AUX_PWR | | N | | N | |
| | LAN0_DIS_N | | Y | | Y | |
| | LAN1_DIS_N | | Y | | Y | |
| | MAIN_PWR_OK | | N | | N | |
| | JTCK | | N | | N | |
| | JTDI | | N | | N | |
| | JTDO | | N | | N | |
| | JTMS | | N | | N | |
| | JRST_N | | Y | | Y | |
| | PE_RST_N | | N | | N | |
| | PE_WAKE_N | | N | | N | |



| Reserved | Pin Name | Reserved | Internal Pull Up at Power Up | | Internal Pull Up at Nominal Active State | |
|---|---|---|---|---|---|---|
| | | | PUP | Comment | PUP | Comment |
| | OSC_SEL | | Y | | Y | |
| | POR_BYPASS | | Y | | N | |

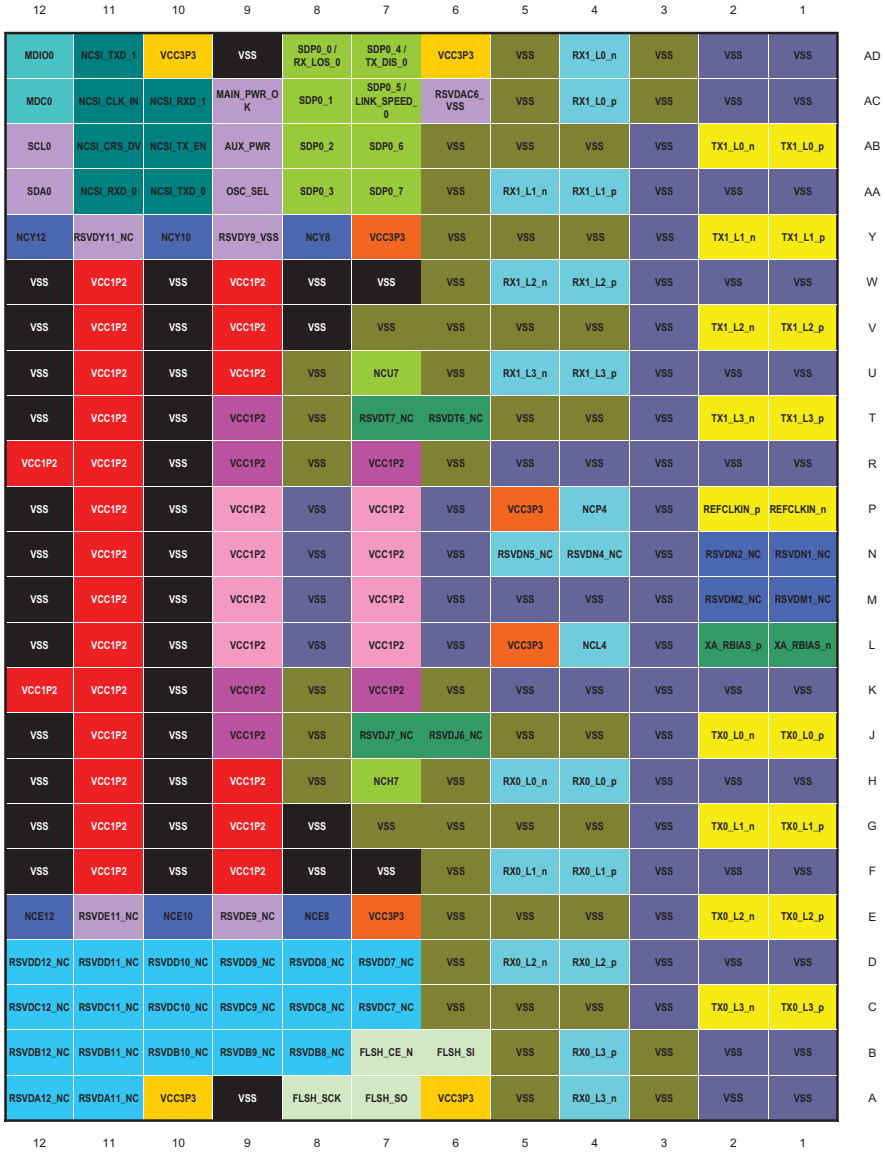
**Note:** Refer to the reference schematics for implementation details.

# 2.2 Ball Out — Top Level

Top view, through package.

| | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AD | VSS | VSS | VSS | LAN0_DIS_N | LAN1_DIS_N | VCC3P3 | PE_RST_N | SCL1 | VSS | VCC3P3 | LED0_0 | LED1_0 |
| AC | VSS | VSS | VSS | PER_0_n | PER_0_p | SMBCLK | SDA1 | MDIO1 | SDP1_0 / RX_LOS_1 | SDP1_5 / LINK_SPEED_1 | LED0_1 | LED1_1 |
| AB | PE_CLK_n | PE_CLK_p | VSS | VSS | VSS | SMBD | MDC1 | SDP1_2 | SDP1_1 | SDP1_6 | LED0_2 | LED1_2 |
| AA | VSS | VSS | VSS | PER_1_n | PER_1_p | SMBALRT_N | PE_WAKE_N | SDP1_3 | SDP1_4 / TX_DIS_1 | SDP1_7 | LED0_3 | LED1_3 |
| Y | PET_0_n | PET_0_p | VSS | VSS | VSS | VSS | RSVDY18_NC | RSVDY17_NC | NCY16 | RSVDY15_NC | NCY14 | RSVDY13_NC |
| W | VSS | VSS | VSS | RSVDW21_NC | RSVDW20_NC | VSS | VSS | VSS | RSVDW16_VSS | VSS | VCC1P2 | VSS |
| V | PET_1_n | PET_1_p | VSS | VSS | VSS | VSS | VSS | VSS | RSVDV16_VSS | VSS | VCC1P2 | VSS |
| U | VSS | VSS | VSS | PER_2_n | PER_2_p | VSS | VCC1P2 | VSS | VCC1P2 | VSS | VCC1P2 | VSS |
| T | PET_2_n | PET_2_p | VSS | VSS | VSS | VSS | VCC1P2 | VSS | VCC1P2 | VSS | VCC1P2 | VSS |
| R | VSS | VSS | VSS | PER_3_n | PER_3_p | VSS | VCC1P2 | VSS | VCC1P2 | VSS | VCC1P2 | VCC1P2 |
| P | PET_3_n | PET_3_p | VSS | VSS | VSS | VSS | VCC1P2 | VSS | VCC1P2 | VSS | VCC1P2 | VSS |
| N | PE_RSENSE | VSS | VSS | RSVDN21_NC | RSVDN20_NC | VSS | VCC1P2 | VSS | VCC1P2 | VSS | VCC1P2 | VSS |
| M | PE_RBIAS | VSS | VSS | RSVDM21_NC | RSVDM20_NC | VSS | VCC1P2 | VSS | VCC1P2 | VSS | VCC1P2 | VSS |
| L | RSVDL24_NC | RSVDL23_NC | VSS | VSS | VSS | VSS | VCC1P2 | VSS | VCC1P2 | VSS | VCC1P2 | VSS |
| K | VSS | VSS | VSS | PER_4_n | PER_4_p | VSS | VCC1P2 | VSS | VCC1P2 | VSS | VCC1P2 | VCC1P2 |
| J | PET_4_n | PET_4_p | VSS | VSS | VSS | VSS | VCC1P2 | VSS | VCC1P2 | VSS | VCC1P2 | VSS |
| H | VSS | VSS | VSS | PER_5_n | PER_5_p | VSS | VCC1P2 | VSS | VCC1P2 | VSS | VCC1P2 | VSS |
| G | PET_5_n | PET_5_p | VSS | VSS | VSS | VSS | VSS | VSS | VCC1P2 | VSS | VCC1P2 | VSS |
| F | VSS | VSS | VSS | RSVDF21_VSS | NCF20 | VSS | VSS | VSS | VCC1P2 | VSS | VCC1P2 | VSS |
| E | PET_6_n | PET_6_p | VSS | VSS | VSS | VSS | NCE18 | RSVDE17_VSS | NCE16 | RSVDE15_NC | NCE14 | RSVDE13_NC |
| D | VSS | VSS | VSS | PER_6_n | PER_6_p | POR_BYPASS | RSVDD18_NC | RSVDD17_NC | RSVDD16_NC | RSVDD15_NC | RSVDD14_NC | RSVDD13_NC |
| C | PET_7_n | PET_7_p | VSS | VSS | VSS | EE_CS_N | RSVDC18_NC | RSVDC17_NC | RSVDC16_NC | RSVDC15_NC | RSVDC14_NC | RSVDC13_NC |
| B | VSS | VSS | VSS | PER_7_n | PER_7_p | EE_SK | EE_DI | RSVDB17_NC | JTCK | JTDO | JRST_N | JTMS |
| A | VSS | VSS | VSS | RSVDA21_NC | RSVDA20_NC | VCC3P3 | EE_DO | RSVDA17_NC | VSS | VCC3P3 | LAN_PWR_GOOD | JTDI |
| | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 |

**Figure 2-1 Package Layout - Left View**



| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MDIO0 | NCSI_TXD_1 | VCC3P3 | VSS | SDP0_0 / RX_LOS_0 | SDP0_4 / TX_DIS_0 | VCC3P3 | VSS | RX1_L0_n | VSS | VSS | VSS | AD |
| MDC0 | NCSI_CLK_IN | NCSI_RXD_1 | MAIN_PWR_OK | SDP0_1 | SDP0_5 / LINK_SPEED_0 | RSVDAC6_VSS | VSS | RX1_L0_p | VSS | VSS | VSS | AC |
| SCL0 | NCSI_CRS_DV | NCSI_TX_EN | AUX_PWR | SDP0_2 | SDP0_6 | VSS | VSS | VSS | VSS | TX1_L0_n | TX1_L0_p | AB |
| SDA0 | NCSI_RXD_0 | NCSI_TXD_0 | OSC_SEL | SDP0_3 | SDP0_7 | VSS | RX1_L1_n | RX1_L1_p | VSS | VSS | VSS | AA |
| NCY12 | RSVDY11_NC | NCY10 | RSVDY9_VSS | NCY8 | VCC3P3 | VSS | VSS | VSS | VSS | TX1_L1_n | TX1_L1_p | Y |
| VSS | VCC1P2 | VSS | VCC1P2 | VSS | VSS | VSS | RX1_L2_n | RX1_L2_p | VSS | VSS | VSS | W |
| VSS | VCC1P2 | VSS | VCC1P2 | VSS | VSS | VSS | VSS | VSS | VSS | TX1_L2_n | TX1_L2_p | V |
| VSS | VCC1P2 | VSS | VCC1P2 | VSS | NCU7 | VSS | RX1_L3_n | RX1_L3_p | VSS | VSS | VSS | U |
| VSS | VCC1P2 | VSS | VCC1P2 | VSS | RSVDT7_NC | RSVDT6_NC | VSS | VSS | VSS | TX1_L3_n | TX1_L3_p | T |
| VCC1P2 | VCC1P2 | VSS | VCC1P2 | VSS | VCC1P2 | VSS | VSS | VSS | VSS | VSS | VSS | R |
| VSS | VCC1P2 | VSS | VCC1P2 | VSS | VCC1P2 | VSS | VCC3P3 | NCP4 | VSS | REFCLKIN_p | REFCLKIN_n | P |
| VSS | VCC1P2 | VSS | VCC1P2 | VSS | VCC1P2 | VSS | RSVDN5_NC | RSVDN4_NC | VSS | RSVDN2_NC | RSVDN1_NC | N |
| VSS | VCC1P2 | VSS | VCC1P2 | VSS | VCC1P2 | VSS | VSS | VSS | VSS | RSVDM2_NC | RSVDM1_NC | M |
| VSS | VCC1P2 | VSS | VCC1P2 | VSS | VCC1P2 | VSS | VCC3P3 | NCL4 | VSS | XA_RBIAS_p | XA_RBIAS_n | L |
| VCC1P2 | VCC1P2 | VSS | VCC1P2 | VSS | VCC1P2 | VSS | VSS | VSS | VSS | VSS | VSS | K |
| VSS | VCC1P2 | VSS | VCC1P2 | VSS | RSVDJ7_NC | RSVDJ6_NC | VSS | VSS | VSS | TX0_L0_n | TX0_L0_p | J |
| VSS | VCC1P2 | VSS | VCC1P2 | VSS | NCH7 | VSS | RX0_L0_n | RX0_L0_p | VSS | VSS | VSS | H |
| VSS | VCC1P2 | VSS | VCC1P2 | VSS | VSS | VSS | VSS | VSS | VSS | TX0_L1_n | TX0_L1_p | G |
| VSS | VCC1P2 | VSS | VCC1P2 | VSS | VSS | VSS | RX0_L1_n | RX0_L1_p | VSS | VSS | VSS | F |
| NCE12 | RSVDE11_NC | NCE10 | RSVDE9_NC | NCE8 | VCC3P3 | VSS | VSS | VSS | VSS | TX0_L2_n | TX0_L2_p | E |
| RSVDD12_NC | RSVDD11_NC | RSVDD10_NC | RSVDD9_NC | RSVDD8_NC | RSVDD7_NC | VSS | RX0_L2_n | RX0_L2_p | VSS | VSS | VSS | D |
| RSVDC12_NC | RSVDC11_NC | RSVDC10_NC | RSVDC9_NC | RSVDC8_NC | RSVDC7_NC | VSS | VSS | VSS | VSS | TX0_L3_n | TX0_L3_p | C |
| RSVDB12_NC | RSVDB11_NC | RSVDB10_NC | RSVDB9_NC | RSVDB8_NC | FLSH_CE_N | FLSH_SI | VSS | RX0_L3_p | VSS | VSS | VSS | B |
| RSVDA12_NC | RSVDA11_NC | VCC3P3 | VSS | FLSH_SCK | FLSH_SO | VCC3P3 | VSS | RX0_L3_n | VSS | VSS | VSS | A |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | |

**Figure 2-2 Package Layout - Right View**

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

# 3.0 Interconnects

## 3.1 PCI-Express* (PCIe*)

### 3.1.1 Overview

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

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

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

Figure 3-1 shows the PCIe architecture.

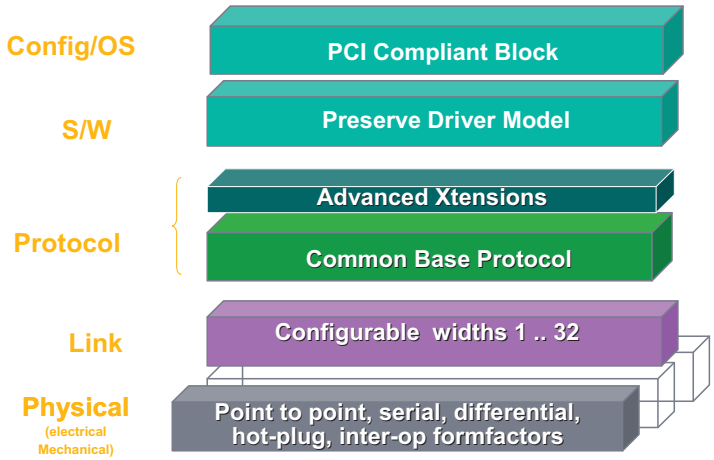



**Figure 3-1 PCIe Stack Structure**

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

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

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

#### 3.1.1.1 Architecture, Transaction and Link Layer Properties

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

- Baseline messaging:
  - In-band messaging of formerly side-band legacy signals (interrupts, etc.)
  - System-level power management supported via messages
- Power management:
  - Full support for PCIm
  - Wake capability from D3cold state
  - Compliant with ACPI, PCIm software model
  - Active state power management
- Support for PCIe V2.0 (2.5GT/s or 5GT/s)
  - Support for completion time out control
  - Support for additional registers in the PCIe capability structure

### 3.1.1.2 Physical Interface Properties

- Point-to-point interconnect
  - Full-duplex; no arbitration
- Signaling technology:
  - Low Voltage Differential (LVD)
  - Embedded clock signaling using 8b/10b encoding scheme
- Serial frequency of operation: PCIe V2.0 (2.5GT/s or 5GT/s).
- Interface width of 1, 2, 4, or 8 PCIe lanes.
- DFT and DFM support for high-volume manufacturing

### 3.1.1.3 Advanced Extensions

PCIe defines a set of optional features to enhance platform capabilities for specific usage modes. The 82599 supports the following optional features:

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

## 3.1.2 General Functionality

### 3.1.2.1 Native/Legacy

All 82599 PCI functions are native PCIe functions.

### 3.1.2.2 Locked Transactions

The 82599 does not support locked requests as a target or a master.

## 3.1.3 Host Interface

PCIe device numbers identify logical devices within the physical device (the 82599 is a physical device). The 82599 implements a single logical device with two separate PCI Functions: LAN 0 and LAN 1. The device number is captured from each type 0 configuration write transaction.

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

### 3.1.3.1 TAG ID Allocation

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

#### 3.1.3.1.1 TAG ID Allocation for Read Transactions

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

**Table 3-1 TAG ID Allocation Table for Read Transactions**

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

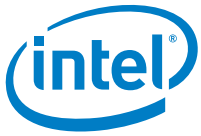


**Table 3-1 TAG ID Allocation Table for Read Transactions (Continued)**

| TAG ID | Description | TAG ID | Description |
|---|---|---|---|
| 0x8 | Data Request 0x8 | 0x18 | Rx Descriptor 0 |
| 0x9 | Data Request 0x9 | 0x19 | Rx Descriptor 1 |
| 0xA | Data Request 0xA | 0x1A | Rx Descriptor 2 |
| 0xB | Data Request 0xB | 0x1B | Rx Descriptor 3 |
| 0xC | Data Request 0xC | 0x1C | Rx Descriptor 4 |
| 0xD | Data Request 0xD | 0x1D | Rx Descriptor 5 |
| 0xE | Data Request 0xE | 0x1E | Rx Descriptor 6 |
| 0xF | Data Request 0xF | 0x1F | Rx Descriptor 7 |

#### 3.1.3.1.2 TAG ID Allocation for Write Transactions

Request tag allocation depends on these system parameters:

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

Case 1 — DCA Disabled in the System:

The following table lists the write requests tags:

| Tag ID | Description |
|---|---|
| 2 | Write-back descriptor Tx /write-back head. |
| 4 | Write-back descriptor Rx. |
| 6 | Write data. |
| 30 | MSI and MSI-X. |

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

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

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

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

— Bits[7:4] — Reserved

- DCA 1.0 Platforms: the request tag (all eight bits) is taken from the *CPU ID* field of the DCA_RXCTRL or DCA_TXCTRL registers

### 3.1.3.2 Completion Timeout Mechanism

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

The completion timeout mechanism is activated for each request that requires one or more completions when the request is transmitted. The 82599 provides a programmable range for the completion timeout, as well as the ability to disable the completion timeout altogether. The completion timeout is programmed through an extension of the PCIe capability structure.

The 82599's reaction to a completion timeout is listed in Table 3-8.

The 82599 controls the following aspects of completion timeout:

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

**Table 3-2 Completion Timeout Programming**

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

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

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

#### 3.1.3.2.1 Completion Timeout Period

Programmed through the PCI configuration. Visible through bits 3:0 in the Device Capabilities 2 Register (0xC4; RO) register (see Section 9.3.10.10). The 82599 supports all four ranges defined by PCIe V2.0 (2.5GT/s or 5GT/s):

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

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

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

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

## 3.1.4 Transaction Layer

The upper layer of the PCIe architecture is the transaction layer. The transaction layer connects to 82599's core using an implementation-specific protocol. Through this core-to-transaction-layer protocol, the application-specific parts of the 82599 interact with the PCIe subsystem and transmits and receives requests to or from the remote PCIe agent, respectively.

### 3.1.4.1 Transaction Types Accepted by the 82599

**Table 3-3 Transaction Types Accepted by the Transaction Layer**

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

Flow Control Types Legend:

CPLD — Completion Data Payload

CPLH — Completion Headers

NPD — Non-Posted Request Data Payload

NPH — Non-Posted Request Headers

PD — Posted Request Data Payload

PH — Posted Request Headers

### 3.1.4.2 Transaction Types Initiated by the 82599

**Table 3-4 Transaction Types Initiated by the Transaction Layer**

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

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

#### 3.1.4.2.1 Data Alignment

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

The 82599 breaks requests into 4 KB-aligned requests (if needed). This does not pose any requirement on software. However, if software allocates a buffer across a 4 KB boundary, hardware issues multiple requests for the buffer. Software should consider aligning buffers to a 4 KB boundary in cases where it improves performance.

The general rules for packet alignment are as follows. Note that these apply to all the 82599 requests (read/write):

- The length of a single request does not exceed the PCIe limit of MAX_PAYLOAD_SIZE for write and MAX_READ_REQ for read.
- The length of a single request does not exceed the 82599 internal limitations.
- A single request does not span across different memory pages as noted by the 4 KB boundary alignment previously mentioned.

If a request can be sent as a single PCIe packet and still meet the general rules for packet alignment, then it is not broken at the cache line boundary but rather sent as a single packet (motivation is that the chipset can break the request along cache line boundaries, but the 82599 should still benefit from better PCIe use). However, if any of the three general rules require that the request is broken into two or more packets, then the request is broken at the cache line boundary.

#### 3.1.4.2.2 Multiple Tx Data Read Requests (MULR)

The 82599 supports 16 multiple pipelined requests for transmit data. In general, requests can belong to the same packet or to consecutive packets. However, the following restrictions apply:

- All requests for a packet must be issued before a request is issued for a consecutive packet.
- Read requests can be issued from any of the supported queues, as long as the previous restriction is met. Pipelined requests can belong to the same queue or to separate queues. However, as previously noted, all requests for a certain packet are issued (from the same queue) before a request is issued for a different packet (potentially from a different queue).
- The PCIe specification does not insure that completions for separate requests return in-order. Read completions for concurrent requests are not required to return in the order issued. The 82599 handles completions that arrive in any order. Once all completions arrive for a given request, it can issue the next pending read data request.
- The 82599 incorporates a reorder buffer to support re-ordering of completions for all issued requests. Each request/completion can be up to 512 bytes long. The maximum size of a read request is defined as the minimum {2 KB bytes, Max_Read_Request_Size}.
- In addition to the transmit data requests, the 82599 can issue eight pipelined read requests for Tx descriptors and eight pipelined read requests for Rx descriptors. The requests for Tx data, Tx descriptors, and Rx descriptors are independently issued.

### 3.1.4.3 Messages

#### 3.1.4.3.1 Received Messages

Message packets are special packets that carry a message code. The upstream device transmits special messages to the 82599 by using this mechanism. The transaction layer decodes the message code and responds to the message accordingly.

**Table 3-5 Supported Message in the 82599 - as a Receiver**

| Message Code [7:0] | Routing r2r1r0 | Message | 82599 Later Response |
|---|---|---|---|
| 0x14 | 100b | PM_Active_State_NAK | Internal Signal Set |
| 0x19 | 011b | PME_Turn_Off | Internal Signal Set |
| 0x50 | 100b | Slot power limit support (has one Dword data) | Silently Drop |
| 0x7E | 010b, 011b,100b | Vendor_defined type 0 No data | Unsupported Request |
| 0x7E | 010b,011b,100b | Vendor_defined type 0 data | Unsupported Request |

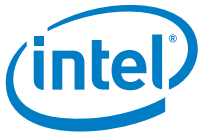


**Table 3-5 Supported Message in the 82599 - as a Receiver (Continued)**

| Message Code [7:0] | Routing r2r1r0 | Message | 82599 Later Response |
|---|---|---|---|
| 0x7F | 010b,011b,100b | Vendor_defined type 1 no data | Silently Drop |
| 0x7F | 010b, 011b,100b | Vendor_defined type 1 data | Silently Drop |
| 0x00 | 011b | Unlock | Silently Drop |

### 3.1.4.3.2 Transmitted Messages

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

**Table 3-6 Supported Message in the 82599 - as a Transmitter**

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

### 3.1.4.4 Ordering Rules

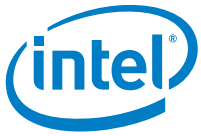
The 82599 meets the PCIe ordering rules by following the PCI simple device model:

1. Deadlock Avoidance – The 82599 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 82599 are allowed to proceed.
2. Descriptor/Data Ordering — The 82599 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 can change the MSI or MSI-X tables during run-time. Software expects that interrupt messages issued after the table has been updated are using the updated contents of the tables.
   a. Since software doesn't know when the tables are actually updated in the 82599, 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 82599 meets the rules relating to independence between target and master accesses:
   a. The acceptance of a target posted request does not depend upon the transmission of any TLP.
   b. The acceptance of a target Non-posted Request does not depend upon the transmission of a non-posted request.
   c. Accepting a completion does not depend upon the transmission of any TLP.

#### 3.1.4.4.1 Out of Order Completion Handling

In a split transaction protocol, when using multiple read requests in a multi-processor environment, there is a risk that completions for separate requests arrive from the host memory out of order and interleaved. In this case, the 82599 sorts the completions and transfers them to the network in the correct order.

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



# 3.1.4.5 Transaction Definition and Attributes

## 3.1.4.5.1 Max Payload Size

The 82599's policy for determining Max Payload Size (MPS) is as follows:

1. Master requests initiated by the 82599 (including completions) limit Max Payload Size to the value defined for the function issuing the request.
2. Target write accesses to the 82599 are accepted only with a size of one Dword or two Dwords. Write accesses in the range of three Dwords (MPS) are flagged as unreliable. Write accesses above MPS are flagged as malformed.

## 3.1.4.5.2 Traffic Class (TC) and Virtual Channels (VC)

The 82599 only supports TC = 0 and VC = 0 (default).

## 3.1.4.5.3 Relaxed Ordering

The 82599 takes advantage of the relaxed ordering rules in PCIe. By setting the relaxed ordering bit in the packet header, the 82599 enables the system to optimize performance in the following cases:

1. Relaxed ordering for descriptor and data reads — When the 82599 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 82599 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 82599 cannot relax ordering for descriptor writes or an MSI write.

Relaxed ordering is enabled globally in the 82599 by clearing the CTRL_EXT.RO_DIS bit and further enabled per queue in the DCA_RXCTRL[n] registers.

# 3.1.4.6 Flow Control

## 3.1.4.6.1 Flow Control Rules

The 82599 only implements the default Virtual Channel (VC0). A single set of credits is maintained for VC0.

**Table 3-7 Flow Control Credits Allocation**

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

Rules for FC updates:

- The 82599 maintains two credits for NPD at any given time. It increments the credit by one after the credit is consumed, and sends an UpdateFC packet as soon as possible. UpdateFC packets are scheduled immediately after a resource is available.
- The 82599 provides **16** credits for PH (such as for concurrent target writes) and four credits for NPH (such as for four concurrent target reads). UpdateFC packets are scheduled immediately after a resource is available.
- The 82599 follows the PCIe recommendations for frequency of UpdateFC FCPs.

#### 3.1.4.6.2 Upstream Flow Control Tracking

The 82599 issues a master transaction only when the required flow control credits are available. Credits are tracked for posted, non-posted, and completions (the later to operate against a switch).

#### 3.1.4.6.3 Flow Control Update Frequency

In all cases, Update Flow Control Packets (FCPs) are scheduled immediately after a resource is available.

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

#### 3.1.4.6.4 Flow Control Timeout Mechanism

The 82599 implements the optional flow control update timeout mechanism.

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

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

## 3.1.5 Link Layer

### 3.1.5.1 ACK/NAK Scheme

The 82599 supports two alternative schemes for ACK/NAK rate:

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

The *PCIe Error Recovery* bit (loaded from the EEPROM) determines which of the two schemes is used.

### 3.1.5.2 Supported DLLPs

The following DLLPs are supported by the 82599 as a receiver:

- ACK
- NAK
- PM_Request_Ack
- InitFC1-P
- InitFC1-NP
- InitFC1-Cpl
- InitFC2-P
- InitFC2-NP
- InitFC2-Cpl
- UpdateFC-P
- UpdateFC-NP
- UpdateFC-Cpl

The following DLLPs are supported by the 82599 as a transmitter:

- ACK
- NAK

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

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

#### 3.1.5.3 Transmit EDB Nullifying (End Bad)

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

### 3.1.6 Physical Layer

#### 3.1.6.1 Link Speed

The 82599 supports PCIe V2.0 (2.5GT/s or 5GT/s). The following configuration controls link speed:

- PCIe *Supported Link Speeds* bit — Indicates the link speeds supported by the 82599. Loaded from the PCIe *Link Speed* field in the EEPROM.

| EEPROM Word Offset (Starting at Odd Word) | Allow PCIe V2.0(Default) | Force PCIe V2.0 Setting | Description |
|---|---|---|---|
| 2*N+1 | 0x094 | | MORIA6 register offset (lower word). |
| 2*N+2 | 0x0000 | 0x0100 | Disabling PCIe V2.0 is controlled by setting bit[8] in this register. When the bit is set the 82599 does not advertise PCIe V2.0 link-speed support. |

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

The 82599 does not initiate a hardware autonomous speed change.

The 82599 supports entering compliance mode at the speed indicated in the *Target Link Speed* field in the PCIe Link Control 2 register. Compliance mode functionality is controlled via the PCIe Link Control 2 register.

### 3.1.6.2 Link Width

- The 82599 supports a maximum link width of x8, x4, x2, or x1 as determined by the PCIe Analog Configuration Module in the EEPROM and can be set as follows. Note that these settings might not be needed during normal operation:

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

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

During link configuration, the platform and the 82599 negotiate on a common link width. The link width must be one of the supported PCIe link widths (x1, 2x, x4, x8), such that:

- If Maximum Link Width = x8, then the 82599 negotiates to either x8, x4, x2 or x1[1]
- If Maximum Link Width = x4, then the 82599 negotiates to either x4 or x1
- If Maximum Link Width = x1, then the 82599 only negotiates to x1

The 82599 does not initiate a hardware autonomous link width change.

**Note:** Some PCIe x8 slots are actually configured as x4 slots. These slots have insufficient bandwidth for full 10 GbE line rate with dual port 10 GbE devices. If a solution suffers bandwidth issues when both 10 GbE ports are active, it is recommended to verify that the PCIe slot is indeed a true PCIe x8.

### 3.1.6.3 Polarity Inversion

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

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

1. See restriction in Section 3.1.6.6.

### 3.1.6.4 L0s Exit Latency

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

### 3.1.6.5 Lane-to-Lane De-Skew

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

The 82599 supports de-skew of up to 12 symbols time [48 ns for PCIe v2.0 (2.5GT/s) and 24 ns for PCIe V2.0 (5GT/s)].

### 3.1.6.6 Lane Reversal

Auto lane reversal is supported by the 82599 at its hardware default setting. The following lane reversal modes are supported:

- Lane configurations x8, x4, x2, and x1
- Lane reversal in x8 and in x4
- Degraded mode (downshift) from x8 to x4 to x2 to x1 and from x4 to x1, with one restriction — if lane reversal is executed in x8, then downshift is only to x1 and not to x4.

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

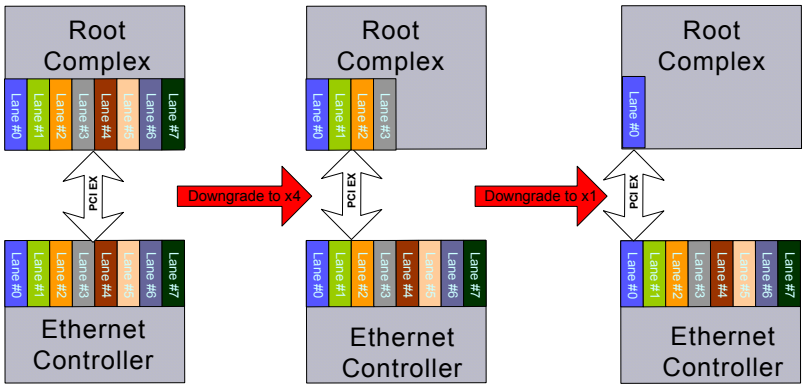



**Figure 3-2 Lane Downshift in an x8 Configuration**

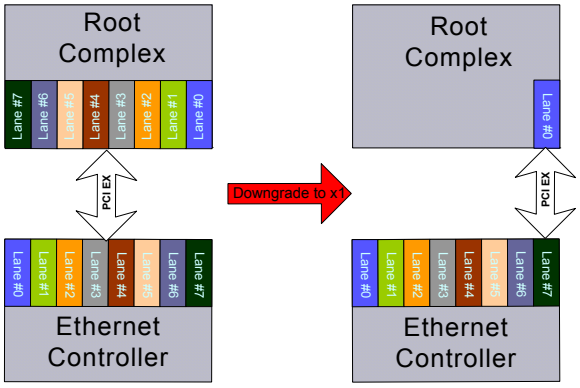



**Figure 3-3 Lane Downshift in a Reversal x8 Configuration**

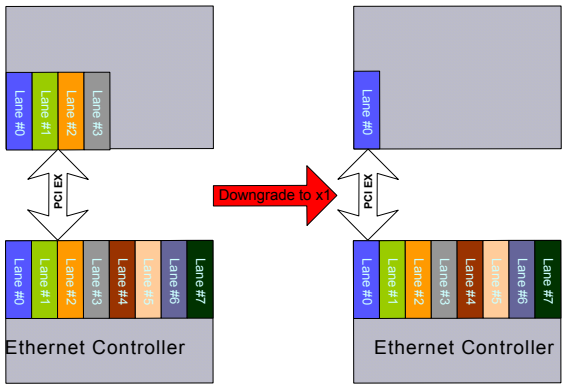



**Figure 3-4 Lane Downshift in a x4 Configuration**

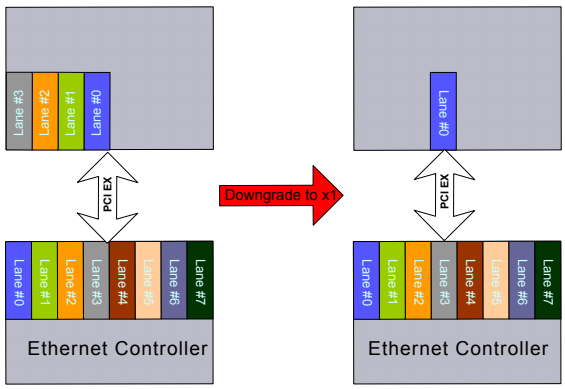



**Figure 3-5 Lane Downshift in an x4 Reversal Configuration**

#### 3.1.6.7 Reset

The PCIe PHY supplies the core reset to the 82599. The reset can be caused by the following events:

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

#### 3.1.6.8 Scrambler Disable

The scrambler/de-scrambler functionality in the 82599 can be eliminated by two mechanisms:

- Upstream according to the PCIe specification
- EEPROM bit — Scram_dis.

### 3.1.7 Error Events and Error Reporting

#### 3.1.7.1 General Description

PCIe defines two error reporting paradigms: the baseline capability and the Advanced Error Reporting (AER) capability. The baseline error reporting capabilities are required of all PCIe devices and define the minimum error reporting requirements. The AER capability is defined for more robust error reporting and is implemented with a specific PCIe capability structure. Both mechanisms are supported by the 82599.

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. These include the following cases of Unsupported Request (UR):

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

Figure 3-6 shows, in detail, the flow of error reporting in the 82599.

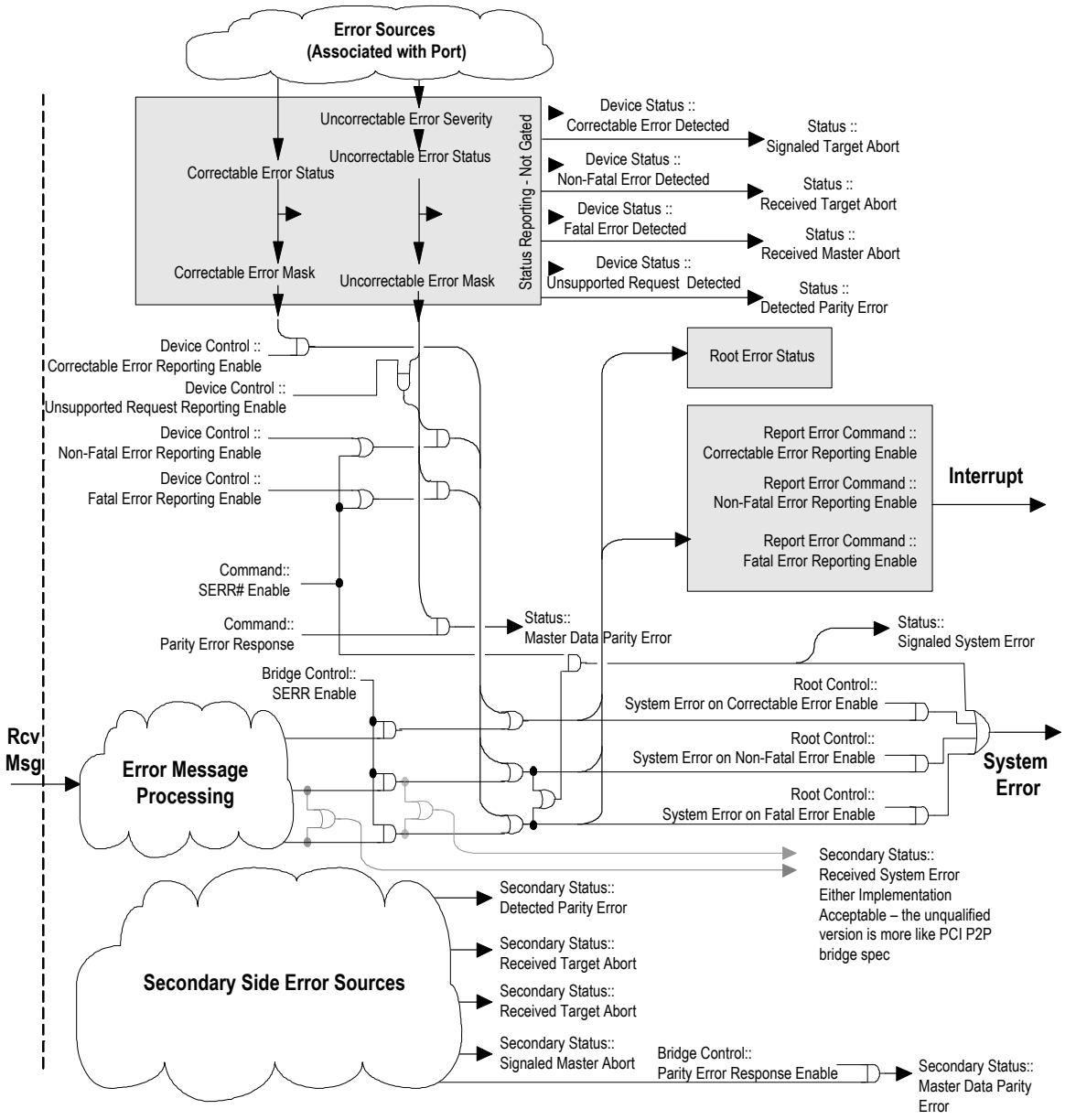



**Figure 3-6 Error Reporting Mechanism**

### 3.1.7.2 Error Events

Table 3-8 lists the error events identified by the 82599 and the response in terms of logging, reporting, and actions taken. Refer to the PCIe specification for the effect on the PCI Status register.

**Table 3-8 Response and Reporting of PCIe Error Events**

| Error Name | Error Events | Default Severity | Action |
|---|---|---|---|
| **Physical Layer Errors** | | | |
| Receiver Error | • 8b/10b Decode Errors<br>• Packet Framing Error | Correctable<br>Send ERR_CORR | TLP to Initiate NAK, Drop Data<br>DLLP to Drop |
| **Data Link Errors** | | | |
| Bad TLP | • Bad CRC<br>• Not Legal EDB<br>• Wrong Sequence Number | Correctable<br>Send ERR_CORR | TLP to Initiate NAK, Drop Data |
| Bad DLLP | • Bad CRC | Correctable<br>Send ERR_CORR | DLLP το Drop |
| Replay Timer Timeout | • REPLAY_TIMER expiration | Correctable<br>Send ERR_CORR | Follow LL Rules |
| REPLAY NUM Rollover | • REPLAY NUM Rollover | Correctable<br>Send ERR_CORR | Follow LL Rules |
| Data Link Layer Protocol Error | • Violations of Flow Control Initialization Protocol | Uncorrectable<br>Send ERR_FATAL | |
| TLP Errors | | | |
| Poisoned TLP Received | • TLP With Error Forwarding | Uncorrectable<br>ERR_NONFATAL<br>Log Header | If completion TLP:<br>Error is non-fatal (default case)<br>• Send error message if advisory<br>• Retry the request once and send advisory error message on each failure<br>• If fails, send uncorrectable error message<br>Error is defined as fatal<br>• Send uncorrectable error message |
| Unsupported Request (UR) | • Wrong Config Access<br>• MRdLk<br>• Config Request Type1<br>• Unsupported Vendor Defined Type 0 Message<br>• Not Valid MSG Code<br>• Not Supported TLP Type<br>• Wrong Function Number<br>• Received TLP Outside Address Range | Uncorrectable<br>ERR_NONFATAL<br>Log header | Send Completion With UR |

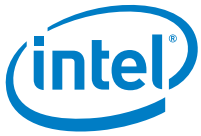


**Table 3-8 Response and Reporting of PCIe Error Events (Continued)**

| Error Name | Error Events | Default Severity | Action |
|---|---|---|---|
| Completion Timeout | • Completion Timeout Timer Expired | Uncorrectable<br>ERR_NONFATAL | Error is non-fatal (default case)<br>• Send error message if advisory<br>• Retry the request once and send advisory error message on each failure<br>• If fails, send uncorrectable error message<br>Error is defined as fatal<br>• Send uncorrectable error message |
| Completer Abort | • Received Target Access With Data Size >64 bits | Uncorrectable.<br>ERR_NONFATAL<br>Log header | Send completion with CA |
| Unexpected Completion | • Received Completion Without a Request For It (Tag, ID, etc.) | Uncorrectable<br>ERR_NONFATAL<br>Log Header | Discard TLP |
| Receiver Overflow | • Received TLP Beyond Allocated Credits | Uncorrectable<br>ERR_FATAL | Receiver Behavior is Undefined |
| Flow Control Protocol Error | • Minimum Initial Flow Control Advertisements<br>• Flow Control Update for Infinite Credit Advertisement | Uncorrectable.<br>ERR_FATAL | Receiver Behavior is Undefined |
| Malformed TLP (MP) | • Data Payload Exceed Max_Payload_Size<br>• Received TLP Data Size Does Not Match Length Field<br>• TD field value does not correspond with the observed size<br>• PM Messages That Don't Use TC0.<br>• Usage of Unsupported VC | Uncorrectable<br>ERR_FATAL<br>Log Header | Drop the Packet, Free FC Credits |
| Completion with Unsuccessful Completion Status | | No Action (already done by originator of completion) | Free FC Credits |

### 3.1.7.3 Error Forwarding (TLP Poisoning)

If a TLP is received with an error-forwarding trailer, the packet is dropped and is not delivered to its destination. The 82599 then reacts as described in Table 3-8.

The 82599 does not initiate any additional master requests for that PCI function until it detects an internal software reset for the associated LAN port. Software is able to access device registers after such a fault.

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

### 3.1.7.4 End-to-End CRC (ECRC)

The 82599 supports ECRC as defined in the PCIe specification. The following functionality is provided:

- Inserting ECRC in all transmitted TLPs:
  - The 82599 indicates support for inserting ECRC in the *ECRC Generation Capable* bit of the PCIe configuration registers. This bit is loaded from the *ECRC Generation* EEPROM bit.
  - Inserting ECRC is enabled by the *ECRC Generation Enable* bit of the PCIe configuration registers.
- ECRC is checked on all incoming TLPs. A packet received with an ECRC error is dropped. Note that for completions, a completion timeout occurs later (if enabled), which results in re-issuing the request.
  - The 82599 indicates support for ECRC checking in the *ECRC Check Capable* bit of the PCIe configuration registers. This bit is loaded from the *ECRC Check* EEPROM bit.
  - Checking of ECRC is enabled by the *ECRC Check Enable* bit of the PCIe configuration registers.
- ECRC errors are reported
- System software can configure ECRC independently per each LAN function

### 3.1.7.5 Partial Read and Write Requests

#### Partial memory accesses

The 82599 has limited support of read and write requests with only part of the byte enable bits set:

- Partial writes with at least one byte enabled are silently dropped.
- Zero-length writes have no internal impact (nothing written, no effect such as clear-by-write). The transaction is treated as a successful operation (no error event).
- Partial reads with at least one byte enabled are handled as a full read. Any side effect of the full read (such as clear by read) is also applicable to partial reads.
- Zero-length reads generate a completion, but the register is not accessed and undefined data is returned.

**Note:** The 82599 does not generate an error indication in response to any of the previous events.

#### Partial I/O accesses

- Partial access on address
  - A write access is discarded
  - A read access returns 0xFFFF

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

### 3.1.7.6 Error Pollution

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

### 3.1.7.7 Completion With Unsuccessful Completion Status

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

### 3.1.7.8 Error Reporting Changes

The PCIe Rev. 1.1 specification defines two changes to advanced error reporting. A (new) *Role Based Error Reporting* bit in the Device Capabilities register is set to 1b to indicate that these changes are supported by the 82599.

1. Setting the *SERR# Enable* bit in the PCI Command register also enables UR reporting (in the same manner that the *SERR# Enable* bit enables reporting of correctable and uncorrectable errors). In other words, the *SERR# Enable* bit overrides the *Unsupported Request Error Reporting Enable* bit in the PCIe Device Control register.
2. Changes in the response to some uncorrectable non-fatal errors detected in non-posted requests to the 82599. These are called Advisory Non-Fatal Error cases. For each of the errors listed, the following behavior is defined:
   - — The *Advisory Non-Fatal Error Status* bit is set in the Correctable Error Status register to indicate the occurrence of the advisory error and the *Advisory Non-Fatal Error Mask* corresponding bit in the Correctable Error Mask register is checked to determine whether to proceed further with logging and signaling.
   - — If the *Advisory Non-Fatal Error Mask* bit is clear, logging proceeds by setting the corresponding bit in the Uncorrectable Error Status register, based upon the specific uncorrectable error that's being reported as an advisory error. If the corresponding *Uncorrectable Error* bit in the Uncorrectable Error Mask register is clear, the First Error Pointer and Header Log registers are updated to log the error, assuming they are not still occupied by a previous unserviced error.
   - — An ERR_COR Message is sent if the *Correctable Error Reporting Enable* bit is set in the Device Control register. An ERROR_NONFATAL message is not sent for this error.

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

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

## 3.1.8 Performance Monitoring

The 82599 incorporates PCIe performance monitoring counters to provide common capabilities to evaluate performance. The 82599 implements four 32-bit counters to correlate between concurrent measurements of events as well as the sample delay and interval timers. The four 32-bit counters can also operate in a two 64-bit mode to count long intervals or payloads. Software can reset, stop, or start the counters (all at the same time).

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

Counters operate in one of the following modes:

- Count mode — the counter increments when the respective event occurred
- Leaky bucket mode — the counter increments only when the rate of events exceeded a certain value. See Section 3.1.8.1 for more details.

The list of events supported by the 82599 and the counters *Control* bits are described in Section 8.2.3.3.

### 3.1.8.1 Leaky Bucket Mode

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

- One of four 16-bit Leaky Bucket Counters (LBC) is enabled via the *LBC Enable* [3:0] bits in the PCIe Statistic Control register #1.
- The LBC is controlled by the GIO_COUNT_START, GIO_COUNT_STOP, GIO_COUNT_RESET bits in the PCIe Statistic Control register #1.

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

# 3.2 SMBus

SMBus is a management interface for pass through and/or configuration traffic between an external Management Controller (MC) and the 82599.

## 3.2.1 Channel Behavior

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

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

## 3.2.2 SMBus Addressing

The number of SMBus addresses that the 82599 responds to depends on the LAN mode (teaming/non-teaming). If the LAN is in teaming mode (fail-over mode), the 82599 is presented over the SMBus as one device and has one SMBus address. If the LAN is in non-teaming mode, the SMBus is presented as two SMBus devices on the SMBus (two SMBus addresses). In dual-address mode, all pass through functionality is duplicated on the SMBus address, where each SMBus address is connected to a different LAN port.

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

The SMBus address method is defined through the *SMB Addressing Mode* bit in the EEPROM. The SMBus addresses are set using the *SMBus 0 Slave Address* and *SMBus 1 Slave Address* fields in the EEPROM.

**Note:** If single-address mode is selected, only the *SMBus 0 Slave Address* field is valid.

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

Besides the SMBus address values, all the previous parameters of the SMBus (SMBus channel selection, addressing mode, and address enable) can be set only through the EEPROM.

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

## 3.2.3 SMBus Notification Methods

The 82599 supports three methods of signaling the external MC that it has information that needs to be read by the external MC:

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

The notification method that is used by the 82599 can be configured from the SMBus using the Receive Enable command. The default method is set from the *Notification Method* field in the LRXEN1 word from the EEPROM.

The following events cause the 82599 to send a notification event to the external MC:

- Receiving a LAN packet designated for the MC.
- Receiving a Request Status command from the MC that initiates a status response.
- The 82599 is configured to notify the external MC upon status changes (by setting the EN_STA bit in the Receive Enable command) along with one of the following events:
  - TCO Command Aborted
  - Link Status changed
  - Power state change
  - LinkSec indication

There can be cases where the external MC is hung and cannot respond to the SMBus notification. The 82599 has a timeout value defined in the EEPROM (refer to Section 6.4.4.3) to avoid hanging while waiting for the notification response. If the MC does not respond until the timeout expires, the notification is de-asserted.

### 3.2.3.1 SMBus Alert and Alert Response Method

The SMBus Alert# signal is an additional SMBus signal, which acts as an asynchronous interrupt signal to an external SMBus master. The 82599 asserts this signal each time it has a message that it needs the external MC to read and if the chosen notification method is the SMBus alert method.

**Note:** SMBus Alert# is an open-drain signal, which means that other devices beside the 82599 can be connected to the same alert pin and the external MC requires a mechanism to distinguish between the alert sources as follows:

The external MC responds to the alert by issuing an ARA cycle to detect the alert source device. The 82599 responds to the ARA cycle (if it was the SMBus alert source) and de-asserts the alert when the ARA cycle completes. Following the ARA cycle, the MC issues a Read command to retrieve the the 82599 message.

**Note:** Some MCs do not implement the ARA cycle transaction. These MCs respond to an alert by issuing a Read command to the 82599 (0xC0/0xD0 or 0xDE). The 82599 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 82599 is the source of the SMBus alert, it replies to the read transaction.

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

**Note:** The ARA transaction does not support PEC.

The alert response address transaction format is as follows:

| 1 | 7 | 1 | 1 | 8 | 1 | 1 |
|---|---|---|---|---|---|---|
| S | Alert Response Address | Rd | A | Slave Device Address | A | P |
| | **0001 100** | **0** | **0** | | **1** | |

**Figure 3-7 SMBus ARA Cycle Format**

## 3.2.3.2 Asynchronous Notify Method

When configured using the asynchronous notify method, the 82599 acts as an SMBus master and notifies the external MC by issuing a modified form of the write word transaction. The asynchronous notify transaction SMBus address and data payload are configured using the Receive Enable command or by using the EEPROM defaults (see Section 6.4.3.19).

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

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

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

**Figure 3-8 Asynchronous Notify Command Format**

### 3.2.3.3 Direct Receive Method

If configured, the 82599 has the capability to send the message it needs to transfer to the external MC, as a master over the SMBus instead of alerting the MC and waiting for it to read the message.

The message format is shown Figure 3-9. Note that the command that should be used is the same command that should be used by the MC in the Block Read command and the opcode that the 82599 puts in the data is the same as it would have put in the Block Read command of the same functionality. The rules for the *F* an *L* flags are also the same as in the Block Read command.

| 1 | 7 | 1 | 1 | 1 | 1 | 6 | 1 | |
|---|---|---|---|---|---|---|---|---|
| S | Target Address | Wr | A | F | L | Command | A | • • • |
| | MC Slave Address | 0 | 0 | First Flag | Last Flag | Receive TCO Command 01 0000b | 0 | |

| 8 | 1 | 8 | 1 | | 1 | 8 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|
| Byte Count | A | Data Byte 1 | A | • • • | A | Data Byte N | A | P |
| N | 0 | | 0 | | 0 | | 0 | |

**Figure 3-9 Direct Receive Transaction Format**

## 3.2.4 Receive TCO Flow

The 82599 is used as a channel for receiving packets from the network link and passing them to the external MC. The MC can configure the 82599 to pass specific packets to the MC (see Section 10.2). Once a full packet is received from the link and identified as a manageability packet that should be transferred to the MC, the 82599 starts the receive TCO transaction flow to the MC.

The maximum SMBus fragment length is defined in the EEPROM (see Section 6.4.4.2). The 82599 uses the SMBus notification method to notify the MC that it has data to deliver. The packet is divided into fragments, where the 82599 uses the maximum fragment size allowed in each fragment. The last fragment of the packet transfer is always the status of the packet. As a result, the packet is transferred in at least two fragments. The data of the packet is transferred in the receive TCO LAN packet transaction.

When SMBus alert is selected as MC notification method, the 82599 notifies the MC on each fragment of a multi-fragment packet.

When asynchronous notify is selected as the MC notification method, the 82599 notifies the MC only on the first fragment of a received packet. It is the MC's responsibility to read the full packet including all the fragments.

Any timeout on the SMBus notification results in discarding of the entire packet. Any NACK by the MC on one of the 82599's receive bytes also causes the packet to be silently discarded.

Since SMBus throughput is lower than the network link throughput, the 82599 uses an 8 KB internal buffer per LAN port, which stores incoming packets prior to being sent over the SMBus interface. The 82599 services back-to-back management packets as long as the buffer does not overflow.

The maximum size of the received packet is limited by the 82599 hardware to 1536 bytes. Packets larger then 1536 bytes are silently discarded. Any packet smaller than 1536 bytes is processed by the 82599.

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

## 3.2.5 Transmit TCO Flow

The 82599 is used as a channel for transmitting packets from the external MC to the network link. The network packet is transferred from the external MC over the SMBus, and then, when fully received by the 82599, is transmitted over the network link.

In dual-address mode, each SMBus address is connected to a different LAN port. When a packet received in SMBus transactions using the *SMBus 0 Slave Address*, it is transmitted to the network using LAN port 0 and is transmitted through LAN port 1 if received on *SMBus 1 Slave Address*. In single-address mode, the transmitted port is chosen according to the fail-over algorithm (see Section 10.2.2.2).

The 82599 supports packets up to an Ethernet packet length of 1536 bytes. SMBus transactions can be up to 240 bytes in length, which means that packets can be transferred over the SMBus in more than one fragment. In each command byte there are the *F* and *L* bits. When the *F* bit is set, it means that this is the first fragment of the packet and *L* means that it is the last fragment of the packet (when both are set, it means that the entire packet is in one fragment). The packet is sent over the network link only after all its fragments have been received correctly over the SMBus.

The 82599 calculates the L2 CRC on the transmitted packet, and adds its four bytes at the end of the packet. Any other packet field (such as XSUM) must be calculated and inserted by the external MC controller (the 82599 does not change any field in the transmitted packet, besides adding padding and CRC bytes). If the packet sent by the MC is bigger than 1536 bytes, then the packet is silently discard by the 82599.

The minimum packet length defined by the 802.3 specification is 64 bytes. The 82599 pads packets that are less than 64 bytes to meet the specification requirements (no need for the MC to do it). There is one exception, that is if the packet sent over the SMBus is less than 32 bytes, the MC must pad it for at least 32 bytes. The passing bytes value should be zero. Packets which are smaller then 32 bytes (including padding) are silently discarded by the 82599.

If the network link is down when the 82599 has received the last fragment of the packet, it silently discards the packet.

**Note:** Any link down event while the packet is being transferred over the SMBus does not stop the operation, since the 82599 waits for the last fragment to end to see whether the network link is up again.

The transmit SMBus transaction is described in Section 10.5.2.1.

### 3.2.5.1 Transmit Errors in Sequence Handling

Once a packet is transferred over the SMBus from the MC to the 82599 the *F* and *L* flags should follow specific rules. The *F* flag defines that this is the first fragment of the packet, and the *L* flag defines that the transaction contains the last fragment of the packet.

Table 3-9 lists the different option of the flags in transmit packet transactions.

**Table 3-9 SMBus Transmit Sequencing**

| Previous | Current | Action/Notes |
|---|---|---|
| Last | First | Allowed – accept both. |
| Last | Not First | Error for current transaction. Current transaction is discarded and an abort status is asserted. |
| Not Last | First | Error for previous transaction. The previous transaction (until previous First) is discarded. The current packet is processed.<br>No abort status is asserted. |
| Not Last | Not First | The 82599 can 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.

### 3.2.5.2 TCO Command Aborted Flow

Bit 6 in first byte of the status returned from the 82599 to the external MC indicates that there was a problem with previous SMBus transactions or with the completion of the operation requested in previous transaction.

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

- Any error in the SMBus protocol (NACK, SMBus time outs).
- Any error in compatibility due to required protocols to specific functionality (RX Enable command with byte count not 1/14 as defined in the command specification).
- If the 82599 does not have space to store the transmit packet from the MC (in an internal buffer) before sending it to the link. In this case, all transactions are completed but the packet is discarded and the MC is notified through the *Abort* bit.
- Error in *F/L* bit sequence during multi-fragment transactions.
- The *Abort* bit is asserted after an internal reset to the 82599 manageability unit.

**Note:** The abort in the status does not always imply that the last transaction of the sequence was incorrect. There is a time delay between the time the status is read from the 82599 and the time the transaction has occurred.

## 3.2.6 Concurrent SMBus Transactions

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

## 3.2.7 SMBus ARP Functionality

The 82599 supports the SMBus ARP protocol as defined in the SMBus 2.0 specification. Note that the 82599 is a persistent slave address device each time its SMBus address is valid after power-up and loaded from the EEPROM. The 82599 supports all SMBus ARP commands defined in the SMBus specification, both general and directed.

**Note:** SMBus ARP can be disabled through EEPROM configuration (See Section 6.4.4.3).

### 3.2.7.1 SMBus ARP in Dual-/Single-Mode

The 82599 can operate in either single SMBus address mode or in dual SMBus address mode. These modes reflect on its SMBus-ARP behavior.

When working in single-address mode, the 82599 presents itself on the SMBus as one device, and responds to SMBus-ARP as only one device. In this case its SMBus address is SMBus address 0 as defined in EEPROM SMBus ARP addresses word (see Section 6.4.4.4). The device has only one Address Resolved (AR) and one Address Valid (AV) flag each. The vendor ID that is the Ethernet MAC address of the LAN's port, is taken from port 0 address.

In dual-address mode, the 82599 responds as two SMBus devices, meaning it has two sets of AR/AV flags (one for each port). The 82599 should respond twice to the SMBus-ARP master, one time for each port. Both SMBus addresses are taken from the SMBus ARP addresses word of the EEPROM. The Unique Device Identifier (UDID) is different between the two ports in the version ID field, which represent the Ethernet MAC address, which is different between the two ports. It is recommended for the 82599 to first answer as port 0, and only when the address is assigned, to answer as port 1 to the Get UDID command.

### 3.2.7.2 SMBus ARP Flow

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

- Address Valid — This flag is set when the 82599 has a valid SMBus address.
- Address Resolved — This flag is set when the 82599 SMBus address is resolved: SMBus address was assigned by the SMBus-ARP process.

**Note:** These flags are internal the 82599 flags and not shown to external SMBus devices.

Since the 82599 is a Persistent SMBus Address (PSA) device, the *AV* flag is always set, while the *AR* flag is cleared after power-up until the SMBus-ARP process completes. Since *AV* is always set, it means that the 82599 always has a valid SMBus address. The entire SMBus ARP Flow is described in Figure 3-10.

When the SMBus master needs to start the SMBus-ARP process, it resets (in terms of ARP functionality) all the devices on the SMBus, by issuing either Prepare to ARP or Reset Device commands. When the 82599 accepts one of these commands, it clears its *AR* flag (if set from previous SMBus-ARP process), but not its *AV* flag (The current SMBus address remains valid until the end of the SMBus ARP process).

A cleared *AR* flag means that the 82599 answers the following SMBus ARP transactions that are issued by the master. The SMBus master then issues a Get UDID command (General or Directed), to identify the devices on the SMBus. The 82599 responds to the Directed command all the time, and to the General command only if its *AR* flag is not set. After the Get UDID, the master assigns the 82599 SMBus address, by issuing Assign Address command. The 82599 checks whether the UDID matches its own UDID, and if there is a match it switches its SMBus address to the address assigned by the command (byte 17). After accepting the Assign Address command, the *AR* flag is set, and from this point (as long as the *AR* flag is set), the 82599 does not respond to the Get UDID General command, while all other commands should be processed even if the *AR* flag is set.

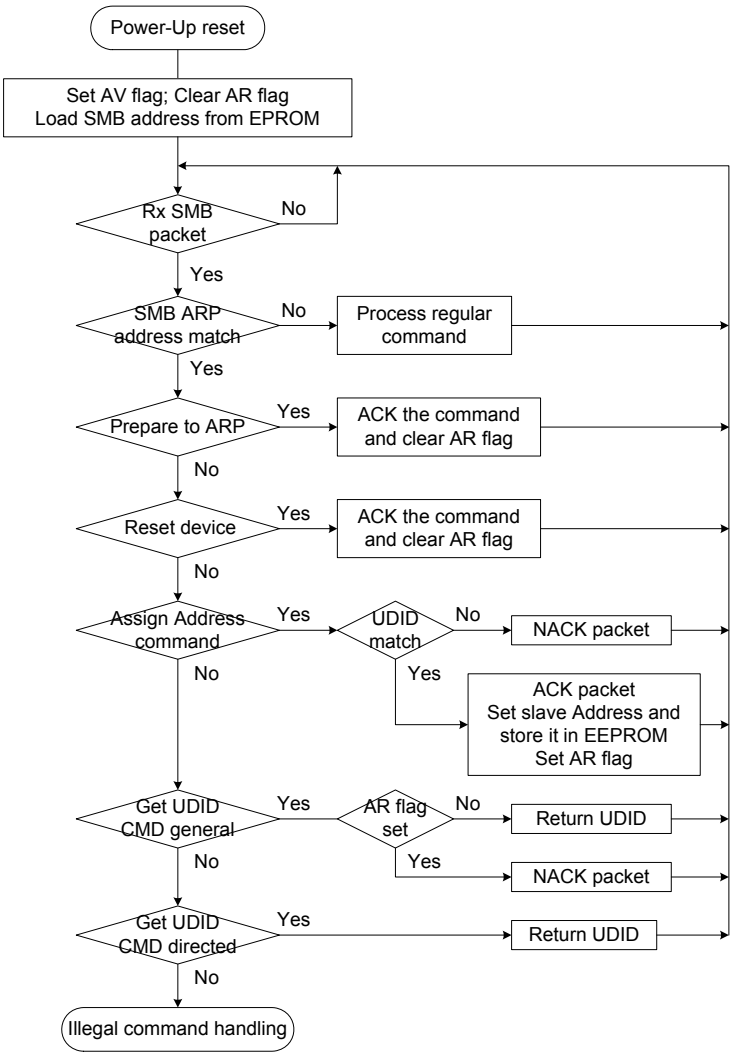



**Figure 3-10 SMBus ARP Flow**

#### 3.2.7.2.1 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:

| 1 Byte | 1 Byte | 2 Bytes | 2 Bytes | 2 Bytes | 2 Bytes | 2 Bytes | 4 Bytes |
|---|---|---|---|---|---|---|---|
| Device Capabilities | Version/ Revision | Vendor ID | Device ID | Interface | Subsystem Vendor ID | Subsystem Device ID | Vendor Specific ID |
| See as follows | See as follows | 0x8086<br>As reflected in the Device ID field in the PCI config space | As reflected in the Device ID field in the PCI config space | 0x0004 | 0x0000 | 0x0000 | See as follows |
| MSB | | | | | | | LSB |

Where:

- Interface: Identifies the protocol layer interfaces supported over the SMBus connection by the device.
  In this case, SMBus Version 2.0
  Constant value: 0x0004.
- Subsystem Fields: These fields are not supported and return zeros.

**Device Capabilities: Dynamic and Persistent Address, *PEC Support* bit**

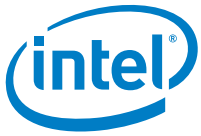
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Address Type | | Reserved (0) | Reserved (0) | Reserved (0) | Reserved (0) | Reserved (0) | PEC Supported |
| 0b | 1b | 0b | 0b | 0b | 0b | 0b | 0b |
| MSB | | | | | | | LSB |

**Version/Revision: UDID Version 1, Silicon Revision**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved (0) | Reserved (0) | UDID Version | | | Silicon Revision ID | | |
| 0b | 0b | 001b | | | See as follows | | |
| MSB | | | | | | | LSB |

**Silicon Revision ID:**

| Silicon Version | Revision ID |
|---|---|
| A0 | 000b |
| B0 | 001b |



**Vendor Specific ID:**

Four LSB bytes of the device Ethernet MAC address. The device Ethernet MAC address is taken from the EEPROM LAN Core 0/1 Modules in the EEPROM (see Section 6.3.7.2). Note that in the 82599 there are two Ethernet MAC Addresses (one for each port).

| 1 Byte | 1 Byte | 1 Byte | 1 Byte |
|---|---|---|---|
| Ethernet MAC Address, byte 3 | Ethernet MAC Address, byte 2 | Ethernet MAC Address, byte 1 | Ethernet MAC Address, byte 0 |
| MSB | | | LSB |

### 3.2.8 LAN Fail-Over Through SMBus

In fail-over mode, the 82599 determines which ports are used for transmit/reactive (according to the configuration). LAN fail-over is tied to the SMBus addressing mode. When the SMBus is dual-address mode, the 82599 does not activate its fail-over mechanism (it ignores the fail-over register), and operates in two individual LAN ports. When the SMBus is in single-address mode, in PT mode, the 82599 operates in fail-over mode as described in section Section 10.2.2.2.

## 3.3 Network Controller — Sideband Interface (NC-SI)

In the 82599, the NC-SI interface is connected to an external MC. Note that the 82599 NC-SI interface meets the NC-SI specification as a PHY-side device.

### 3.3.1 Electrical Characteristics

The 82599 complies with the electrical characteristics defined in the NC-SI specification.

The 82599 NC-SI behavior is configured by the 82599 on power-up:

- The output driver strength for the NC-SI output signals is configured by the EEPROM *RMM Out Buffer Strength* field (default = 0x1F).
- The NC-SI topology is loaded from the EEPROM (point-to-point or multi-drop with the default being point-to-point).

The 82599 dynamically drives its NC-SI output signals as required by the sideband protocol:

- On power up, the 82599 floats the NC-SI outputs.
- If the 82599 operates in point-to-point mode, then the 82599 starts driving the NC-SI outputs at some time following power up.
- If the 82599 operates in a multi-drop mode, the 82599 drives the NC-SI outputs as configured by the MC.

### 3.3.2 NC-SI Transactions

Compatible with the NC-SI specification.

## 3.4 EEPROM

### 3.4.1 General Overview

The 82599 uses an EEPROM device for storing product configuration information. The EEPROM is divided into three general regions:

Hardware accessed — loaded by the 82599 hardware after power-up, PCI reset de-assertion, D3 to D0 transition, or software reset. Different hardware sections in the EEPROM are loaded at different events. See further details on power-up and reset sequences in Section 4.0.

Firmware Area — Includes structures used by the firmware for management configuration in its different modes.

Software accessed — used by software only. The meaning of these registers as listed here is a convention for software only and is ignored by the 82599.

### 3.4.2 EEPROM Device

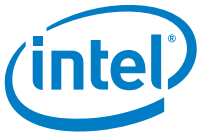
The EEPROM interface supports an SPI interface and. It expects the EEPROM to be capable of 5 MHz operation.

The 82599 is compatible with many sizes of 4-wire serial EEPROM devices. All EEPROM's are accessed in 16-bit data words only. For EEPROM size information, see Section 11.6.2.

The 82599 automatically determines the address size to be used with the SPI EEPROM it is connected to, and sets the EEPROM *Size* field of the EEPROM/FLASH Control and Data register (EEC.EE_ADDR_SIZE) field appropriately. Software can use this size to determine how to access the EEPROM. The exact size of the EEPROM is determined within one of the EEPROM words.

### 3.4.3 EEPROM Vital Content

The EEPROM contains several main types of vital content: pre-boot, pre-operating system and pre-driver parameters, manageability related structures, hardware default parameters, and driver default parameters. The 82599 must have the EEPROM to auto-load these settings.



## 3.4.4 Software Accesses

The 82599 provides two different methods for software access to the EEPROM.

- Use the built-in controller to read the EEPROM
- Access the EEPROM directly using the EEPROM's 4-wire interface

In addition, the Vital Product Data (VPD) area of the EEPROM can be accessed via the VPD capability structure of the PCIe.

Software can use the EEPROM Read (EERD) register to cause the 82599 to read a word from the EEPROM that the software can then use. To do this, software writes the address to read to the *Read Address* (EERD.ADDR) field and then simultaneously writes a 1b to the *Start Read* bit (EERD.START). The 82599 reads the word from the EEPROM, sets the *Read Done* bit (EERD.DONE), and puts the data in the *Read Data* field (EERD.DATA). Software can then poll the EEPROM Read register until it sees the *Read Done* bit set, then use the data from the *Read Data* field.

**Note:** Any words read this way are not written to the 82599's internal registers.

Software can also directly access the EEPROM's 4-wire interface through the EEPROM/Flash Control (EEC) register. It can use this for reads, writes, or other EEPROM operations.

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

1. Write a 1b to the *EEPROM Request* bit (EEC.EE_REQ).
2. Read the *EEPROM Grant* bit (EEC.EE_GNT) until it becomes 1b. It remains 0b as long as hardware is accessing the EEPROM.
3. Write or read the EEPROM using the direct access to the 4-wire interface as defined in the EEPROM/Flash Control and Data (EEC) register. The exact protocol used depends on the EEPROM placed on the board and can be found in the appropriate EEPROM datasheet.
4. Write a 0b to the *EEPROM Request* bit (EEC.EE_REQ).

**Note:** Each time the EEPROM is not valid (blank EEPROM or wrong signature), software should use the direct access to the EEPROM through the EEC register.

## 3.4.5 Signature Field

The only way the 82599 has to tell if an EEPROM is present is by trying to read the EEPROM. The 82599 first reads the EEPROM Control word at word address 0x000000 and at address 0x000800. It then checks the signature value at bits 7 and 6 in both addresses. If bit 7 is 0b and bit 6 is 1b in one of the two addresses, it considers the EEPROM to be present and valid. It then reads the additional EEPROM words and programs its internal registers based on the values read. Otherwise, it ignores the values it reads from that location and does not read any other words.

## 3.4.6 Protected EEPROM Space

The 82599 provides a mechanism for a hidden area in the EEPROM of the host. The hidden area cannot be read or write accessed via the EEPROM registers in the CSR space. It can be accessed only by the manageability subsystem.

After the EEPROM was configured to be protected, changing bits that are protected require specific manageability instructions with an authentication mechanism. This mechanism is defined in the firmware documentation.

### 3.4.6.1 Initial EEPROM Programming

In most applications, initial EEPROM programming is done directly on the EEPROM pins. Nevertheless, it is desired to enable existing software utilities (accessing the EEPROM via the host interface) to initially program the entire EEPROM without breaking the protection mechanism. Following a power up sequence, the 82599 reads the hardware initialization words in the EEPROM. If the signature in both word addresses 0x000000 and 0x000800 is not equal to 01b the EEPROM is assumed as non-programmed. There are two effects of a non-valid signature:

1. The 82599 does not read any further EEPROM data and sets the relevant registers to default.
2. The 82599 enables host write (and read) access to any location in the EEPROM via the EEPROM CSR registers.

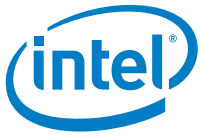
### 3.4.6.2 EEPROM Protected Areas

The 82599 defines two protected areas in the EEPROM. The first area is words 0x00-0x0F. These words hold the basic configuration and the pointers to all other configuration sections. The second area is a programmable size area located at the end of the EEPROM and assigned with protecting the appropriate sections that should be blocked for changes.

### 3.4.6.3 Activating the Protection Mechanism

Following a device initialization, the 82599 reads the Init control 1 word from the EEPROM sectors 0 and 1. The 82599 turns on the protection mechanism if this word contains a valid signature (equals to 01b) and bit 4 (EEPROM protection) is set to 1b. Once the protection mechanism is turned on, words 0x00-0x0F area become write-protected and the hidden area that is defined by word 0x0 becomes read/write protected to host access.

**Note:** Although possible by configuration, it is prohibited that the software sections in the EEPROM is included as part of the EEPROM protected area.



#### 3.4.6.4 Non Permitted Access to Protected Areas in the EEPROM

This section refers to EEPROM accesses by the host via the EEC (bit banging) or EERD (parallel read access) registers. Following a write access to the protected areas in the EEPROM (word 0x0 and the hidden area defined by word 0x0), hardware responds properly on the PCIe bus but does not initiate any access to the EEPROM. Following a read access to the hidden area in the EEPROM (as defined by word 0x0), hardware does not access the EEPROM and returns meaningless data to the host.

**Notes:** Using the bit banging access, the SPI EEPROM can be accessed in a burst mode by providing opcode, address, and then read or write data for multiple bytes. Hardware inhibits any attempt to access the protected EEPROM locations even in burst accesses.

Software should not access the EEPROM in a burst write mode starting in a non-protected area and continue to a protected one, or vice versa. In such a case, it is not guaranteed that the write access to the non-protected area takes place.

### 3.4.7 EEPROM Recovery

The EEPROM contains fields that if programmed incorrectly might affect the functionality of the 82599. The impact might range from an incorrect setting of some function (like LED programming), via disabling of entire features (such as no manageability) and link disconnection, to inability to access the device via the regular PCIe interface.

The 82599 implements a mechanism that enables recovery from a faulty EEPROM no matter what the impact is, using an SMBus message that instructs the firmware to invalidate the EEPROM.

This mechanism uses an SMBus message that the firmware is able to receive in all modes, no matter what is the content of the EEPROM. After receiving this message, firmware clears word 0x0 including the signature. Afterwards, the BIOS/operating system initiates a reset to force an EEPROM auto-load process that fails and enables access to the device.

Firmware is programmed to receive such a command only from PCIe reset until one of the functions changes its status from D0u to D0a. Once one of the functions moves to D0a it can be safely assumed that the device is accessible to the host and there is no further need for this function. This reduces the possibility of malicious software using this command as a back door and limits the time the firmware must be active in non-manageability mode.

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

| Function | Command | Data Byte |
|---|---|---|
| Release EEPROM | 0xC7 | 0xB6 |

**Notes:** This solution requires a controllable SMBus connection to the 82599.

In case more than one the 82599 is in a state to accept this solution, all of the the 82599 devices connected to the same SMBus accept the command. The devices in D0u state release the EEPROM.

After receiving a release EEPROM command, firmware should keep its current state. It is the responsibility of the programmer updating the EEPROM to send a firmware reset if required after the full EEPROM update process is done.

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

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

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

The most significant bit in Configuration address 2 indicates which port is the target of the access (0 or 1). The 82599 always enables the manageability block after power up. The manageability clock is stopped only if the manageability function is disabled in the EEPROM and one of the functions had transitioned to D0a; otherwise, the manageability block gets the clock and is able to wait for the new command.

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

## 3.4.8 EEPROM Deadlock Avoidance

The EEPROM is a shared resource between the following clients:

1. Hardware auto read.
2. LAN port 0 and LAN port 1 software accesses.
3. Manageability-firmware accesses.

When accessing the EEPROM, software and manageability-firmware should use the EEPROM parallel access. On this interface, hardware schedules the actual accesses to the EEPROM, avoiding starvation of any client. The bit banging interface does not guarantee fairness between the clients, therefore it should be avoided in nominal operation as much as possible. When write accesses to the EEPROM are required the software or manageability should access the EEPROM one word at a time releasing the interface after each word.

## 3.4.9 VPD Support

The EEPROM image can contain an area for VPD. This area is managed by the OEM vendor and does not influence the behavior of the hardware. Word 0x2F of the EEPROM image contains a pointer to the VPD area in the EEPROM. A value of 0xFFFF means VPD is not supported and the VPD capability does not appear in the configuration space.

The maximum area size is 256 bytes but can be smaller. The VPD block is built of a list of resources. A resource can be either large or small. The structure of these resources are listed in the following tables.

**Table 3-10 Small Resource Structure**

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

**Table 3-11 Large Resource Structure**

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

The 82599 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 82599 assumes the following VPD fields with the limitations listed in Table 3-12.

**Table 3-12 VPD Structure**

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

VPD structure limitations:

- The structure must start with a tag equal to 0x82. If the 82599 does not detect a value of 0x82 in the first byte of the VPD area or the structure does not follow the information listed inTable 3-12, it assumes the area is not programmed and the entire 256 bytes area is read only.
- The RO area and R/W area are both optional and can appear in any order. A single area is supported per tag type. See PCI 3.0 specification Appendix I for details of the different tags.
- If a VPD-W tag is found, the area defined by its size is writable via the VPD structure.

- Both read and write sections on the VPD area must be Dword aligned (for example, each tag must start on Dword boundaries, and each data field must end on Dword boundary). Write accesses to Dwords that are only partially in the R/W area are ignored. VPD software is responsible to make the right alignment to enable a write to the entire area.
- The structure must end with a tag equal to 0x78. The tag must be word aligned.
- The VPD area is accessible for read and write via the regular EEPROM mechanisms pending the EEPROM protection capabilities enabled. The VPD area can be accessed through the PCIe configuration space VPD capability structure listed in Table 3-12. Write accesses to a read only area or any accesses outside of the VPD area via this structure are ignored.

# 3.5 Flash

The 82599 provides an interface to an external serial Flash/ROM memory device. This Flash/ROM device can be mapped into memory space for each LAN device through the use of Base Address Registers (BARs). EEPROM bits associated with each LAN device selectively disable/enable whether the Flash can be mapped for each LAN device by controlling the BAR register advertisement and write ability.

## 3.5.1 Flash Interface Operation

The 82599 provides two different methods for software access to the Flash.

Using the legacy Flash transactions the Flash is read from, or written to, each time the host CPU performs a read or a write operation to a memory location that is within the Flash address mapping or upon boot via accesses in the space indicated by the Expansion ROM Base Address register. All accesses to the Flash require the appropriate command sequence for the device used. Refer to the specific Flash data sheet for more details on reading from or writing to Flash. 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 82599 controls accesses to the Flash when it decodes a valid access.

**Note:** Flash read accesses must always be assembled by the 82599 each time the access is greater than a byte-wide access.

The 82599 byte reads or writes to the Flash take about 2 µs time. The device continues to issue retry accesses during this time.

The 82599 supports only byte writes to the Flash.

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

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

1. Write a 1b to the *Flash Request* bit (FLA.FL_REQ).
2. Read the *Flash Grant* bit (FLA.FL_GNT) until it becomes 1b. It remains 0b as long as there are other accesses to the Flash.
3. Write or read the Flash using the direct access to the 4-wire interface as defined in the FLA register. The exact protocol used depends on the Flash placed on the board and can be found in the appropriate Flash datasheet.
4. Write a 0b to the *Flash Request* bit (FLA.FL_REQ).

## 3.5.2 Flash Write Control

The Flash is write controlled by the bits in the EEPROM/FLASH Control and Data (EEC.FWE) register. Note that attempts to write to the Flash device when writes are disable (FWE=01b) should not be attempted. Behavior after such an operation is undefined, and might result in component and/or system hangs.

After sending one byte to write to the Flash, software can check if it can send the next byte to write (check if the write process in the Flash had finished) by reading the FLA register. If bit (FLA.FL_BUSY) in this register is set, the current write did not finish. If bit (FLA.FL_BUSY) is cleared, then software can continue and write the next byte to the Flash.

## 3.5.3 Flash Erase Control

When software needs to erase the Flash it should set the FLA.FL_ER bit in the FLA register to 1b (Flash erase and set bits EEC.FWE in the EEPROM/Flash Control register to 0b).

Hardware gets this command and sends the Erase command to the Flash. The erase process then finishes by itself. Software should wait for the end of the erase process before any further access to the Flash. This can be checked by using the Flash Write control mechanism previously described (see Section 3.5.2).

The opcode used for the erase operation is defined in the FLOP register.

**Note:** Sector erase by software is not supported. To delete a sector, the serial (bit bang) interface should be used.

## 3.5.4 Flash Access Contention

The 82599 implements internal arbitration between Flash accesses initiated through the LAN 0 device and those initiated through the LAN 1 device. If accesses from both LAN devices are initiated during the same approximate size window, The first one is served first and only then the next one. Note that the 82599 does not synchronize between the two entities accessing the Flash though contentions caused from one entity reading and the other modifying the same locations is possible.

To avoid this contention, accesses from both LAN devices should be synchronized using external software synchronization of the memory or I/O transactions responsible for the access. It is possible to ensure contention-avoidance simply by the nature of the software sequence.

# 3.6 Configurable I/O Pins — Software-Definable Pins (SDP)

The 82599 has eight Software-Defined Pins (SDP pins) per port that can be used for miscellaneous hardware or software-controllable purposes (see Figure 3-11). 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 eight pins are defined as input pins.

Some SDP pins have specific functionality:

- The default direction of the lower SDP pins (SDP0-SDP3) is loaded from the SDP Control word in the EEPROM.
- The lower SDP pins (SDP0-SDP3) 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.

Certain SDP pins can be allocated to hardware functions. For example SDP2, SDP3, SDP6 and SDP7 can be defined to support IEEE1588 auxiliary devices. In addition, the functionality of the I/O pins are programmed by the TimeSync Auxiliary Control (TSAUXC) register.

Table 3-13 defines an **example** of possible usage of SDP I/O pins, MDIO pins, and $I^2C$ pins as a function of an optical module or the PHY being interfaced. 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 by setting the *SDPxxx_NATIVE* bits in the ESDP register. Native mode in those SDP I/O pins designed for PHY and optical module specific usages, defines the pin functionality while in an inactive state (reset or power down) while behavior in an active state is controlled by software. The hardware functionality of these SDP I/O pins differ mainly by the active behavior controlled by software.

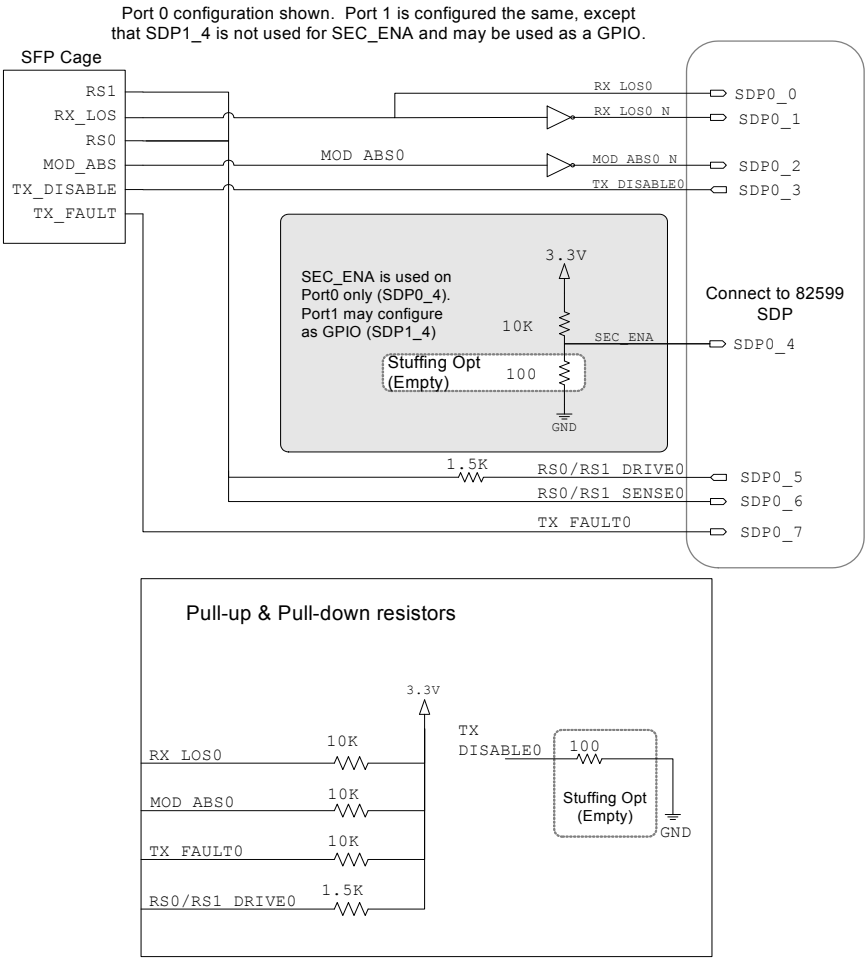
**Table 3-13 Example for SDP, MDIO and $I^2C$ Ports Usage**

| | SFP+ | Reserved | Copper PHY | X2/XPAK[1] |
|---|---|---|---|---|
| SDP0 | GPIO: RX_LOS | | GPIO: INTR_L | GPIO: LASI |
| SDP1 | GPIO: RX_LOS_N | | | |
| SDP2 | GPIO: MOD_ABS_N | | | |
| SDP3 | GPIO: TX_DISABLE | | NATIVE: TS_SDP3 | NATIVE: TS_SDP3 |
| SDP4 Port 0 | IN: SEC_ENA | | IN: SEC_ENA | IN: SEC_ENA |

## Table 3-13 Example for SDP, MDIO and I²C Ports Usage (Continued)

| | SFP+ | Reserved | Copper PHY | X2/XPAK[1] |
|---|---|---|---|---|
| SDP4 Port 1 | GPIO | | GPIO | GPIO |
| SDP5 | NATIVE: RS0/RS1 drive | | NATIVE: RESET | NATIVE: TX ON/OFF[2] |
| SDP6 | GPIO: RS0/RS1 sense | | NATIVE: TS_SDP6 | GPIO: RESET_N |
| SDP7 | GPIO: TX_FAULT | | NATIVE: TS_SDP7 | NATIVE: TS_SDP7 |
| MDIO | | | MDIO | MDIO |
| I²C | I²C | | | |

1. To Support XENPAK, X2 or XPAK modules, 3.3V to 1.2V level shifters are required between the 82599 and an optical module.
2. When TX ON/OFF is low in XENPAK, X2 and XPAK modules transmission is disabled. The *SDP5_Function* bit in the ESDP register should be set to 0b enabling the pin to be at a HiZ state while the 82599 is in an inactive state (as defined in the register). Board designers should populate with an external pull-down resistor forcing a low level during an inactive state.

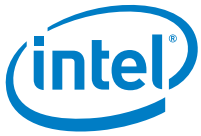



## Figure 3-11 SDP Connections

Table 3-14 lists the signals defined in Table 3-13 and behavior during reset and power down state (D3) without management.

### Table 3-14 SDP Assigned Signals Description

| Signal | Description | Software I/O Programming | Default Values at (Reset, D3 no WoL and no MNG) |
|---|---|---|---|
| RX_LOS, RX_LOS_N | RX_LOS high and RX_LOS_N low indicate insufficient optical power for reliable signal reception. | GPIO: Input | Input, no change. |
| LASI, INTR_L | INTR_L or Link Alarm Status Interrupt (LASI) — when low, indicates possible module operational fault or a status critical to the host system. | GPIO: Input (Interrupt) | Input, no change. |
| RS0/RS1 drive | Short-circuit protected. | Native: Output | Output, autonomous high or tri-state with pull-up. |
| RS0/RS1 sense | Directly connected input. | GPIO: Input | Input, no change. |
| P_DOWN/RST | When held high by the host, places the module in standby (low power) mode.<br>The negative edge of P_DOWN/RST signal initiates a complete module reset. | GPIO: Output | Input, no change. In order to minimize PHY power, software should drive the SDP to high or set to input while populating a pull-up. |
| RESET_N | When low, XENPAK, X2 or XPAK optical module is reset. | GPIO: Output | Output, no change. In order to minimize PHY power software should drive the SDP to low or set to input while populating a pull-down. |
| RESET | When high, the copper PHY is reset. | Native: Output | Output, autonomous high or tri-state with pull-up. |
| TX_DISABLE | When TX_DISABLE is asserted high, optical module transmitter is turned off. | GPIO: Output | Output, no change. In order to minimize PHY power software should drive the SDP to high or set to input while populating a pull-up. |
| TX_DIS | When TX_DIS is asserted high, optical module transmitter is turned off. | Native: Output | Output, autonomous high or tri-state with pull-up. |
| TX ON/OFF | 1b = Transmitter on.<br>0b = Transmitter off. | Native: Output | Output, autonomous low or tri-state with pull-down. |
| MOD_DET_N | Inverted mode detect. | GPIO: Input (Interrupt) | Input, no change. |
| TS_SDPX | Time sync support pins, can be used as event in or event out. | Native: According to programmed functionality | Tri-state during reset. No change in D3. External pull-up / pull-down as required by the system designer. |
| TX_FAULT | When high, indicates that the module transmitter has detected a fault condition related to laser operation or safety. | GPIO: Input (Interrupt) | Input, no change. |
| MOD_NR | When high, indicates that the module has detected a condition that renders transmitter and or receiver data invalid. | GPIO: Input (Interrupt) | Input, no change. |
| MOD_ABS | | GPIO: Input (Interrupt) | Input, no change. |
| FAN_Status | Optional health indication of the fan. | GPIO: Input (Interrupt) | Input, no change. |



# 3.7 Network Interface (MAUI Interface)

The 82599 supports 10 GbE operation, 1 GbE operation and 100 Mb/s Ethernet operation on the MAUI interface. The 82599 can support different or the same link speeds (10 Gb/s, 1 Gb/s and 100 Mb/s) and protocols on each of the two MAUI ports. The 82599 also supports automatic crossover and polarity correction on each of the MAUI ports to eliminate the need for crossover cables between similar devices. The 82599 also supports auto-negotiation when configured for backplane Ethernet to automatically select between KX, KX4 and KR.

When in 10 GbE operating mode, the MAUI interface can be configured as any of the following:

- A four lane XAUI interface.
- A four lane 10GBASE-BX4 interface.
- A four lane 10GBASE-KX4 interface.
- A four lane 10GBASE-CX4 interface.
- A single lane 10GBASE-KR interface.
- A single lane SFI interface.

When in 1 GbE operating mode, the MAUI interface can be configured as any of the following:

- A single lane 1000BASE-KX interface.
- A single lane 1000BASE-BX interface.
- A single SGMII (1 Gb/s or 100 Mb/s) lane over a KX or BX compliant electrical interface.

The device implements all features required for transmission and reception defined for the XAUI, BX4, CX4, KX4, KX, KR, SFI and BX Media interface. The MAUI interface supports the IEEE 802.3ae
(10 GbE — XAUI), IEEE 802.3ap (KX, KX4 and KR), IEEE802.3ak (10GBASE-CX4), PICMG3.1 (1000BASE-BX and 10GBASE-BX4), and SFI standards.

In 10 GbE BX4, KX4, CX4 or XAUI operating modes, data passes on all four MAUI lanes complying with the BX4, KX4, CX4 or XAUI protocol. In 10GBASE-KR, SFI, SGMII, 1000BASE-KX, 1000BASE-BX, or 10GBASE-BX4 operation, data passes on MAUI lane 0 complying with the 10 GbE KR, SFI protocols, the 1 GbE KX or BX protocols or the 100 Mb/s and 1 GbE SGMII protocol over a KX or BX electrical interface.

## 3.7.1 10 GbE Interface

The 82599 provides complete functionality to support up to two 10 Gb/s ports. The device 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 82599 enables 10 GbE operation compliant to the XAUI, CX4, KX4, KR, SFI specifications by programming the appropriate bits in the AUTOC register.

### 3.7.1.1 XAUI Operating Mode

The Ten Gigabit Attachment Unit Interface (XAUI) supports data rates of 10 Gb/s over four differential paths in each direction for a total of eight pairs, with each path operating at 3.125 Gb/s. The interface is used to connect the 82599 to an external 10 GbE PHY device with a XAUI interface. XAUI operating mode can be forced by software by setting the relevant bits in the AUTOC register and disabling auto-negotiation (see Section 3.7.4.2).

#### 3.7.1.1.1 XAUI Overview

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

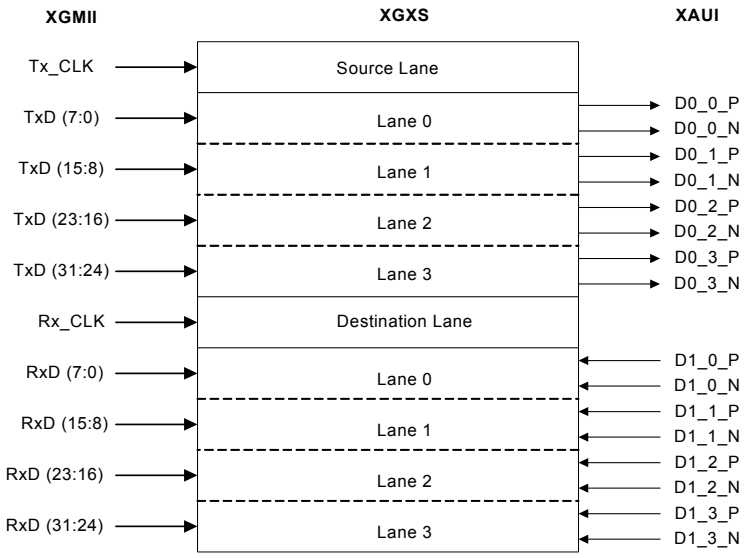



**Figure 3-12 XGMII to XAUI at the XGXS**

The XAUI interface has the following characteristics:

a. Simple signal mapping to the XGMII.

b. Independent transmit and receive data paths.

c. Four lanes conveying the XGMII 32-bit data and control.

d. Differential signaling with low voltage swing.

e. Self-timed interface enables jitter control to the PCS.

f. Using 8B/10B coding.

Figure 3-13 shows the architectural positioning of XAUI.

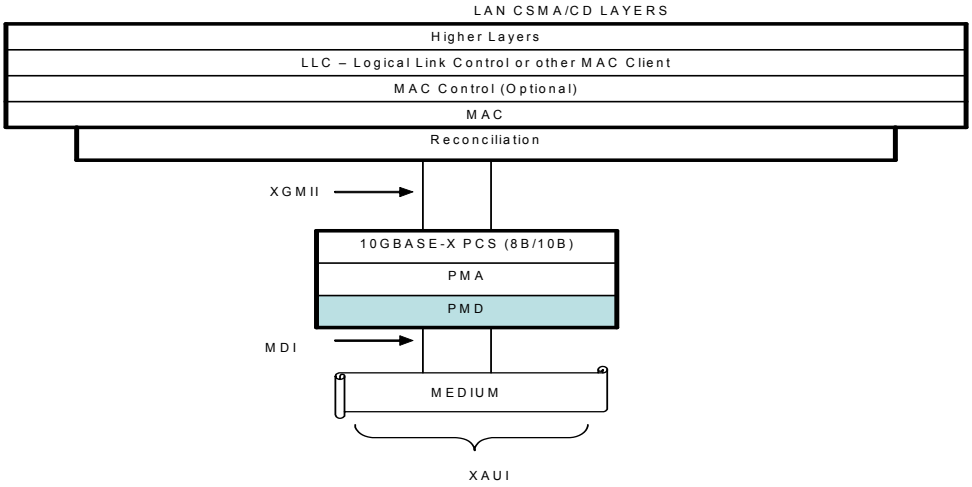



**Figure 3-13 Architectural Positioning of XAUI**

### 3.7.1.1.2 XAUI Operation

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

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

1. The XGMII is organized into four lanes with each lane conveying a data octet or control character on each edge of the associated clock. The source XGXS converts bytes on an XGMII lane into a self clocked, serial, 8B/10B encoded data stream. Each of the four XGMII lanes is transmitted across one of the four XAUI lanes.
2. The source XGXS converts XGMII Idle control characters (inter-frame) into an 8B/10B code sequence.
3. The destination XGXS recovers clock and data from each XAUI lane and de-skews the four XAUI lanes into the single-clock XGMII.
4. The destination XGXS adds to or deletes from the inter-frame gap as needed for clock rate disparity compensation prior to converting the inter-frame code sequence back into XGMII idle control characters.
5. The XGXS uses the same code and coding rules as the 10GBASE-X PCS and PMA specified in IEEE 802.3 Clause 48.

#### 3.7.1.1.3 XAUI Electrical Characteristics

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

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

### 3.7.1.2 10GBASE-KX4 Operating Mode

The KX4 interface supports data rates of 10 Gb/s over copper traces in improved FR4 PCBs. Data is transferred over four differential paths in each direction for a total of eight pairs, with each path operating at 3.125Gbaud to support overhead of 8B/10B coding. The interface is used to connect the 82599 to a KX4 switch port over the backplane or to an external 10 GbE PHY device with a KX4 interface.

The MAUI interface is configured as a KX4 interface while auto-negotiation to a KX4 link partner is detected. KX4 operation can also be forced by EEPROM or software by setting the relevant bits in the AUTOC register and disabling auto-negotiation (see Section 3.7.4.2).

#### 3.7.1.2.1 KX4 Overview

10GBASE-KX4 definition is based on XAUI with 10GBASE-CX4 extensions and specifies 10 Gb/s operation over four differential paths in each direction for a total of eight pairs, or 16 connections. This system uses the 10GBASE-X PCS and PMA as defined in IEEE802.3 Clause 48 with amendments for auto-negotiation as specified in IEEE802.3ap. The 10GBASE-KX4 PMD is defined in IEEE802.3ap Clause 71.

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

Figure 3-14 shows the architectural positioning of 10GBASE-KX4.

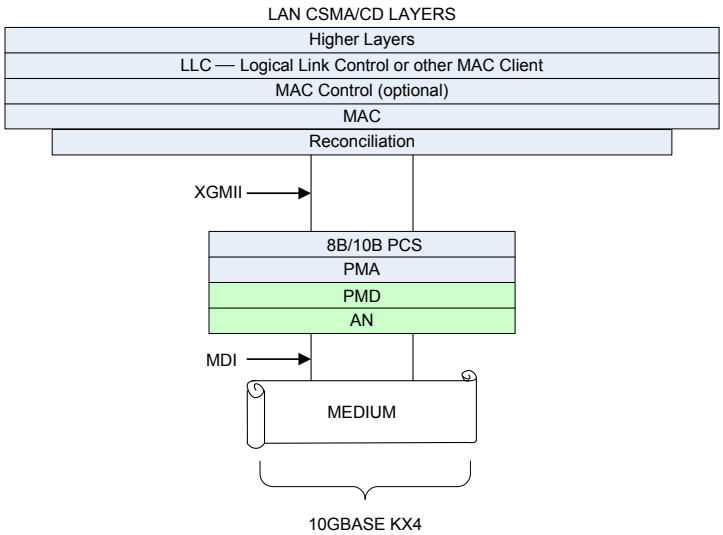



**Figure 3-14 Architectural Positioning of 10GBASE-KX4**

#### 3.7.1.2.2 KX4 Electrical Characteristics

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

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

### 3.7.1.3 10GBASE-KR Operating Mode

The KR interface supports data rates of 10 Gb/s over copper traces in improved FR4 PCBs. Data is transferred over a single differential path in each direction for a total of two pairs, with each path operating at 10.3125 Gbaud ± 100 ppm to support overhead of 64B/66B coding. The interface is used to connect the 82599 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 EEPROM or software by setting the relevant bits in the AUTOC register and disabling auto-negotiation (see Section 3.7.4.2). When in 10GBASE-KR operating mode, MAUI lane 0 is used for receive and transmit activity while lanes 1 to 3 of the MAUI interface are powered down.

#### 3.7.1.3.1 KR Overview

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

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

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

Figure 3-15 shows the architectural positioning of 10GBASE-KR.

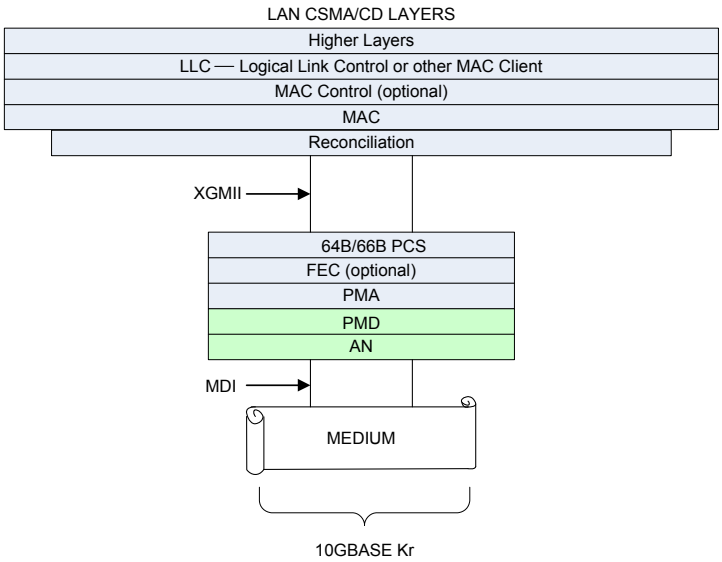



**Figure 3-15 Architectural Positioning of 10GBASE-KR**

#### 3.7.1.3.2 KR Electrical Characteristics

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

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

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

#### 3.7.1.3.3 KR Reverse Polarity

The 82599 supports reverse polarity of the KR transmit and receive lanes. It is enabled by the following EEPROM setting in the Core 0/1 Analog Configuration Modules:

**Reverse Tx polarity setting:**

| EEPROM Word Offset (Starting at Odd Word) | Reserved | KR / SFI Reverse Polarity | Description |
|---|---|---|---|
| 2*N+1 | | 0x0101 | Set page 1. |
| 2*N+2 | | 0x1E12 | Write register 0x1E the data 0x12 to invert Tx polarity. |

**Reverse Rx polarity setting**

| EEPROM Word Offset (Starting at Odd Word) | Reserved | KR / SFI Reverse Polarity | Description |
|---|---|---|---|
| 2*N+1 | | 0x0101 | Set page 1. |
| 2*N+2 | | 0x1FC0 | Write register 0x1F the data 0xC0 to invert Rx polarity. |

### 3.7.1.4 10GBASE-CX4 Operating Mode

The CX4 interface supports data rates of 10 Gb/s over twinaxial cable. Data is transferred over four differential paths in each direction for a total of eight pairs, with each path operating at 3.125Gbaud to support overhead of 8B/10B coding. The interface is used to connect the 82599 to a CX4 switch. CX4 operation can be forced by EEPROM or software by setting the relevant bits in the AUTOC register and disabling auto-negotiation (see Section 3.7.4.2).

### 3.7.1.4.1 CX4 Overview

10GBASE-CX4 definition specifies 10 Gb/s operation over four differential paths in each direction for a total of eight pairs, or 16 connections. This system uses the 10GBASE-X PCS and PMA as defined in IEEE802.3 Clause 48. The 10GBASE-CX4 PMD is defined in IEEE802.3 Clause 54.

CX4 is a full-duplex interface that uses four self-clocked serial differential links in each direction to achieve 10 Gb/s data throughput. Each serial link operates at 3.125 Gbaud to accommodate both data and the overhead associated with 8B/10B coding. The self-clocked nature eliminates skew concerns between clock and data.

Figure 3-16 shows the architectural positioning of 10GBASE-CX4.

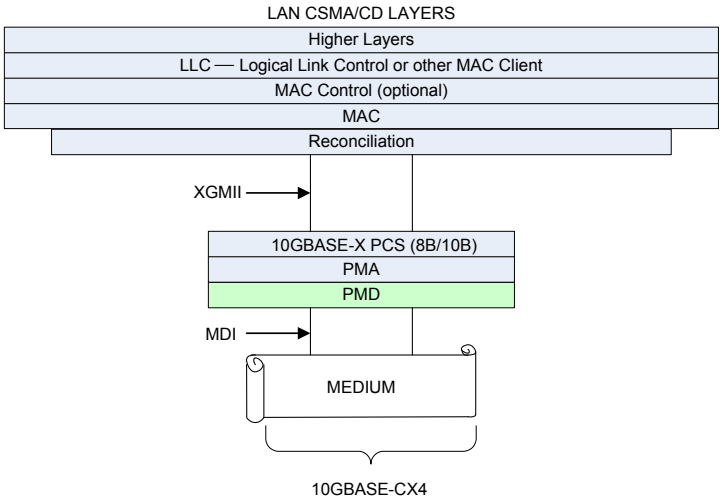



**Figure 3-16 Architectural Positioning of 10GBASE-CX4**

### 3.7.1.4.2 CX4 Electrical Characteristics

The CX4 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 CX4 signal paths are point-to-point connections. Each path corresponds to a CX4 lane and is comprised of two complementary signals making a balanced differential pair. There are four differential paths in each direction for a total of eight pairs, or 16 connections. The signal paths are intended to operate on twinaxial cable assemblies up to 15 m in length.

### 3.7.1.5 10GBASE-BX4 Operating Mode

10 GbE is supported within PICMG 3.1 by adopting a subset of the IEEE 802.3 XAUI specifications. Where XAUI is a chip-to-chip interface between test points TP-1 and TP-4, the PICMG 3.1 specifies what goes into the backplane at TP-T and what comes out of the backplane at TP-R. When implementing a
10 Gb/s PICMG 3.1 channel, board designers must implement this channel with compliant TP-T and TP-R test points.

**Note:** The channel-to-channel skew is handled by the XAUI protocol.

#### 3.7.1.5.1 10GBASE-BX4 Electrical Characteristics

Transmitted Electrical Specifications at TP-1:

PICMG 3.1 specifies the compliance point TP-T. System designers are required to implement additional margin at TP-1 to ensure compliance at TP-T.

The impedance at termination must be 100 Ω ± 10%.

Transmitted Electrical Specifications at TP-T:

The PICMG 3.1 drive levels into the backplane must conform to the following specifications as listed in Table 3-15.

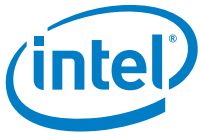
**Table 3-15 Transmit Specifications at TP-T**

| Parameter | Value | Units |
|---|---|---|
| Baud rate | 3.125 | GBd |
| Clock tolerance | ± 100 | ppm |
| Differential amplitude maximum | 1600 | mV p-p |
| Absolute output voltage limits | -0.4 min, 1.6 max | V |
| Differential output return loss | See footnote[1] | dB |
| Output jitter | | |
| Near-end maximums (TP-T) | | |
| Total jitter | ± 0.075 peak from the mean | UI |
| Deterministic jitter | ± 0.085 peak from the mean | UI |

1. s11 = –10 dB for 312.5 MHz < Freq (f) < 625 MHz, and –10 + 10log(f/625) dB for 625 MHz <= Freq (f) = < 3.125 GHz; where f is frequency in MHz.

**Note:** All measurements are made through a mated pair connector.

To maintain inter-operability between older and newer technologies and to avoid damage to the components, the maximum drive amplitude of any PICMG 3.1 driver must not exceed 1600 mV P-P.



The output impedance requirement applies to all valid output levels. The reference impedance for differential return loss measurements is 100 Ω.

Receiver Electrical Specifications at TP-R:

Table 3-16 lists the receiver specifications at TP-R.

**Table 3-16 Receiver Specifications at TP-R**

| Parameter | Value | Units |
|---|---|---|
| Baud rate | 3.125 | GBd |
| Clock tolerance | ± 100 | ppm |
| Differential return loss | 10 | dB |
| Common mode return loss | 6 | dB |
| Jitter amplitude tolerance (p-p) | 0.65 | UI |
| Differential skew | 75 | ps |

Receiver input impedance must result in a differential return loss better than 10 dB and a common mode return loss better than 6 dB from 100 MHz to 2.5 GHz. This includes contributions from all components related to the receiver including coupling components. The return loss reference impedance is 100 Ω for differential return loss and 25 Ω for common mode.

Receiver Electrical Specifications at TP-4:

PICMG 3.1 specifies the compliance point TP-R. System designers are required to ensure the additional losses to TP-4 are accounted for.

The AC coupling capacitors at the receiver must be no more than 470 pF +1% and matched within 2% with each other.

A 10GBASE-BX4 interface between two GbE ports is shown in Figure 3-17.

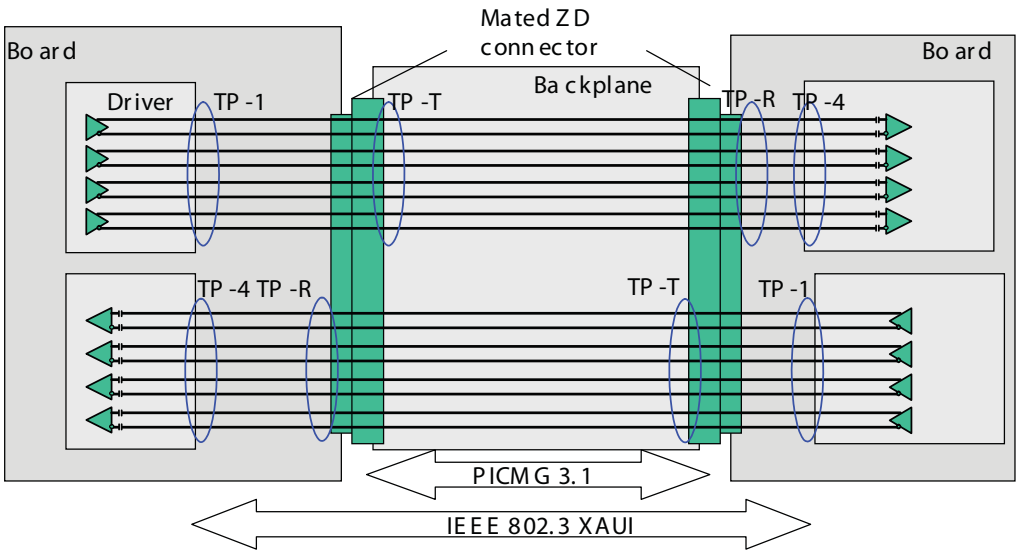



**Figure 3-17 10GBASE-BX4 Electrical Environment**

## 3.7.1.6 SFI Operating Mode

The MAUI interface is configured as SFI by EEPROM or software by setting the relevant bits in the AUTOC register and disabling auto-negotiation (see Section 3.7.4.2). When in SFI operating mode, only the operation of the 82599 Analog Front End (AFE) is modified, while the rest of the 82599 logic and circuitry operates similar to 10GBASE-KR. When in SFI operating mode, MAUI lane 0 is used for receive and transmit activity while lanes 1 to 3 of the MAUI interface are powered down.

### 3.7.1.6.1 SFI Overview

SFI definition enables 10 Gb/s operation over a single differential path in each direction for a total of two pairs, or four connections. When in SFI operating mode the 82599 uses the 10GBASE-R PCS and 10 Gigabit PMA as defined in IEEE802.3 Clause 49 and 51, respectively.

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

#### 3.7.1.6.2 SFI Electrical Characteristics

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

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

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

## 3.7.2 GbE Interface

The 82599 provides complete support for up to two 1 Gb/s port implementations. The device performs all functions required for transmission and reception defined by the different standards.

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

When operating in 1 GbE operation mode, the 82599 uses Lane 0 of the XAUI interface for 1 GbE operation while the other three XAUI lanes are powered down.

The 82599 enables 1 GbE operation compliant with the KX, BX or SGMII specifications by programming the appropriate bits in the AUTOC register.

### 3.7.2.1 1000BASE-KX Operating Mode

The MAUI interface, when operating as a KX Interface, supports data rates of 1 Gb/s over copper traces on improved FR4 PCBs. Data is transferred over a single differential path in each direction for a total of two pairs (Lane 0 of MAUI interface and Lanes 1 to 3 powered down), with each path operating at 1.25 Gbaud to support overhead of 8B/10B coding. The interface is used to connect the 82599 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 IEEE802.3ap clause 73 ending with 1 Gb/s as the HCD, the MAUI interface is configured as a KX interface. KX operating mode can also be forced by software by setting the relevant bits in the AUTOC register and disabling auto-negotiation (see Section 3.7.4.2).

#### 3.7.2.1.1 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 together with the amendments placed in IEEE802.3ap. The 1000BASE-KX PMD is defined in IEEE802.3ap Clause 70.

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

Figure 3-18 shows the architecture positioning of 1000BASE-KX.

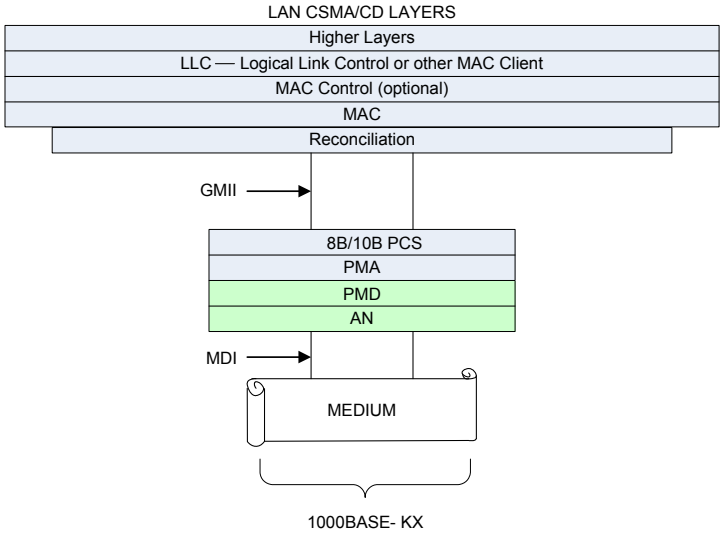
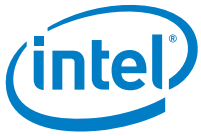



**Figure 3-18 Architectural Positioning of 1000BASE-KX**

#### 3.7.2.1.2 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. The signal paths are intended to operate up to approximately one meter over controlled impedance traces on improved FR4 PCBs.



#### 3.7.2.2 1000BASE-BX Operating Mode

1000BASE-BX is the PICMG 3.1 electrical specification for transmission of 1 Gb/s Ethernet encoded data over a 100 Ω differential backplane. The 1000BASE-BX standard defines a full-duplex interface that uses a single serial differential link in each direction (one pair for receive and one for transmit) 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. BX operating mode can be forced by software by setting the relevant bits in the AUTOC register and disabling auto-negotiation (see Section 3.7.4.2).

##### 3.7.2.2.1 BX Electrical Characteristics

The BX lane is a low swing AC coupled differential interface. 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 BX signal paths are point-to-point connections. Each path corresponds to a BX 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.

### 3.7.3 SGMII Support

The 82599 supports 1 Gb/s and 100 Mb/s operation using the SGMII protocol over the KX and BX electrical interface (AC coupling, no source synchronous TX clock, etc.).

#### 3.7.3.1 SGMII Overview

SGMII interface supported by the 82599 enables operation at 1 Gb/s over two differential, controlled impedance pairs of traces (one pair for transmit, one pair for receive). When operating in SGMII, the MAUI interface uses the 1000BASE-X PCS and PMA as defined in IEEE802.3 Clause 36 and the 1000BASE-KX PMD as defined in IEEE802.3ap Clause 70 or the 1000BASE-BX as defined in the PCIMG 3.1 standard. In SGMII operating mode, the MAUI interface can support data rates of 1 Gb/s and 100 Mb/s.

SGMII, supported by the 82599, is a full-duplex interface that uses a single serial differential link in each direction to achieve 1 Gb/s data throughput. Each serial link operates at 1.25 GBaud to accommodate both data and the overhead associated with 8B/10B coding. The self-clocked nature eliminates skew concerns between clock and data.

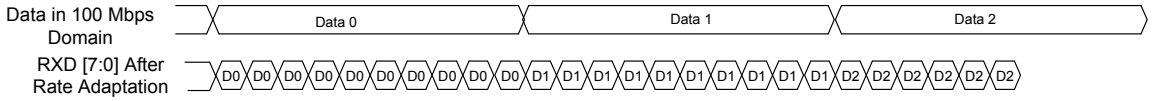
SGMII control information, as listed in Table 3-17 is transferred from the PHY to the MAC to signal change of link speed (100 Mb/s or 1 Gb/s). This is achieved by using the auto-negotiation functionality defined in Clause 37 of the IEEE Specification 802.3z. Instead of the ability advertisement, the PHY sends the control information via its tx_config_reg[15:0] as listed in Table 3-17 each time the link speed information changes. Upon receiving control information, the MAC acknowledges the update of the control information by asserting bit 14 of its tx_config_reg[15:0] as listed in Table 3-17.

Compared to the definition in IEEE802.3 clause 37, the link_timer inside the auto-negotiation has been changed from 10 ms to 1.6 ms to ensure a prompt update of the link status.

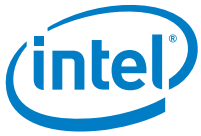
### Table 3-17 SGMII Link Control Information

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

When operating in 100 Mb/s the SGMII interface elongates the frame by replicating each frame byte 10 times for 100 Mb/s. This frame elongation takes place above the 802.3z PCS layer, thus the start frame delimiter only appears once per frame. Note that the 802.3z PCS layer might remove the first byte of the elongated frame. An example of a 100 Mb/s elongated frame can be seen in Figure 3-19.




### Figure 3-19 Data Sampling in 100 Mb/s Mode



## 3.7.4 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 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 is defined in IEEE802.3ap Clause 73 and includes support for parallel detection of 1000BASE-KX and 10GBASE-KX4 links in addition to transmission and reception of extended base page and next page auto-negotiation frames. The 82599 supports reception of extended base page and next page auto-negotiation frames but does not transmit next page auto-negotiation frames only NULL frames.

### 3.7.4.1 Link Configuration

The 82599 network interface meets industry specifications for:

- 10 GbE:
  - — XAUI (IEEE 802.3ae)
  - — SFI (SFF-8431 Specifications for Enhanced 8.5 and 10 Gigabit Small Form Factor Pluggable Module SFP+)
- 10 GbE — 10GBASE-CX4 (IEEE 802.3ak)
- 1 GbE backplane:
  - — Ethernet 1000BASE-KX (IEEE 802.3ap)
  - — Ethernet 1000BASE-BX (PICMG3.1)
  - — SFI (SFF-8431 Specifications for Enhanced 8.5 and 10 Gigabit Small Form Factor Pluggable Module SFP+)
- 10 GbE backplane:
  - — Ethernet 10GBASE-KX4 (IEEE 802.3ap)
  - — Ethernet 10GBASE-KR (IEEE 802.3ap)

The MAUI AFE is configured at start up to support the appropriate protocol as a function of the negotiation process and pre-defined control bits that are either loaded from the EEPROM or configured by software.

### 3.7.4.2 MAC Link Setup and Auto Negotiation

The MAC block in the 82599 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 82599 supports 10GBase-KX4, 10GBase-CX4,10GBase-KR, SFI or XAUI (10 GbE Attachment Unit Interface). While in 1 GbE operating mode, the 82599 supports 1000Base-KX, 1000Base-BX or SGMII (SGMII also supports both 100 Mb/s and 1 Gb/s data rates) protocols. 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 IEEE802.3ap clause 73 enables selection between KR (10G), KX4 (10G) and KX (1G) compliant link partners and defining link characteristics and link speed. While the auto-negotiation process defined in IEEE802.3 clause 37 enables detection of the BX (1 GbE) link characteristics but not the link speed.

Link setting is done by configuring the speed configuration in the AUTOC.LMS field, defining the appropriate physical interface by programming AUTOC.1G_PMA_PMD, AUTOC.10G_PMA_PMD_PARALLEL and AUTOC2.10G_PMA_PMD_Serial and restarting auto-negotiation by setting AUTOC.Restart_AN to 1b.

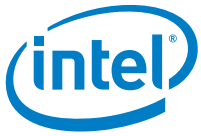
**Note:** Auto-negotiation logic will reset the data pipeline on AUTOC.Restart_AN assertion only if AUTOC.LMS field is changed. If the user wants to change link configuration parameters but keep the same AUTOC.LMS field value, link configuration must take these steps:

1. Read AUTOC register. Write back AUTOC register content with LMS[2] bit inverted (AUTOC bit 15) and Restart_AN bit asserted.
2. Read ANAS field in ANLP1 register. Check that it is not zero (or idle), indicating that autonegotiation was restarted.
3. Write AUTOC register with original LMS field and Restart_AN bit asserted.

### 3.7.4.3 Hardware Detection of Legacy Link Partner (Parallel Detection)

The 82599 supports the IEEE802.3ap 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 (KX4 or KX as defined in IEEE802.3ap clause 73) by activating KX4 and KX alternately and attempting to achieve link synchronization by the related PCS block.

Parallel detection is enabled as part of clause 73 backplane auto-negotiation process by appropriately configuring the link speed and auto-negotiation mode in the AUTOC.LMS register field, clearing AUTOC2.PDD to 0b and restarting auto-negotiation by setting the AUTOC.Restart_AN bit to 1b.



### 3.7.4.4 MAUI Link Setup Flow

The 82599 MAUI interface is configured at start up (before the driver is loaded) in the following manner:

1. If the link is statically configured by programming the appropriate AUTOC (LMS, 1G_PMA_PMD, 10G_PMA_PMD_PARALLEL) register fields and AUTOC2.10G_PMA_PMD_Serial field, the 82599 attempts to synchronize on incoming data and if successful updates the relevant Link status registers (LINKS, ANLP1 and ANLP2) and sets up the link. If link synchronization is not successful, the 82599 does not report link-up in the LINKS register and continuously attempts to set up the link according to the static configuration.
2. If the Link is not statically configured and parallel detection is enabled (auto-negotiation enabled in AUTOC.LMS and the AUTOC2.PDD parallel detect disable is 0b) the 82599 starts IEEE802.3ap clause 73 negotiation protocol by attempting to parallel detect the protocol on the MAUI interface by enabling KX and KX4 receive circuitry and trying to synchronize on incoming data. If synchronization succeeds in either KX or KX4 modes, the 82599 updates the relevant link status registers (LINKS, ANLP1 and ANLP2) and commences with setting up the link.
3. If parallel detect fails, the 82599 attempts to auto-negotiate according to IEEE802.3ap clause 73 using the data written to the AUTOC, AUTOC2 and AUTOC3 registers. If auto-negotiation succeeds, the 82599 updates the link status registers (LINKS, ANLP1 and ANLP2). If auto-negotiation fails, the 82599 does not report link up in the LINKS register and retries acquiring the link by parallel detection and auto-negotiation continuously (the receiver goes through a continuous cycle of 1 GbE parallel detect, 10 GbE parallel detect and clause 73 auto-negotiation).
4. If parallel detect or static configuration succeeds and the link rate is 1 Gb/s, AUTOC.LMS enables IEEE802.3 clause 37 auto-negotiation. The 82599 auto-negotiates to define link characteristics according to IEEE802.3 clause 37 using information placed in registers PCS1GANA and PCS1GANNP. On completion of clause 37 auto-negotiation, the 82599 updates the status in the LINKS, PCS1GLSTA, PCS1GANLPNP and PCS1GANLP registers.
5. If parallel detect or static configuration succeeds and the link rate is 1 Gb/s, SGMII is enabled in the AUTOC.LMS field (LMS = 101b). If the 82599 detects the SGMII negotiation control information sent by the PHY, the 82599 auto-negotiates to define link characteristics (1 Gb/s or 100 Mb/s and full duplex capability) according to the SGMII specification. On completing SGMII auto-negotiation, the 82599 updates the status in the LINKS, PCS1GLSTA and PCS1GANLP registers.

When AUTOC.LMS is set to 1b of the auto-negotiation modes and the *Link Up* bit is set to 1b in the LINKS register, the final link speed can be read from the LINK_SPEED field of LINKS.

If *LINK_SPEED* is 10 Gb/s, the *MLINK_MODE* field is used to differentiate between KX4 (10 GbE parallel) and KR (10 GbE serial).

**Note:** AUTOC.AN_RESTART must be set on every AUTOC.LMS change.

### 3.7.4.5 Next Page Support

Next Page (NP) support in the 82599 is compliant with IEEE802.3ap.

The 82599 acts as receiver of NP each time the link partner needs to transmit NP data through the KX/KX4 auto-negotiation process.

The 82599 does not support transmission of configurable NP. It transmits a null NP each time the auto-negotiation arbitration state machine is required to go through the NP handshake. There is a possibility to configure the *Acknowledge2* field in the NP through the AUTOC.ANACK2 bit.

### 3.7.4.6 Forcing Link Up

Forcing link up can be accomplished by software by setting the AUTOC.FLU bit to 1b, which forces the MAC to the appropriate MAC link speed as defined by the AUTOC.LMS field and the appropriate protocol as defined by the AUTOC.10G_PMA_PMD_PARALLEL, AUTOC2.10G_PMA_PMD_Serial and AUTOC.1G_PMA_PMD bits. The Force-Link-Up mode enables loopback operation (when HLREG0.LPBK is set to 1b) by setting the link_up indication regardless of the XGXS/PCS_1G/KR_locked status. Link indication in register LINKS should be ignored when in this mode.

### 3.7.4.7 Crossover

The 82599 supports crossover on each of the two MAUI ports to eliminate the need for crossover cables between similar devices. This has historically been accomplished using special crossover cables (patch cables), magnetic pinouts or PCB wiring. The 82599 supports crossover configuration in both 10 GbE and 1 GbE operating modes via the SERDESC register.

Having established that there is a problem with the link connection, the driver detects and corrects crossovers and arbitrary polarity swaps for several configurations of pair swaps. Crossover can also be set by EEPROM following power up.

The following receiver pairs:

A — MI_QL0 (MIP_QL0 and MIN_QL0)

B — MI_QL1 (MIP_QL1 and MIN_QL1)

C — MI_QL2 (MIP_QL2 and MIN_QL2)

D — MI_QL3 (MIP_QL3 and MIN_QL3)

can be connected to the corresponding link partner’s transmit pairs in any of the following ways with arbitrary polarity (positive and negative wires exchanged):

- No crossover
- A/B crossover only
- C/D crossover only
- A/B crossover and C/D crossover

Crossover operation is controlled by programming the relevant bits in the SerDes Interface Control (SERDESC) register. The SERDESC register supports correction of all combinations of crossover scenarios, in addition to the scenarios previously described.

## 3.7.5 Transceiver Module Support

The 82599 MAUI interface with additional usage of low speed interface pins (SDP, $I^2C$ and MDIO I/Os) supports a connection to transceiver modules compliant with the following Multi Source Agreements (MSAs):

- XENPAK — A cooperation agreement for 10 Gigabit Ethernet Transceiver package Rev 3.0
- X2 — A cooperation agreement for a small Versatile 10 Gigabit Ethernet Transceiver package Rev 2.0b
- XPAK — A cooperation agreement for a small form factor pluggable 10 Gigabit Ethernet Transceiver package Rev 2.2
- SFP+ — SFF-8431 Specifications for Enhanced 8.5 and 10 Gigabit Small Form Factor Pluggable Module SFP+ rev 1.0

Figure 3-20 shows the various transceiver module architecture.

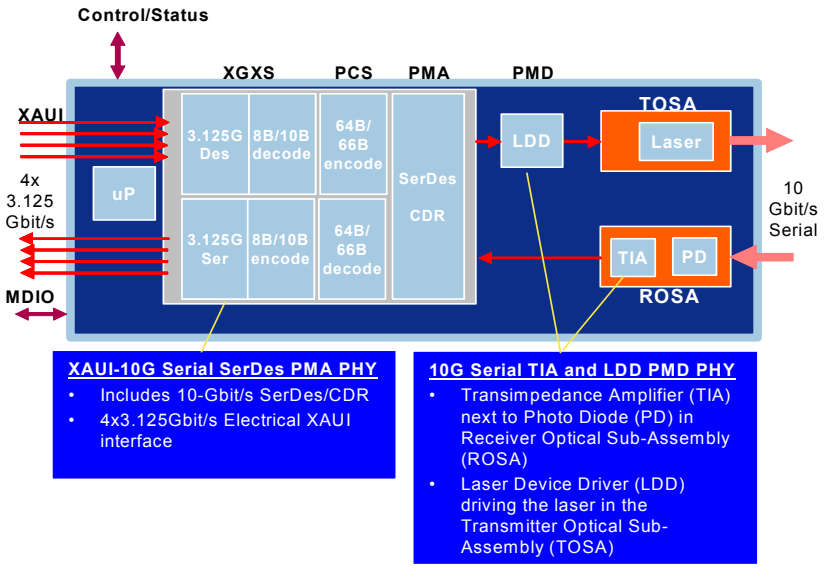



**Figure 3-20 XENPAK, XPAK and X2 Transceiver Architecture**

Table 3-18 lists the required interface (per port) for supporting the various modules. The 82599 supports the high speed interface using the MAUI port and the low speed interface using the SDP pins.

**Table 3-18 Optical Module Interface Support**

| Module Type | High Speed MAUI Protocol | Low Speed Interface (per port) |
|---|---|---|
| XENPAK | XAUI | MDC[1] (1.2V OUT), MDIO[1] (1.2V I/O),<br>TX ON/OFF[2] (1.2V OUT), RESET[2] (1.2V OUT)<br>LASI (1.2V IN — Interrupt) |
| X2 | XAUI | MDC[1] (1.2V OUT), MDIO[1] (1.2V I/O),<br>TX ON/OFF[2] (1.2V OUT), RESET[2] (1.2V OUT)<br>LASI (1.2V IN — Interrupt) |
| XPAK | XAUI | MDC[1] (1.2V OUT), MDIO[1] (1.2V I/O),<br>TX ON/OFF[2] (1.2V OUT), RESET[2] (1.2V OUT)<br>LASI (1.2V IN — Interrupt) |
| SFP+ | SFI | SCL[1] (I2C — OD), SDL[1] (I2C — OD)<br>TX Disable[3] (LVTTL — OUT), RS0/1 (LVTTL — OUT)<br>TX Fault (LVTTL IN), RX_LOS (LVTTL — IN) |

1. Single management interface can be used for two ports.
2. Output low during reset and power down.

The 82599 enables interfacing optical modules using the MAUI pins, MDIO pins, $I^2C$ pins and SDP pins. When interfacing with XENPAK, XPAK and X2 modules, level translators from LVTTL to 1.2V need to be added on the MDIO pins and the relevant SDP pins.

## 3.7.6 Management Data Input/Output (MDIO) Interface

The 82599 supports a MDIO interface (per port) to control PHY functionality through the interface. PHY configuration registers are mapped into the MDIO space and can be accessed by the MAC or any other MDIO-master device.

The 82599 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 monitor and control of PHY functionality. The 82599 is compliant with the IEEE802.3 clause 45 in both 10 GbE and
1 GbE operation. The 82599 also supports IEEE 802.3 clause 22 frame formats and register address space for accessing legacy PHY registers. 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 need to be added on board.

Figure 3-21 shows the basic connectivity between the PHY and MAC.

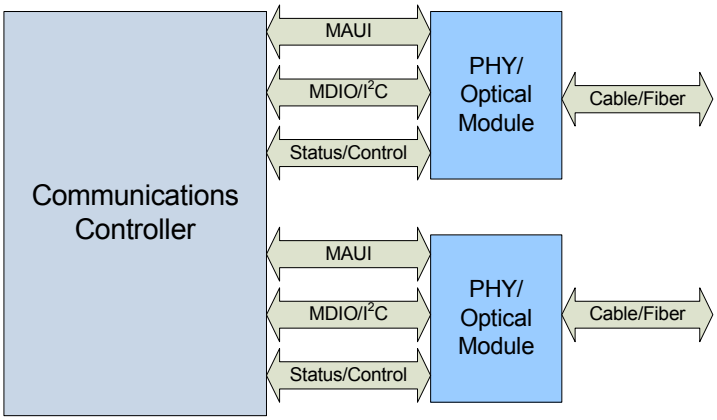



**Figure 3-21 Basic PHY MAC Connectivity**

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

### 3.7.6.1 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 3-19.

**Table 3-19 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 |
| 100 Mb/s | 240 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 3-22 (measured at the MII connector).

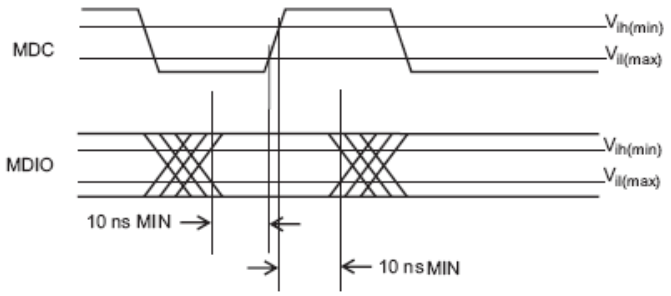



**Figure 3-22 MDIO Timing Sourced by the MAC**

When the MDIO signal is sourced by the PHY, it is sampled by the MAC (STA) synchronously with respect to the rising edge of MDC. The clock to output delay from the PHY, as measured at the MII connector, must be a minimum of 0 ns, and a maximum of 300 ns, as shown in Figure 3-23.

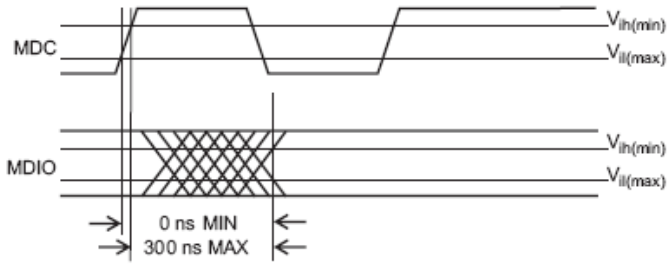



**Figure 3-23 MDIO Timing Sourced by the PHY**

## 3.7.6.2 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.

### 3.7.6.3 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 82599 supports both frame structures to enable interfacing PHYs that support either protocol.

The basic frame format as defined in IEEE802.3 clause 22 can optionally be used for accessing legacy PHY registers is listed in Table 3-20.

**Table 3-20 Clause 22 Basic MDIO Frame Format**

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

The MDIO interface defined in clause 45 uses indirect addressing to create an extended address space enabling access to a large number of registers within each MDIO Managed Device (MMD). The MDIO management frame format is listed in Table 3-21.

**Table 3-21 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 postread-increment-address transaction. During a write or address transaction, the STA must drive a one bit for the first bit time of the TA and a zero bit for the second bit time of the TA. Figure 3-24 shows the behavior of the MDIO signal during the *TA* field of a read transaction.

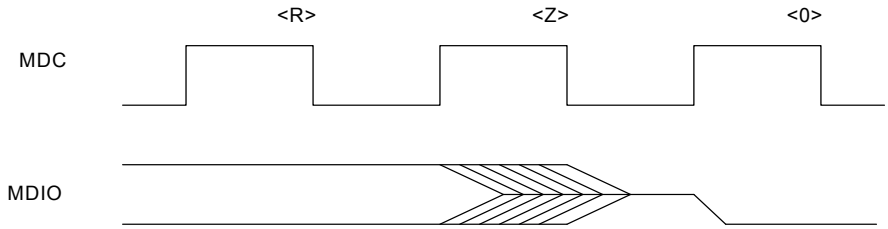



**Figure 3-24 Behavior of MDIO During TA Field of a Read Transaction**

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

### 3.7.6.4 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:

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

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

## 3.7.7 Ethernet Flow Control (FC)

The 82599 supports flow control as defined in 802.3x, as well as the specific operation of asymmetrical flow control defined by 802.3z. The 82599 also supports Priority Flow Control (PFC), sometimes referred to as Class Based Flow Control or (CBFC), as part of the DCB architecture.

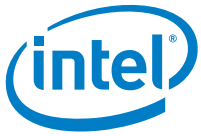
**Note:** The 82599 can either be configured to receive regular flow control packets or Priority Flow Control (PFC) packets. The 82599 does not support the reception of both types of packets simultaneously.

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

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

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

- MAC Flow Control (MFLCN) register — Enables flow control and passing of control packets to the host.
- Flow Control Configuration (FCCFG) — Determines mode for Tx flow control (no FC vs. link based versus priority based). Note that if Tx flow control is enabled then Tx CRC by hardware should be enabled as well (HLREG0.TXCRCEN = 1b).
- Flow Control Address Low, High (RAL[0], RAH[0]) — 6-byte flow control multicast address.
- Priority Flow Control Type Opcode (PFCTOP) — Contains the type and opcode values for priority FC.
- 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 priority FC 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 priority FC 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 priority FC mode.
- Flow Control Refresh Threshold Value (FCRTV) — 16-bit PAUSE refresh threshold value (in legacy FC FCRTV[0] must be smaller than FCTTV[0]).



# 3.7.7.1 MAC Control Frames and Reception of Flow Control Packets

## 3.7.7.1.1 MAC Control Frame — Other than FC

IEEE reserved the Ethertype value of 0x8808 for MAC control frames as listed in Table 3-22.

**Table 3-22 MAC Control Frame Format**

| | |
|---|---|
| DA | The *Destination Address* field can be an individual or multicast (including broadcast) address. Permitted values for the *Destination Address* field can be specified separately for a specific control opcode such as FC packets. |
| SA | Port Ethernet MAC address (6 bytes). |
| Type | 0x8808 (2 bytes). |
| Opcode | The MAC control opcode indicates the MAC control function. |
| Parameters | The *MAC Control Parameters* field must contain MAC control opcode-specific parameters. This field can contain none, one, or more parameters up to a maximum of minFrameSize =20 bytes. |
| Reserved field = 0x00 | The *Reserved* field is used when the MAC control parameters do not fill the fixed length MAC control frame. |
| CRC | 4 bytes. |

## 3.7.7.1.2 Structure of 802.3X FC Packets

802.3X FC packets are defined by the following three fields (see Table 3-23):

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

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

**Table 3-23 802.3X Packet Format**

| | |
|---|---|
| DA | 01_80_C2_00_00_01 (6 bytes). |
| SA | Port Ethernet MAC address (6 bytes). |
| Type | 0x8808 (2 bytes). |
| Opcode | 0x0001 (2 bytes). |
| Time | XXXX (2 bytes). |
| Pad | 42 bytes. |
| CRC | 4 bytes. |

#### 3.7.7.1.3 PFC

DCB introduces support for multiple traffic classes assigning different priorities and bandwidth per TC. Link level Flow Control (PAUSE) stops all the traffic classes. PFC or CBFC allows more granular flow control on the Ethernet link in an 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. Board designers can implement PFC by sending a timer indication to the transmitting station traffic class (XOFF) of a nearly full receive buffer condition at the 82599. At this point the transmitter would stop transmitting packets for that TC until the XOFF timer expires or a XON message is received for the stopped TC.

Similarly, once the 82599 receives a priority-based XOFF it stops transmitting packets for that specific TC until the XOFF timer expires or XON packet for that TC is received.

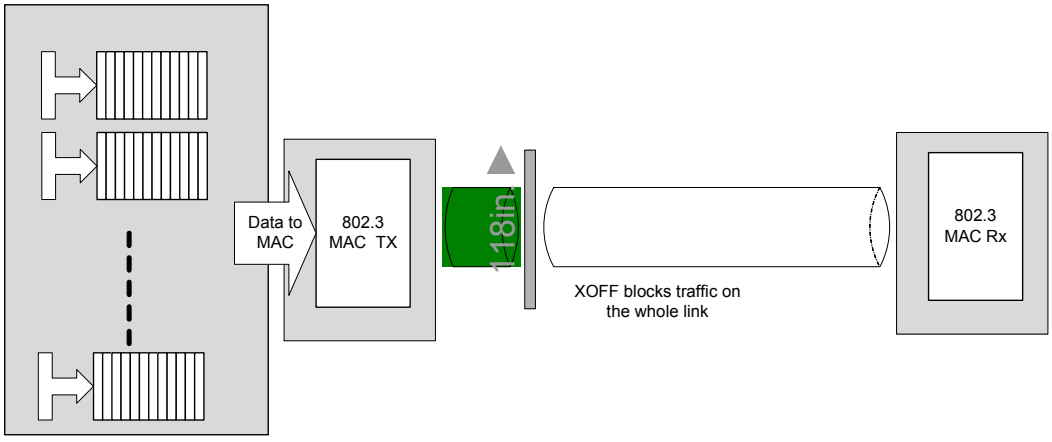



**Figure 3-25 802.3X Link Flow Control (PAUSE)**

Link flow control (802.3X) causes all traffic to be stopped on the link. DCB uses the same mechanism of flow control but provides the ability to do PFC on TCs as shown in Figure 3-26.

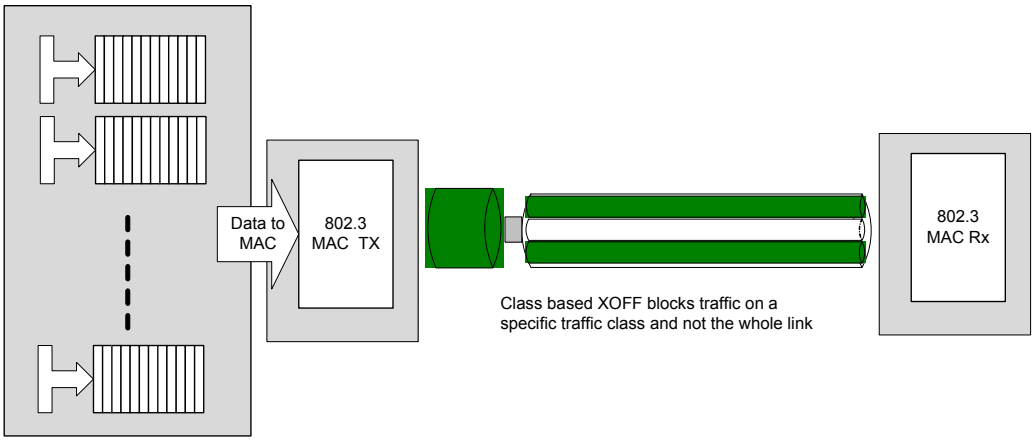



**Figure 3-26 Priority Flow Control**

**Table 3-24 Packet Format for Priority Flow Control**

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

**Table 3-25 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<br>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). For example, Timer 0 refers to UP 0, etc. The 82599 binds a UP (and therefore the 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 whenever receiving PFC frames.

When a PFC frame is formatted by the 82599, the same values are replicated into every *Timer* field and priority enable vector bit of all the UPs bound to the concerned TC. These values as configured in the 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 previous UPs, the traffic on the associated TC must be paused by the highest XOFF timer's value.

#### 3.7.7.1.4 Operation and Rules

The 82599 operates in either link FC or in PFC mode. Enabling both modes concurrently is not allowed:

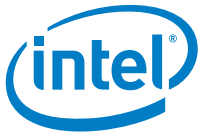
- Link FC is enabled by the *RFCE* bit in the MFLCN register.
- PFC is enabled by the *RPFCE* bit in the MFLCN register.

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

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.

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

- Increments the appropriate statistics register(s)
- Initialize the pause timer based on the packet's PAUSE *Timer* field (overwriting any current timer's value)



  - In case of PFC, this is done per TC. If several UPs are associated with a TC, then the device sets the timer to the maximum value among all enabled timer fields associated with the TC.

- Disable packet transmission or schedule the disabling of transmission after the current packet completes.
  - In case of PFC, this is done per paused TC
  - 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
  - In case of PFC, this is done per TC
- Reception of an XON frame (a frame with its PAUSE timer set to 0b)
  - In case of PFC, this is done per TC

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

#### 3.7.7.1.5 Timing Considerations

When operated at 10 Gb/s line speed, the 82599 must not begin to transmit a (new) frame more than 60 pause quanta after receiving a valid Link XOFF frame, as measured at the wires (a pause quantum is 512 bit times). When connected to an external 10GBASE-KR PHY with FEC or to an external 10GBASE-T PHY, the response time requirement decreases to 74 pause quanta, because of extra delays consumed by these external PHYs.

When operating at 1 Gb/s line speed, the 82599 must not begin to transmit a (new) frame more than 2 pause quanta after receiving a valid Link XOFF frame, as measured at the wires.

The 802.1Qbb draft 1.0, proposes that the tolerated response time for Priority XOFF frames are the same as Link XOFF frames with extra budget of 19072 bit times if MACSec is used, or of 2 pause quanta otherwise. This extra budget is aimed to compensate the fact that decision to stop new transmissions from a specific TC must be taken earlier in the transmit data path than for the Link Flow Control case.

### 3.7.7.2 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 transfer of PAUSE packets to the host. The same policy applies to both link FC and priority FC packets as listed in Table 3-26. Note that any packet must pass the L2 filters as well.

- The *PMCF* bit controls transfer of non-PAUSE packets to the host. Note that when link FC frames are not enabled (RFCE = 0b) then link FC frames are considered as MAC Control (MC) frames for this matter. Similarly, when PFC frames are not enabled (RPFCE = 0b) then PFC frames are considered as MC frames as well.

**Note:** When virtualization is enabled, forwarded control packets are queued according to the regular switching procedure defined in Section 7.10.3.4.

**Table 3-26 Transfer of PAUSE Packet to Host (DPF Bit)**

| RFCE | RPFCE | DPF | Link FC handling | Priority FC handling |
|---|---|---|---|---|
| 0b | 0b | X | Treat as MC (according to PMCF setting). | Treat as MC (according to PMCF setting). |
| 1b | 0b | 0b | Accept. | Treat as MC (according to PMCF setting). |
| 1b | 0b | 1b | Reject. | Treat as MC (according to PMCF setting). |
| 0b | 1b | 0b | Treat as MC (according to PMCF setting). | Accept. |
| 0b | 1b | 1b | Treat as MC (according to PMCF setting). | Reject. |
| 1b | 1b | X | Unsupported setting. | Unsupported setting. |

## 3.7.7.3 Transmitting PAUSE Frames

The 82599 generates PAUSE packets to insure there is enough space in its receive packet buffers to avoid packet drop. The 82599 monitors the fullness of its receive FIFOs and compares it with the contents of a programmable threshold. When the threshold is reached, the 82599 sends a PAUSE frame. The 82599 supports both link flow control and PFC — but not both concurrently. When DCB is enabled, it sends only PFC, and when DCB is disabled, it send only link flow control.

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

### 3.7.7.3.1 Priority Flow Control

The same flow control mechanism is used for PFC and for 802.3X flow control to determine when to send XOFF and XON packets. When PFC is used in the receive path, Priority PAUSE packets are sent instead of 802.3X PAUSE packets. The format of priority PAUSE packets is described in Section 3.7.7.1.3.

Specific considerations for generating PFC packets:

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

### 3.7.7.3.2 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 flow control mode and the PFC mode.

The content of the Flow Control Receive Threshold High (FCRTH) register determines at what point the 82599 transmits the first PAUSE frame. The 82599 monitors the fullness of the receive FIFO and compares it with the contents of FCRTH. When the threshold is reached, the 82599 sends a PAUSE frame with its pause time field equal to FCTTV.

At this time, the 82599 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 82599 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 82599 sends a PAUSE frame if it has previously sent one and the FIFO overflows. This is intended to minimize the amount of packets dropped if the first PAUSE frame did not reach its target.

### 3.7.7.3.3 Flow Control High Threshold — FCRTH

The 82599 sends a PAUSE frame when a 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. This latency is composed of the following elements:

- Threshold Cross to XOFF Transmission + Round-trip Latency + XOFF Reception to Link Partner Response, where:

| Latency Parameter | Affected by. . . | Value at 10 GbE with Jumbo |
|---|---|---|
| Trigger to XOFF transmission. | Max packet size at all TCs. | 9.5 KB (example). |
| Link partner XOFF to transmission hold. | Max packet size on the specific TC. | 9.5 KB (example). |
| Round-trip Latency. | The latencies on the wire and the LAN devices at both sides of the wire. | 8 KB (see the calculation that follows). |

- Round-trip Latency Calculation:
  - Pause Quanta (PQ) = 512 bit time (bt)
  - Round trip for 10 GbE MAC + XAUI + 10 GbE PHY = 16+8+50 PQ ≅ 4.7 KB (using another PHY a lower latency can be taken)
  - Round trip capable (2x100 m) = 200 m x 50 bt/m = 10000 bt ≅ 1.25 KB (at other known topologies lower latency can be taken)
  - Plus 2 KB for some guard-band and processing latency of transmission and reception pause frames

The internal architecture of the Rx packet buffer is as follows:

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).

Translating the latencies to possible consumed Rx packet buffer at worst case is:

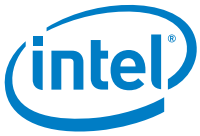
| Latency Parameter | Value | Consumed Rx Packet Buffer |
|---|---|---|
| Trigger to XOFF transmission | 9.5 KB | 1.44 x 9.5 KB ≅ 14 KB |
| Link partner XOFF to transmission hold | 9.5 KB | 9.5 KB |
| Round-trip latency | 8 KB | 1.44 x 8 KB ≅ 11.5 KB |

The FCRTH should be set to the size of the Rx packet buffer minus (14 + 9.5 + 11.5 = 35 KB). As previously indicated, these numbers are valid if jumbo frames are enabled in all traffic classes. When it is required to avoid packet lost, software must follow this requirement and enable flow control functionality.

When Tx to Rx switching is enabled, packets can be received to the Rx packet buffer by local VM-to-VM traffic. Once the Rx packet buffer gets full and is above the high threshold it might receive up to one additional packet from a local VM. Therefore, FCRTH should be set to the size of the Rx packet buffer minus (the size previously explained plus one additional max packet size).

#### 3.7.7.3.4 Flow Control 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 can be set high enough so that the Rx packet buffer does not get empty before new whole 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 as twice the bandwidth on the wire (when only a single port is active at a given time). Therefore, on 10 GbE network with jumbo packets a threshold that guarantee that the Rx packet buffer is not emptied should be set larger than: 2 x (2 x 9.5 KB + 8 KB) ≅ 54 KB. 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.



#### 3.7.7.3.5 Packet Buffer Size

When flow control is enabled, the total size of a 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 (9728-byte) jumbo frames are enabled) and larger than a few frames in other cases. If the available Rx packet is large enough, it is recommended to increase as much as possible the hysteresis budget. If the available Rx packet is not large enough it might be required to cut both the low threshold as well as the hysteresis budget. The following table lists a few examples while it is recommended to validate the values for a given use case.

| Latency Parameter | Flow Control High Threshold | Flow Control Low Threshold | Total Packet Buffer Size |
|---|---|---|---|
| 9.5 KB (9728-byte) jumbo enabled with no DCB with flow control. | 477 KB | 54 KB | 512 KB |
| 9.5 KB (9728-byte) jumbo enabled x 8 TCs with flow control. | 29 KB | 19.5 KB | 64 KB |
| 9.5 KB (9728-byte) jumbo enabled x 8 TCs with flow control and flow director table enabled with 128 KB. | 13 KB | 9 KB | 48 KB |
| 9.5 KB (9728-byte) jumbo enabled x 4 TCs with flow control and 1500-byte (no jumbo) x 4 TCs with flow control and flow director table enabled with 128 KB. | 21 KB<br>14 KB | 11.5 KB<br>9 KB | 56 KB (jumbo)<br>40 KB (1.5 KB) |
| 9.5 KB (9728-byte) jumbo enabled x 4 TCs WITHOUT flow control and<br>1500-byte (no jumbo) x 4 TCs WITH flow control and flow director table enabled with 128 KB. | N/A<br>30 KB | N/A<br>20 KB | 40 KB (jumbo)<br>56 KB (1.5 KB) |

When Tx-to-Rx switching is enabled (in virtualization mode) the high threshold should take into account potential VM-to-VM reception. As a result, the Rx packet buffer's sizes should be increased, respectively.

### 3.7.7.4 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 not support (or enable) PFC, the 82599 throttles the traffic using link FC. Following is the required device setting and functionality:

- The 82599 should be set to legacy link FC by setting MFLCN.RFCE.
- Reception of 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 Flow Control Transmit Timer Value of all TCs must be set to the same value.

## 3.7.8 Inter Packet Gap (IPG) Control and Pacing

The 82599 supports transmission pacing by extending the IPG (the gap between consecutive packets). The pacing mode allows the average data rate to be slowed in systems that cannot support the full link rate (10 Gb/s, 1Gb/s or 100 Mb/s). As listed in Table 3-27, the pacing modes work by stretching the IPG in proportion to the data sent. In this case the data sent is measured from the end of preamble to the last byte of the packet. No allowance is made for the preamble or default IPG when using pacing mode.

Example 1:

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

Example 2:

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

Table 3-27 lists the pacing configurations supported by the 82599 at link rates of 10 Gb/s. When operating at lower link speeds the pacing speed is proportional to the link speed.

**Table 3-27 Pacing Speeds at 10 Gb/s Link Speed**

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

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

**Note:** The IPG pacing feature is a parallel feature to the Tx rate scheduler where IPG pacing is applied to the entire Tx data flow while the Tx rate scheduler is applied separately to each Tx queue. Therefore, if a single queue is used, either feature can be used to limit the Tx data rate; however, if multiple queues are used, the IPG pacing feature is a better choice for a homogeneous Tx data rate limitation.

## 3.7.9 MAC Speed Change at Different Power Modes

Normal speed negotiation drives to establish a link at the Highest Common Denominator (HCD) link speed. The 82599 supports an additional mode of operation, where the MAC establishes a link at the Lowest Common Denominator (LCD) link speed. The link-up process enables a link to come up at any possible speed in cases where power is more important than performance. Different behavior is defined for the D0 state and non-D0 states as a function of the AUTOC.D10GMP, AUTOC.RATD and MMNGC.MNG_VETO register bits.

The 82599 can initiate auto-negotiation without direct driver command in the following cases:

- When the state of MAIN_PWR_OK pin changes.
- When the MNG_VETO bit value changes.
- On a transition from D0a state to a non-D0a state, or from a non-D0a state to D0a state.

Figure 3-27 shows the 82599 behavior when entering low power mode and Figure 3-28 shows the 82599 behavior when going to power-up mode.

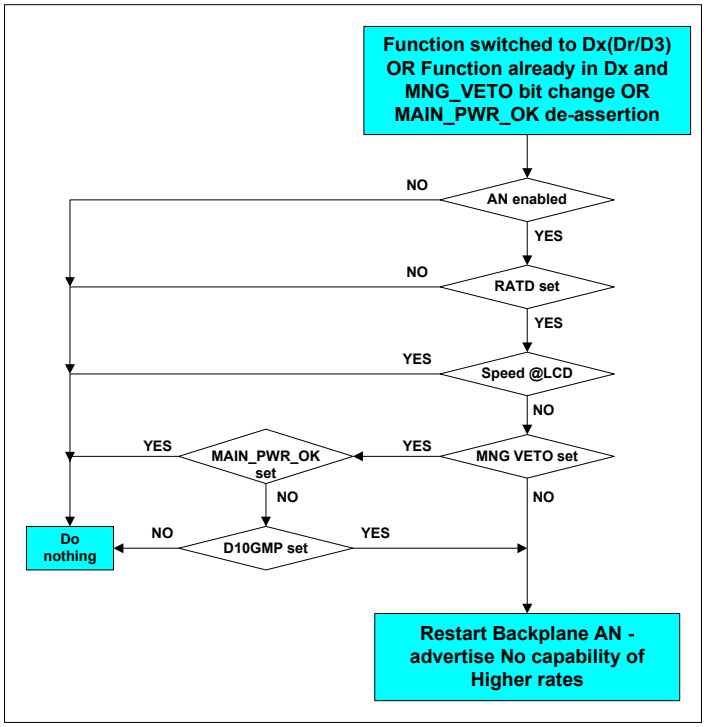



**Figure 3-27 MAC Speed Change When Entering Power Down Mode**

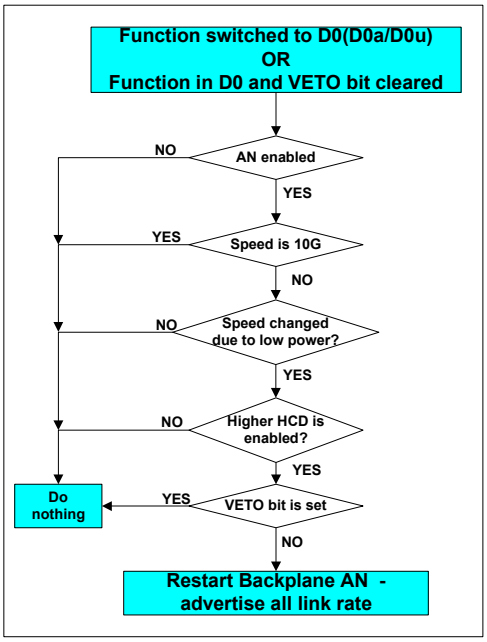



**Figure 3-28 MAC Speed Change on Entering Power-up Mode**

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

# 4.0 Initialization

## 4.1 Power Up

### 4.1.1 Power-Up Sequence

The figure below shows the 82599 power-up sequence from power ramp up until the 82599 is ready to accept host commands.

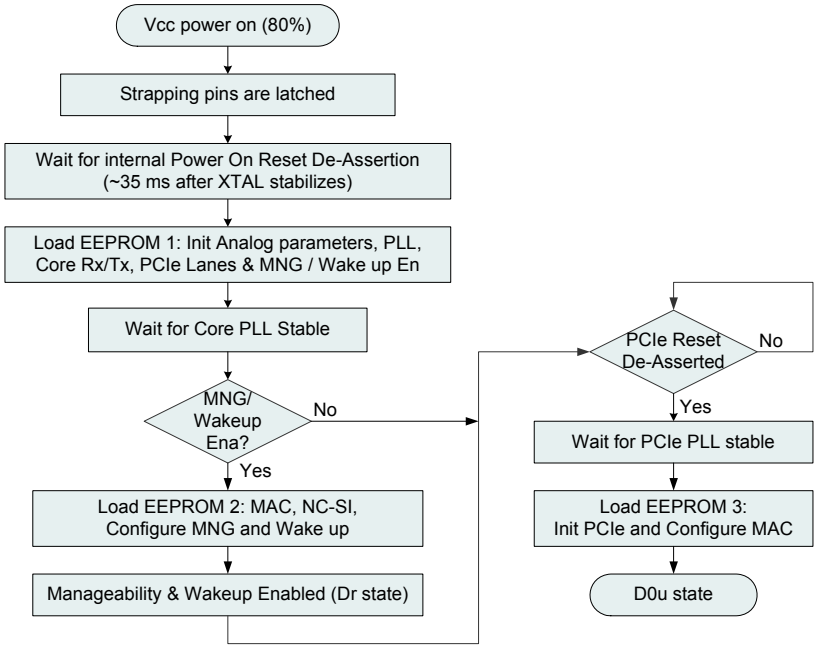



**Figure 4-1 82599 Power-Up Sequence**

## 4.1.2 Power-Up Timing Diagram

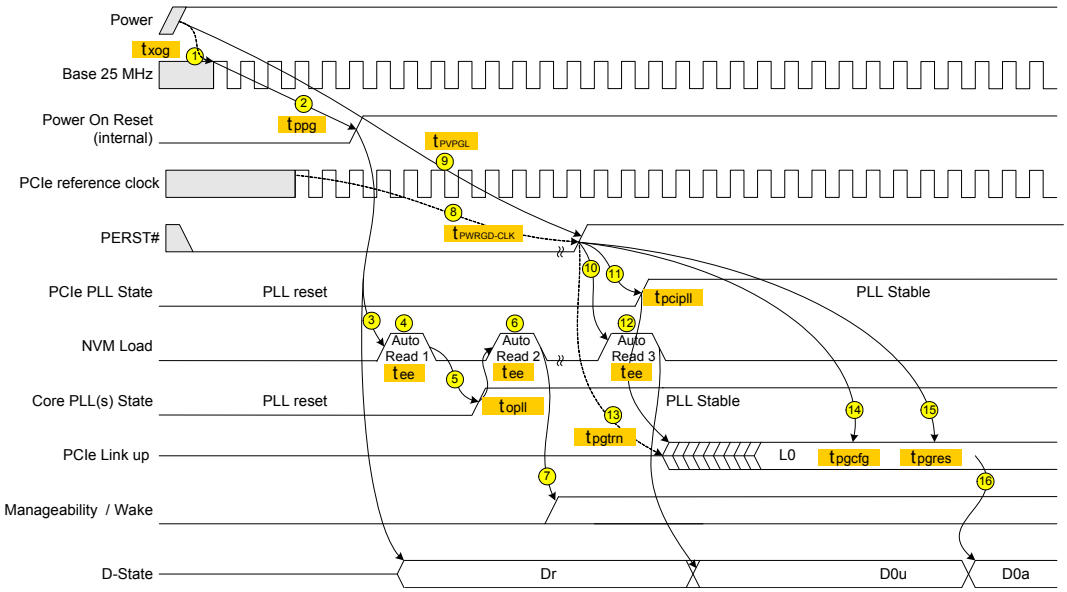



**Figure 4-2 Power-Up Timing Diagram**

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

| Note | |
|---|---|
| 1 | Base 25 clock is stable $t_{xog}$ after power is stable. |
| 2 | Internal Reset is released $t_{ppg}$ after Base 25 is stable (also power supplies are good). |
| 3 | NVM read starts following the rising edge of the internal Power On Reset or external LAN Power Good. |
| 4 | EEPROM auto-load 1: EEPROM Init Section; PCIe Analog; Core Analog. |
| 5 | EEPROM auto-load 1 completion to Core PLL(s) stable — $t_{opll}$. |
| 6 | EEPROM auto-load 2: MAC module manageability and wake up (if manageability / wake up enabled). |
| 7 | APM wake up and/or manageability active, based on NVM contents (if enabled). |
| 8 | The PCIe reference clock is valid $t_{PWRGD-CLK}$ before the de-assertion of PCIe Reset (PCIe specification). |
| 9 | PCIe Reset is de-asserted $t_{PVPGL}$ after power is stable (PCIe specification). |
| 10 | De-assertion of PCIe Reset invokes the EEPROM auto-load 3. |
| 11 | De-assertion of PCIe Reset to PCIe PLL stable $t_{PCIPLL}$. |
| 12 | EEPROM auto-load 3: PCIe General Configuration; PCIe Configuration Space; LAN Core Modules and MAC module if manageability is not enabled. |
| 13 | Link training starts after $t_{pgtrn}$ from PCIe Reset de-assertion (PCIe specification). |

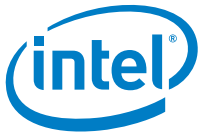


**Table 4-1 Notes for Power-Up Timing Diagram (Continued)**

| Note | |
|---|---|
| 14 | A first PCIe configuration access might arrive after $t_{pgcfg}$ from PCIe Reset de-assertion (PCIe specification). |
| 15 | A first PCI configuration response can be sent after $t_{pgres}$ from PCIe Reset de-assertion (PCIe specification). |
| 16 | Setting the *Memory Access Enable* or *Bus Master Enable* bits in the PCI Command register transitions the 82599 from D0u to D0 state. |

### 4.1.2.1 Timing Requirements

The 82599 requires the following start-up and power state transitions.

**Table 4-2 Power-Up Timing Requirements**

| Parameter | Description | Min | Max. | Notes |
|---|---|---|---|---|
| $t_{xog}$ | Base 25 MHz clock stable from power stable. | | 10 ms | |
| $t_{PWRGD-CLK}$ | PCIe clock valid to PCIe power good. | 100 µs | - | According to PCIe specification. |
| $t_{PVPGL}$ | Power rails stable to PCIe Reset inactive. | 100 ms | - | According to PCIe specification. |
| $t_{pgcfg}$ | External PCIe Reset signal to first configuration cycle. | 100 ms | | According to PCIe specification. |

**Note:** It is assumed that the external 25 MHz clock source is stable after the power is applied; the timing for that is part of txog.

#### 4.1.2.2 Timing Guarantees

The 82599 guarantees the following start-up and power state transition related timing parameters.

**Table 4-3 Power-Up Timing Guarantees**

| Parameter | Description | Min | Max. | Notes |
|---|---|---|---|---|
| $t_{xog}$ | Xosc stable from power stable. | | 10 ms | |
| $t_{ppg}$ | Internal power good delay from valid power rail. | | 35 ms | Use internal counter for external devices stabilization. |
| $t_{ee}$ | EEPROM read duration. | | 20 ms | Actual time depends on the EEPROM content. |
| topll | EEPROM auto-load 1 completion to Core PLL(s) stable | | 10 ms | |
| tpciplI | De-assertion of PCIe Reset to PCIe PLL stable. | | 5 ms | |
| tpgtrn | PCIe Reset to start of link training. | | 20 ms | According to PCIe specification. |
| tpgres | PCIe Reset to first configuration cycle. | 100 ms | 1 sec | According to PCIe specification. |

# 4.2 Reset Operation

## 4.2.1 Reset Sources

The 82599 reset sources are described in the sections that follow:

### 4.2.1.1 LAN_PWR_GOOD

The 82599 has an internal mechanism for sensing the power pins. Once the power is up and stable, the 82599 creates an internal reset, which acts as a master reset of the entire chip. It is level sensitive, and while it is 0b, all of the registers are held in reset. LAN_PWR_GOOD is interpreted to be an indication that device power supplies are all stable. LAN_PWR_GOOD changes state during system power up.

### 4.2.1.2 PE_RST_N (PCIe Reset)

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

### 4.2.1.3 In-band PCIe Reset

The 82599 generates an internal reset in response to a physical layer message from PCIe or when the PCIe link goes down (entry to polling or detect state). This reset is equivalent to PCI reset in previous (PCI) GbE controllers.

### 4.2.1.4 D3hot to D0 Transition

This is also known as ACPI reset. The 82599 generates an internal reset on the transition from D3hot power state to D0 (caused after configuration writes from D3 to D0 power state). Note that this reset is per function and resets only the function that transitioned from D3hot to D0.

### 4.2.1.5 Function Level Reset (FLR) Capability

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

#### 4.2.1.5.1 FLR in Non-IOV Mode

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

#### 4.2.1.5.2 Physical Function FLR (PFLR)

An FLR reset to the PF function in an IOV mode is equivalent to a FLR 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 the VFMailbox[n] is set following the reset (per VF) to indicate to the VFs that a PF FLR took place. Each VF is responsible to probe this bit (such as after a timeout).

#### 4.2.1.5.3 Virtual Function FLR (VFLR)

A VF operating in an IOV mode can issue a FLR. The VFLR resets the resources allocated to the VF (such as 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 of the *IOV Enable* bit in the IOV structure is equivalent to a VFLR to all the VFs in the same port.

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

### 4.2.1.6 Software Resets

#### 4.2.1.6.1 Software Reset

Software reset is done by writing to the *Device Reset* bit of the Device Control register (CTRL.RST). The 82599 re-reads the per-function EEPROM fields after a software reset. Bits that are not normally read from the EEPROM are reset to their default hardware values.

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

Fields controlled by the LED, SDP and Init3 words of the EEPROM are not reset and not re-read after a software reset.

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

Prior to issuing software reset, the driver needs to execute the master disable algorithm as defined in Section 5.2.5.3.2.

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

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

#### 4.2.1.6.2 Physical Function (PF) Software Reset

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

The procedure for a PF software reset is as follows:

- The PF driver disables master accesses by the device through the master disable mechanism (see Section 5.2.5.3.2). Master disable affects all VFs traffic.
- Execute the procedure described in Section 4.2.2 to synchronize between the PF and VFs.

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

#### 4.2.1.6.3 VF Software Reset

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

#### 4.2.1.6.4 Force TCO

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

#### 4.2.1.7 Link Reset

Also referred to as MAC reset.

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

A link reset is equivalent to a software reset + reset of the MAC. The 82599 re-reads the per-function EEPROM fields after link reset. Bits that are normally read from the EEPROM 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 also generate a link reset.

Prior to issuing link reset, the driver needs to execute the master disable algorithm as defined in Section 5.2.5.3.2.

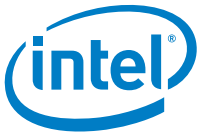
### 4.2.2 Reset in PCI-IOV Environment

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

#### 4.2.2.1 (RSTI)/(RSTD)

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

- One of the following reset cases takes place:
  - LAN Power Good
  - PCIe Reset (PERST and in-band)
  - D3hot --> D0
  - FLR
  - Software reset by the PF
- The 82599 sets the *RSTI* bits in all the VFMailbox registers. Once the reset completes, each VF can read its VFMailbox register to identify a reset in progress.
  - The VF might 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 82599 clears the *RSTI* bits in all the VFMailbox registers and sets the Reset Done (*RSTD*) bits in all the VFMailbox registers.
  - The VF might read the *RSTD* bit to detect that a reset has occurred. The *RSTD* bit is cleared on read.



#### 4.2.2.2 VF Receive Enable — PFVFRE / VF Transmit Enable — PFVFTE

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

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

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

### 4.2.3 Reset Effects

Table 4-4 through Table 4-6 list how resets affect the following registers and logic:

**Table 4-4 Reset Effects — Common Resets**

| Reset Activation | LAN Power Good | PCIe PERST# | In-band PCIe Reset | FW Reset | Force TCO | Notes |
|---|---|---|---|---|---|---|
| EEPROM Read | See Section 6.3.1 | | | | | |
| LTSSM (back to detect/polling) | X | X | X | | | |
| PCIe Link Data Path | X | X | X | | | |
| PCI Configuration Registers RO | X | X | X | | | 9 |
| PCI Configuration Registers RW | X | X | X | | | 9 |
| PCIe Local Registers | X | X | X | | | 8 |
| Data Path | X | X | X | | X | 2 |
| On-die Memories | X | X | X | | X | 7 |
| MAC, PCS, Auto-Negotiation, LinkSec, IPsec | X | X 6 | X 6 | | X | |
| Wake Up (PM) Context | X | 1 | | | | 3 |
| Wake Up/Manageability Control/Status Regs | X | | | | | 4, 5 |

### Table 4-4 Reset Effects — Common Resets (Continued)

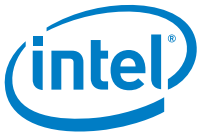
| Reset Activation | LAN Power Good | PCIe PERST# | In-band PCIe Reset | FW Reset | Force TCO | Notes |
|---|---|---|---|---|---|---|
| Manageability Unit | X | | | X | | |
| LAN Disable Strapping Pins | X | | | | | |
| All Other Strapping Pins | X | | | | | |

### Table 4-5 Reset Effects — Per Function Resets

| Reset Activation | D3 or Dr | FLR or PFLR | SW Reset | Link Reset or Exit from LAN Disable | Notes |
|---|---|---|---|---|---|
| EEPROM Read | See Section 6.3.1 | | | | |
| LTSSM (back to detect/polling) | | | | | |
| PCIe Link Data Path | | | | | |
| PCI Configuration Registers RO | | | | | 9 |
| PCI Configuration Registers RW | X | X | | | 9 |
| Data path, Memory Space | X | X | X | X | 2 |
| On-die Memories | X | X | X | | 7 |
| MAC, PCS, Auto-Negotiation, LinkSec, IPsec | X 6 | X 6 | X 6 | X | |
| Virtual Function Resources | X | X | X | | 10 |
| Wake Up (PM) Context | | | | | 3 |
| Wake Up/Manageability Control/Status Regs | | | | | 4,5 |
| Manageability Unit | | | | | |
| Strapping Pins | | | | | |

### Table 4-6 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 |



**Note:** *VFLR* won't clear the VFMAILBOX.VFU bit. This bit should be cleared by a direct write access or by setting PFMailbox.RVFU bit. Refer to Section 8.3.5.1.5 for more details.

### Notes For Table 4-4 through Table 4-6:

1. If AUX_PWR = 0b the wake up context is reset (*PME_Status* and *PME_En* bits should be 0b at reset if the 82599 does not support PME from D3cold).
2. The following register fields do not follow the previous general rules:
   - ESDP registers- reset on LAN Power Good only.
   - LED configuration registers.
   - The *Aux Power Detected* bit in the PCIe Device Status register is reset on LAN Power Good and PCIe Reset only.
   - FLA — reset on LAN Power Good only.
   - RAH/RAL[n, where n>0], MTA[n], VFTA[n], FHFT_*[n], TDBAH/TDBAL, and RDBAH/RDVAL registers have no default value. If the functions associated with the registers are enabled they must be programmed by software. Once programmed, their value is preserved through all resets as long as power is applied.
   - Statistic registers (physical function)
3. The wake up context is defined in the PCI Bus Power Management Interface Specification (sticky bits). It includes:
   - *PME_En* bit of the Power Management Control/Status Register (PMCSR)
   - *PME_Status* bit of the Power Management Control/Status Register (PMCSR)
   - Aux_En in the PCIe registers
   - The device requester ID (since it is required for the PM_PME TLP)

   The shadow copies of these bits in the Wakeup Control Register are treated identically.
4. Refers to bits in the Wake Up Control Register that are not part of the Wake-Up Context (the PME_En and PME_Status bits). Note that the WUFC and WUC registers are not part of the Wake Up Context and are reset as part of the data path. Include also the SW_FW_SYNC and the FWSM registers.
5. The Wake Up Status Registers include the following:
   - Wake Up Status Register
   - Wake Up Packet Length
   - Wake Up Packet Memory
6. The MAC cluster is reset by the appropriate event only if manageability unit is disabled and the host is in a low power state with WoL disabled.
7. The contents of the following memories are cleared to support the requirements of PCIe FLR:
   - The Tx packet buffers
   - The Rx packet buffers
   - IPsec Tx SA tables
   - IPsec Rx SA tables
8. The following registers are part of this group:
   - SWSM
   - GCR (only bit 9 is cleared by this reset while all other fields are cleared at LAN Power Good reset)
   - GSCL_1/GSCL_2
   - GSCN_0/1/2/3
9. Sticky bits and hardware init bits (indicated as HwInit) in the PCI Configuration registers are cleared only by LAN Power Good reset.
10. These registers include:
    - VFEICS
    - VFEIMS
    - VFEIAC
    - VFEIAM
    - VFEITR 0-2
    - VTIVAR0
    - VFIVAR_MISC
    - VFPBACL
    - VFMailbox
11. These registers include:
    - VFEICS

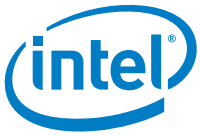
- VFEIMS
- VFEIMC
- VFEIAC
- VFEIAM
- VFEICR
- EITR 0-2
- VTIVAR0
- VFIVAR_MISC
- VFPBACL
- VFMailbox

12. These registers include:
    - Specific VF bits in the FVRE and FVTE are cleared as well
13. These registers include:
    - MSI/MSI-X enable bits
    - BME
    - Error indications
14. Rx and Tx counters might miss proper counting due to VFLR indicating more packets than those ones actually transferred. It could happen if the VFLR happened after counting occurred but before Tx or Rx were completed.

# 4.3 Queue Disable

See Section 4.6.7.1 for details on disabling and enabling an Rx queue.

See Section 4.6.8.1 for details on disabling and enabling a Tx queue.



# 4.4 Function Disable

## 4.4.1 General

For a LAN on Motherboard (LOM) design, it might be desirable for the system to provide BIOS-setup capability for selectively enabling or disabling LAN functions. It enables the end-user more control over system resource-management and avoids conflicts with add-in NIC solutions. The 82599 provides support for selectively enabling or disabling one or both LAN device(s) in the system.

## 4.4.2 Overview

Device presence (or non-presence) must be established early during BIOS execution, 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 82599 LAN-disable mechanism is implemented in order to be compatible with such a solution.

The 82599 provides two mechanisms to disable each of its LAN ports:

- The LANx_DIS_N pins (one pin per LAN port) are sampled on reset to determine the LAN enablement.
- One of the LAN ports can be disabled using EEPROM configuration.

Disabling a LAN port affects the PCI function it resides on. When function 0 is disabled (either LAN0 or LAN1), two different behaviors are possible:

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

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

### Table 4-7 PCI Functions Mapping (Legacy Mode)

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

### Table 4-8 PCI Functions Mapping (Dummy Function Mode)

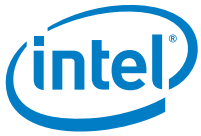
| PCI Function # | LAN Function Select | Function 0 | Function 1 |
|---|---|---|---|
| Both LAN functions are enabled. | 0 | LAN 0 | LAN 1 |
| | 1 | LAN 1 | LAN 0 |
| LAN 0 is disabled. | 0 | Dummy | LAN1 |
| | 1 | LAN 1 | Disable |
| LAN 1 is disabled. | 0 | LAN 0 | Disable |
| | 1 | Dummy | LAN 0 |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

The following rules apply to function disable:

- When function 0 is disabled in legacy mode, the LAN port associated originally with function 1 appears in function 0. Function 1 disappears from the PCI configuration space.
- When function 0 is disabled in dummy function mode, it is converted into a dummy PCI function. Function 1 is not affected.
- When function 1 is disabled, it disappears from the PCI configuration space.
- The disabled LAN port is still available for manageability purposes if disabled through the EEPROM mechanism. The disabled LAN port is not available for manageability purposes if disabled through the pin mechanism.
- Dummy function mode should not be used in PCI IOV mode (since PF0 is required to support certain functionality)

The following EEPROM bits control function disable:

- One PCI function can be enabled or disabled according to the EEPROM *LAN PCI Disable* bit.
- The *LAN Disable Select* EEPROM field indicates which function is disabled.
- The *LAN Function Select* EEPROM bit defines the correspondence between LAN Port and PCI function
- The *Dummy Function Enable* EEPROM bit enables the dummy function mode. Default value is disabled.



## 4.4.3 Control Options

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

LAN0 **or** LAN 1 can be disabled in the EEPROM by setting the *LAN PCI Disable* bit in the PCIe Control 2 word at offset 0x05. The *LAN Disable Select* bit in the same word in the EEPROM selects which LAN is disabled. Furthermore, if the LAN port at function 0 is disabled, the *Dummy Function Enable* bit in the same word chooses between filling the disabled function by a dummy function, or moving the other LAN port to function 0.

**Note:** Mapping LAN0 and LAN1 to PCI function 0 and PCI function 1 is controlled by the EEPROM *LAN Function Select* bit in the PCIe Control 2 word at offset 0x05.

LAN0 **and** LAN 1 can be disabled on the board level by driving the LAN0_Dis_N and LAN1_Dis_N pins to low. These I/O pins have internal weak pull-up resistors so leaving them unconnected or driving them to high enables the respective LAN port. These pins are strapping options, sampled at LAN Power Good, PCIe reset or in-band PCIe reset.

## 4.4.4 Event Flow for Enable/Disable Functions

This section describes the driving levels and event sequence for device functionality. Following a Power on Reset / LAN Power Good/ PCIe Reset/ In-Band Reset, the LANx_DIS_N signals should be driven high (or left open) for normal operation. If any of the LAN functions are not required statically, its associated disable strapping pin can be tied statically to low.

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

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

1. PCIe is established following PCIe reset.
2. BIOS recognizes that a LAN function in the 82599 should be disabled.
3. The BIOS drives the LANx_DIS_N signal to the low level.
4. BIOS issues PCIe reset or an in-band PCIe reset.
5. As a result, the 82599 samples the LANx_DIS_N signals and disables the LAN function and issues an internal reset to this function.
6. BIOS might start with the device enumeration procedure (the disabled LAN function is invisible — changed to dummy function).
7. Proceed with normal operation.
8. Re-enable could be done by driving the LANx_DIS_N signal high and then requesting the user to issue a warm boot to initialize new bus enumeration.

#### 4.4.4.2 Multi-Function Advertisement

If one of the LAN devices is disabled and function 0 is the only active function, the 82599 is no longer a multi-function device. The 82599 normally reports a 0x80 in the PCI configuration header, indicating multi-function capability. However, if a LAN is disabled and only function 0 is active, the 82599 reports a 0x0 in this field to signify single-function capability.

#### 4.4.4.3 Interrupt Utilization

When both LAN devices are enabled, the 82599 uses the PCI legacy interrupts of both ports for interrupt reporting. The EEPROM 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 EEPROM configuration must be set accordingly. Under these circumstances, the *Interrupt Pin* field of the PCI configuration header always reports a value of 0x1, indicating INTA# pin usage, which means legacy PCI interrupt of port A is used.

#### 4.4.4.4 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 an EEPROM 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 82599 appears as a single-function device, the common power value, if selected, reports 0x0 (undefined value), as common power is undefined for a single-function device.

## 4.5 Device Disable

### 4.5.1 Overview

When both LAN ports are disabled following a Power on Reset / LAN Power Good/ PCIe Reset/ In-Band Reset, the LANx_DIS_N signals should be tied statically to low. In this state the device is disabled, LAN ports are powered down, all internal clocks are shut, and the PCIe connection is powered down (similar to L2 state).

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

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

1. PCIe is established following PCIe reset.
2. BIOS recognizes that the 82599 should be disabled.
3. The BIOS drives the LANx_DIS_N signals to the low level.
4. BIOS issues PCIe reset or an in-band PCIe reset.
5. As a result, the 82599 samples the LANx_DIS_N signals and disables the LAN ports and the PCIe connection.
6. Re-enable can be done by driving at least one of the LANx_DIS_N signals high and then issuing a PCIe reset to restart the device.

## 4.6 Software Initialization and Diagnostics

### 4.6.1 Introduction

This section discusses general software notes for the 82599, especially initialization steps. This includes:

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

### 4.6.2 Power-Up State

When the 82599 powers up, it automatically reads the EEPROM. The EEPROM contains sufficient information to bring the link up and configure the 82599 for manageability and/or APM wakeup. However, software initialization is required for normal operation.

## 4.6.3 Initialization Sequence

The following sequence of commands is typically issued to the device by the software device driver in order to initialize the 82599 for normal operation. The major initialization steps are:

1. Disable interrupts.
2. Issue global reset and perform general configuration (see Section 4.6.3.2).
3. Wait for EEPROM auto read completion.
4. Wait for DMA initialization done (RDRXCTL.DMAIDONE).
5. Setup the PHY and the link (see Section 4.6.4).
6. Initialize all statistical counters (see Section 4.6.5).
7. Initialize receive (see Section 4.6.7).
8. Initialize transmit (see Section 4.6.8).
9. Enable interrupts (see Section 4.6.3.1).

### 4.6.3.1 Interrupts During Initialization

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

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

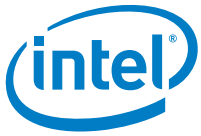
After initialization completes, a typical driver enables the desired interrupts by writing to the IMS register.

### 4.6.3.2 Global Reset and General Configuration

**Note:** Global Reset = software reset + link reset.

Device initialization typically starts with a software reset that puts the device into a known state and enables the device driver to continue the initialization sequence. Following a Global Reset the Software driver should wait at least 10msec to enable smooth initialization flow.

To enable flow control, program the FCTTV, FCRTL, FCRTH, FCRTV and FCCFG registers. If flow control is not enabled, these registers should be written with 0x0. If Tx flow control is enabled then Tx CRC by hardware should be enabled as well (HLREG0.TXCRCEN = 1b). Refer to Section 3.7.7.3.2 through Section 3.7.7.3.5 for the recommended setting of the Rx packet buffer sizes and flow control thresholds. Note that FCRTH[n].RTH fields must be set by default regardless if flow control is enabled or not. Typically, FCRTH[n] default value should be equal to RXPBSIZE[n]-0x6000. FCRTH[n].FCEN should be set to 0b if flow control is not enabled as all the other registers previously indicated.



The link interconnect 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 EEPROM by applying the appropriate settings to the link interconnect block.

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

### 4.6.4.1 BX/ SGMII Link Setup Flow

1. BX link electrical setup is done according to EEPROM configuration to set the analog interface to the appropriate setting.
2. Configure the *Link Mode Select* field in the AUTOC register to the appropriate operating mode.
3. Configure any interface fields in the SERDESC register if necessary.
4. Restart the link using the *Restart Auto Negotiation* field in the AUTOC register.
5. Verify correct link status (sync, link_up, speed) using the LINKS register.

### 4.6.4.2 XAUI / BX4 / CX4 / SFI Link Setup Flow

1. XAUI / BX4 / CX4 / SFI link electrical setup is done according to EEPROM configuration to set the analog interface to the appropriate setting.
2. Configure the *Link Mode Select* field in the AUTOC register, AUTOC.10G_PARALLEL_PMA_PMD and AUTOC2.10G_PMA_PMD_Serial to the appropriate operating mode.
3. Configure any interface fields in the SERDESC register if necessary.
4. Restart the link using the *Restart Auto Negotiation* field in the AUTOC register.
5. Verify correct link status (align, link_up, speed) using the LINKS register.

### 4.6.4.3 KX / KX4 / KR Link Setup Flow Without Auto-Negotiation

1. KX / KX4 / KR link electrical setup is done according to EEPROM configuration to set the analog interface to the appropriate setting.
2. Configure the *Link Mode Select* field in the AUTOC register, AUTOC.10G_PARALLEL_PMA_PMD and AUTOC2.10G_PMA_PMD_Serial to the appropriate operating mode.
3. Configure any interface fields in the SERDESC register if necessary.
4. Restart the link using the *Restart Auto Negotiation* field in the AUTOC register.
5. Verify correct link status (sync, align, link_up, speed) using the LINKS register.

#### 4.6.4.4 KX / KX4 / KR Link Setup Flow With Auto-Negotiation

1. KX / KX4 / KR link electrical setup is done according to EEPROM configuration to set the analog interface to the appropriate setting.
2. Configure the *Link Mode Select* field in the AUTOC register, AUTOC.10G_PARALLEL_PMA_PMD and AUTOC2.10G_PMA_PMD_Serial to the appropriate operating mode.
3. Configure any interface fields in the SERDESC register if necessary.
4. Configure the *KX_Support* field and any other auto-negotiation related fields in the AUTOC register.
5. Restart the link using the *Restart Auto Negotiation* field in the AUTOC register.
6. Verify correct link status (sync, align, link_up, speed) using the LINKS register.

### 4.6.5 Initialization of Statistics

Statistics registers are hardware-initialized to values as detailed in each particular register's description. The initialization of these registers begins upon transition to 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 be completed within 1 ms of this transition. Note that access to statistics registers prior to this interval might return indeterminate values.

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

Queue counters are mapped using the RQSMR registers for Rx queues, and TQSM registers for Tx queues. Refer to Section 8.2.3.23.71 for RQSMR setup, and Section 8.2.3.23.73 for TQSM setup. Note that if software requires the queue counters, the RQSMR and TQSM registers must be re-programmed following a device reset.

### 4.6.6 Interrupt Initialization

**Operating with Legacy or MSI Interrupts:**

- The software driver associates between Tx and Rx interrupt causes and the EICR register by setting the IVAR[n] registers.
- Program SRRCTL[n].RDMTS (per receive queue) if software uses the receive descriptor minimum threshold interrupt.
- All interrupts should be set to 0b (no auto clear in the EIAC register). Following an interrupt, software might 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 EITR[n] and GPIE according to the preferred mode of operation.

- Software clears EICR by writing all ones to clear old interrupt causes.
- Software enables the required interrupt causes by setting the EIMS register.

**Operating with MSI-X:**

- The operating system / BIOS sets the hardware to MSI-X mode and programs the MSI-X table as part of the device enumeration procedure.
- The software driver associates between interrupt causes and MSI-X vectors and the throttling timers EITR[n] by programming the IVAR[n] and IVAR_MISC registers.
- Program 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 the auto mask in the EIAM and EIAM[n] registers according to the preferred mode of operation.
- Set the interrupt throttling in EITR[n] and GPIE according to the preferred mode of operation.
- Software enables the required interrupt causes by setting the EIMS[n] registers.

## 4.6.7 Receive Initialization

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

- Receive Address (RAL[n] and RAH[n]) for used addresses.
- 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 EEPROM or from any other means (for example, it could be stored anywhere in the EEPROM or even in the platform PROM for LOM design).

Set up the Multicast Table Array (MTA) registers. This entire table should be zeroed and 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. This entire table should be zeroed and only the desired VLAN addresses should be permitted (by writing 0x1 to the corresponding bit location). Set the VLNCTRL.VFE bit if VLAN filtering is required.

Initialize the flexible filters 0…5 — Flexible Host Filter Table registers (FHFT).

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

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

Note that RDRXCTL.CRCStrip and HLREG0.RXCRCSTRP must be set to the same value. At the same time the RDRXCTL.RSCFRSTSIZE should be set to 0x0 as opposed to its hardware default.

Program RXPBSIZE, MRQC, PFQDE, RTRUP2TC, MFLCN.RPFCE, and MFLCN.RFCE according to the DCB and virtualization modes (see Section 4.6.11.3).

Enable jumbo reception by setting HLREG0.JUMBOEN in one of the following two cases:

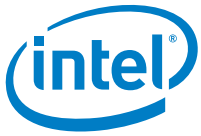
1. Jumbo packets are expected. Set the MAXFRS.MFS to expected max packet size.
2. LinkSec encapsulation is expected.

In these cases set the MAXFRS.MFS bit in the Max Frame Size register to the expected maximum packet size plus 32 bytes for the LinkSec encapsulation. Refer to Section 8.2.3.22.13 for details about the correct handling of VLAN and double VLAN headers.

Enable receive coalescing if required as described in Section 4.6.7.2.

The following should be done per each receive queue:

1. Allocate a region of memory for the receive descriptor list.
2. Receive buffers of appropriate size should be allocated and pointers to these buffers should be stored in the descriptor ring.
3. Program the descriptor base address with the address of the region (registers RDBAL, RDBAL).
4. Set the length register to the size of the descriptor ring (register RDLEN).
5. Program SRRCTL associated with this queue according to the size of the buffers and the required header control.
6. If header split is required for this queue, program the appropriate PSRTYPE for the appropriate headers.
7. Program RSC mode for the queue via the RSCCTL register.
8. Program RXDCTL with appropriate values including the queue *Enable* bit. Note that packets directed to a disabled queue are dropped.
9. Poll the RXDCTL register until the *Enable* bit is set. The tail should not be bumped before this bit was read as 1b.
10. Bump the tail pointer (RDT) to enable descriptors fetching by setting it to the ring length minus one.
11. Enable the receive path by setting RXCTRL.RXEN. This should be done only after all other settings are done following the steps below.
    - Halt the receive data path by setting SECRXCTRL.RX_DIS bit.
    - Wait for the data paths to be emptied by HW. Poll the SECRXSTAT.SECRX_RDY bit until it is asserted by HW.
    - Set RXCTRL.RXEN
    - Clear the SECRXCTRL.SECRX_DIS bits to enable receive data path



— If software uses the receive descriptor minimum threshold Interrupt, that value should be set.

Set bit 16 of the CTRL_EXT register and clear bit 12 of the DCA_RXCTRL[n] register[n].

### 4.6.7.1 Dynamic Enabling and Disabling of Receive Queues

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

#### 4.6.7.1.1 Enabling

Follow the per queue initialization described in the previous section.

#### 4.6.7.1.2 Disabling

- Disable the routing of packets to this queue by re-configuring the Rx filters. In order to ensure that the receive packet buffer does not contain any packets to the specific queue it is required to follow the Flushing the Packet Buffers procedure described later in this section.
- If RSC is enabled on the specific queue and VLAN strip is enabled as well then wait 2 ITR expiration time (ensure all open RSC are completed).
- Disable the queue by clearing the RXDCTL.ENABLE bit. The 82599 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 82599 completes the current buffer write. If the packet spreads over more than one data buffer, all subsequent buffers are not written.
- The 82599 clears the RXDCTL.ENABLE bit only after all pending memory accesses to the descriptor ring are done. The driver should poll this bit before releasing the memory allocated to this queue.
- Once the RXDCTL.ENABLE bit is cleared the driver should wait additional amount of time
(~100 μs) before releasing the memory allocated to this queue.

Software might re-configure the Rx filters back to the original setting.The Rx path can be disabled only after all the receive queues are disabled.

#### 4.6.7.1.3 Flushing the Packet Buffers

Because 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, a Virtual Machine (VM) goes down and a different VM gets the queue.

The 82599 provides a mechanism for software to identify when the packet buffers were drained of such stale packets. The read-only RXMEMWRAP register contains a set of counters (one per packet buffer) that increments each time a buffer is overtaken by the tail pointer. Software must read a counter repeatedly until its count is incremented at least by two, to insure that the buffer made at least one complete wrap-around. Software should also check the *Empty* bit for the counter. If the bit is set, the buffer is empty and there is no further need to sample the buffer counter.

### 4.6.7.2 RSC Enablement

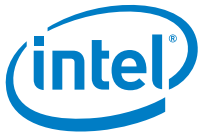
RSC enablement as well as RSC parameter settings are assumed as static. It should be enabled prior to reception and can be disabled only after the relevant Rx queue(s) are disabled.

#### 4.6.7.2.1 Global Setting

- In SR-IOV mode RSC must be disabled globally by setting the RFCTL.RSC_DIS bit. In this case the following steps in this section are not required.
- Enable global CRC stripping via HLREG0 (hardware default setting)
- Software should set the RDRXCTL.RSCACKC bit that forces RSC completion on any change of the ACK bit in the Rx packet relative to the RSC context.
- The SRRCTL[n].BSIZEHEADER (header buffer size) must be larger than the packet header (even if header split is not enabled). A minimum size of 128 bytes for the header buffer addresses this requirement.
- NFS packet handling:
  - NFS header filtering should be disabled if NFS packets coalescing are required (at the TCP layer). The RFCTL.NFSW_DIS and RFCTL.NFSR_DIS bits should be set to 1b. Furthermore, the PSR_type1 bit in the PSRTYPE[n] registers (header split on NFS) must be turned off in all queues.
  - Both RFCTL.NFSW_DIS and RFCTL.NFSR_DIS bits should be cleared to 0b if NFS coalescing is not required. The PSR_type1 can be set per queue according to the required header split.

#### 4.6.7.2.2 Per Queue Setting

- Enable RSC and configure the maximum allowed descriptors per RSC by setting the *MAXDESC* and *RSCEN* fields in the RSCCTL[n].
- Use non-legacy descriptor type by setting SRRCTL[n].DESCTYPE to non-zero values.
- TCP header recognition — the PSR_type4 in the PSRTYPE[n] registers should be set.
- Interrupt setting:
  - Interrupt moderation must be enabled by setting EITR[n].ITR_INTERVAL to a value greater than zero. Note that the ITR Interval must be larger than the RSC Delay. Also, if the CNT_WDIS bit is cleared (write enable), then the ITR counter should be set to zero.



— 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 encounters many instances that RSC did not complete as expected following EITR interrupt assertion, *RSC Delay* might need to be increased.

— Map the relevant Rx queues to an interrupt by setting the relevant IVAR registers.

## 4.6.8 Transmit Initialization

- Program the HLREG0 register according to the required MAC behavior.
- Program TCP segmentation parameters via registers DMATXCTL (while maintaining *TE* bit cleared), DTXTCPFLGL, and DTXTCPFLGH; and DCA parameters via DCA_TXCTRL.
- Set RTTDCS.ARBDIS to 1b.
- Program DTXMXSZRQ, TXPBSIZE, TXPBTHRESH, MTQC, and MNGTXMAP, according to the DCB and virtualization modes (see Section 4.6.11.3).
- Clear RTTDCS.ARBDIS to 0b.

The following steps should be done once per 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, TDBAH).
3. Set the length register to the size of the descriptor ring (TDLEN).
4. Program the TXDCTL register with the desired TX descriptor write back policy (see Section 8.2.3.9.10 for recommended values).
5. If needed, set TDWBAL/TWDBAH to enable head write back.
6. Enable transmit path by setting DMATXCTL.TE. This step should be executed only for the first enabled transmit queue and does not need to be repeated for any following queues.
7. Enable the queue using TXDCTL.ENABLE. Poll the TXDCTL register until the *Enable* bit is set.

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

### 4.6.8.1 Dynamic Enabling and Disabling of Transmit Queues

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

#### 4.6.8.1.1 Enabling

Follow the per queue initialization described in the previous section.

#### 4.6.8.1.2 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. Therefore, its *RS* bit must be set or WTHRESH must be greater than zero. If none of these 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 *DD* bit in ring or polling the Head_WB content). It might be required to flush the transmit queue by setting TXDCTL[n].SWFLSH if the *RS* bit in the last fetched descriptor is not set or if WTHRESH is greater than zero.
5. Disable the queue by clearing TXDCTL.ENABLE.
6. Any packets waiting for transmission in the packet buffer would still be sent at a later time.

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

## 4.6.9 FCoE Initialization Flow

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

- The FCoE DDP context table should be initialized clearing the FCBUFF.Valid and FCFLT.Valid bits of all contexts.
- EType Queue Filter — ETQF[n]: Select a filter by setting the FCoE bit. The *EType* field should be set to 0x8906 (FCoE Ethernet Type). If FCoE traffic is expected on multiple VLAN priorities then multiple ETQF filters might be required.
- EType Queue Select — ETQS[n]: Each ETQF filter is associated to a queue select register. The ETQS registers can be used to direct the FCoE traffic to specific receive queues. Up to one queue per Traffic Class (TC) as programmed in the ETQF.
- Multiple receive queues can be enabled by setting FCRECTL.ENA and programming the FCRETA[n] registers.
- Low Latency Interrupts (LLI) for critical FCoE frames can be enabled by setting the FCRXCTRL.FCOELLI bit.
- Set the RDRXCTL.FCOE_WRFIX bit that forces a DDP write exchange context closure after receiving the last packet in a sequence with an active *Sequence Initiative* bit in the *F_CTL* field.
- Follow the rules indicated in Section 7.13.2.1 and Section 7.13.3.1 for Tx and Rx cross functionality requirements. These sections include requirements on Ethernet CRC and padding handling, LinkSec offload, Legacy Rx buffers, and more.
- Software is not expected to access the EOF and SOF setting. If there is some error in the implementation, these settings might need to be modified. If so, software must program the REOFF and TEOFF registers by the same values. This same rule applies to the RSOFF and TSOFF registers.

# 4.6.10 Virtualization Initialization Flow

## 4.6.10.1 VMDq Mode

### 4.6.10.1.1 Global Filtering and Offload Capabilities

- Select one of the VMDQ pooling methods — MAC/VLAN filtering for pool selection and either DCB or RSS for the queue in pool selection. MRQC.Multiple Receive Queues Enable = 1000b, 1010b, 1011b, 1100b, or 1101b.
- DCB should be initiated as described in Section 4.6.11. In RSS mode, the RSS key (RSSRK) and redirection table (RETA) should be programmed. Note that the redirection table is common to all the pools and only indicates the queue inside the pool to use once the pool is chosen. Each pool can decide if it uses DCB.
- Configure PFVTCTL to define the default pool.
- Enable replication via PFVTCTL.Rpl_En.
- If needed, enable padding of small packets via HLREG0.TXPADEN.
- The MPSAR registers are used to associate Ethernet MAC addresses to pools. Using the MPSAR registers, software must reprogram RAL[0] and RAH[0] by their values (software could read these registers and then write them back with the same content).

### 4.6.10.1.2 Mirroring Rules

For each mirroring rule to be activated:

- Set the type of traffic to be mirrored in the PFMRCTL[n] register.
- Set the mirror pool in PFMRCTL[n].MP.
- For pool mirroring, set the PFMRVM[n] register with the pools to be mirrored.
- For VLAN mirroring, set PFMRVLAN[n] with the indexes from the PFVLVF registers of the VLANs to be mirrored.

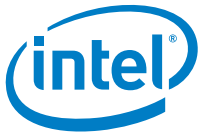
### 4.6.10.1.3 Security Features

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

#### 4.6.10.1.4 Per Pool Settings

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

- Associate the unicast Ethernet MAC address of the VM by enabling the pool in the MPSAR registers.
- If all the Ethernet MAC addresses are used, the Unicast Hash Table (PFUTA) can be used. Pools servicing VMs whose address is in the hash table should be declared as so by setting PFVML2FLT.ROPE. Packets received according to this method didn't pass perfect filtering and are indicated as such.
- Enable the pool in all the 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 PFVML2FLT.ROMPE. Packets received according to this method didn't pass perfect filtering and are indicated as such.
- Define whether this VM should get all multicast/broadcast packets in the same VLAN via PFVML2FLT.MPE and PFVML2FLT.BAM, and whether it should accept untagged packets via PFVML2FLT.AUPE.
- Enable the pool in each 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.
- 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 offloads application such as VLAN stripor LinkSec decrypting.
- Set which header split is required via the PSRTYPE register.
- In RSS mode, define if the pool uses RSS via the proper MRQC.MRQE mode.
- Enable the Pool in the PFVFRE register to allow Rx Filtering
- To Enable Multiple Tx queues, Set the MTQC as described in Section 7.2.1.2.1
- Enable the Pool in the PFVFTE register to allow Tx Filtering
- Enable Rx and Tx queues as described in Section 4.6.7 and Section 4.6.8.
- For each Rx queue a drop/no drop flag can be set in SRRCTL.DROP_EN and via the PFQDE register, controlling the behavior if no receive buffers are available in the queue to receive packets. The usual behavior is to allow drop in order to avoid head of line blocking. Setting PFQDE per queue is made by using the *Queue Index* field in the PFQDE register.



## 4.6.10.2 IOV Initialization

### 4.6.10.2.1 Physical Function (PF) Driver Initialization

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

**Note:** The link setup might include the authentication process (802.1X or other), and setup of the LinkSec channel, and setup of the DCB parameters.

In IOV mode, VMDq + RSS mode is not available.

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

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

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

#### 4.6.10.2.1.1 VF Specific Reset Coordination

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

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

### 4.6.10.2.2 VF Driver Initialization

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

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

### 4.6.10.2.3 Full Reset Coordination

A mechanism is provided to synchronize reset procedures between the PF and the VFs. It is provided specifically for PF software reset but can be used in other reset cases. These reset cases are described in the following procedure.

One of the following reset cases takes place:

- LAN Power Good

- PCIe reset (PERST and in-band)
- D3hot --> D0
- FLR
- Software reset by the PF

The 82599 sets the *RSTI* bits in all the 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 82599 clears the *RSTI* bits in all the VFMailbox registers and sets the *RSTD* (*Reset Done*) bits in all the VFMailbox registers.

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

## 4.6.11 DCB Configuration

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

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

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

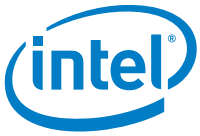
Rate parameters and bandwidth allocation to VMs can be modified on the fly without disturbing traffic flows.

### 4.6.11.1 CPU Latency Considerations

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

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

Each time the CPU is in low power, received packets need to be stored (or dropped) in the 82599 for the duration of the exit time. Given 64 KB Rx packet buffers per TC in the 82599, Priority Flow Control (PFC) does not spread (or a packet is not dropped) provided that the CPU exits its low power state within
50 µs.



### 4.6.11.2 Link Speed Change Procedure

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

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

### 4.6.11.3 Initial Configuration Flow

Only the following configuration modes are allowed.

#### 4.6.11.3.1 General Case: DCB-on, VT-on

1. Configure packet buffers, queues, and traffic mapping:
   - 8 TCs mode — Packet buffer size and threshold, typically

     RXPBSIZE[0-7].SIZE=0x40

     TXPBSIZE[0-7].SIZE=0x14

     but non-symmetrical sizing is also allowed (see Section 8.2.3.9.13 for rules)

     TXPBTHRESH.THRESH[0-7]=TXPBSIZE[0-7].SIZE — Maximum expected Tx packet length in this TC
   - 4 TCs mode — Packet buffer size and threshold, typically

     RXPBSIZE[0-3].SIZE=0x80, RXPBSIZE[[4-7].SIZE=0x0

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

     but non-symmetrical sizing among TCs[0-3] is also allowed (see Section 8.2.3.9.13 for rules)

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

     TXPBTHRESH.THRESH[4-7]=0x0
   - Multiple Receive and Transmit Queue Control (MRQC and MTQC)
     - Set MRQC.MRQE to 1xxxb, with the three least significant bits set according to the number of VFs, TCs, and RSS mode as described in Section 8.2.3.7.12.
     - Set both *RT_Ena* and *VT_Ena* bits in the MTQC register.
     - Set MTQC.NUM_TC_OR_Q according to the number of TCs/VFs enabled as described in Section 8.2.3.7.16.
   - Set the PFVTCTL.VT_Ena (as the MTQC.VT_Ena)
   - Queue Drop Enable (PFQDE) — In SR-IO the *QDE* bit should be set to 1b in the PFQDE register for all queues. In VMDq mode, the *QDE* bit should be set to 0b for all queues.

— Split receive control (SRRCTL[0-127]): Drop_En=1 — drop policy for all the queues, in order to avoid crosstalk between VMs

— Rx User Priority (UP) to TC (RTRUP2TC)

— Tx UP to TC (RTTUP2TC)

— DMA TX TCP Maximum Allowed Size Requests (DTXMXSZRQ) — set Max_byte_num_req = 0x010 = 4 KB

2. Enable PFC and disable legacy flow control:

— Enable transmit PFC via: FCCFG.TFCE=10b

— Enable receive PFC via: MFLCN.RPFCE=1b

— Disable receive legacy flow control via: MFLCN.RFCE=0b

— Refer to Section 8.2.3.3 for other registers related to flow control

3. Configure arbiters, per TC[0-1]:

— Tx descriptor plane T1 Config (RTTDT1C) per queue, via setting RTTDQSEL first. Note that the RTTDT1C for queue zero must always be initialized.

— Tx descriptor plane T2 Config (RTTDT2C[0-7])

— Tx packet plane T2 Config (RTTPT2C[0-7])

— Rx packet plane T4 Config (RTRPT4C[0-7])

4. Enable TC and VM arbitration layers:

— Tx Descriptor plane Control and Status (RTTDCS), bits:

TDPAC=1b, VMPAC=1b, TDRM=1b, BDPM=1b, BPBFSM=0b

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

— Rx Packet Plane Control and Status (RTRPCS): RAC=1b, RRM=1b

5. Set the SECTXMINIFG.SECTXDCB field to 0x1F.

#### 4.6.11.3.2 DCB-On, VT-Off

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

- Configure packet buffers, queues, and traffic mapping:

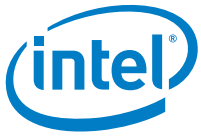
— MRQC and MTQC

  - Set MRQE to 0xxxb, with the three least significant bits set according to the number of TCs and RSS mode
  - Set *RT_Ena* bit and clear the *VT_Ena* bit in the MTQC register.
  - Set MTQC.NUM_TC_OR_Q according to the number of TCs enabled

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

- Allow no-drop policy in Rx:

— PFQDE: The *QDE* bit should be set to 0b in the PFQDE register for all queues enabling per queue policy by the SRRCTL[n] setting.



  - Split Receive Control (SRRCTL[0-127]): The *Drop_En* bit should be set per receive queue according to the required drop / no-drop policy of the TC of the queue.
- Tx descriptor plane control and status (RTTDCS) bits:
  - TDPAC=1b, VMPAC=1b, TDRM=1b, BDPM=0b if Tx rate limiting is not enabled and 1b if Tx rate limiting is enabled, BPBFSM=0b.
- Disable VM arbitration layer:
  - Clear RTTDT1C register, per each queue, via setting RTTDQSEL first
  - RTTDCS.VMPAC=0b

#### 4.6.11.3.3 DCB-Off, VT-On

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

- Disable multiple packet buffers and allocate all queues to PB0:
  - RXPBSIZE[0].SIZE=0x200, RXPBSIZE[1-7].SIZE=0x0
  - TXPBSIZE[0].SIZE=0xA0, TXPBSIZE[1-7].SIZE=0x0
  - TXPBTHRESH.THRESH[0]=0xA0 — Maximum expected Tx packet length in this TC TXPBTHRESH.THRESH[1-7]=0x0
  - MRQC and MTQC
    - Set MRQE to 1xxxb, with the three least significant bits set according to the number of VFs and RSS mode
    - Clear *RT_Ena* bit and set the *VT_Ena* bit in the MTQC register.
    - Set MTQC.NUM_TC_OR_Q according to the number of VFs enabled
  - Set PFVTCTL.VT_Ena (as the MRQC.VT_Ena)
  - Rx UP to TC (RTRUP2TC), UPnMAP=0b, n=0,...,7
  - Tx UP to TC (RTTUP2TC), UPnMAP=0b, n=0,...,7
  - DMA TX TCP Maximum Allowed Size Requests (DTXMXSZRQ) — set Max_byte_num_req = 0xFFF = 1 MB
- Disable PFC and enabled legacy flow control:
  - Disable receive PFC via: MFLCN.RPFCE=0b
  - Enable transmit legacy flow control via: FCCFG.TFCE=01b
  - Enable receive legacy flow control via: MFLCN.RFCE=1b
- Configure VM arbiters only, reset others:
  - Tx Descriptor Plane T1 Config (RTTDT1C) *per pool*, via setting RTTDQSEL first for the pool index. Clear RTTDT1C for other queues. Note that the RTTDT1C for queue zero must always be initialized.
  - Clear RTTDT2C[0-7] registers
  - Clear RTTPT2C[0-7] registers

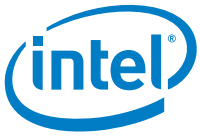
  - Clear RTRPT4C[0-7] registers
- Disable TC arbitrations while enabling the packet buffer free space monitor:
  - Tx Descriptor Plane Control and Status (RTTDCS), bits:

    TDPAC=0b, VMPAC=1b, TDRM=0b, BDPM=1b, BPBFSM=0b
  - Tx Packet Plane Control and Status (RTTPCS): TPPAC=0b, TPRM=0b, ARBD=0x224
  - Rx Packet Plane Control and Status (RTRPCS): RAC=0b, RRM=0b

#### 4.6.11.3.4 DCB-Off, VT-Off

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

- Disable multiple packet buffers and allocate all queues and traffic to PB0:
  - RXPBSIZE[0].SIZE=0x200, RXPBSIZE[1-7].SIZE=0x0
  - TXPBSIZE[0].SIZE=0xA0, TXPBSIZE[1-7].SIZE=0x0
  - TXPBTHRESH.THRESH[0]=0xA0 — Maximum expected Tx packet length in this TC TXPBTHRESH.THRESH[1-7]=0x0
  - MRQC and MTQC
    - Set MRQE to 0xxxb, with the three least significant bits set according to the RSS mode
    - Clear both *RT_Ena* and *VT_Ena* bits in the MTQC register.
    - Set MTQC.*NUM_TC_OR_Q* to 00b.
  - Clear *PFVTCTL.VT_Ena* (as the MRQC.VT_Ena)
  - Rx UP to TC (RTRUP2TC), UPnMAP=0b, n=0,...,7
  - Tx UP to TC (RTTUP2TC), UPnMAP=0b, n=0,...,7
  - DMA TX TCP Maximum Allowed Size Requests (DTXMXSZRQ) — set Max_byte_num_req = 0xFFF = 1 MB
- Allow no-drop policy in Rx:
  - PFQDE: The *QDE* bit should be set to 0b in the PFQDE register for all queues enabling per queue policy by the SRRCTL[n] setting.
  - Split Receive Control (SRRCTL[0-127]): The *Drop_En* bit should be set per receive queue according to the required drop / no-drop policy of the TC of the queue.
- Disable PFC and enable legacy flow control:
  - Disable receive PFC via: MFLCN.RPFCE=0b
  - Enable receive legacy flow control via: MFLCN.RFCE=1b
  - Enable transmit legacy flow control via: FCCFG.TFCE=01b
- Reset all arbiters:
  - Clear RTTDT1C register, per each queue, via setting RTTDQSEL first



  - Clear RTTDT2C[0-7] registers
  - Clear RTTPT2C[0-7] registers
  - Clear RTRPT4C[0-7] registers
- Disable TC and VM arbitration layers:
  - Tx Descriptor Plane Control and Status (RTTDCS), bits:

    TDPAC=0b, VMPAC=0b, TDRM=0b, BDPM=1b, BPBFSM=1b
  - Tx Packet Plane Control and Status (RTTPCS): TPPAC=0b, TPRM=0b, ARBD=0x224
  - Rx Packet Plane Control and Status (RTRPCS): RAC=0b, RRM=0b

### 4.6.11.4 Transmit Rate Scheduler

In some applications it might be useful to setup rate limiters on Tx queues for other usage models (rate-limiting VF traffic for instance). In all cases, setting a rate limiter on Tx queue N to a TargetRate requires the following settings:

**Global Setting**

- The Transmit Rate-scheduler memory for all transmit queues must be cleared before rate limiting is enabled on any queue. This memory is accessed by the RTTBCNRC register mapped by the RTTDQSEL.TXDQ_IDX.
- Set global transmit compensation time to the MMW_SIZE in RTTBCNRM register. Typically MMW_SIZE=0x014 if 9.5 KB (9728-byte) jumbo is supported and 0x004 otherwise.

**Per Queue Setting**

- Select the requested queue by programming the queue index - RTTDQSEL.TXQ_IDX
- Program the desired rate as follow:
  - Compute the Rate_Factor which equals Link_Speed / Target_Rate. Link_Speed could be either 10 Gb/s or 1 Gb/s. Note that the Rate_Factor is composed of an integer number plus a fraction. The integer part is a 10 bit number field and the fraction part is a 14 bit binary fraction number.
  - Integer (Rate_Factor) is programmed by the RTTBCNRC.RF_INT[9:0] field
  - Fraction (Rate_Factor) is programmed by the RTTBCNRC.RF_DEC[13:0] field. It equals RF_DEC[13] * $2^{-1}$ + RF_DEC[12] * $2^{-2}$ + ... + RF_DEC[0] * $2^{-14}$
- Enable Rate Scheduler by setting the RTTBCNRC. RS_ENA

Numerical Example

- Target_Rate = 240 Mb/s; Link_Speed = 10 Gb/s
- Rate_Factor = 10 / 0.24 = 41.6666... = 101001.10101010101011b
- RF_DEC = 10101010101011b; RF_INT = 0000101001b
- Therefore, set RTTBCNRC to 0x800A6AAB

**Note:** The IPG pacing feature is a parallel feature to the Tx rate scheduler where IPG pacing is applied to the entire Tx data flow while the Tx rate scheduler is applied separately to each Tx queue. Therefore, if a single queue is used, either feature can be used to limit the Tx data rate; however, if multiple queues are used, the IPG pacing feature is a better choice for a homogeneous Tx data rate limitation.

## 4.6.11.5 Configuration Rules

### 4.6.11.5.1 TC Parameters

#### Traffic Class

Per 802.1p, priority #7 is the highest priority.

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

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

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

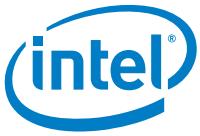
#### Bandwidth Group (BWGs)

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

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

#### Refill Credits

Refill credits regulate the bandwidth allocated to BWG and TC. The ratio between the credits of the BWG's represents the relative bandwidth percentage allocated to each BWG. The ratio between the credits of the TC's represents the relative bandwidth percentage allocated to each TC within a BWG.



Credits are configured and calculated using 64 bytes granularity.

1. In any case, the number of refill credits assigned per TC should be as small as possible but must be larger than the maximum frame size used and larger than 1.5 KB. Using a lower refill value causes more refill cycles before a packet can be sent. These extra cycles unnecessarily increase the latency.
2. Refill credits ratio between TCs should be equal to the desired ratio of bandwidth allocation between the different TCs. Applying rule #1, means bandwidth shares are sorted from the smaller to the bigger, and just one maximum sized frame is allocated to the smallest.
3. The ratio between the refill credits of any two TCs should not be greater than 100.
4. Exception to rule #2 — TCs that require low latency should be configured so that they are under subscribed. For example, credit refill value should provide these TCs somewhat more bandwidth than what they actually need. Low latency TCs should always have credits so they can be next in line for the WSP arbitration.

   This exception causes the low latency TC to always have maximum credits (as it starts with maximum credits and on average cycle uses less than the refill credits).

The end point that is sending/receiving packets of 127 bytes eventually gets double the bandwidth it was configured to, as we do all the credit calculation by rounding the values down to the next 64 byte aligned value.

### Maximum Credit Limit

The maximum credit limit value establishes a limit for the number of credits that a TC or BWG can own at any given time. This value prevents stacking up stale credits that can be added up over a relatively long period of time and then used by TCs all at once, altering fairness and latency.

Maximum credits limits are configured and calculated using 64 bytes granularity.

1. Maximum credit limit should be bigger than the refill credits allocated to the TC.
2. Maximum credit limit should be set to be as low as possible while still meeting other rules to minimize the latency impact on low latency TCs.
3. If a low latency TC generates a burst that is larger than its maximum credit limit this TC might experience higher latency since the TC needs to wait for allocation of additional credits because it finished all its credits for this cycle. Therefore maximum credit limit for a low latency TC must be set bigger than the maximum burst length of traffic expected on that TC (for all the VMs at once). If TC7 and TC6 are for low latency traffic, it leads to:

   ```
   Max(TC7,6) >= MaxBurst(TC7,6) served with low latency
   ```

4. An arbitration cycle can extend when one or more TCs accumulate credits more than their refill values (up to their maximum credit limit). For such a case, a low latency TC should be provided with enough credits to cover for the extended cycle duration. Since the low latency TC operates at maximum credits (see rule #3) its maximum credit limit should meet the following formula:

   ```
   {Max(TCx)/SUMi=0..7[Max(TCi)]} >= {BW(TCx)/Full BW}
   ```

   The formula applies to both descriptor arbiter and data arbiter.

5. When in a virtualized environment, the low latency TC condition checked by the VM WRR arbiter (see Section 7.7.2.3.2) induces the following relation between the maximum credits of a low latency TC and the refill credits of its attached VM arbiter:

```
Max(TCx) >= 2 x {SUMi=0...15[Refill(VMi)]}
```

6. To ensure bandwidth for low priority TC (when those are allocated with most of the bandwidth) the maximum credit value of the low priority TC in the data arbiter needs to be high enough to ensure sync between the two arbiters. In the equation that follows the bandwidth numbers are from the descriptor arbiter while the maximum values are of the data arbiter.

```
{Max(TCx)/SUMi=x+1..7[Max(TCi)]} >= {BW(TCx)/Full_PCIE_BW}
```

   Note that the previous equation is worst case and covers the assumption that all higher TCs have the full maximum to transmit.

**Tip:** A simplified maximum credits allocation scheme would be to find the minimum number N >= 2 such that rules #3 and #5 are respected, and allocate

```
Max(TCi) = N x Refill(TCi), for i=0...7
```

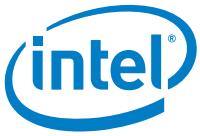
By maintaining the same ratios between the maximum credits and the bandwidth shares, the bandwidth allocation scheme is made more immune to disturbing events such as reception of priority pause frames with short timer values.

### GSP and LSP

*TC Link Strict Priority* (TC.LSP): This bit specifies that the configured TC can transmit without any restriction of credits. This effectively means that the TC can take up entire link bandwidth, unless preempted by higher priority traffic. The Tx queues associated with LSP TC must be set as *Strict Low Latency* in the TXLLQ[n] registers.

*TC Strict Priority* within group (TC.GSP): This bit defines whether strict priority is enabled or disabled for this TC within its BWG. If TC.GSP is set to 1b, the TC is scheduled for transmission using strict priority. It does not check for availability of credits in the TC. It does check whether the BWG of this TC has credits. For example, the amount of traffic generated from this TC is still limited by the BWG allocated for the BWG.

1. TC's with the *LSP* bit set should be the first to be considered by the scheduler. This implies that *LSP* should be configured to the highest priority TC's. For example, starting from priority 7 and down. The other TC's should be used for groups with bandwidth allocation. It is recommended to use LSP only for one TC (TC7) as the first LSP TC takes its bandwidth and there are no guarantees to the lower priority LSPs.
2. GSP can be set to more than one TC in a BWG, always from the highest priority TC within that BWG downward. For the LAN scenario, all TCs could be configured to be GSP as their bandwidth needs are not known.
3. To a low latency TC for which the *GSP* bit is set, non-null refill credits must be set for at least one maximum sized frame. It ensures that even after having been quiet for a while, some BWG credits are left available to the GSP TC, for serving it with minimum latency (without waiting for replenishing). Bigger refill credits values ensure longer burst of GSP traffic served with minimum latency.



## 4.6.11.5.2 VM Parameters

### Refill Credits

Refill credits regulate the fraction of the TC's bandwidth that is allocated to a VM. The ratio between the credits of the VMs represents the relative TC bandwidth percentage allocated to each VM.

Credits are configured and calculated using 64 bytes granularity.

1. The number of refill credits assigned per VM should be as small as possible but still larger than the maximum frame size used and larger than 1.5 KB in any case. Using a lower refill value causes more refill cycles before a packet can be sent. These extra cycles increase the latency unnecessarily.
2. Refill credits ratio between VMs should be equal to the desired ratio of bandwidth allocation between the different TCs. Applying rule #1, means bandwidth shares are sorted from the smaller to the bigger, and just one maximum sized frame is allocated to the smallest.
3. The ratio between the refill credits of any two VMs within the TC should not be greater than 10.

VMs that are sending/receiving packets of 127 bytes eventually gets double the bandwidth it was configured to as we do all the credit calculation by rounding the values down to the next 64 byte aligned value.

4. In a low latency TC, non-null refill credits must be set to a VSP VM, for at least one maximum sized frame. It ensures that even after having been quiet for a while, some TC credits are left available to the VSP VM, for serving it with minimum latency (without waiting for TC to replenish). Bigger refill credits values ensure longer burst of VSP traffic served with minimum latency.

### Example 4-1 Refill and MaxCredits Setting Example

This example assumes a system with only four TCs and three VMs present, and with the following bandwidth allocation scheme. Also, full PCIe bandwidth is evaluated to 15 G.

### Table 4-9 Bandwidth Share Example

| TCs and VMs | | Bandwidth Share% | Notes |
|---|---|---|---|
| TC0 | Total | 40 | 9.5 KB (9728-byte) jumbo allowed. |
| | VM0 | 60 | |
| | VM1 | 30 | |
| | VM2 | 10 | |
| TC1 | Total | 20 | No jumbo. |
| | VM0 | 34 | |
| | VM1 | 33 | |
| | VM2 | 33 | |

### Table 4-9 Bandwidth Share Example (Continued)

| TCs and VMs | | Bandwidth Share% | Notes |
|---|---|---|---|
| TC2 | Total | 30 | Low latency TC. No jumbo.<br>Bandwidth share already increased.<br>MaxBurstTC2=120 KB |
| | VM0 | 80 | |
| | VM1 | 10 | |
| | VM2 | 10 | |
| TC3 | Total | 10 | Low latency LSP TC.<br>No jumbo.<br>MaxBurstTC3=36 KB |
| | VM0 | 20 | |
| | VM1 | 60 | |
| | VM2 | 20 | |

The ratios between TC refills were driven by TC0, which was set as 152 for supporting 9.5 KB jumbos.

The ratio between MaxCredits and Refill were taken as 17 for all the TCs, as driven by TC2 relation between MaxCredits and MaxBurstTC2.

### Table 4-10 Refill and MaxCredits Setting

| TCs and VMs | | Refill (64-Byte Units) | MaxCredits (64-Byte Units) |
|---|---|---|---|
| TC0 | Total | 152 | 2584 |
| | VM0 | 912 | |
| | VM1 | 456 | |
| | VM2 | 152 | |
| TC1 | Total | 76 | 1292 |
| | VM0 | 25 | |
| | VM1 | 24 | |
| | VM2 | 24 | |

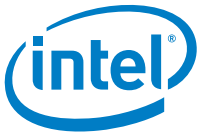


**Table 4-10 Refill and MaxCredits Setting (Continued)**

| TCs and VMs | | Refill (64-Byte Units) | MaxCredits (64-Byte Units) |
|---|---|---|---|
| TC2 | Total | 114 | 1938 |
| | VM0 | 192 | |
| | VM1 | 24 | |
| | VM2 | 24 | |
| TC3 | Total | 38 | 646 |
| | VM0 | 24 | |
| | VM1 | 72 | |
| | VM2 | 24 | |

## 4.6.12 Security Initialization

After power up or device reset, security offload is disabled by default (both LinkSec and IPsec), and the content of SA tables must be cleared by software.

Security offload cannot be enabled if internal security fuses are not enabled or the SDP0_4 pin is set to 0b. In this case, both IPsec and LinkSec are disabled and the following security related fields are not writable:

- SECTXCTRL.SECTX_DIS is set to 1b and read as 1b.
- SECRXCTRL.SECRX_DIS is set to 1b and read as 1b.
- IPSTXIDX.IPS_TX_EN is cleared to 0b and read as 0b.
- IPSRXIDX.IPS_RX_EN is cleared to 0b and read as 0b.
- LSECTXCTRL bits 1:0 are cleared to 00b and read as 00b.
- LSECRXCTRL bits 3:2 are cleared to 00b and read as 00b.

Security offload can be used when enabled by internal security fuses and when the SDP0_4 pin is set to 1b. In this case, the security offload can be enabled/disabled via the flows described as follows.

### 4.6.12.1 Security Enablement Flow

To enable one of the security modes perform the following steps:

1. Stop the data paths by setting the SECTXCTRL.TX_DIS and SECRXCTRL.RX_DIS bits.
2. Wait for hardware to empty the data paths. Poll the SECTXSTAT.SECTX_RDY and SECRXSTAT.SECRX_RDY bits until they are both asserted by hardware.
3. Clear the SECTXCTRL.SECTX_DIS and SECRXCTRL.SECRX_DIS bits to enable the Tx and Rx crypto engines.

When enabling IPsec or LinkSec offload, set SECTXMINIFG.MINSECIFG to 0x3 extending back-to-back gap to the security block required for its functionality.

When enabling IPsec, set the SECTXCTRL.STORE_FORWARD bit, since a store and forward IPsec buffer is required for the processing of AH packets (*ICV* field insertion is done at the beginning of the frame). Otherwise, clear this bit.

When enabling IPsec, write the SEC buffer almost full threshold, register SECTXBUFFAF.buff_af_thresh, with the value of 0x15.

4. Enable SA lookup:

   For IPsec, set the IPSTXIDX.IPS_TX_EN and the IPSRXIDX.IPS_RX_EN bits.

   For LinkSec, set the enable bits in the LSECTXCTRL and LSECRXCTRL registers.

5. Restart the data paths by clearing the SECTXCTRL.TX_DIS and SECRXCTRL.RX_DIS bits.

### 4.6.12.2 Security Disable Flow

To disable one of the security modes perform the following steps:

1. Stop the data paths by setting the SECTXCTRL.TX_DIS and SECRXCTRL.RX_DIS bits.
2. Wait for hardware to empty the data paths. Poll the SECTXSTAT.SECTX_RDY and SECRXSTAT.SECRX_RDY bits until they are both asserted by hardware.
3. Disable SA lookup:

   For IPsec, clear the IPSTXIDX.IPS_TX_EN and the IPSRXIDX.IPS_RX_EN bits.

   For LinkSec, clear the enable bits in the LSECTXCTRL and LSECRXCTRL registers.

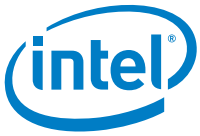
4. Set the SECTXCTRL.SECTX_DIS and SECRXCTRL.SECRX_DIS bits to disable the Tx and Rx crypto engines.

   When disabling IPsec, clear the SECTXCTRL.STORE_FORWARD bit, to avoid using the IPsec buffer and thus reducing Tx internal latency.

   When disabling IPsec, write the SEC buffer almost full threshold, register SECTXBUFFAF.buff_af_thresh, with the value of 0x250.

5. Restart the data paths by clearing the SECTXCTRL.TX_DIS and SECRXCTRL.RX_DIS bits.

**Note:** Disabling the crypto engine reduces the 82599's power consumption.



## 4.6.13 Alternate MAC Address Support

In some systems, the MAC address used by a port needs to be replaced with a temporary MAC address in a way that is transparent to the software layer. One possible usage is in blade systems, to allow a standby blade to use the MAC address of another blade that failed, so that the network image of the entire blade system does not change.

In order to allow this mode, a management console might change the MAC address in the NVM image. It is important in this case to be able to keep the original MAC address of the device as programmed at the factory.

In order to support this mode, the 82599 provides the Alternate Ethernet MAC Address structure in the NVM to store the original MAC addresses. This structure is described in Section 6.2.7. 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.

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

# 5.0 Power Management and Delivery

This section defines how power management is implemented in the 82599.

## 5.1 Power Targets and Power Delivery

See Section 11.2.1 for the current consumption and see Section 11.4.1 for the power supply specification.

## 5.2 Power Management

### 5.2.1 Introduction to the 82599 Power States

The 82599 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 82599 supports a Dr state that is entered when PE_RST_N is asserted (including the D3cold state).

Figure 5-1 shows the power states and transitions between them.

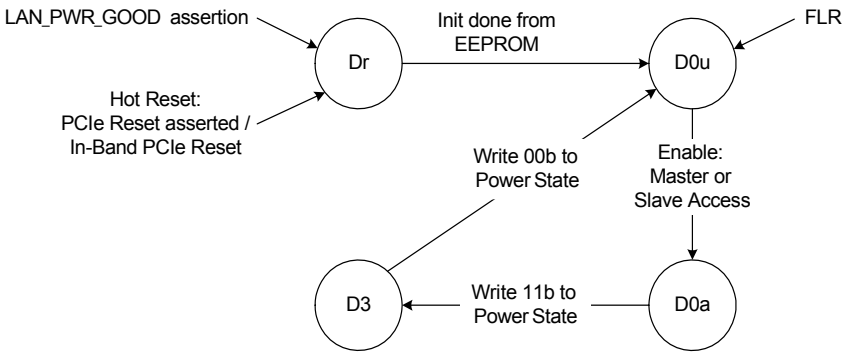



**Figure 5-1 Power Management State Diagram**

## 5.2.2 Auxiliary Power Usage

The 82599 uses the AUX_PWR indication that auxiliary power is available to it, and therefore advertises D3cold wake up support. The amount of power required for the function, which includes the entire Network Interface Card (NIC) is advertised in the Power Management Data register, which is loaded from the EEPROM.

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 (WUC) register are reset only by the power up reset (detection of power rising).

The only effect of setting AUX_PWR to 1b is advertising D3cold wake up support and changing the reset function of PME_En and PME_Status. AUX_PWR is a strapping option in the 82599.

The 82599 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 might consume (and therefore its power state) in the D3cold state (internal Dr state). Note that the actual amount of power differs between form factors.

## 5.2.3 Power Limits by Certain Form Factors

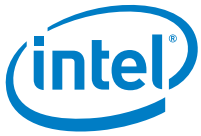
Table 5-1 lists the power limitations introduced by different form factors.

**Table 5-1 Power Limits by Form-Factor**

| | Form Factor | |
|---|---|---|
| | **LOM** | **PCIe Card** |
| **Main** | N/A | 25 W |
| **Auxiliary (aux enabled)** | 375 mA @ 3.3V | 375 mA @ 3.3V |
| **Auxiliary (aux disabled)** | 20 mA @ 3.3V | 20 mA @ 3.3V |

**Note:** Auxiliary current limit only applies when the primary 3.3V voltage source is not available (the card is in a low power D3 state).

The 82599 exceeds the allocated auxiliary power in some configuration (such as both ports running at GbE or 10 GbE speed). The 82599 must be configured such that it meets the previously described requirements. To do so, link speed can be restricted to GbE and one of the LAN ports can be disabled when operating on auxiliary power. See Section 5.2.5.4.



## 5.2.4 Interconnects Power Management

This section describes the power reduction techniques used by the 82599's main interconnects.

### 5.2.4.1 PCIe Link Power Management

The PCIe link state follows the power management state of the device. Since the 82599 incorporates multiple PCI functions, the device power management state is defined as the power management state of the most awake function:

- If any function is in D0 state (either D0a or D0u), the PCIe link assumes the device is in D0 state.

  Else,

- If the functions are in D3 state, the PCIe link assumes the device is in D3 state.

  Else,

- The device is in Dr state (PE_RST_N is asserted to all functions).

The 82599 supports all PCIe power management link states:

- L0 state is used in D0u and D0a states.
- The L0s state is used in D0a and D0u states each time link conditions apply.
- The L1 state is used in D0a and D0u states each time link conditions apply, as well as in the D3 state.
- The L2 state is used in the Dr state following a transition from a D3 state if PCI-PM PME is enabled.
- The L3 state is used in the Dr state following power up, on transition from D0a and also if PME is not enabled in other Dr transitions.

The 82599 support for Active State Link Power Management (ASLPM) is reported via the PCIe Active State Link PM Support register loaded from EEPROM.

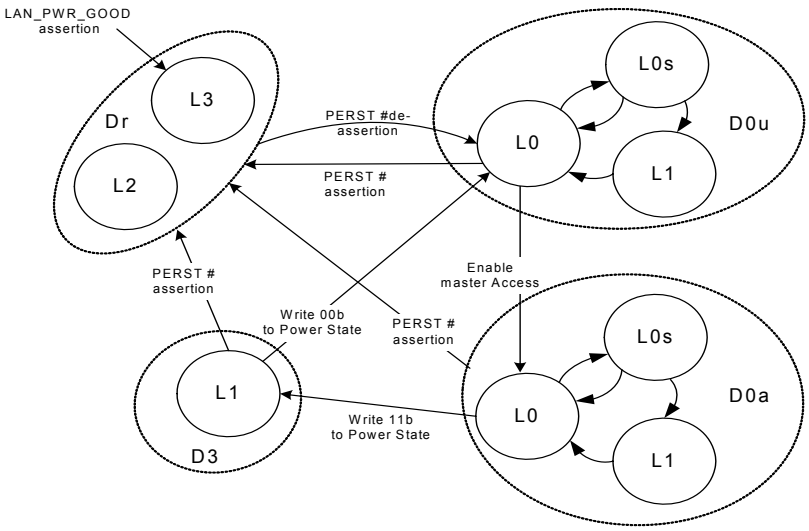



**Figure 5-2 Link Power Management State Diagram**

While in L0 state, the 82599 transitions the transmit lane(s) into L0s state once the idle conditions are met for a period of time defined as follows.

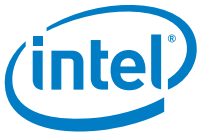
L0s configuration fields are:

- L0s enable — The default value of the *Active State Link PM Control* field in the PCIe Link Control register is set to 00b (both L0s and L1 disabled). System software can later write a different value into the Link Control register. The default value is loaded on any reset of the PCI configuration registers.
- The *L0S_ENTRY_LAT* bit in the PCIe Control Register (GCR), determines L0s entry latency. When set to 0b, L0s entry latency is the same as L0s exit latency of the device at the other end of the link. When set to 1b, L0s entry latency is (L0s exit Latency of the device at the other end of the link /4). Default value is 0b (entry latency is the same as L0s exit latency of the device at the other end of the link).
- L0s exit latency (as published in the *L0s Exit Latency* field of the Link Capabilities register) is loaded from the EEPROM. Separate values are loaded when the 82599 shares the same reference PCIe clock with its partner across the link, and when the 82599 uses a different reference clock than its partner across the link. The 82599 reports whether it uses the slot clock configuration through the *PCIe Slot Clock Configuration* bit loaded from the *Slot_Clock_Cfg* EEPROM bit.
- L0s acceptable latency (as published in the *Endpoint L0s Acceptable Latency* field of the Device Capabilities register) is loaded from the EEPROM.

While in L0s state, the 82599 transitions the link into L1 state once the transmit lanes or both directions of the link have been in L0s state for a period of time defined in PCI configuration space loaded from the PCIe Init Configuration 1 word in the EEPROM.

The following EEPROM fields control L1 behavior:

- Act_Stat_PM_Sup — Indicates support for ASPM L1 in the PCIe configuration space (loaded into the *Active State Link PM Support* field)
- PCIe PLL Gate Disable — Controls PCIe PLL gating while in L1 or L2 states



- L1_Act_Ext_Latency — Defines L1 active exit latency
- L1_Act_Acc_Latency — Defines L1 active acceptable exit latency
- Latency_To_Enter_L1 — Defines the period (in the L0s state) before transitioning into an L1 state

#### 5.2.4.2 Network Interfaces Power Management

The 82599 transitions any of the XAUI interfaces into a low-power state in the following cases:

- The respective LAN function is in LAN disable mode using LANx_DIS_N pin.
- The 82599 is in 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 EEPROM is cleared), then LAN ports are not allocated for manageability.
- If manageability is enabled:
  - Power up — Following EEPROM 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 (like 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 (like teaming) determine which port is used.

Enabling a port as a result of the previous causes an internal reset of the port.

When a XAUI interface is in low-power state, the 82599 asserts the respective SDP pin to enable an external PHY device to power down as well.

### 5.2.5 Power States

#### 5.2.5.1 D0uninitialized State

The D0u state is a low-power state used after PE_RST_N is de-asserted following power up (cold or warm), on hot reset (in-band reset through PCIe physical layer message) or on D3 exit.

When entering D0u, the 82599 disables wake ups. If the *APM Mode* bit in the EEPROM's Control Word 3 is set, then APM wake up is enabled.

#### 5.2.5.1.1 Entry to a D0u State

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

De-assertion of internal PE_RST_N means that the entire state of the device is cleared, other than sticky bits. State is loaded from the EEPROM, followed by establishment of the PCIe link. Once this is done, configuration software can access the device.

On a transition from D3 to D0u state, the 82599 requires that software perform a full re-initialization of the function including its PCI configuration space.

### 5.2.5.2 D0active State

Once memory space is enabled, the 82599 enters an active state. It can transmit and receive packets if properly configured by the driver. Any APM wake up previously active remains active. The driver can deactivate APM wake up by writing to the Wake Up Control (WUC) register, or activate other wake up filters by writing to the Wake Up Filter Control (WUFC) register.

#### 5.2.5.2.1 Entry to D0a State

D0a is entered from the D0u state by writing a 1b to the *Memory Access Enable* or the *I/O Access Enable* bit of the PCI Command register. The DMA, MAC, and PHY of the appropriate LAN function are enabled.

### 5.2.5.3 D3 State (PCI-PM D3hot)

The 82599 transitions to D3 when the system writes a 11b to the *Power State* field of the PMCSR. Any wake-up filter settings that were enabled before entering this reset state are maintained. Upon transitioning to D3 state, the 82599 clears the *Memory Access Enable* and *I/O Access Enable* bits of the PCI Command register, which disables memory access decode. In D3, the 82599 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 (such as an unsupported request) is detected while in D3hot, and reporting is enabled, the link must be returned to L0 if it is not already in L0 and an error message must be sent. See section 5.3.1.4.1 in the PCIe Base Specification.

A D3 state is followed by either a D0u state (in preparation for a D0a state) or by a transition to Dr state (PCI-PM D3cold state). To transition back to D0u, the system writes a 00b to the *Power State* field of the PMCSR. Transition to Dr state is through PE_RST_N assertion.

#### 5.2.5.3.1 Entry to D3 State

Transition to D3 state is through a configuration write to the *Power State* field of the PCI-PM registers.

Prior to transition from D0 to the D3 state, the device driver disables scheduling of further tasks to the 82599; it masks all interrupts, it does not write to the Transmit Descriptor Tail register or to the Receive Descriptor Tail register and operates the master disable algorithm as defined in Section 5.2.5.3.2. If wake-up capability is needed, the driver should set up the appropriate wake-up registers and the system should write a 1b to the *PME_En* bit of the PMCSR or to the *Auxiliary (AUX) Power PM Enable* bit of the PCIe Device Control register prior to the transition to D3.

If all PCI functions are programmed into D3 state, the 82599 brings its PCIe link into the L1 link state. As part of the transition into L1 state, the 82599 suspends scheduling of new TLPs and waits for the completion of all previous TLPs it has sent. The 82599 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, the device driver might disable one of the LAN ports (LAN disable) and/or transition the link(s) to GbE speed (if supported by the network interface). See Section 5.2.4.2 for a description of network interface behavior in this case.

#### 5.2.5.3.2 Master Disable

System software can disable master accesses on the PCIe link by either clearing the *PCI Bus Master* bit or by bringing the function into a D3 state. From that time on, the device must not issue master accesses for this function. Due to the full-duplex nature of PCIe, and the pipelined design in the 82599, it might happen that multiple requests from several functions are pending when the master disable request arrives. The protocol described in this section insures that a function does not issue master requests to the PCIe link after its master enable bit is cleared (or after entry to D3 state).

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

- *PCIe Master Disable* bit in the Device Control (CTRL) register — When the *PCIe Master Disable* bit is set, the 82599 blocks new master requests by this function. The 82599 then proceeds to issue any pending requests by this function. This bit is cleared on master reset (LAN Power Good all the way to software reset) to enable master accesses.
- *PCIe Master Enable Status* bits in the Device Status register — Cleared by the 82599 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 82599 clears the *PCIe Master Enable Status* bit:
  - — Master requests by the transmit and receive engines.
  - — All pending completions to the 82599 are received.

The device driver disables any reception to the Rx queues as described in Section 4.6.7.1. Then the device driver sets the *PCIe Master Disable* bit when notified of a pending master disable (or D3 entry).

The 82599 then blocks new requests and proceeds to issue any pending requests by this function. The driver then reads the change made to 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 driver might time out if the *PCIe Master Enable Status* bit is not cleared within a given time. Examples for cases that the device might not clear the *PCIe Master Enable Status* bit for a long time are cases of flow control, link down, or DMA completions not making it back to the DMA block.

In these cases, the driver should check that the *Transaction Pending* bit (bit 5) in the Device Status register in the PCI config space is clear before proceeding. In such cases the driver might need to initiate two consecutive software resets with a larger delay than 1 µs between the two of them.

In the above situation, the data path must be flushed before the software resets the 82599. The recommended method to flush the transmit data path is as follows:

1. Inhibit data transmission by setting the HLREG0.LPBK bit and clearing the RXCTRL.RXEN bit. This configuration avoids transmission even if flow control or link down events are resumed.
2. Set the GCR_EXT.Buffers_Clear_Func bit for 20 microseconds to flush internal buffers.
3. Clear the HLREG0.LPBK bit and the GCR_EXT.Buffers_Clear_Func
4. It is now safe to issue a software reset.

### 5.2.5.4 Dr State

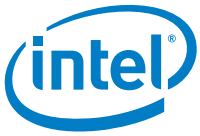
Transition to Dr state is initiated on several occasions:

- On system power up — Dr state begins with the assertion of the internal power detection circuit (LAN_PWR_GOOD) and ends with de-assertion of PE_RST_N.
- On transition from a D0a state — During operation the system can assert PE_RST_N at any time. In an ACPI system, a system transition to the G2/S5 state causes a transition from D0a to Dr state.
- On transition from a D3 state — The system transitions the device into the Dr state by asserting PCIe PE_RST_N.

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

The system can maintain PE_RST_N asserted for an arbitrary time. The de-assertion (rising edge) of PE_RST_N causes a transition to D0u state.

While in Dr state, the 82599 can maintain functionality (for WoL or manageability) or can enter a Dr Disable state (if no WoL and no manageability) for minimal device power. The Dr Disable mode is described in the sections that follow.



#### 5.2.5.4.1 Dr Disable Mode

The 82599 enters a Dr disable mode on transition to D3cold state when it does not need to maintain any functionality. The conditions to enter either state are:

- The device (all PCI functions) is in Dr state
- APM WOL is inactive for both LAN functions
- Pass-through manageability is disabled
- ACPI PME is disabled for all PCI functions

Entry into Dr disable is usually done on assertion of PCIe PE_RST_N. It can also be possible to enter Dr disable mode by reading the EEPROM while already in Dr state. The usage model for this later case is on system power up, assuming that manageability and wake up are not required. Once the device enters Dr state on power up, the EEPROM is read. If the EEPROM contents determine that the conditions to enter Dr disable are met, the device then enters this mode (assuming that PCIe PE_RST_N is still asserted).

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

If Dr disable mode is entered from D3 state, the platform can remove the 82599 power. If the platform removes the 82599 power, it must remove all power rails from the device if it needs to use this capability. Exit from this state is through power-up cycle to the 82599. Note that the state of the SDP pins is undefined once power is removed from the device.

#### 5.2.5.4.2 Entry to Dr State

Dr-entry on platform power up is as follows:

- Assertion of the internal power detection circuit (LAN_PWR_GOOD). Device power is kept to a minimum by keeping the XAUI interfaces in low power.
- The EEPROM is then read and determines device configuration.
- If the *APM Enable* bit in the EEPROM's Control Word 3 is set, then APM wake up is enabled (for each port independently).
- If the *MNG Enable* bit in the EEPROM word is set, pass-through manageability is not enabled.
- Each of the LAN ports can be enabled if required for WoL or manageability. See Section 5.2.4.2 for exact condition to enable a port.
- The PCIe link is not enabled in Dr state following system power up (since PE_RST_N is asserted).

Entry to Dr state from D0a state is through assertion of the PE_RST_N signal. An ACPI transition to the G2/S5 state is reflected in a device transition from D0a to Dr state. The transition can be orderly (such as a user selected a shut down operating system option), in which case the device driver can have a chance to intervene. Or, it can be an emergency transition (like power button override), in which case, the device driver is not notified.

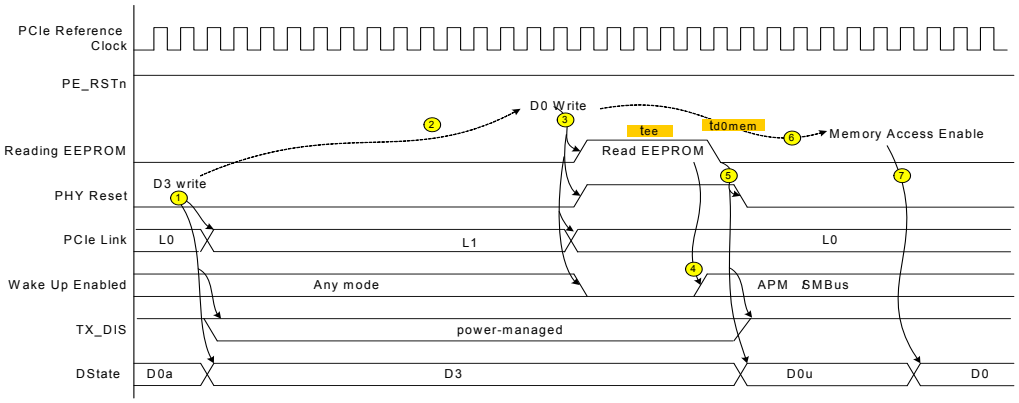
Transition from D3 state to Dr state is done by assertion of PE_RST_N signal. Prior to that, the system initiates a transition of the PCIe link from L1 state to either the L2 or L3 state (assuming all functions were already in D3 state). The link enters L2 state if PCI-PM PME is enabled.

## 5.2.6 Timing of Power-State Transitions

The following sections give detailed timing for the state transitions. In the diagrams the dotted connecting lines represent the 82599 requirements, while the solid connecting lines represent the 82599 guarantees.

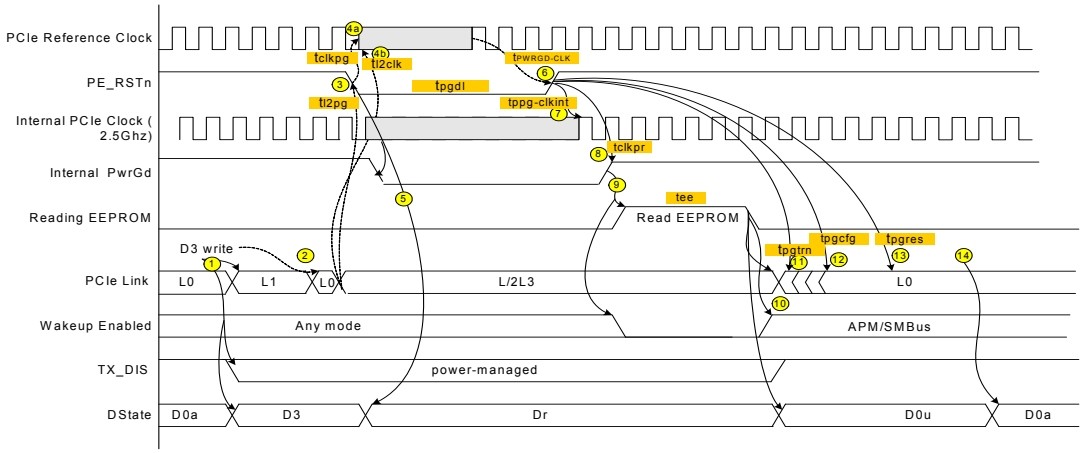
**Note:** The timing diagrams are not to scale. The clocks edges are shown to indicate running clocks only and are not used to indicate the actual number of cycles for any operation.

### 5.2.6.1 Transition from D0a to D3 and Back without PE_RST_N



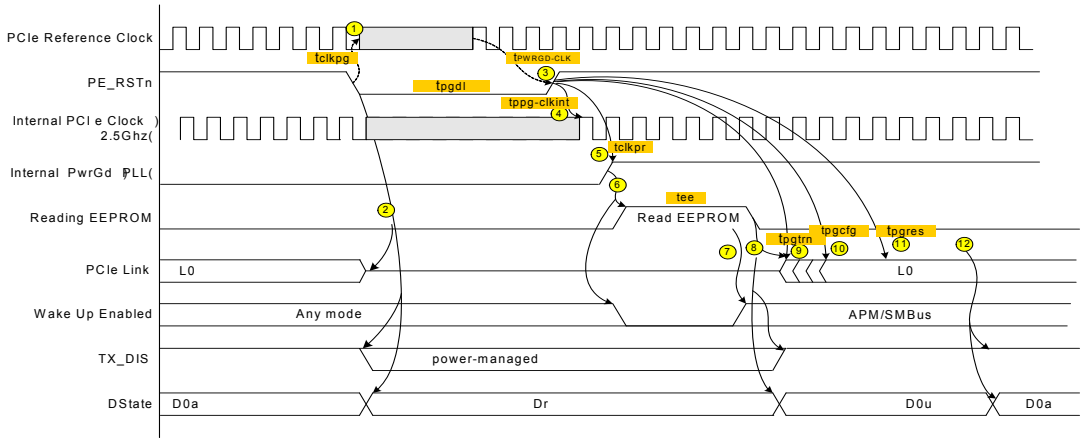

| Note | |
|---|---|
| 1 | Writing 11b to the *Power State* field of the PMCSR transitions the 82599 to D3. |
| 2 | The system can keep the 82599 in D3 state for an arbitrary amount of time. |
| 3 | To exit D3 state the system writes 00b to the *Power State* field of the PMCSR. |
| 4 | APM wake up or manageability can be enabled based on what is read in the EEPROM. |
| 5 | After reading the EEPROM, the LAN ports are enabled and the 82599 transitions to D0u state. |
| 6 | The system can delay an arbitrary time before enabling memory access. |
| 7 | Writing a 1b to the *Memory Access Enable* bit or to the *I/O Access Enable* bit in the PCI Command register transitions the 82599 from D0u to D0 state. |

### 5.2.6.2 Transition from D0a to D3 and Back with PE_RST_N

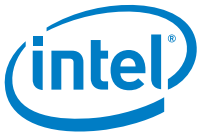



| Note | |
|---|---|
| 1 | Writing 11b to the *Power State* field of the PMCSR transitions the 82599 to D3. PCIe link transitions to L1 state. Possible indication to external PHYs to enter low-power mode. |
| 2 | The system can delay an arbitrary amount of time between setting D3 mode and transitioning the link to an L2 or L3 state. |
| 3 | Following link transition, PE_RST_N is asserted. |
| 4 | The system must assert PE_RST_N before stopping the PCIe reference clock. It must also wait tl2clk after link transition to L2/L3 before stopping the reference clock. |
| 5 | On assertion of PE_RST_N, the 82599 transitions to Dr state. |
| 6 | The system starts the PCIe reference clock $t_{PWRGD-CLK}$ before de-assertion PE_RST_N. |
| 7 | The internal PCIe clock is valid and stable $t_{ppg-clkint}$ from PE_RST_N de-assertion. |
| 8 | The PCIe internal PWRGD signal is asserted tclkpr after the external PE_RST_N signal. |
| 9 | Assertion of internal PCIe PWRGD causes the EEPROM to be re-read and disables wake up. |
| 10 | APM wake-up mode can be enabled based on what is read from the EEPROM. External PHYs are enabled. |
| 11 | Link training starts after tpgtrn from PE_RST_N de-assertion. |
| 12 | A first PCIe configuration access can arrive after $t_{pgcfg}$ from PE_RST_N de-assertion. |
| 13 | A first PCI configuration response can be sent after tpgres from PE_RST_N de-assertion. |
| 14 | Writing a 1b to the *Memory Access Enable* bit in the PCI Command register transitions the device from D0u to D0 state. |

### 5.2.6.3 Transition from D0a to Dr and Back without Transition to D3




| Note | |
|---|---|
| 1 | The system must assert PE_RST_N before stopping the PCIe reference clock. It must also wait tl2clk after link transition to L2/L3 before stopping the reference clock. |
| 2 | On assertion of PE_RST_N, the 82599 transitions to Dr state and the PCIe link transition to electrical idle. Possible indication to external PHYs to enter low-power mode. |
| 3 | The system starts the PCIe reference clock $t_{PWRGD-CLK}$ before de-assertion PE_RST_N. |
| 4 | The internal PCIe clock is valid and stable $t_{ppg-clkint}$ from PE_RST_N de-assertion. |
| 5 | The PCIe internal PWRGD signal is asserted tclkpr after the external PE_RST_N signal. |
| 6 | Assertion of internal PCIe PWRGD causes the EEPROM to be re-read and disables wake up. |
| 7 | APM wake-up mode can be enabled based on what is read from the EEPROM. |
| 8 | After reading the EEPROM, external PHYs are enabled. |
| 9 | Link training starts after tpgtrn from PE_RST_N de-assertion. |
| 10 | A first PCIe configuration access can arrive after $t_{pgcfg}$ from PE_RST_N de-assertion. |
| 11 | A first PCI configuration response can be sent after tpgres from PE_RST_N de-assertion |
| 12 | Writing a 1b to the *Memory Access Enable* bit in the PCI Command register transitions the device from D0u to D0 state. |



### 5.2.6.4 Timing Requirements

The 82599 requires the following start up and power state transitions.

**Table 5-2 Start Up and Power State Transitions**

| Parameter | Description | Min | Max. | Notes |
|---|---|---|---|---|
| $t_{xog}$ | Xosc stable from power stable | | 10 ms | |
| $t_{PWRGD-CLK}$ | PCIe clock valid to PCIe power good | 100 µs | - | According to PCIe specification. |
| $t_{PVPGL}$ | Power rails stable to PCIe PE_RST_N inactive | 100 ms | - | According to PCIe specification. |
| $t_{pgcfg}$ | External PE_RST_N signal to first configuration cycle | 100 ms | | According to PCIe specification. |
| td0mem | Device programmed from D3h to D0 state to next device access | 10 ms | | According to PCI power management specification. |
| tl2pg | L2 link transition to PE_RST_N assertion | 0 ns | | According to PCIe specification. |
| tl2clk | L2 link transition to removal of PCIe reference clock | 100 ns | | According to PCIe specification. |
| tclkpg | PE_RST_N assertion to removal of PCIe reference clock | 0 ns | | According to PCIe specification. |
| tpgdl | PE_RST_N assertion time | 100 µs | | According to PCIe specification. |

### 5.2.6.5 Timing Guarantees

The 82599 guarantees the following start up and power state transition related timing parameters.

**Table 5-3 Start-up and Power State Transition Timing Parameters[1]**

| Parameter | Description | Min | Max. | Notes |
|---|---|---|---|---|
| $t_{ee}$ | EEPROM read duration | | 20 ms | |
| $t_{ppg-clkint}$ | PCIe PE_RST_N to internal PLL lock | - | 50 µs | |
| $t_{clkpr}$ | Internal PCIe PWGD from external PCIe PE_RST_N | | 50 µs | |
| $t_{pgtrn}$ | PCIe PE_RST_N to start of link training | | 20 ms | According to PCIe specification. |
| $t_{pgres}$ | External PE_RST_N to response to first configuration cycle | 100 ms | 1 sec | According to PCIe specification. |

1. See also: Table 4-3, Power-Up Timing Guarantees.

# 5.3 Wake Up

## 5.3.1 Advanced Power Management Wake Up

Advanced Power Management Wake Up, or APM Wake Up, was previously known as Wake on LAN (WoL). It is a feature that has existed in the 10/100 Mb/s NICs for several generations. The basic premise is to receive a broadcast or unicast packet with an explicit data pattern, and then to assert a signal to wake up the system. In the earlier generations, this was accomplished by using a special signal that ran across a cable to a defined connector on the motherboard. The NIC would assert the signal for approximately 50 ms to signal a wake up. The 82599 uses (if configured to) an in-band PM_PME message for this.

At power up, the 82599 reads the *APM Enable* bit from the EEPROM into the *APM Enable (APME)* bits of the GRC register. This bit control the enabling of APM wake up.

When APM wake up is enabled, the 82599 checks all incoming packets for Magic Packets.

Once the 82599 receives a matching Magic Packet, it:

- Sets the *PME_Status* bit in the PMCSR.
- Asserts PE_WAKE_N.
- Issues a PM_PME message.

APM wake up is supported in all power states and only disabled if a subsequent EEPROM read results in the *APM Wake Up* bit being cleared.

## 5.3.2 ACPI Power Management Wake Up

The 82599 supports ACPI power management-based wake up. It can generate system wake-up events from three sources:

- Reception of a Magic Packet.
- Reception of a network wake-up packet.
- Detection of a link change of state.
- Activating ACPI power management wake up requires the following steps:
  - The operating system (at configuration time) writes a 1b to the *PME_En* bit of the PMCSR (bit 8).
  - The driver programs the Wake Up Filter Control (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) registers. 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. If the SW driver enables any of the wakeup options above it should also set the WUC.PME_En bit as well.
  - Once the 82599 wakes the system, the driver needs to clear WUFC until the next time the system goes to a low power state with wake up.

Normally, after enabling wake up, the operating system writes (11b) to the lower two bits of the PMCSR to put the 82599 into low-power mode.

Once wake up is enabled, the 82599 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 82599:

- Sets the *PME_Status* bit in the PMCSR.
- If the *PME_En* bit in the PMCSR is set, asserts PE_WAKE_N.

If enabled, a link state change wake up causes similar results, setting *PME_Status*, asserting *PE_WAKE_N* when the link goes up or down.

*PE_WAKE_N* remains asserted until the operating system either writes a 1b to the *PME_Status* bit of the PMCSR register or writes a 0b to the *PME_En* bit.

After receiving a wake-up packet or link change event, the 82599 ignores any subsequent wake-up packets or link change events until the driver clears the WUS register.

## 5.3.3 Wake-Up Packets

The 82599 supports various wake-up packets using two types of filters:

- Pre-defined filters
- Flexible filters

Each of these filters are enabled if the corresponding bit in the WUFC register is set to 1b.

**Note:** When VLAN filtering is enabled, packet s that passed any of the receive wake-up filters should only cause a wake-up event if it also passed the VLAN filtering.

### 5.3.3.1 Pre-Defined Filters

The following packets are supported by the 82599'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 showing which bytes at which offsets are compared to determine if the packet passes the filter. Both VLAN frames and LLC/SNAP can increase the given offsets if they are present.

#### 5.3.3.1.1 Directed Exact Packet

The 82599 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 Wake Up Filter Control (WUFC.EX) register.

| Offset | # of bytes | Field | Value | Action | Comment |
|---|---|---|---|---|---|
| 0 | 6 | Destination Address | | Compare | Match any pre-programmed address. |

#### 5.3.3.1.2 Directed Multicast Packet

For multicast packets, the upper bits of the incoming packet's destination address index a bit vector, the Multicast Table Array (MTA) that indicates whether to accept the packet. If the *Directed Multicast Wake Up Enable* bit set in the Wake Up Filter Control (WUFC.MC) register and the indexed bit in the vector is one, then the 82599 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 (MCSTCTRL.MO) register.

| Offset | # of bytes | Field | Value | Action | Comment |
|---|---|---|---|---|---|
| 0 | 6 | Destination Address | | Compare | See previous paragraph. |

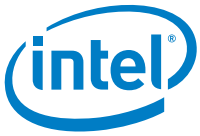
#### 5.3.3.1.3 Broadcast

If the *Broadcast Wake Up Enable* bit in the Wake Up Filter Control (WUFC.BC) register is set, the 82599 generates a wake-up event when it receives a broadcast packet.

| Offset | # of bytes | Field | Value | Action | Comment |
|---|---|---|---|---|---|
| 0 | 6 | Destination Address | FF*6 | Compare | |

#### 5.3.3.1.4 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 (FCTRL.BAM) register is 0b. If APM wake up is enabled in the EEPROM, the 82599 starts up with the Receive Address Register 0 (RAH0, RAL0) loaded from the EEPROM. This is to enable the 82599 to accept packets with the matching IEEE address before the driver comes up.



| Offset | # of Bytes | Field | Value | Action | Comment |
|---|---|---|---|---|---|
| 0 | 6 | Destination Address | | Compare | MAC header.<br>Processed by main address filter. |
| 6 | 6 | Source Address | | Skip | |
| 12 | 4 | Possible VLAN Tag | | Skip | |
| 12 | 8 | Possible Len/LLC/SNAP Header | | Skip | |
| 12 | 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) |

**Note:** Accepting broadcast Magic Packets for wake-up purposes when the *Broadcast Accept* bit of the Receive Control (FCTRL.BAM) register is 0b is a change from 82544, which initialized FCTRL.BAM to 1b if APM was enabled in the EEPROM, but then required that bit to be 1b to accept broadcast Magic Packets, unless broadcast packets passed another perfect or multicast filter.

### 5.3.3.1.5 ARP/IPv4 Request Packet

The 82599 supports reception of ARP request packets for wake up if the *ARP* bit is set in the WUFC register. Four IPv4 addresses are supported, which are programmed in the IPv4 Address Table (IP4AT). A successfully matched packet must pass L2 address filtering, a protocol type of 0x0806, an ARP opcode of 0x01, and one of the four programmed IPv4 addresses. The 82599 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.<br>Processed by main address filter. |
| 6 | 6 | Source Address | | Skip | |
| 12 | 4 | Possible VLAN Tag | | Compare | |
| 12 | 4 | Possible Len/LLC/SNAP Header | | Skip | |
| 12 | 2 | Type | 0x0806 | Compare | ARP. |
| 14 | 2 | Hardware Type | 0x0001 | Compare | |

| Offset | # of Bytes | Field | Value | Action | Comment |
|---|---|---|---|---|---|
| 16 | 2 | Protocol Type | 0x0800 | Compare | |
| 18 | 1 | Hardware Size | 0x06 | Compare | |
| 19 | 1 | Protocol Address Length | 0x04 | Compare | |
| 20 | 2 | Operation | 0x0001 | Compare | |
| 22 | 6 | Sender Hardware Address | - | Ignore | |
| 28 | 4 | Sender IP Address | - | Ignore | |
| 32 | 6 | Target Hardware Address | - | Ignore | |
| 38 | 4 | Target IP Address | IP4AT | Compare | Can match any of four values in IP4AT. |

#### 5.3.3.1.6 Directed IPv4 Packet

The 82599 supports reception of directed IPv4 packets for wake up if the *IPV4* 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 0x0800, and one of the four programmed IPv4 addresses. The 82599 also handles directed IPv4 packets that have VLAN tagging on both Ethernet II and Ethernet SNAP types.

| Offset | # of Bytes | Field | Value | Action | Comment |
|---|---|---|---|---|---|
| 0 | 6 | Destination Address | | Compare | MAC header.<br>Processed by main address filter. |
| 6 | 6 | Source Address | | Skip | |
| 12 | 4 | Possible VLAN Tag | | Compare | |
| 12 | 8 | Possible Len/LLC/SNAP Header | | Skip | |
| 12 | 2 | Type | 0x0800 | Compare | IP |
| 14 | 1 | Version/ HDR length | 0x4X | Compare | Check IPv4. |
| 15 | 1 | Type of Service | - | Ignore | |
| 16 | 2 | Packet Length | - | Ignore | |
| 18 | 2 | Identification | - | Ignore | |
| 20 | 2 | Fragment Information | - | Ignore | |
| 22 | 1 | Time to Live | - | Ignore | |

| Offset | # of Bytes | Field | Value | Action | Comment |
|---|---|---|---|---|---|
| 23 | 1 | Protocol | - | Ignore | |
| 24 | 2 | Header Checksum | - | Ignore | |
| 26 | 4 | Source IP Address | - | Ignore | |
| 30 | 4 | Destination IP Address | IP4AT | Compare | Can match any of four values in IP4AT. |

#### 5.3.3.1.7 Directed IPv6 Packet

The 82599 supports reception of directed IPv6 packets for wake up if the *IPV6* bit is set in the WUFC register. One IPv6 address is supported, which 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 82599 also handles directed IPv6 packets that have VLAN tagging on both Ethernet II and Ethernet SNAP types.

| Offset | # of Bytes | Field | Value | Action | Comment |
|---|---|---|---|---|---|
| 0 | 6 | Destination Address | | Compare | MAC header.<br>Processed by main address filter. |
| 6 | 6 | Source Address | | Skip | |
| 12 | 4 | Possible VLAN Tag | | Compare | |
| 12 | 8 | Possible Len/LLC/SNAP Header | | Skip | |
| 12 | 2 | Type | 0x0800 | Compare | IP. |
| 14 | 1 | Version/ Priority | 0x6X | Compare | Check IPv6. |
| 15 | 3 | Flow Label | - | Ignore | |
| 18 | 2 | Payload Length | - | Ignore | |
| 20 | 1 | Next Header | - | Ignore | |
| 21 | 1 | Hop Limit | - | Ignore | |
| 22 | 16 | Source IP Address | - | Ignore | |
| 38 | 16 | Destination IP Address | IP6AT | Compare | Match value in IP6AT. |

#### 5.3.3.2 Flexible Filter

The 82599 supports a total of **six** 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). 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 the *PME_En* bit in the PMCSR or the WUC register.

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 FHFT 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.

Packet that passed a wake-up flexible filter should cause a wake-up event only if it is directed to the 82599 (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 completes.

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

##### 5.3.3.2.1 IPX Diagnostic Responder Request Packet

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

| Offset | # of Bytes | Field | Value | Action | Comment |
|---|---|---|---|---|---|
| 0 | 6 | Destination Address | | Compare | |
| 6 | 6 | Source Address | | Skip | |
| 12 | 4 | Possible VLAN Tag | | Compare | |
| 12 | 8 | Possible Len/LLC/SNAP Header | | Skip | |
| 12 | 2 | Type | 0x8137 | Compare | IPX. |
| 14 | 16 | Some IPX Information | - | Ignore | |
| 30 | 2 | IPX Diagnostic Socket | 0x0456 | Compare | |

#### 5.3.3.2.2 Directed IPX Packet

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

| Offset | # of Bytes | Field | Value | Action | Comment |
|---|---|---|---|---|---|
| 0 | 6 | Destination Address | | Compare | MAC header.<br>Processed by main address filter. |
| 6 | 6 | Source Address | | Skip | |
| 12 | 4 | Possible VLAN Tag | | Compare | |
| 12 | 8 | Possible Len/LLC/SNAP Header | | Skip | |
| 12 | 2 | Type | 0x8137 | Compare | IPX. |
| 14 | 10 | Some IPX Information | - | Ignore | |
| 24 | 6 | IPX Node Address | Receive Address 0 | Compare | Must match Receive Address 0. |

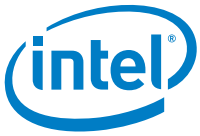
#### 5.3.3.2.3 IPv6 Neighbor Discovery Filter

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

### 5.3.4 Wake Up and Virtualization

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

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



# 6.0 Non-Volatile Memory Map

## 6.1 EEPROM General Map

The following table lists the EEPROM map used by the 82599. This table lists common modules for the EEPROM including: hardware pointers, software and firmware. Blocks are detailed in the following sections. All addresses and pointers in this table are absolute in word units.

| Word Address | Used By | Field Name | LAN 0 / 1 | Reserved |
|---|---|---|---|---|
| 0x00 | HW | EEPROM Control Word 1 — Section 6.3.2.1 | Shared Logic | |
| 0x01 | HW | EEPROM Control Word 2 — Section 6.3.2.2 | Shared Logic | |
| 0x03 | HW | PCIe Analog Configuration Module Pointer — Section 6.3.3 | Shared Logic | |
| 0x04 | HW | Core 0 Analog Configuration Module Pointer — Section 6.3.4 | Port 0 | |
| 0x05 | HW | Core 1 Analog Configuration Module Pointer — Section 6.3.4 | Port 1 | |
| 0x06 | HW | PCIe General Configuration Module Pointer — Section 6.3.5 | Shared Logic | |
| 0x07 | HW | PCIe Configuration Space 0 Module Pointer — Section 6.3.6 | Function 0 | |
| 0x08 | HW | PCIe Configuration Space 1 Module Pointer — Section 6.3.6 | Function 1 | |
| 0x09 | HW | LAN Core 0 Module Pointer — Section 6.3.7 | Port 0 | |
| 0x0A | HW | LAN Core 1 Module Pointer — Section 6.3.7 | Port 1 | |
| 0x0B | HW | MAC 0 Module Pointer — Section 6.3.8 | Port 0 | |
| 0x0C | HW | MAC 1 Module Pointer — Section 6.3.8 | Port 1 | |
| 0x0D | HW | CSR 0 Auto Configuration Module Pointer — Section 6.3.9 | Port 0 | |
| 0x0E | HW | CSR 1 Auto Configuration Module Pointer — Section 6.3.9 | Port 1 | |
| 0x0F | FW | Firmware Module Pointer — Section 6.4 | FW | |
| 0x10 – 0x14 | SW | SW Compatibility Module — Section 6.2.1 | SW | |
| 0x15 — 0x16 | SW | PBA Bytes 1...4 — Section 6.2.2 | SW | |

| Word Address | Used By | Field Name | LAN 0 / 1 | Reserved |
|---|---|---|---|---|
| 0x17 | SW | iSCSI Boot Configuration Start Address — Section 6.2.3 | SW | |
| 0x18 – 0x19 | SW | Software Reserved | SW | |
| 0x20 | SW | PXE VLAN Configuration Pointer | SW | |
| 0x21 – 0x25 | SW | Software Reserved | SW | |
| 0x26 | SW | External Thermal Sensor Block — Section 6.2.9 | SW | |
| 0x27 | SW | Alternate SAN MAC Address Block — Section 6.2.10 | SW | |
| 0x28 | SW | Active SAN MAC Address Block — Section 6.2.11 | SW | |
| 0x29 - 0x2E | SW | Software Reserved | SW | |
| 0x2F | OEM | VPD Pointer — Section 6.2.5 | Shared Logic | |
| 0x30 – 0x36 | PXE | PXE Word 0 (Software Use) Configuration — Section 6.2.6 | SW | |
| 0x37 | SW | Alternate Ethernet MAC Addresses Pointer — Section 6.2.7 | SW | |
| 0x38 | HW | EEPROM Control Word 3 — Section 6.3.2.3 | Shared Logic | |
| 0x39 – 0x3E | HW | Hardware Reserved | Reserved | |
| 0x3F | SW | Software Checksum, Words 0x00 — 0x3F | Shared Logic | |

# 6.2 EEPROM Software

## 6.2.1 SW Compatibility Module — Word Address 0x10-0x14

Five words in the EEPROM 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.

## 6.2.2 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 EEPROM.

Note that through the course of hardware ECOs, the suffix field is incremented. The purpose of this information is to enable customer support (or any user) to identify the revision level of a product.

Network driver software should not rely on this field to identify the product or its capabilities.

Current PBA numbers have exceeded the length that can be stored as hex values in these two words. For these PBA numbers the high word is a flag (0xFAFA) indicating that the PBA is stored in a separate PBA block. The low word is a pointer to a PBA block.

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

The PBA block is pointed to by word 0x16.

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

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

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

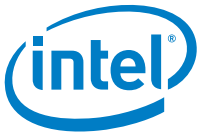
Older PBA numbers starting with (A,B,C,D,E) are stored directly in words 0x15 and 0x16. The dash itself is not stored nor is the first digit of the 3-digit suffix, as it is always 0b for relevant products.

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

## 6.2.3 iSCSI Boot Configuration — Word Address 0x17

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

| Configuration Item | Offset (Bytes) | Size in Bytes | Comments |
|---|---|---|---|
| **Shared Words** | | | |
| Boot Signature | 0x1:0x0 | 2 | 'i', 'S' (0x5369). |
| Block Size | 0x3:0x2 | 2 | Total byte size of the boot configuration block. |
| Structure Version | 0x4 | 1 | Version of this structure. Should be set to 1b. |
| Reserved | 0x5 | 1 | Reserved for future use, should be set to zero. |
| iSCSI Initiator Name | 0x105:0x6 | 255 + 1 | iSCSI initiator name. This field is optional and built by manual input, DHCP host name, or with MAC address. |
| iSCSI Configuration Block | 0x107:0x106 | 2 | Bits 15:8 (Major) - Combo image major version.<br>Bits 7:0 (Build) - Combo image build number (15:8). |
| | 0x109:0x108 | 2 | Bits 15:8 (Build) - Combo image build number (7:0).<br>Bits 7:0 (Minor) - Combo image minor version. |
| Reserved | 0x127:0x10A | 30 | Reserved for future use, should be set to zero. |



| Configuration Item | Offset (Bytes) | Size in Bytes | Comments |
|---|---|---|---|
| **Port 0 Configuration** | | | |
| iSCSI Flags | 0x129:0x128 | 2 | Bit 0x00: Initiator DHCP flag<br>0b = Use static configurations from this structure.<br>1b = Use DHCP to get initiator IP configuration.<br>Bit 0x01: Enable DHCP for getting iSCSI target DHCP flag.<br>0b = Use static target configuration.<br>1b = Use DHCP to get target information by the Option 17 Root Path.<br>Bit 0x02 – 0x03: Authentication Type<br>0x00 = None.<br>0x01 = One way chap.<br>0x02 = Mutual chap.<br>Bit 0x04 – 0x05: Ctrl-D setup menu<br>0x00 = Enabled<br>0x03 = Disabled, skip Ctrl-D entry<br>Bit 0x06 - 0x07: Reserved<br>Bit 0x08 - 0x09: ARP Retries<br>Retry value<br>Bit 0x0A – 0x0F: ARP Timeout<br>Timeout value for each try |
| iSCSI Initiator IP | 0x12D:0x12A | 4 | Initiator DHCP flag<br>Not set = This field should contain the initiator IP address.<br>Set = This field is ignored. |
| Subnet Mask | 0x131:0x12E | 4 | Initiator DHCP flag<br>Not set = This field should contain the subnet mask.<br>Set = This field is ignored. |
| Gateway IP | 0x135:0x132 | 4 | Initiator DHCP flag<br>Not set = This field should contain the gateway IP address.<br>Set = This field is ignored. |
| iSCSI Boot LUN | 0x137:0x136 | 2 | Target DHCP flag<br>Not set = iSCSI target LUN number should be specified.<br>Set = This field is ignored. |
| iSCSI Target IP | 0x13B:0x138 | 4 | Target DHCP flag;<br>Not set = IP address of iSCSI target.<br>Set = This field is ignored. |
| iSCSI Target Port | 0x13D:0x13C | 2 | Target DHCP flag<br>Not set = TCP port used by iSCSI target. Default is 3260.<br>Set = This field is ignored. |
| iSCSI Target Name | 0x23D:0x13E | 255 + 1 | Target DHCP flag<br>Not set = iSCSI target name should be specified.<br>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. |

| Configuration Item | Offset (Bytes) | Size in Bytes | Comments |
|---|---|---|---|
| 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. |
| FCoE Flags | 0x2E5:0x2E4 | 2 | Bit 1 used for Disable FCoE Ctrl-D Menu. Default = 0.<br>0b = FCoE Ctrl-D menu is enabled and user can configure FCoE ports.<br>1b = FCoE Ctrl-D menu is disabled and user cannot configure FCoE ports. |
| Reserved | 0x2EB:0x2E6 | 6 | Reserved for future use, should be set to zero. |
| **FCoE Target 0** | | | |
| Target Worldwide Port Name (WWPN) | 0x2F3:0x2EC | 8 | Byte string of target WWPN |
| Boot LUN | 0x2F5:0x2F4 | 2 | Target LUN |
| VLAN ID | 0x2F7:0x2F6 | 2 | VLAN ID for the Port. Default is 0. |
| Target Boot Order | 0x2F8 | 1 | Target Boot Order.<br>Valid range is 0-4, with 0 meaning no boot order. |
| Reserved | 0x2FB:0x2F9 | 3 | Reserved for future use, should be set to zero. |
| **FCoE Target 1** | | | |
| Target Worldwide Port Name (WWPN) | 0x303:0x2FC | 8 | Byte string of target WWPN |
| Boot LUN | 0x305:0x304 | 2 | Target LUN |
| VLAN ID | 0x307:0x306 | 2 | VLAN ID for the Port. Default is 0. |
| Target Boot Order | 0x308 | 1 | Target Boot Order.<br>Valid range is 0-4, with 0 meaning no boot order. |
| Reserved | 0x30B:0x309 | 3 | Reserved for future use, should be set to zero. |
| **FCoE Target 2** | | | |
| Target Worldwide Port Name (WWPN) | 0x313:0x30C | 8 | Byte string of target WWPN |
| Boot LUN | 0x315:0x314 | 2 | Target LUN |
| VLAN ID | 0x317:0x316 | 2 | VLAN ID for the Port. Default is 0. |

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

## 6.2.4 Software Reserved Word — PXE VLAN Configuration Pointer — Word Address 0x20

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

### Table 6-1 PXE VLAN Configuration Section Summary Table

| Word Offset | Word Name | Description |
|---|---|---|
| 0x0000 | VLAN Block Signature | ASCII 'V', 'L'. |
| 0x0001 | Version and Size | Contains version and size of structure. |
| 0x0002 | Port 0 VLAN Tag | VLAN tag value for the first port of the 82599. Contains PCP, CFI and VID fields.<br>A value of 0 means no VLAN is configured for this port. |
| 0x0003 | Port 1 VLAN Tag | VLAN tag value for the second port of the 82599. Contains PCP, CFI and VID fields.<br>A value of 0 means no VLAN is configured for this port. |

**Table 6-2 VLAN Block Signature - 0x0000**

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

**Table 6-3 Version and Size- 0x0001**

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

**Table 6-4 Port 0 VLAN Tag - 0x0002**

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

**Table 6-5 Port 1VLAN Tag - 0x0003**

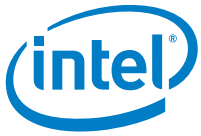
| Bits | Name | Default | Description |
|---|---|---|---|
| 15:13 | Priority (0-7) | 0x0 | Priority 0-7 |
| 12 | Reserved | 0x0 | Always 0 |
| 11:0 | VLAN ID (1- 4095) | 0x0 | VLAN ID (1- 4095) |

## 6.2.5 VPD Module Pointer — Word Address 0x2F

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

## 6.2.6 EEPROM PXE Module — Word Address 0x30-0x36

Words 0x30 through 0x36 are reserved for configuration and version values used by PXE code.



The configuration of the Boot Agent software is controlled by several words in the EEPROM . 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.

| Word Address | Description | Reserved |
|---|---|---|
| 0x30 | PXE Word 0 (Software Use) Configuration | |
| 0x31 | PXE Word 1 (Software Use) Configuration | |
| 0x32 | PXE Word (Software Use) PXE Version | |
| 0x33 | PXE Word (Software Use) EFI Version | |
| 0x34 – 0x36 | Additional PXE Reserved words (Software Use) | |

### 6.2.6.1 PXE Setup Options PCI Function 0 — Word Address 0x30

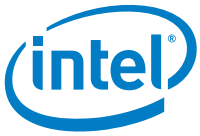
The main setup options for Port 0 are stored in this word. These options are those that can be changed by the user using the Control-S setup menu.

| Bits | Name | Default | Description |
|---|---|---|---|
| 15:13 | | | Reserved. Must be 0x0. |
| 12:10 | FSD | | Bits 12:10 control forcing speed and duplex during driver operation.<br>Valid values are:<br>000b = Auto-negotiate. (default)<br>001b = Reserved.<br>010b = 100 Mb/s half duplex.<br>011b = Not valid (treated as 000b).<br>100b = Not valid (treated as 000b).<br>101b = Reserved.<br>110b = 100 Mb/s full duplex.<br>111b = 1000 Mb/s full duplex.<br>*Note:* Only applicable for copper-based adapters. Not applicable to 10 GbE. |
| 9 | LWS | | Legacy OS Wakeup Support.<br>If set to 1b, the agent enables PME in the adapter's PCI configuration space during initialization. This allows remote wake up under legacy operating systems that don't normally support it. Note that enabling this makes the adapter technically non-compliant with the ACPI specification, which is why the default is disabled.<br>Must be set to 0b for 1 GbE and 10 GbE adapters.<br>0b = Disabled (default).<br>1b = Enabled. |
| 8 | DSM | | Display Setup Message.<br>If the bit is set to 1b, the "Press Control-S" message is displayed after the title message. Default value is 1b. |

| Bits | Name | Default | Description |
|---|---|---|---|
| 7:6 | PT | | Prompt Time.<br>These bits control how long the "Press Control-S" setup prompt message is displayed, if enabled by DIM.<br>00b = 2 seconds (default).<br>01b = 3 seconds.<br>10b = 5 seconds.<br>11b = 0 seconds.<br>*Note:* The Ctrl-S message is not displayed if 0 seconds prompt time is selected. |
| 5 | Disable iSCSI Setup Menu | 0x0 | This controls the iSCSI init message (Ctrl+D menu prompt) when iSCSI is disabled.<br>When iSCSI option ROM is disabled and this bit is set to 1b, the init message is not displayed for the port.<br>When iSCSI option ROM is disabled and this bit is set to 0b, the init message is displayed for the port.<br>When iSCSI option ROM is enabled this bit does not matter, as the init message is displayed for the port. |
| 4:3 | DBS | | Default Boot Selection. These bits select which device is the default boot device. These bits are only used if the agent detects that the BIOS does not support boot order selection or if the MODE field of word 31h is set to MODE_LEGACY.<br>00b = Network boot, then local boot (default)<br>01b = Local boot, then network boot<br>10b = Network boot only<br>11b = Local boot only |

Bits 2:0 are defined as follows:

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



## 6.2.7 Alternate Ethernet MAC Address — Word Address 0x37

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

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

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

## 6.2.8 Checksum Word Calculation (Word 0x3F)

The checksum word (0x3F) is used to ensure that the base EEPROM image is a valid image. The value of this word should be calculated such that after adding all the words (0x00:0x3F), including the checksum word itself, the sum should be 0xBABA. This word is used strictly by software. Hardware does not calculate or check its content but instead checks the Signaturefield in EEPROM Control Word 1.

The first 63 words of the EEPROM have a collection of pointers to other sections of the EEPROM. To ensure the integrity of the additional configuration parameters, their content should be included in the EEPROM checksum word at offset 0x3F. As a result, the new algorithm for determining the checksum is as follows:

1. Perform an EEPROM checksum of the first 63 words at 0x0-0x3E.
2. Starting with word offset 0x03 (PCIE_ANALOG_PTR) read the pointer value.
3. If the value of the pointer is 0x0 or 0xFFFF then move on to the next pointer.
4. If the pointer has a value, then read the content of where the pointer points to.

For example, if the pointer is 0x308, read the value at word offset 0x308. The value in the first word pointed to is the length of that particular configuration data section. Note that the length value is NOT added to the checksum value. If the value at 0x308 was 5, don’t add 5 to the checksum value. Instead the length is used to determine how many words after the count value should be added to the checksum. In this example, 5 words are added to the checksum, starting at word offset 0x309. This same logic applies to the pointers in locations 0x4 through 0xE. The result of the checksum is then subtracted from 0xBABA and compared to the value at word offset 0x3F. If the values match then the checksum is valid, if not, then the checksum is invalid.

The checksum word (0x3F) is used to ensure that the base EEPROM image is a valid image. The following documents the calculation.

```
#define IXGBE_EEPROM_CHECKSUM     0x3F
#define IXGBE_EEPROM_SUM          0xBABA
#define IXGBE_PCIE_ANALOG_PTR     03
#define IXGBE_FW_PTR             0F
static u16 ixgbe_eeprom_calc_checksum(struct ixgbe_hw *hw)
{
    u16 i;
    u16 j;
    u16 checksum = 0;
    u16 length = 0;
    u16 pointer = 0;
    u16 word = 0;

    /* Include 0x0-0x3F in the checksum */
    for (i = 0; i < IXGBE_EEPROM_CHECKSUM; i++) {
        if (ixgbe_eeprom_read(hw, i, &word) != IXGBE_SUCCESS) {
            DEBUGOUT("EEPROM read failed\n");
            break;
        }
        checksum += word;
    }

    /* Include all data from pointers except for the fw pointer */
    for (i = IXGBE_PCIE_ANALOG_PTR; i < IXGBE_FW_PTR; i++) {
        ixgbe_eeprom_read(hw, i, &pointer);

        /* Make sure the pointer seems valid */
        if (pointer != 0xFFFF && pointer != 0) {
            ixgbe_eeprom_read(hw, pointer, &length);

            if (length != 0xFFFF && length != 0) {
                for (j = pointer+1; j <= pointer+length; j++) {
                    ixgbe_eeprom_read(hw, j, &word);
                    checksum += word;
                }
            }
        }
    }

    checksum = (u16)IXGBE_EEPROM_SUM - checksum;

    return checksum;
}
```

## 6.2.9 Software Reserved Word 15 — Ext. Thermal Sensor Configuration Block Pointer — Word Address 0x26

Pointer to  External Thermal Sensor Configuration block.

| Bits | Name | Default | Description |
|---|---|---|---|
| 15:0 | External Thermal Sensor | 0xFFFF | Pointer to  External Thermal Sensor Configuration block.<br>0x0000 and 0xFFFF indicates an invalid pointer. |

## 6.2.10 Software Reserved Word 16 — Alternate SAN MAC Block Pointer — Word Address 0x27

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

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

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

### 6.2.10.1 Alternate SAN MAC Address Block

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

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

### 6.2.10.2 Capabilities - Offset 0x0

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

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

## 6.2.11 Software Reserved Word 17 — Active SAN MAC Block Pointer — Word Address 0x28

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

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

### 6.2.11.1 Active SAN MAC Address Block

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

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

# 6.3 EEPROM Hardware Sections

This module contains address control words and hardware pointers indicated as HW in the table of Section 6.1.

## 6.3.1 EEPROM Hardware Section — Auto-Load Sequence

The following table lists sections of auto-read following device reset events.

**Table 6-6 EEPROM Section Auto-Read**

| | LAN_PWR_GOOD | PCIe Reset or PCIe Inband Reset | D3 to D0 transit or FLR (per port) | SW Reset (per port) | Link Reset (per port) | Force TCO |
|---|---|---|---|---|---|---|
| PCIe Analog Configuration | X | | | | | |
| PCIe General Configuration | | X | | | | |
| PCIe Function 0/1 Config Space (for each LAN port) | | X | X | | | |
| LAN Core and CSRs (for each LAN port) | | X | X | X | X | X |
| MAC Module (for each LAN port) | X (1) | X (2) | X (2) | X (2) | X | |

1. The MAC module is auto-read only if manageability or wake up are enabled.
2. The MAC module is auto-read only if the manageability unit is not enabled.

## 6.3.2 EEPROM Init Module

The init section (EEPROM control word 0x1, 0x2, and 0x38) are read after a LAN_PWR_GOOD reset and PCIe Reset.

### 6.3.2.1 EEPROM Control Word 1 — Address 0x00

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:12 | Reserved | 0x0 | Reserved. | |
| 11:8 | EEPROM Size | 0010b | These bits indicate the EEPROM's actual size.<br>Mapped to EEC.EE_Size (see field definition in the EEC register section). | |

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 7:6 | Signature | 01b | The Signature field indicates to the 82599 that there is a valid EEPROM present.<br>If the Signature field is not 01b, the other bits in this word are ignored, no further EEPROM read is performed, and the default values are used for the configuration space IDs. | |
| 5 | MNG Enable | 0b | Manageability Enable.<br>When set, indicates that the manageability block is enabled. When cleared, the manageability block is disabled (clock gated).<br>Mapped to GRC.MNG_EN | |
| 4 | EEPROM Protection | 0b | If set to 1, EEPROM protection schemes are enabled. | |
| 3:0 | HEPSize | 0b | Hidden EEPROM Block Size.<br>This field defines the EEPROM area accessible only by manageability firmware. It can also be used to store secured data and other manageability functions. The size in bytes of the secured area equals:<br>0 bytes (if HEPSize equals zero), or 2^ HEPSize bytes (2 bytes, 4 bytes, …32 KB.) | |

### 6.3.2.2 EEPROM Control Word 2 — Address 0x01

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:7 | Reserved | 0x0 | Reserved. | |
| 6 | Core KR PLL Gate Disable | 0b | When set disable the gating of the Core KR_PLL in device low power states and Non-KR Modes<br>*Note:* In case KR_Dis bit is set, KR-PLL is disabled regardless the value of this bit | |
| 5:3 | Reserved | 001b | Reserved. | |
| 2 | Core XAUI Gate Disable | 0b | When set disables the gating of the Core XAUI PLL in device low power states and Non-XAUI Modes. | |
| 1 | Core Clocks Gate Disable | 0b | During nominal operation this bit should be zero enabling core clock gating.<br>When set disables the gating of the core clock in low power state. Setting this bit also has side affects disabling auto link down (when both MNG and WOL are disabled) and also keeps the LEDs active. | |
| 0 | PCIe PLL Gate Disable | 0b | When set disables the gating of the PCIe PLL in L1/2 states. | |

#### 6.3.2.3 EEPROM Control Word 3 — Address 0x38

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:2 | Reserved | 0x0 | Reserved. | |
| 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.<br>If the LAN PCI disable bit in the NVM is set for Port 1, then the APM bit must be cleared. | |
| 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.<br>If the LAN PCI disable bit in the NVM is set for Port 0, then the APM bit must be cleared. | |

**Note:** Bits 1:0 should not be set if the respective port is disabled by PCIe Control 2 word bits 1:0.

### 6.3.3 PCIe Analog Configuration Module

These sections are loaded only after LAN_PWR_GOOD only. These sections contain the analog default configurations for the 82599's PCIe analog parts. Word 0x3 is the pointer for this section (the EEPROM address, in words).

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

| Word Offset | Description | |
|---|---|---|
| 0x0 | Section Length | |
| 0x1 | PCIe Analog Address 1 | |
| 0x2 | PCIe Analog Data 1 | |
| 0x3 | PCIe Analog Address 2 | |
| 0x4 | PCIe Analog Data 2 | |
| …. | …… | |

#### 6.3.3.1 Section Length — Offset 0x00

The section length word contains the length of the section in words. Note that section length does not include a count for the section length word.

#### 6.3.3.2 PCIe Analog Address — Offset 0x01, 0x03, 0x05...

Each odd offset word in the PCIe analog section is the register address. The PCIe analog registers are 2 words wide with a 12-bit address width. Bits 11:2 are the register address (in Dwords) and bit 1 select the upper/lower word of the Dword register.

#### 6.3.3.3 PCIe Analog Data — Offset 0x02, 0x04, 0x06...

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

### 6.3.4 Core 0/1 Analog Configuration Modules

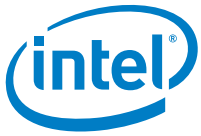
These modules are loaded after LAN_PWR_GOOD only. They contain the analog default configurations for the 82599's XAUI/KR analog parts. Words 0x4-0x5 are the pointers for these sections (the EEPROM address, in words).

The structure of all sections is similar to the following structure.

| Word offset | Bits | Description | Reserved |
|---|---|---|---|
| 0x0 | 15:0 | Section Length — Section 6.3.4.1. | |
| 0x1 | 15:8 | Configuration Address — Section 6.3.4.2 | |
| 0x1 | 7:0 | Configuration Data — Section 6.3.4.2 | |
| 0x2 | 15:8 | Configuration Address — Section 6.3.4.2 | |
| 0x2 | 7:0 | Configuration Data — Section 6.3.4.2 | |
| .... | | ..... | |

#### 6.3.4.1 Section Length — Offset 0x00

The section length word contains the length of the section in words. Note that section length does not include a count for the section length word.



#### 6.3.4.2 Data and Address Words — Offset 0x01, 0x02, 0x03...

Each word in the analog configuration section has the same structure: bits 15:8 are the register address and bits 7:0 are the register's data. The analog registers are eight bits wide with an 8-bit address width. After reading the EEPROM word, the register specified in bits 15:8 is loaded with the data from bits 7:0.

### 6.3.5 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 EEPROM (full-byte address; must be word aligned).

| Offset | Description |
|---|---|
| 0x00 | Section Length Section 6.3.5.1. |
| 0x01 | PCIe Init Configuration 1 Section 6.3.5.2 |
| 0x02 | PCIe Init Configuration 2 Section 6.3.5.3 |
| 0x03 | PCIe Init Configuration 3 Section 6.3.5.4 |
| 0x04 | PCIe Control 1 Section 6.3.5.5 |
| 0x05 | PCIe Control 2 Section 6.3.5.6 |
| 0x06 | PCIe LAN Power Consumption Section 6.3.5.7 |
| 0x07 | PCIe Control 3 Section 6.3.5.8 |
| 0x08 | PCIe Sub-System ID Section 6.3.5.9 |
| 0x09 | PCIe Sub-System Vendor ID Section 6.3.5.10 |
| 0x0A | PCIe Dummy Device ID Section 6.3.5.11 |
| 0x0B | PCIe Device Revision ID Section 6.3.5.12 |
| 0x0C | IOV Control Word 1 Section 6.3.5.13 |
| 0x0D | IOV Control Word 2 Section 6.3.5.14 |
| 0x0E | Reserved |
| 0x0F | Reserved |
| 0x10 | Reserved |
| 0x11 | Serial Number Ethernet MAC Address Section 6.3.5.15 |

| Offset | Description |
|---|---|
| 0x12 | Serial Number Ethernet MAC Address Section 6.3.5.16 |
| 0x13 | Serial Number Ethernet MAC Address Section 6.3.5.17 |
| 0x14 | PCIe L1 Exit latencies Section 6.3.5.18 |
| 0x15 | Spare Section 6.3.5.19 |

### 6.3.5.1 Section Length — Offset 0x00

The section length word contains the length of the section in words. Note that section length does not include a count for the section length word.

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

### 6.3.5.2 PCIe Init Configuration 1 — Offset 0x01

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15 | Reserved | 0b | Reserved. | |
| 14:12 | L0s acceptable latency | 011b | Loaded to the *Endpoint L0s Acceptable Latency* field in the Device Capabilities register as part of the PCIe Configuration registers at power up. | |
| 11:9 | L0s G2 Sep exit latency | 111b | L0s exit latency G2S. Loaded to *L0s Exit Latency* field in the Link Capabilities register as part of the PCIe Configuration registers in PCIe V2.0 (5GT/s) system with a separate clock setting. | |
| 8:6 | L0s G2 Com exit latency | 100b | L0s exit latency G2C. Loaded to *L0s Exit Latency* field in the Link Capabilities register as part of the PCIe Configuration registers in PCIe V2.0 (5GT/s) system with a common clock setting. | |
| 5:3 | L0s G1 Sep exit latency | 111b | L0s exit latency G1S. Loaded to *L0s Exit Latency* field in the Link Capabilities register as part of the PCIe Configuration registers in PCIe v2.0 (2.5GT/s) system with a separate clock setting. | |
| 2:0 | L0s G1 Com exit latency | 011b | L0s exit latency G1C. Loaded to L0s Exit Latency field in the Link Capabilities register as part of the PCIe Configuration registers in PCIe v2.0 (2.5GT/s) system with a common clock setting. | |

### 6.3.5.3 PCIe Init Configuration 2 — Offset 0x02

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:13 | Reserved | 0x0 | Reserved. | |
| 12 | FLR delay disable | 0b | Disable the FLR value in the *FLR Delay* field in this word. | |
| 11:8 | FLR delay | 0x1 | FLR response time in cycles defines the delay from FLR assertion to its affect. | |
| 7:6 | PCI-E capability version | 10b | PCIe Capability Version.<br>This field must be set to 10b to use extended configuration capability (used for a timeout mechanism). This field is mapped to GCR.PCIe_Capability_Version. | |
| 5 | ECRC generation enable | 1b | Loaded into the *ECRC Generation Capable* bit of the PCIe Configuration registers.<br>At 1b the device is capable of generating ECRC. | |
| 4 | ECRC check enable | 1b | Loaded into the *ECRC Check Capable* bit of the PCIe Configuration registers.<br>At 1b the device is capable of checking ECRC. | |
| 3 | FLR capability enable | 1b | *FLR Capability Enable* bit is loaded to the PCIe Configuration registers via the Device Capabilities register. | |
| 2 | Cache line size | 0b | Cache Line Size<br>0b = 64 bytes.<br>1b = 128 bytes. | |
| 1:0 | Max payload size | 10b | Maximum payload size support for TLPs. Loaded to the PCIe Configuration registers via the Device Capabilities register. | |

### 6.3.5.4 PCIe Init Configuration 3 — Offset 0x03

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:4 | Reserved | 0x0 | Reserved. | |
| 3 | PCIe Down Reset Disable | 0b | Disables a core and reset when the PCIe link goes down. | |
| 2:1 | Act_Stat_PM_Sup | 11b | *Active State Link PM Support* is loaded to the PCIe Configuration registers via the *Link Capabilities* field. | |
| 0 | Slot_Clock_Cfg | 1b | Slot Clock Configuration.<br>When set, the 82599 uses the PCIe reference clock supplied on the connector (for add-in solutions). This bit is loaded to the "PCIe configuration registers" -> "Link Status". | |

### 6.3.5.5 PCIe Control 1 —Offset 0x04

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:5 | Reserved | 0x0 | Reserved. | |
| 4 | DIS Clock Gating in DISABLE | 1b | Disable clock gating when LTSSM is in a disable state. | |
| 3 | DIS Clock Gating in L2 | 1b | Disable clock gating when LTSSM is at L2 state. | |
| 2 | DIS Clock Gating in L1 | 1b | Disable clock gating when LTSSM is at L1 state. | |
| 1 | DIS Clock Gating in G2 | 1b | Disable clock gating in PCIe V2.0 (5GT/s). | |
| 0 | DIS Clock Gating in G1 | 1b | Disable clock gating in PCIe v2.0 (2.5GT/s). | |

### 6.3.5.6 PCIe Control 2 — Offset 0x05

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15 | Completion Timeout Resend | 0b | When set, enables a response to a request once the completion timeout expired.<br>This bit is mapped to GCR.Completion_Timeout_Resend.<br>0b = Do not resend request on completion timeout.<br>1b = Resend request on completion timeout. | |
| 14:4 | Reserved | 0x0 | Reserved. | |
| 3 | LAN Function Select | 0b | When the LAN Function Select field = 0b, LAN 0 is routed to PCI function 0 and LAN 1 is routed to PCI function 1.<br>If the LAN Function Select field = 1b, LAN 0 is routed to PCI function 1 and LAN 1 is routed to PCI function 0.<br>This bit is mapped to FACTPS[30]. | |
| 2 | Dummy Function Enable | 0b | Controls the behavior of function 0 when disabled.<br>0b = Legacy Mode.<br>1b = Dummy Function Mode.<br>See Section 4.4. | |
| 1 | LAN PCI Disable | 0b | LAN PCI Disable. When set to 1b, one LAN port is disabled.<br>The function that is disabled is determined by the *LAN Disable Select* bit. If the disabled function is function 0, it acts as a dummy function or the other LAN function depending on the *Dummy Function Enable* setting. | |
| 0 | LAN Disable Select | 0b | LAN Disable Select<br>0b = LAN 0 is disabled.<br>1b = LAN 1 is disabled. | |

### 6.3.5.7 PCIe LAN Power Consumption — Offset 0x06

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | LAN D0 Power | | The value in this field is reflected in the PCI Power Management Data register of the LAN functions for D0 power consumption and dissipation (Data_Select = 0 or 4).<br>Power is defined in 100 mW units. The power includes also the external logic required for the LAN function. | |
| 7:5 | Function 0 Common Power | | The value in this field is reflected in the PCI Power Management Data register of function 0 when the Data_Select field is set to 8 (common function).<br>The MSBs in the Data register that reflects the power values are padded with zeros. When one port is used this field should be set to 0. | |
| 4:0 | LAN D3 Power | | The value in this field is reflected in the PCI Power Management Data register of the LAN functions for D3 power consumption and dissipation (Data_Select = 3 or 7).<br>Power is defined in 100 mW units. The power includes also the external logic required for the LAN function. The MSBs in the Data register that reflects the power values are padded with zeros. | |

### 6.3.5.8 PCIe Control 3 — Offset 0x07

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15 | Reserved | 0b | Reserved. | |
| 14 | PREFBAR | 0b | Prefetchable bit indication in the memory BARs:<br>0b = BARs are marked as non-prefetchable<br>1b = BARs are marked as prefetchable | |
| 13 | CSR_Size | 0b | The CSR_Size and FLASH_Size fields define the usable Flash size and CSR mapping window size.<br>*Note:* When the CSR_Size and Flash_Size fields in EEPROM are set to 0, Flash Bar in the PCI configuration space is disabled. | |
| 12 | IO_Sup | 1b | I/O Support (affects I/O BAR request).<br>When set to 1b, I/O is supported. When cleared the *I/O Access Enable* bit in the Command register (as part of the Mandatory PCI Configuration) is RO at 0b. | |
| 11 | Reserved | 0b | Reserved. | |
| 10:8 | Flash_Size | 010b | Indicates a Flash size of 64 KB * 2 ^Flash_Size.<br>The Flash_Size impacts the requested memory space for the Flash and expansion ROM BARs in PCIe configuration space.<br>See Table 6-7, Usable Flash_Size below.<br>*Note:* When the CSR_Size and Flash_Size fields in EEPROM are set to 0, Flash Bar in the PCI configuration space is disabled. | |
| 7:2 | Reserved | 0x0 | Reserved | |
| 1 | Load Subsystem IDs | 1b | When set to 1b, indicates that the function is to load its PCIe sub-system ID and sub-system vendor ID from the EEPROM (offset 0x8 and 0x9 in this section). | |

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 0 | Load Device ID | 1b | When set to 1b, indicates that the function is to load its PCI device ID from the EEPROM (offset 0x0A in this section for dummy device ID and offset 2 in PCIe configuration space 0/1 section for active functions). | |

**Table 6-7 Usable Flash_Size**

| Flash_Size | CSR_Size | Resulted CSR + Flash BAR Size | Installed Flash Device | Usable Flash Space |
|---|---|---|---|---|
| 000b | 0 | 128 KByte | No Flash | 0 |
| 000b | 1 | N/A | N/A | Reserved |
| 001b | 0 | 256 KByte | 128 KByte | 128 KByte |
| 001b | 1 | N/A | N/A | Reserved |
| 010b | 0 | 256 KByte | 256 KByte | 256 KByte minus 128 KByte |
| 010b | 1 | 512 KByte | 256 KByte | 256 KByte |
| 011b | 0 | 512 KByte | 512 KByte | 512 KByte minus 128 KByte |
| 011b | 1 | 1 MByte | 512 KByte | 512 KByte |
| 100b | 0 | 1 MByte | 1 MByte | 1 MByte minus 128 KByte |
| 100b | 1 | 2 MByte | 1 MByte | 1 MByte |
| 101b | 0 | 2 MByte | 2 MByte | 2 MByte minus 128 KByte |
| 101b | 1 | 4 MByte | 2 MByte | 2 MByte |
| 110b | 0 | 4 MByte | 4 MByte | 4 MByte minus 128 KByte |
| 110b | 1 | 8 MByte | 4 MByte | 4 MByte |
| 111b | 0 | 8 MByte | 8 MByte | 8 MByte minus 128 KByte |
| 111b | 1 | 16 MByte | 8 MByte | 8 MByte |

### 6.3.5.9 PCIe Sub-System ID — Offset 0x08

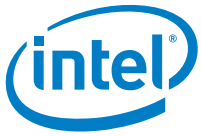
If the load sub-system IDs in offset 0x7 of this section is set, this word is read in to initialize the sub-system ID. The default value is 0x0.

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

### 6.3.5.10 PCIe Sub-System Vendor ID — Offset 0x09

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

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



### 6.3.5.11 PCIe Dummy Device ID — Offset 0x0A

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

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

### 6.3.5.12 PCIe Device Revision ID —Offset 0x0B

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | Reserved | 0x0 | Set to 0x0 | |
| 7:0 | DEVREVID | 0x1 | Device Rev ID.<br>The actual device revision ID is the EEPROM value XORed with the hardware value (0x0 for the 82599 A-0 and 0x1 for the 82599 B-0). | |

### 6.3.5.13 IOV Control Word 1 — Offset 0x0C

This word controls the behavior of IOV functionality.

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

### 6.3.5.14 IOV Control Word 2 — Offset 0x0D

This word defines the device ID for virtual functions.

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | VDev ID | 0x10ED | Virtual function device ID. | |

### 6.3.5.15 Serial Number Ethernet MAC Address — Offset 0x11

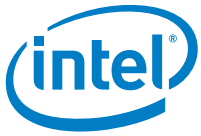
| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | Serial Number Ethernet MAC Address 0, Byte 1 | | Part of the Ethernet MAC address used to generate the PCIe serial number.<br>See Section 9.4.2. | |
| 7:0 | Serial Number Ethernet MAC Address 0, Byte 0 | | Part of the Ethernet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

### 6.3.5.16 Serial Number Ethernet MAC Address — Offset 0x12

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

### 6.3.5.17 Serial Number Ethernet MAC Address — Offset 0x13

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



### 6.3.5.18 PCIe L1 Exit latencies — Offset 0x14

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15 | Reserved | 0b | Reserved. | |
| 14:12 | L1_Act_Acc_Latency | 110b | Loaded to the *Endpoint L1 Acceptable Latency* field in the Device Capabilities register as part of the PCIe Configuration registers at power up. | |
| 11:9 | L1 G2 Sep exit latency | 101b | L1 exit latency G2S. Loaded to the Link Capabilities register via the *L1 Exit Latency* field in PCIe V2.0 (5GT/s) systems that have a separate clock setting. | |
| 8:6 | L1 G2 Com exit latency | 011b | L1 exit latency G2C. Loaded to the Link Capabilities register via the *L1 Exit Latency* field in PCIe V2.0 (5GT/s) systems that have a common clock setting. | |
| 5:3 | L1 G1 Sep exit latency | 100b | L1 exit latency G1S. Loaded to the Link Capabilities register via the *L1 Exit Latency* field in PCIe v2.0 (2.5GT/s) systems that have a separate clock setting. | |
| 2:0 | L1 G1 Com exit latency | 010b | L1 exit latency G1C. Loaded to the *L*ink Capabilities register via the *L1 Exit Latency* field in PCIe v2.0 (2.5GT/s) systems that have a common clock setting. | |

### 6.3.5.19 Reserved — Offset 0x15

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:2 | Reserved | 0x1 | Reserved. | |
| 1 | MSIX Memory | 1b | MSIX memory ECC enable. | |
| 0 | CDQ Memory | 1b | CDQ memory ECC enable. | |

## 6.3.6 PCIe Configuration Space 0/1 Modules

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

| Offset | Description |
|---|---|
| 0x00 | Section LengthSection 6.3.6.1. |
| 0x1 | Control WordSection 6.3.6.2 |
| 0x2 | Device IDSection 6.3.6.3 |
| 0x3 | CDQM Memory Base 0/1 LowSection 6.3.6.4 |
| 0x4 | CDQM Memory Base 0/1 HighSection 6.3.6.5 |
| 0x5 | ReservedSection 6.3.6.6 |

### 6.3.6.1 Section Length — Offset 0x00

The section length word contains the length of the section in words. Note that section length does not include a count for the section length word.

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

### 6.3.6.2 Control Word — Offset 0x01

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:14 | Reserved | 00b | Reserved. | |
| 13:12 | Interrupt Pin | 0b for LAN0<br>1b for LAN1 | Controls the value advertised in the *Interrupt Pin* field of the PCI configuration header for this device/function.<br>Values of 00b, 01b, 10b and 11b correspond to INTA#, INTB#, INTC# and INTD# respectively. When one port is used this field must be set to 00b (using INTA#) to comply with PCI spec requirements. | |
| 11 | Storage Class | 0b | When set, the class code of this port is set to 0x010000 (SCSI). When cleared, the class code of this port is set to 0x020000 (LAN). | |
| 10 | MSI Mask | 1b | MSI per-vector masking setting.<br>This bit is loaded to the masking bit (bit 8) in the Message Control of the MSI Configuration Capability structure. | |
| 9 | Reserved | 1b | Reserved. | |

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 8 | LAN Boot Disable | 1b | A value of 1b disables the expansion ROM BAR in the PCI configuration space. | |
| 7 | Reserved | 0b | Reserved. | |
| 6:0 | MSI_X_N | 0x3F | This field specifies the number of entries in the MSI-X tables for this function.<br>MSI_X_N is equal to the number of entries minus one. For example, a return value of 0x7 means eight vectors are available. This field is loaded into the PCIe MSI-X capabilities structure. The MSI_X_N must not exceed 0x3F (64 MSI-X vectors). | |

#### 6.3.6.3 Device ID — Offset 0x02 Device ID

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | Device_ID | 0x10D8 | If the *Load Device ID* in offset 0x7 of the PCIe General configuration section is set, this word is loaded to the device ID of the LAN function. | |

#### 6.3.6.4 CDQM Memory Base 0/1 Low — Offset 0x03 [Reserved]

#### 6.3.6.5 CDQM Memory Base 0/1 High — Offset 0x04 [Reserved]

#### 6.3.6.6 EEPROM PCIe Configuration Space 0/1 - Offset 0x05 [Reserved]

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | Reserved | 0x0 | Reserved. | |

## 6.3.7 LAN Core 0/1 Modules

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

| Offset | High Byte[15:8] | Low Byte[7:0] | Section |
|---|---|---|---|
| 0x0 | Section Length - Section 6.3.7.1. | | |
| 0x1 | Ethernet MAC Address Byte 2 | Ethernet MAC Address Byte 1 | Section 6.3.7.2.1 |
| 0x2 | Ethernet MAC Address Byte 4 | Ethernet MAC Address Byte 3 | Section 6.3.7.2.2 |
| 0x3 | Ethernet MAC Address Byte 6 | Ethernet MAC Address Byte 5 | Section 6.3.7.2.3 |
| 0x4 | LED 1 configuration | LED 0 Configuration | Section 6.3.7.3.1 |
| 0x5 | LED 3 Configuration | LED 2 Configuration | Section 6.3.7.3.2 |
| 0x6 | SDP Control | | Section 6.3.7.4 |
| 0x7 | Filter Control | | Section 6.3.7.5 |

### 6.3.7.1 Section Length — Offset 0x00

The section length word contains the length of the section in words. Note that section length does not include a count for the section length word.

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

### 6.3.7.2 Ethernet MAC Address Registers

The Ethernet Individual Address (IA) is a 6-byte field that must be unique for each NIC or LOM and must also be unique for each copy of the EEPROM 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 datasheet, the numbering convention is as follows:

| Vendor | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Intel original | 00 | AA | 00 | Variable | Variable | Variable |
| Intel new | 00 | A0 | C9 | Variable | Variable | Variable |

#### 6.3.7.2.1 Ethernet MAC Address Register1 — Offset 0x01

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | Eth_Addr_Byte2 | 0x0 | Ethernet MAC address byte 2. | |
| 7:0 | Eth_Addr_Byte1 | 0x0 | Ethernet MAC address byte 1. | |

#### 6.3.7.2.2 Ethernet MAC Address Register2 — Offset 0x02

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | Eth_Addr_Byte4 | 0x0 | Ethernet MAC address byte 4. | |
| 7:0 | Eth_Addr_Byte3 | 0x0 | Ethernet MAC address byte 3. | |

#### 6.3.7.2.3 Ethernet MAC Address Register3 — Offset 0x03

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | Eth_Addr_Byte6 | 0x0 | Ethernet MAC address byte 6. | |
| 7:0 | Eth_Addr_Byte5 | 0x0 | Ethernet MAC address byte 5. | |

### 6.3.7.3 LED Configuration

The LEDCTL register (Section 8.2.3.1.6) defaults are loaded from two words as listed in the following tables.

#### 6.3.7.3.1 LED Control Lower Word — Offset 0x04

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | LEDCTL1 | 0x0 | LED 1 control. | |
| 7:0 | LEDCTL0 | 0x0 | LED 0 control. | |

#### 6.3.7.3.2 LED control Upper Word — Offset 0x05

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | LEDCTL3 | 0x0 | LED 3 control. | |
| 7:0 | LEDCTL2 | 0x0 | LED 2 control. | |

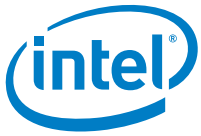
**Note:** The content of the EEPROM words is similar to the register content.

### 6.3.7.4 SDP Control — Offset 0x06

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15 | SDP1_NATIVE | 0b | Defines the SDP1 operating mode that is mapped to ESDP.SDP1_NATIVE loaded at power up:<br>0b = Operates as generic software controlled IO.<br>1b = Native mode operation (hardware function). | |
| 14:12 | Reserved | 000b | Set to 000b. | |
| 11 | SDPDIR[3] | 0b | SDP3 Pin.<br>Initial direction is mapped to ESDP.SDP3_IODIR loaded at power up. | |
| 10 | SDPDIR[2] | 0b | SDP2 Pin.<br>Initial direction is mapped to ESDP.SDP2_IODIR loaded at power up. | |
| 9 | SDPDIR[1] | 0b | SDP1 Pin.<br>Initial direction is mapped to ESDP.SDP1_IODIR loaded at power up. | |
| 8 | SDPDIR[0] | 0b | SDP0 Pin.<br>Initial direction is mapped to ESDP.SDP0_IODIR loaded at power up. | |
| 7:4 | Reserved | 0x0 | Reserved. | |
| 3 | SDPVAL[3] | 0b | SDP3 Pin.<br>Initial output value is mapped to ESDP.SDP3_DATA loaded at power up. | |
| 2 | SDPVAL[2] | 0b | SDP2 Pin.<br>Initial output value is mapped to ESDP.SDP2_DATA loaded at power up. | |
| 1 | SDPVAL[1] | 0b | SDP1 Pin.<br>Initial output value is mapped to ESDP.SDP1_DATA loaded at power up. | |
| 0 | SDPVAL[0] | 0b | SDP0 Pin.<br>Initial output value is mapped to ESDP.SDP0_DATA loaded at power up. | |

### 6.3.7.5 Filter Control — Offset 0x07

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | Reserved | 0x1 | *Note:* Reserved. | |



## 6.3.8 MAC 0/1 Modules

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

| Offset | Content | Section |
|---|---|---|
| | Section Length = 0x5 | |
| 0x1 | Link Mode Configuration | Section 6.3.8.2 |
| 0x2 | Swap Configuration | Section 6.3.8.3 |
| 0x3 | Swizzle and Polarity Configuration | Section 6.3.8.4 |
| 0x4 | Auto Negotiation Default Bits | Section 6.3.8.5 |
| 0x5 | AUTOC2 Upper Half | Section 6.3.8.6 |
| 0x6 | SGMIIC Lower Half | Section 6.3.8.7 |
| 0x7 | KR-PCS configurations | Section 6.3.8.8 |

### 6.3.8.1 Section Length — Offset 0x00

The section length word contains the length of the section in words. Note that the section length does not include a count for the section length word.

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

### 6.3.8.2 Link Mode Configuration – Offset 0x01

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:13 | Link Mode Select | 100b | 000b = 1 Gb/s link (no auto-negotiation).<br>001b = 10 Gb/s parallel link (no auto-negotiation).<br>010b = 1 Gb/s link with clause 37 auto-negotiation enable.<br>011b = 10 Gb/s serial link (no auto-negotiation). Supports SFI without backplane auto-negotiation.<br>100b = KX/KX4/KR backplane auto-negotiation enable. 1 Gb/s (Clause 37) auto-negotiation disable.<br>101b = SGMII 1G/100M link.<br>110b = KX/KX4/KR backplane auto-negotiation enable. 1 Gb/s (Clause 37) auto-negotiation enable.<br>111b = KX/KX4/KR backplane auto-negotiation enable. SGMII 1 Gb/s or 100 Mb/s (in KX) enable. These bits are mapped to AUTOC.LMS | |
| 12 | Restart AN | 0b | Restarts the KX/KX4/KR backplane auto-negotiation process (self-clearing bit).<br>Mapped to AUTOC.Restart_Auto Negotiation. | |
| 11 | RATD | 0b | Restarts backplane auto-negotiation on a transition to Dx.<br>Mapped to AUTOC.RATD and applied to AUTOC.RATD. | |
| 10 | D10GMP | 0b | Disables 10 Gb/s (KX4/KR) on Dx (Dr/D3) without main power.<br>Mapped to AUTOC.D10GMP. | |
| 9 | 1G PMA_PMD | 1b | PMA/PMD used for 1 Gb/s.<br>Mapped to AUTOC.1G_PMA_PMD. | |
| 8:7 | 10G PMA_PMD_ PARALLEL | 01b | PMA/PMD used for 10 Gb/s over four differential pairs for TX and RX each.<br>Mapped to AUTOC.10G_PMA_PMD_PARALLEL. | |
| 6:2 | ANSF | 00001b | AN Selector Field (Debug mode).<br>Mapped to AUTOC.ANSF | |
| 1 | ANACK2 | 0b | *AN Ack2* field.<br>This value is transmitted in the *Achnowledge2* field of the Null Next Page that is transmitted during next page handshake. Mapped to AUTOC.ANACK2 | |
| 0 | Reserved | 0b | Reserved. | |

### 6.3.8.3 SWAP Configuration — Offset 0x02

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:14 | Swap_Rx_Lane_0 | 00b | Determines which core lane is mapped to MAC Rx lane 0.<br>00b = Core Rx lane 0 to MAC Rx lane 0.<br>01b = Core Rx lane 1 to MAC Rx lane 0.<br>10b = Core Rx lane 2 to MAC Rx lane 0.<br>11b = Core Rx lane 3 to MAC Rx lane 0.<br>Mapped to SERDESC.swap_rx_lane_0. | |
| 13:12 | Swap_Rx_Lane_1 | 01b | Determines which core lane is mapped to MAC Rx lane 1.<br>Mapped to SERDESC.swap_rx_lane_1. | |
| 11:10 | Swap_Rx_Lane_2 | 10b | Determines which core lane is mapped to MAC Rx lane 2.<br>Mapped to SERDESC.swap_rx_lane_2. | |
| 9:8 | Swap_Rx_Lane_3 | 11b | Determines which core lane is mapped to MAC Rx lane 3.<br>Mapped to SERDESC.swap_rx_lane_3. | |
| 7:6 | Swap_Tx_Lane_0 | 00b | Determines the core destination Tx lane for MAC Tx lane 0.<br>00b = MAC tx lane 0 to Core Tx lane 0.<br>01b = MAC tx lane 0 to Core Tx lane 1.<br>10b = MAC tx lane 0 to Core Tx lane 2.<br>11b = MAC tx lane 0 to Core Tx lane 3.<br>Mapped to SERDESC.swap_tx_lane_0. | |
| 5:4 | Swap_Tx_Lane_1 | 01b | Determines the core destination Tx lane for MAC Tx lane 1.<br>Mapped to SERDESC.swap_tx_lane_1. | |
| 3:2 | Swap_Tx_Lane_2 | 10b | Determines the core destination Tx lane for MAC Tx lane 2.<br>Mapped to SERDESC.swap_tx_lane_2. | |
| 1:0 | Swap_Tx_Lane_3 | 11b | Determines the core destination Tx lane for MAC Tx lane 3.<br>Mapped to SERDESC.swap_tx_lane_3. | |

### 6.3.8.4 Swizzle and Polarity Configuration — Offset 3

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:12 | Swizzle_Rx | 0x0 | Swizzle_Rx[0] — Swizzles the bits of MAC Rx lane 0.<br>Swizzle_Rx[1] — Swizzles the bits of MAC Rx lane 1.<br>Swizzle_Rx[2] — Swizzles the bits of MAC Rx lane 2.<br>Swizzle_Rx[3] — Swizzles the bits of MAC Rx lane 3.<br>Swizzles the bits if set to 1b.<br>Mapped to SERDESC.Swizzle_Rx_lanes. | |
| 11:8 | Swizzle_Tx | 0x0 | Swizzle_Tx[0] — Swizzles the bits of MAC Tx lane 0.<br>Swizzle_Tx[1] — Swizzles the bits of MAC Tx lane 1.<br>Swizzle_Tx[2] — Swizzles the bits of MAC Tx lane 2.<br>Swizzle_Tx[3] — Swizzles the bits of MAC Tx lane 3.<br>Swizzles the bits if set to 1b.<br>Mapped to SERDESC.Swizzle_Tx_lanes. | |

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 7:4 | Polarity_Rx | 0x0 | Polarity_Rx[0] — Changes the bit polarity of MAC Rx lane 0<br>Polarity_Rx[1] — Changes the bit polarity of MAC Rx lane 1<br>Polarity_Rx[2] — Changes the bit polarity of MAC Rx lane 2<br>Polarity_Rx[3] — Changes the bit polarity of MAC Rx lane 3<br>Changes bit polarity if set to 1b.<br>Mapped to SERDESC.Rx_lanes_polarity. | |
| 3:0 | Polarity_Tx | 0x0 | Polarity_Tx[0] — Changes the bit polarity of MAC Tx lane 0.<br>Polarity_Tx[1] — Changes the bit polarity of MAC Tx lane 1.<br>Polarity_Tx[2] — Changes the bit polarity of MAC Tx lane 2.<br>Polarity_Tx[3] — Changes the bit polarity of MAC Tx lane 3.<br>Changes bit polarity if set to 1b.<br>Mapped to SERDESC.Tx_lanes_polarity. | |

### 6.3.8.5 Auto Negotiation Defaults — Offset 4

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:14 | KX Support | 1b | The value of these EEPROM settings are shown in bits A0:A1 of the *Technology Ability* field of the backplane auto-negotiation word while A2 field is configured in the KR_support bit:<br>00b = A0 = 0 A1 = 0. KX not supported. KX4 not supported. Value is illegal if KR is also not supported (AUTOC.KR_support = 0b).<br>01b = A0 = 1 A1 = 0. KX supported. KX4 not supported.<br>10b = A0 = 0 A1 = 1. KX not supported. KX4 supported.<br>11b = A0 = 1 A1 = 1. KX supported. KX4 supported.<br>Mapped to AUTOC.KX_support. | |
| 13:12 | Pause Bits | 0b | The value of these bits is loaded to bits D11:D10 of the link code word (pause data). Bit 12 is loaded to D11.<br>Mapped to AUTOC.PB. | |
| 11 | RF | 0b | This bit is loaded to the RF bit in the backplane auto-negotiation word.<br>Mapped to AUTOC.RF. | |
| 10:9 | AN Parallel Detect Timer | 00b | Configures the parallel detect counters.<br>00b = 1 ms.<br>01b = 2 ms.<br>10b = 5 ms.<br>11b = 8 ms.<br>Mapped to AUTOC.ANPDT. | |
| 8 | AN RX Loose Mode | 0b | Enables less restricted functionality (allow 9/11 bit symbols).<br>0b = Disables loose mode.<br>1b = Enables loose mode.<br>Mapped to AUTOC.ANRXLM. | |
| 7 | AN RX Drift Mode | 1b | Enables the drift caused by PPM in the RX data.<br>0b = Disables drift mode.<br>1b = Enables drift mode.<br>Mapped to AUTOC.ANRXDM. | |

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 6:3 | AN RX Align Threshold | 0011b | Sets the threshold to determine that the alignment is stable.<br>Sets how many stable symbols to find before declaring the AN_RX.<br>10b = Symbol stable.<br>Mapped to AUTOC.ANRXAT. | |
| 2 | FEC Ability | 1b | FEC Ability. Configures the F0 bit in the backplane auto-negotiation base link code word.<br>Should be set to 1b only if KR ability is set to 1b (AUTOC.KR = 1b).<br>0b = FEC not supported.<br>1b = FEC supported.<br>Mapped to AUTOC.FECA. | |
| 1 | FEC Requested | 0b | FEC requested.<br>Configures the F1 bit in the backplane auto-negotiation base link code word. Should be set to 1b only if KR ability is set to 1b (AUTOC.KR = 1b).<br>0b = FEC not requested from link partner.<br>1b = FEC requested from link partner.<br>Mapped to AUTOC.FECR. | |
| 0 | KR Support | 1b | Configures the A2 bit of the *Technology Ability Field* in the backplane auto-negotiation word while the A0:A1 field is configured according to the KX_support field (bits 31:30):<br>0b = KR not supported. Value is illegal if KX and KX4 are also not supported (AUTOC.KX_support = 00b).<br>1b = KR supported.<br>Mapped to AUTOC.KR_Support. | |

## 6.3.8.6 AUTOC2 Upper Half – Offset 5

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15 | Force FEC Enable | 0b | Force FEC Enable.<br>Enables FEC without dependency on the auto-negotiation resolution. Debug mode only. Mapped to AUTOC2.FORCE_FEC. | |
| 14 | Parallel Detect Disable | 0b | Disables the parallel detect part in the KX/KX4/KR backplane auto-negotiation process.<br>Mapped to AUTOC2.PDD. | |
| 13 | ANIGNRRXRF | 1b | AN Ignore Received RF Field.<br>Mapped to SGMIIC.ANIGNRRXRF | |
| 12 | Reserved | 0b | Reserved | |
| 11:8 | Reserved | 0x0 | Reserved | |
| 7 | Latch High 10G Aligned Indication | 0b | Override any de-skew alignment failures in the 10 Gb/s link (by latching high).<br>Mapped to AUTOC2.LH1GAI. | |
| 6 | Reserved | 0b | Reserved. | |

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 5 | AN 1G TIMEOUT EN | 1b | Auto Negotiation1 Gb/s Timeout Enable.<br>Mapped to PCS1GLCTL.AN 1G TIMEOUT EN | |
| 4 | Reserved | 0b | Reserved | |
| 3 | MAC DFT Override Comma Align | 0b | Override Internal Comma-Align Control.<br>Mapped to MDFTC2. MACDOCA. | |
| 2 | DDPT | 0b | Loaded to the *Disable DME Pages Transmit* bit in the AUTOC2 register. | |
| 1:0 | 10G PMA/PMD serial operation | 00b | PMA/PMD used for 10 Gb/s serial link.<br>Mapped to AUTOC2.10G_PMA_PMD_Serial. | |

### 6.3.8.7 SGMIIC Lower Half — Offset 6

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15 | ANSLNKTMR | 0b | AN SGMII Link-Timer.<br>Mapped to SGMIIC.ANSLNKTMR. | |
| 14 | ANSTRIG | 0b | AN SGMII Trigger.<br>Mapped to SGMIIC.ANSTRIG. | |
| 13 | ANSBYP | 0b | AN SGMII Bypass.<br>Mapped to SGMIIC.ANSBYP. | |
| 12 | ANSFLU100 | 0b | AN SGMII Force Link Up 100 Mb/s.<br>Mapped to SGMIIC.ANSFLU100. | |
| 11:8 | STXRASMP | 0x0 | Shift TX Rate-Adapt Sampling.<br>Mapped to SGMIIC.STXRASMP. | |
| 7:4 | SRXRARSMP | 0x0 | Shift RX Rate-Adapt Replicated Data Sampling.<br>Mapped to SGMIIC.SRXRARSMP. | |
| 3:0 | SRXRASSMP | 0x0 | Shift RX Rate-Adapt Single Data Sampling.<br>Mapped to SGMIIC.SRXRASSMP. | |

### 6.3.8.8 KR-PCS configurations — Offset 7

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15 | IE3_MODE | 1b | IEEE sync mode (debug mode).<br>Mapped to KRPCSFC.IE3_MODE. | |
| 14:11 | Reserved | 0x0 | Reserved. | |
| 10 | BYP_FEC_SIG_DET | 0b | Bypass FEC signal detect (Debug mode).<br>Mapped to KRPCSFC. BYP_FEC_SIG_DET. | |

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 9:0 | Reserved | 0x0 | Reserved. | |

## 6.3.9 CSR 0/1 Auto Configuration Modules

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

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

| Offset | High Byte[15:8] | Low Byte[7:0] | Section |
|---|---|---|---|
| 0x0 | Section Length = 3*n | | |
| 0x1 | CSR Address | | Section 6.3.9.2 |
| 0x2 | Data LSB | | Section 6.3.9.3 |
| 0x3 | Data MSB | | Section 6.3.9.4 |
| | … | | |
| 3*n — 2 | CSR Address | | Section 6.3.9.2 |
| 3*n — 1 | Data LSB | | Section 6.3.9.3 |
| 3*n | Data MSB | | Section 6.3.9.4 |

**Note:** The 82599 blocks any write to the Analog Configuration registers through these sections.

### 6.3.9.1 Section Length — Offset 0x0

The section length word contains the length of the section in words. Note that section length does not include a count for the section length word.

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

### 6.3.9.2 CSR Address — Offset 0x1, 0x4, 0x7...

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | CSR_ADDR | 0x0 | CSR address. | |

### 6.3.9.3 CSR Data LSB — Offset 0x2, 0x5, 0x8...

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | CSR_Data_LSB | 0x0 | CSR data LSB. | |

### 6.3.9.4 CSR Data MSB — Offset 0x3, 0x6, 0x9...

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | CSR_Data_MSB | 0x0 | CSR data MSB. | |

# 6.4 Firmware Module

The following table lists the EEPROM global offsets used by the 82599 firmware.

| Global MNG Word Offset | Description |
|---|---|
| 0x0 | Test Configuration Pointer - Section 6.4.1 |
| 0x1 | Reserved |
| 0x2 | LESM Module Pointer - Appendix B |
| 0x3 | Common Firmware Parameters - Section 6.4.2 |
| 0x4 | Pass Through Patch Configuration Pointer (Patch structure identical to the Loader Patch) - Section 6.4.2 |
| 0x5 | Pass Through LAN 0 Configuration Pointer - Section 6.4.3 |
| 0x6 | SideBand Configuration Pointer - Section 6.4.4 |
| 0x7 | Flexible TCO Filter Configuration Pointer - Section 6.4.5 |
| 0x8 | Pass Through LAN 1 Configuration Pointer - Section 6.4.3 |
| 0x9 | NC-SI Microcode Download Pointer - Section 6.4.6 |
| 0xA | NC-SI Configuration Pointer - Section 6.4.7 |

## 6.4.1 Test Configuration Module

### 6.4.1.1 Section Header — Offset 0x0

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | Block CRC | | | |
| 7:0 | Block Length | | Block length in words | |

### 6.4.1.2 SMBus Address — Offset 0x1

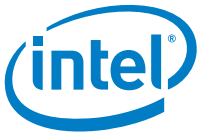
| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:9 | Reserved | | | |
| 8 | SMBus Interface Number | | | |
| 7:0 | SMBus Slave Address | | | |

### 6.4.1.3 Loopback Test Configuration — Offset 0x2

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:2 | Reserved | | | |
| 1 | Loopback Test Use SDP Output | | | |
| 0 | Loopback Test Enable | | | |

## 6.4.2 Common Firmware Parameters — (Global MNG Offset 0x3)

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15 | Reserved | 0b | Reserved, should be set to 0b. | |
| 14 | Redirection Sideband Interface | | 0b = SMBus.<br>1b = NC-SI. | |
| 13:11 | Reserved | 000b | Reserved. | |
| 10:8 | Manageability Mode | | 0x0 = None.<br>0x1 = Reserved.<br>0x2 = Pass Through (PT) mode.<br>0x3 = Reserved.<br>0x4:0x7 = Reserved. | |
| 7 | Port1 Manageability Capable | | 0b = Not capable<br>1b = Bits 3 is applicable to port 1. | |
| 6 | Port0 Manageability Capable | | 0b = Not capable<br>1b = Bits 3 is applicable to port 0. | |
| 5 | LAN1 Force TCO Reset Disable | 0b | 0b = Enable Force TCO reset on LAN1.<br>1b = Disable Force TCO reset on LAN1. | |
| 4 | LAN0 Force TCO Reset Disable | 0b | 0b = Enable Force TCO reset on LAN0.<br>1b = Disable Force TCO reset on LAN0. | |
| 3 | Pass Through Capable | | 0b = Disable.<br>1b = Enable. | |
| 2:0 | Reserved | 000b | Reserved. | |



## 6.4.3 Pass Through LAN 0/1 Configuration Modules

The following sections describe pointers and structures dedicated to pass-through mode for LAN 0 and LAN 1. LAN 0 structure is pointed by the *Firmware Module* pointer at offset 0x5. LAN 1 structure is pointed by the *Firmware Module* pointer at offset 0x8.

### 6.4.3.1 Section Header — Offset 0x0

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | Block CRC8 | | | |
| 7:0 | Block Length | | Block length in words. | |

### 6.4.3.2 LAN 0/1 IPv4 Address 0 (LSB) MIPAF0 — Offset 0x01

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | | | LAN 0/1 IPv4 Address 0, Byte 1. | |
| 7:0 | | | LAN 0/1 IPv4 Address 0, Byte 0. | |

### 6.4.3.3 LAN 0/1 IPv4 Address 0 (MSB) (MIPAF0) — Offset 0x02

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | | | LAN 0/1 IPv4 Address 0, Byte 3. | |
| 7:0 | | | LAN 0/1 IPv4 Address 0, Byte 2. | |

### 6.4.3.4 LAN 0/1 IPv4 Address 1 MIPAF1 — Offset 0x03:0x04

Same structure as LAN0 IPv4 Address 0.

### 6.4.3.5 LAN 0/1 IPv4 Address 2 MIPAF2 — Offset 0x05:0x06

Same structure as LAN0 IPv4 Address 0.

### 6.4.3.6 LAN 0/1 IPv4 Address 3 MIPAF3 — Offset 0x07:0x08

Same structure as LAN0 IPv4 Address 0.

### 6.4.3.7 LAN 0/1 Ethernet MAC Address 0 (LSB) MMAL0 — Offset 0x09

This word is loaded by Firmware to the 16 LS bits of the MMAL[0] register.

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | | | LAN 0/1 Ethernet MAC Address 0, Byte 1. | |
| 7:0 | | | LAN 0/1 Ethernet MAC Address 0, Byte 0. | |

### 6.4.3.8 LAN 0/1 Ethernet MAC Address 0 (Mid) MMAL0 — Offset 0x0A

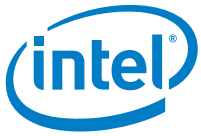
This word is loaded by Firmware to the 16 MS bits of the MMAL[0] register.

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | | | LAN 0/1 Ethernet MAC Address 0, Byte 3. | |
| 7:0 | | | LAN 0/1 Ethernet MAC Address 0, Byte 2. | |

### 6.4.3.9 LAN 0/1 Ethernet MAC Address 0 (MSB) MMAH0 — Offset 0x0B

This word is loaded by Firmware to the MMAH[0] register.

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | | | LAN 0/1 Ethernet MAC Address 0, Byte 5. | |
| 7:0 | | | LAN 0/1 Ethernet MAC Address 0, Byte 4. | |



### 6.4.3.10 LAN 0/1 Ethernet MAC Address 1 MMAL/H1 — Offset 0x0C:0x0E

Same structure as LAN0 Ethernet MAC Address 0. Loaded to MMAL[1], MMAH[1].

### 6.4.3.11 LAN 0/1 Ethernet MAC Address 2 MMAL/H2 — Offset 0x0F:0x11

Same structure as LAN0 Ethernet MAC Address 0. Loaded to MMAL[2], MMAH[2].

### 6.4.3.12 LAN 0/1 Ethernet MAC Address 3 MMAL/H3 — Offset 0x12:0x14

Same structure as LAN0 Ethernet MAC Address 0. Loaded to MMAL[3], MMAH[3].

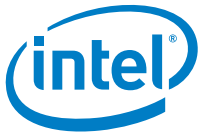
### 6.4.3.13 LAN 0/1 UDP Flexible Filter Ports 0:15 (MFUTP Registers) - Offset 0x15:0x24

| Offset | Bits | Description | Reserved |
|---|---|---|---|
| 0x15 | 15:0 | LAN UDP Flexible Filter Value Port0. | |
| 0x16 | 15:0 | LAN UDP Flexible Filter Value Port1. | |
| 0x17 | 15:0 | LAN UDP Flexible Filter Value Port2. | |
| 0x18 | 15:0 | LAN UDP Flexible Filter Value Port3. | |
| 0x19 | 15:0 | LAN UDP Flexible Filter Value Port4. | |
| 0x1A | 15:0 | LAN UDP Flexible Filter Value Port5. | |
| 0x1B | 15:0 | LAN UDP Flexible Filter Value Port6. | |
| 0x1C | 15:0 | LAN UDP Flexible Filter Value Port7. | |
| 0x1D | 15:0 | LAN UDP Flexible Filter Value Port8. | |
| 0x1E | 15:0 | LAN UDP Flexible Filter Value Port9. | |
| 0x1F | 15:0 | LAN UDP Flexible Filter Value Port10. | |
| 0x20 | 15:0 | LAN UDP Flexible Filter Value Port11. | |
| 0x21 | 15:0 | LAN UDP Flexible Filter Value Port12. | |
| 0x22 | 15:0 | LAN UDP Flexible Filter Value Port13. | |

| Offset | Bits | Description | Reserved |
|---|---|---|---|
| 0x23 | 15:0 | LAN UDP Flexible Filter Value Port14. | |
| 0x24 | 15:0 | LAN UDP Flexible Filter Value Port15. | |

### 6.4.3.14 LAN 0/1 VLAN Filter 0 — 7 (MAVTV Registers) - Offset 0x25:0x2C

| Offset | Bits | Description | Reserved |
|---|---|---|---|
| 0x25 | 15:12 | Reserved | |
| 0x25 | 11:0 | LAN 0/1 VLAN filter 0 value. | |
| 0x26 | 15:12 | Reserved | |
| 0x26 | 11:0 | LAN 0/1 VLAN filter 1 value. | |
| 0x27 | 15:12 | Reserved | |
| 0x27 | 11:0 | LAN 0/1 VLAN filter 2 value. | |
| 0x28 | 15:12 | Reserved | |
| 0x28 | 11:0 | LAN 0/1 VLAN filter 3 value. | |
| 0x29 | 15:12 | Reserved | |
| 0x29 | 11:0 | LAN 0/1 VLAN filter 4 value. | |
| 0x2A | 15:12 | Reserved | |
| 0x2A | 11:0 | LAN 0/1 VLAN filter 5 value. | |
| 0x2B | 15:12 | Reserved | |
| 0x2B | 11:0 | LAN 0/1 VLAN filter 6 value. | |
| 0x2C | 15:12 | Reserved | |
| 0x2C | 11:0 | LAN 0/1 VLAN filter 7 value. | |



### 6.4.3.15 LAN 0/1 Manageability Filters Valid (MFVAL LSB) — Offset 0x2D

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | VLAN | | Indicates if the VLAN filter registers (MAVTV) contain valid VLAN tags.<br>Bit 8 corresponds to filter 0, etc. | |
| 7:4 | Reserved | | | |
| 3:0 | MAC | | Indicates if the MAC unicast filter registers (MMAH, MMAL) contain valid Ethernet MAC Addresses.<br>Bit 0 corresponds to filter 0, etc. | |

### 6.4.3.16 LAN 0/1 Manageability Filters Valid (MFVAL MSB) — Offset 0x2E

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:12 | Reserved | | | |
| 11:8 | IPv6 | | Indicates if the IPv6 address filter registers (MIPAF) contain valid IPv6 addresses.<br>Bit 8 corresponds to address 0, etc. Bit 11 (filter 3) applies only when IPv4 address filters are not enabled (MANC.EN_IPv4_FILTER=0). | |
| 7:4 | Reserved | | Reserved. | |
| 3:0 | IPv4 | | Indicates if the IPv4 address filters (MIPAF) contain a valid IPv4 address.<br>These bits apply only when IPv4 address filters are enabled (MANC.EN_IPv4_FILTER=1). | |

### 6.4.3.17 LAN 0/1 MANC value LSB (LMANC LSB) — Offset 0x2F

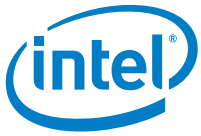
| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | Reserved | 0x0 | Reserved. | |

### 6.4.3.18 LAN 0/1 MANC Value MSB (LMANC MSB) — Offset 0x30

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:9 | Reserved | | Reserved. | |
| 8 | Enable IPv4 Address Filters | | This bit is loaded to the EN_IPv4_FILTER bit in the MANC register. | |
| 7 | Enable Xsum Filtering to MNG | | This bit is loaded to the EN_XSUM_FILTER bit in the MANC register. | |
| 6 | VLAN MNG Filtering | | This bit is loaded to the *Bypass VLAN* bit in the MANC register. | |
| 5 | Enable MNG Packets to Host Memory | | This bit is loaded to the EN_MNG2HOST bit in the MANC register. | |
| 4:0 | Reserved | | Reserved. | |

### 6.4.3.19 LAN 0/1 Receive Enable 1 (LRXEN1) — Offset 0x31

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | Receive Enable Byte 12 | | BMC SMBus slave address. | |
| 7 | Enable BMC Dedicated MAC | | | |
| 6 | Reserved | | Reserved. Must be set to 1b. | |
| 5:4 | Notification Method | | 00b = SMBus alert.<br>01b = Asynchronous notify.<br>10b = Direct receive.<br>11b = Reserved. | |
| 3 | Enable ARP Response | | | |
| 2 | Enable Status Reporting | | | |
| 1 | Enable Receive All | | | |
| 0 | Enable Receive TCO | | | |



### 6.4.3.20 LAN 0/1 Receive Enable 2 (LRXEN2) — Offset 0x32

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | Receive Enable byte 14 | 0x0 | Alert value. | |
| 7:0 | Receive Enable byte 13 | 0x0 | Interface data. | |

### 6.4.3.21 LAN 0/1 MANC2H Value (LMANC2H LSB) — Offset 0x33

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | Reserved | | | |
| 7:0 | Host Enable | | When set, indicates that packets routed by the manageability filters to the manageability block are also sent to the host.<br>Bit 0 corresponds to decision rule 0, etc. | |

### 6.4.3.22 LAN 0/1 MANC2H Value — LMANC2H MSB - Offset 0x34

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | Reserved | 0x0 | Reserved | |

### 6.4.3.23 Manageability Decision Filters — MDEF0 (1) - Offset 0x35

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | MDEF0_L | | Loaded to 16 LS bits of MDEF[0] register. | |

### 6.4.3.24 Manageability Decision Filters — MDEF0 (2) - Offset 0x36

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | MDEF0_M | | Loaded to 16 MS bits of MDEF[0] register. | |

### 6.4.3.25 Manageability Decision Filters — MDEF0 (3) - Offset 0x37

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | MDEFEXT0_L | | Loaded to 16 LS bits of MDEF_EXT[0] register. | |

### 6.4.3.26 Manageability Decision Filters — MDEF0 (4) - Offset 0x38

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | MDEF0EXT_M | | Loaded to 16 MS bits of MDEF_EXT[0] register. | |

### 6.4.3.27 Manageability Decision Filters — MDEF1-6 (1-4) - Offset 0x39:0x50

Same as words 0x035...0x38 for MDEF[1] and MDEF_EXT[1]...MDEF[6] and MDEF_EXT[6]

### 6.4.3.28 Manageability Ethertype Filter 0.1 — METF0 (1) - Offset 0x51

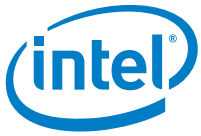
| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | METF0_L | | Loaded to 16 LS bits of METF[0] register. | |

### 6.4.3.29 Manageability Ethertype Filter 0.2 — METF0 (2) - Offset 0x52

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | METF0_M | | Loaded to 16 MS bits of METF[0] register (reserved bits in the METF registers should be set in the EEPROM to the register's default values). | |

### 6.4.3.30 Manageability Ethertype Filter 1...3 (1 and 2) — METF1:3 - Offset 0x53:0x58

Same as words 0x51 and 0x52 for METF[1]...METF[3] registers.



### 6.4.3.31 ARP Response IPv4 Address 0 (LSB) — Offset 0x59

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | | | ARP Response IPv4 Address 0, Byte 1 (firmware use). | |
| 7:0 | | | ARP Response IPv4 Address 0, Byte 0 (firmware use). | |

### 6.4.3.32 ARP Response IPv4 Address 0 (MSB) — Offset 0x5A

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | | | ARP Response IPv4 Address 0, Byte 3 (firmware use). | |
| 7:0 | | | ARP Response IPv4 Address 0, Byte 2 (firmware use). | |

### 6.4.3.33 LAN 0/1 IPv6 Address 0 (n=0...7) (MIPAF.IPV6ADDR0) — Offset 0x5B:0x62

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | | | Loaded to MIPAF registers IPV6ADDR0: Dword offset 'n'/2 to the lower 16 bits for even 'n' and upper 16 bits for odd 'n'. For 'n' = 0...7. | |

### 6.4.3.34 LAN 0/1 IPv6 Address 1 (MIPAF.IPV6ADDR1) — Offset 0x63:0x6A

Same structure as LAN 0/1 IPv6 Address 0.

### 6.4.3.35 LAN 0/1 IPv6 Address 2 (MIPAF) — Offset 0x6B-0x72

Same structure as LAN 0/1 IPv6 Address 0.

## 6.4.4 Sideband Configuration Module

This module is pointed to by global offset 0x06 of the manageability control table.

### 6.4.4.1 Section Header — Offset 0x0

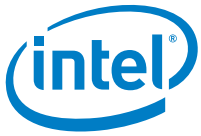
| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | Block CRC8 | 0x0 | | |
| 7:0 | Block Length | 0x0 | Section length in words. | |

### 6.4.4.2 SMBus Maximum Fragment Size — Offset 0x01

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | Max Fragment Size | 0x0 | SMBus Maximum Fragment Size (bytes) | |

### 6.4.4.3 SMBus Notification Timeout and Flags — Offset 0x02

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | SMBus Notification Timeout (ms) | | | |
| 7:6 | SMBus Connection Speed | | 00b = Standard SMBus connection.<br>01b = Reserved.<br>10b = Reserved.<br>11b = Reserved. | |
| 5 | SMBus Block Read Command | | 0b = Block read command is 0xC0.<br>1b = Block read command is 0xD0. | |
| 4 | SMBus Addressing Mode | | 0b = Single address mode.<br>1b = Dual address mode. | |
| 3 | Reserved | | Reserved. | |
| 2 | Disable SMBus ARP Functionality | | | |
| 1 | SMBus ARP PEC | | | |
| 0 | Reserved | | Reserved. | |



### 6.4.4.4 SMBus Slave Addresses — Offset 0x03

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:9 | SMBus 1 Slave Address | | Dual address mode only. | |
| 8 | Reserved | | Reserved. | |
| 7:1 | SMBus 0 Slave Address | | | |
| 0 | Reserved | | Reserved. | |

### 6.4.4.5 Fail-Over Register (Low Word) — Offset 0x04

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:12 | Gratuitous ARP Counter | | | |
| 11:10 | Reserved | | Reserved. | |
| 9 | Enable Teaming Fail-Over on DX | | | |
| 8 | Remove Promiscuous on DX | | | |
| 7 | Enable MAC Filtering | | | |
| 6 | Enable Repeated Gratuitous ARP | | | |
| 5 | Reserved | | Reserved. | |
| 4 | Enable Preferred Primary | | | |
| 3 | Preferred Primary Port | | | |
| 2 | Transmit Port | | | |
| 1:0 | Reserved | | Reserved. | |

### 6.4.4.6 Fail-Over Register (High Word) — Offset 0x05

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | Gratuitous ARP Transmission Interval (seconds) | | | |
| 7:0 | Link Down Fail-Over Time | | | |

#### 6.4.4.7 NC-SI Configuration Offset 0x06

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:14 | Reserved | | Reserved. | |
| 13 | Enable Channel Swap | 0b | 0b = Legacy mode — NCSI channel is aligned with physical port if LFS bit enabled.<br>1b = NCSI channels swapped with the physical port if LFS bit enabled. | |
| 12:11 | Reserved | | Reserved. | |
| 10 | Reserved | 0b | Must be 0. | |
| 9 | NC-SI HW Arbitration Enable | 0b | 0b = Not supported.Must be set to 0b.<br>1b = Supported. | |
| 8 | NC-SI HW-based Packet Copy Enable | 1b | 0b = Disable.<br>1b = Enable. | |
| 7:5 | Package ID | 000b | | |
| 4:0 | Reserved | 0x0 | Must be 0. | |

#### 6.4.4.8 Reserved Words — Offset 0x07 - 0x0C

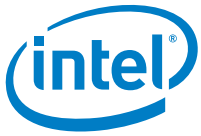
Reserved for future use.

### 6.4.5 Flexible TCO Filter Configuration Module

This module is pointed to by global offset 0x07 of the manageability control section.

#### 6.4.5.1 Section Header — Offset 0x0

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | Block CRC8 | 0x0 | | |
| 7:0 | Block Length | 0x0 | Section length in words. | |



### 6.4.5.2 Flexible Filter Length and Control — Offset 0x01

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | Flexible Filter Length (bytes) | | | |
| 7:5 | Reserved | | Reserved. | |
| 4 | Last Filter | | | |
| 3:2 | Filter Index (0-3) | | | |
| 1 | Apply Filter to LAN 1 | | | |
| 0 | Apply Filter to LAN 0 | | | |

### 6.4.5.3 Flexible Filter Enable Mask — Offset 0x02 - 0x09

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | Flexible Filter Enable Mask | | | |

### 6.4.5.4 Flexible Filter Data — Offset 0x0A - Block Length

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | Flexible Filter Data | | | |

**Note:** This section loads all of the flexible filters, The control + mask + filter data are repeatable as the number of filters. Section length in offset 0 is for all filters.

## 6.4.6 NC-SI Microcode Download Module

This module is pointed to by global offset 0x09 of the manageability control table.

### 6.4.6.1 Patch Data Size — Offset 0x0

### 6.4.6.2 Rx and Tx Code Size — Offset 0x1

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | Rx Code Length | 0x0 | Rx Code length in Dwords. | |
| 7:0 | Tx Code Length | 0x0 | Tx Code length in Dwords. | |

### 6.4.6.3 Download Data — Offset 0x2 - Data Size

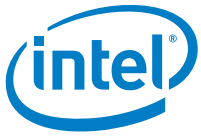
| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | Download Data | 0x0 | Download data. | |

## 6.4.7 NC-SI Configuration Module

This module is pointed to by global offset 0x0A of the manageability control table.

### 6.4.7.1 Section Header — Offset 0x0

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | Block CRC8 | 0x0 | | |
| 7:0 | Block Length | 0x0 | Section length in words. | |



### 6.4.7.2 Rx Mode Control1 (RR_CTRL[15:0]) — Offset 0x1

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:8 | Reserved | | Set to 0x0. | |
| 7:4 | Reserved | | Reserved. | |
| 3 | NC-SI Speed | | When set, the NC-SI MAC speed is 100 Mb/s.<br>When reset, NC-SI MAC speed is 10 Mb/s. | |
| 2 | Receive Without Leading Zeros | | If set, packets without leading zeros (J/K/ symbols) between TXEN assertion and TXD the first preamble byte can be received. | |
| 1 | Clear Rx Error | | Should be set when the Rx path is stuck because of an overflow condition. | |
| 0 | NC-SI Loopback Enable | | When set, enables NC-SI Tx-to-Rx loop. All data that is transmitted from NC-SI is returned to it.<br>No data is actually transmitted from NC-SI. | |

### 6.4.7.3 Rx Mode Control2 (RR_CTRL[31:16]) — Offset 0x2

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | Reserved | 0x0 | | |

### 6.4.7.4 Tx Mode Control1 (RT_CTRL[15:0]) — Offset 0x3

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:3 | Reserved | | Set to 0x0. | |
| 2 | Transmit With Leading Zeros | | When set, sends leading zeros (J/K/ symbols) from CRS_DV assertion to the start of preamble (PHY mode).<br>When de-asserted, does not send leading zeros (MAC mode). | |
| 1 | Clear Tx Error | | Should be set when Tx path is stuck because of an underflow condition.<br>Cleared by hardware when released. | |
| 0 | Enable Tx Pads | | When set, the NC-SI TX pads are driving. Otherwise, they are isolated. | |

### 6.4.7.5 Tx Mode Control2 (RT_CTRL[31:16]) — Offset 0x4

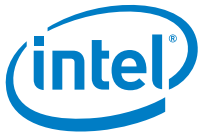
| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | Reserved | 0x0 | Set to 0x0. | |

### 6.4.7.6 MAC Tx Control Reg1 (TxCntrlReg1 (15:0]) — Offset 0x5

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:7 | Reserved | 0x0 | Set to 0x0. | |
| 6 | NC-SI_enable | | Enable the MAC internal NC-SI mode of operation (disables external NC-SI gasket). | |
| 5 | Two_part_deferral | | When set, performs the optional two part deferral. | |
| 4 | Append_fcs | | When set, computes and appends the FCS on Tx frames. | |
| 3 | Pad_enable | | Pad the TX frames, which are less than the minimum frame size. | |
| 2:1 | Reserved | | Reserved. | |
| 0 | Tx_ch_en | | Tx Channel Enable.<br>This bit can be used to enable the Tx path of the MAC. This bit is for debug only and the recommended way to enable the Tx path is via the RT_UCTL_CTRL.TX_enable bit. | |

### 6.4.7.7 MAC Tx Control Reg2 (TxCntrlReg1 (31:16]) — Offset 0x6

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:0 | Reserved | | Reserved. Should be set to 0x0. | |



### 6.4.7.8 NC-SI Settings — Offset 0x7

| Bits | Name | Default | Description | Reserved |
|---|---|---|---|---|
| 15:9 | Reserved | 0x0 | Set to 0x0. | |
| 8:7 | RMM Out Slew Rate | 01b | Configuration of the NC-SI out slew-rate control.<br>00b = Slowest<br>01b = Slow<br>10b = Fast<br>11b = Fastest | |
| 6:1 | RMM Out Buffer Strength | 011111b | Configuration of the NC-SI out buffer strength.<br>000001b = 2 mA<br>000011b = 4 mA<br>000111b = 6 mA<br>001111b = 8 mA<br>011111b = 10 mA<br>111111b = 12 mA | |
| 0 | Reserved | 0b | Set to 0b. | |

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

# 7.0 Inline Functions

## 7.1 Receive Functionality

Packet reception consists of:

- Recognizing the presence of a packet on the wire
- Performing address filtering
- DMA queue assignment
- Storing the packet in the receive data FIFO
- Transferring the data to assigned receive queues in host memory
- Updating the state of a receive descriptor.

A received packet goes through three stages of filtering as depicted in Figure 7-1. The Figure describes a switch-like structure that is used in virtualization mode to route packets between the network port (top of drawing) and one of many virtual ports (bottom of drawing), where each virtual port might be associated with a Virtual Machine (VM), an IOVM, a VMM, or the like. The three stages are:

1. First stage — Ensure that the packet should be received by the port. This is done by a set of L2 filters and is described in detail in Section 7.1.1.
2. Second stage — This stage is specific to virtualization environments and defines the virtual ports (called pools in this document) that are the targets for the Rx packet. A packet can be associated with any number of ports/pools and the selection process is described in Section 7.1.2.2.
3. Third stage — A receive packet that successfully passed the Rx filters is associated with one of many receive descriptor queues as described in this section.

In addition to the filtering rules, a packet must also meet the following criteria:

1. Normally, only good packets are received (packets with none of the following errors: Under Size Error, Over Size Error, Packet Error, Length Error and CRC Error). However, if the store-bad-packet bit is set (FCTRL.SBP), then bad packets that don't pass the filter function are stored in host memory. Packet errors are indicated by error bits in the receive descriptor (RDESC.ERRORS). It is possible to receive all packets, regardless of whether they are bad, by setting the promiscuous enables bit and the store-bad-packet bit.
2. Min Packet Size (Runt packets) — Rx packets, smaller than 21 bytes, cannot be posted to host memory regardless of save bad frame setting.

3. Max Packet Size — Any Rx packet posted from the MAC unit to the DMA unit cannot exceed
   15.5 KB.

**Note:** CRC errors before the SFD are ignored. All packets must have a valid SFD in order to be recognized by the device (even bad packets).

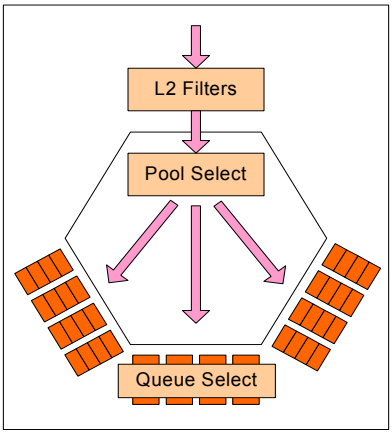



**Figure 7-1 Stages in Packet Filtering**

## 7.1.1 Packet Filtering

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

See Figure 7-2. Host filtering is done in three stages:

1. Packets are filtered by L2 filters (Ethernet MAC address, unicast/multicast/broadcast). See Section 7.1.1.1.
2. Packets are filtered by VLAN if a VLAN tag is present. See Section 7.1.1.2.
3. Packets are filtered by the manageability filters (port, IP, flex, other). See Section 10.3.

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

- The packet does not pass L2 filters, as described in Section 7.1.1.1.
- The packet does not pass VLAN filtering, as described in Section 7.1.1.2.
- The packet passes manageability filtering and the manageability filters determine that the packet should not pass to the host as well (see MANC2H register).

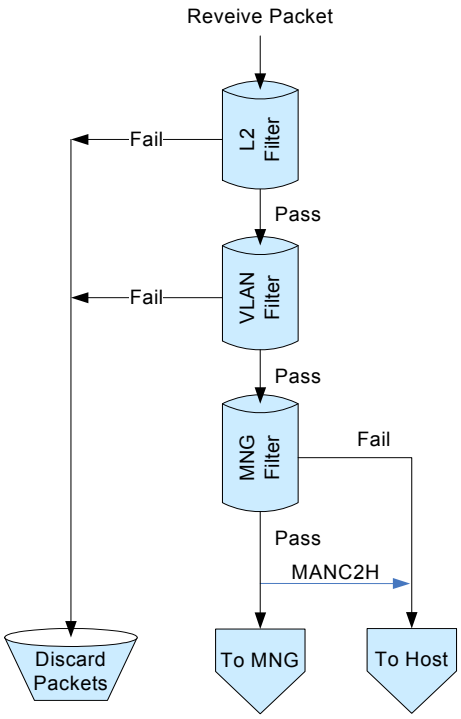



**Figure 7-2 Rx Filtering Flow Chart**

### 7.1.1.1 L2 Filtering

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

- Unicast packet filtering — Promiscuous unicast filtering is enabled (FCTRL.UPE=1b) or the packet passes unicast MAC filters (host or manageability).
- Multicast packet filtering — Promiscuous multicast filtering is enabled by either the host or manageability (FCTRL.MPE=1b or MANC.MCST_PASS_L2 =1b) or the packet matches one of the multicast filters.
- Broadcast packet filtering to host — Promiscuous multicast filtering is enabled (FCTRL.MPE=1b) or Broadcast Accept Mode is enabled (FCTRL.BAM = 1b).
- Broadcast packet filtering to manageability — Always enabled depending on the MDEF filters.

#### 7.1.1.1.1 Unicast Filter

The Ethernet MAC address is checked against the 128 host unicast addresses, 4 KB hash-based unicast address filters and four management unicast addresses (if enabled). The host unicast addresses are controlled by the host interface (the manageability controller must not change them). The other four addresses are dedicated to management functions and are only accessed by the manageability. The destination address of an incoming packet must exactly match one of the pre-configured host address filters or the manageability address filters. These addresses can be unicast or multicast. Those filters are configured through Receive Address Low (RAL), Receive Address High (RAH), Manageability Ethernet MAC Address Low (MMAL) and Manageability Ethernet MAC Address High (MMAH) registers. In addition, there are 4 KB unicast hash filters used for host defined by the PFUTA registers. The unicast hash filters are useful mainly for virtualization settings in those cases that more than 128 filters might be required.

Promiscuous Unicast — Receive all unicasts. Promiscuous unicast mode can be set/cleared only through the host interface (not by the manageability controller) and it is usually used when the LAN device is used as a sniffer.

#### 7.1.1.1.2 Multicast Filter (Partial)

The 12-bit portion of the incoming packet multicast address must exactly match the multicast filter address in order to pass multicast filter. These bits (out of 48 bits of the destination address) can be selected by the *MO* field in the MCSTCTRL register. The entries can be configured only by the host interface and cannot be controlled by the manageability controller.

Promiscuous Multicast — Receive all multicasts. Promiscuous multicast mode can be set/cleared only through the host interface (not by the manageability controller) and it is usually used when the LAN device is used as a sniffer.

### 7.1.1.2 VLAN Filtering

The 82599 provides exact VLAN filtering for host traffic and manageability traffic, as follows:

- Host VLAN filters are programmed by the VFTA[n] registers.
- Manageability VLAN filters are activated by the MDEF filters. One of eight VLAN tags are programmed by the MAVTV[7:0] registers while enabled by the MFVAL register.
- A VLAN match might relate to the *CFI* bit in the VLAN header. It is enabled for host filtering only by the VLNCTRL.CFIEN while the expected value is defined by the VLNCTRL.CFI.

If double VLAN is enabled (see Section 7.4.5), filtering is done on the second (internal) VLAN tag. All the filtering functions of the 82599 ignore the first (external) VLAN in this mode.

Receive packet that passes L2 layer filtering successfully is subjected to VLAN header filtering as illustrated in Figure 7-3:

1. If the packet does not have a VLAN header, it passes to the next filtering stage.
2. Else, if the packet is broadcast and MANC.RCV_TCO_EN bit is set, then it passes to the next filtering stage.

3. Else, if the packet passes a valid manageability VLAN filter and at least one VLAN_AND bit is set in the MDEF[n] registers, then it passes to the next filtering stage.
4. Else, if host VLAN filters are not enabled (VLNCTRL.VFE = 0b), the packet is forwarded to the next filtering stage.
5. Else, if the packet matches an enabled host VLAN filter and CFI checking (if enabled), the packet is forwarded to the next filtering stage.
6. Else, if manageability VLAN filtering is not required (MANC.Bypass_VLAN is set), the packet is forwarded to the next filtering stage as a potential candidate only for manageability.
7. Otherwise, the packet is dropped.

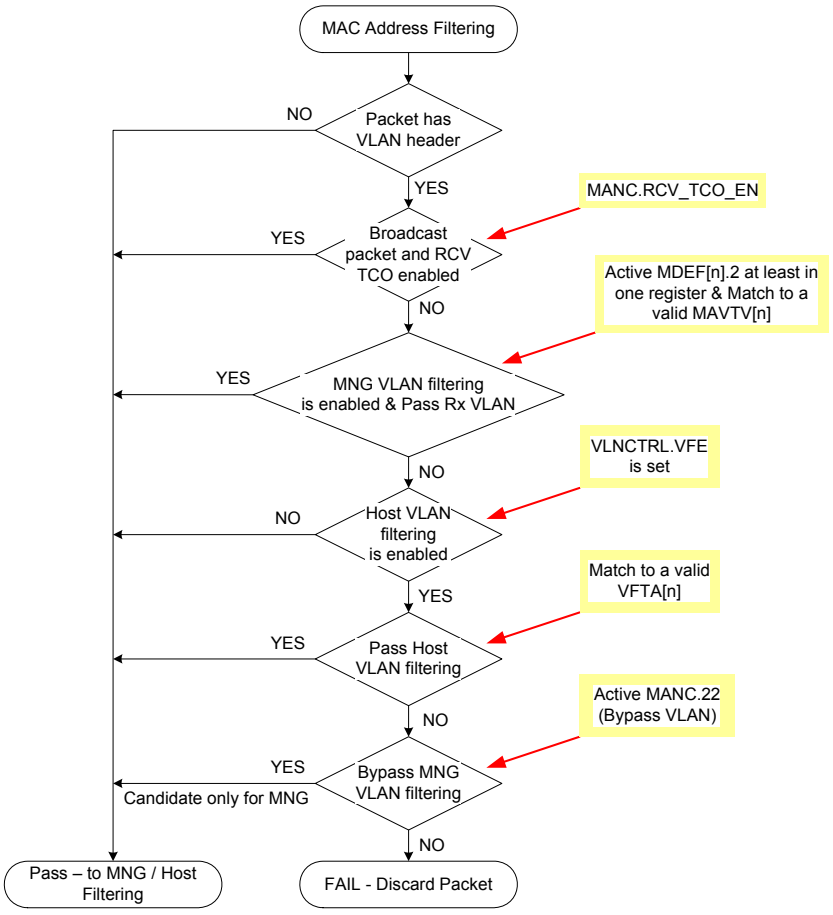



**Figure 7-3 VLAN Filtering**

#### 7.1.1.3 Manageability / Host Filtering

Packets that pass the MAC address filters and VLAN address filters described in the previous sections are subjected to MNG / Host filtering shown in Figure 7-4. The Manageability filters are described in Section 10.3. Packets that are not accepted for Manageability become automatically candidates for the host queue filters described in Section 7.1.2. Packets that pass the Manageability filters may still be posted to the host as well if they match the BMC to host filters defined by the MANC2H register.

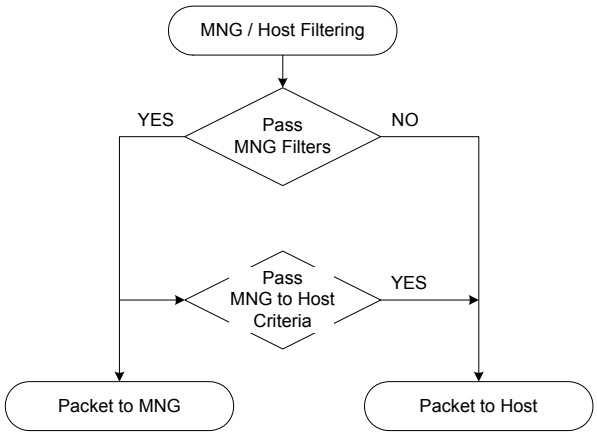
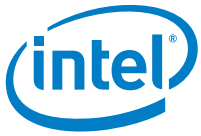



**Figure 7-4 Manageability / Host Filtering**

## 7.1.2 Rx Queues Assignment

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

- Virtualization — In a virtual environment, DMA resources are shared between more than one software entity (operating system and/or device driver). This is done by allocating receive descriptor queues to virtual partitions (VMM, IOVM, VMs, or VFs). Allocating queues to virtual partitions is done in sets, each with the same number of queues, called queue pools, or pools. Virtualization assigns to each received packet one or more pool indices. Packets are routed to a pool based on their pool index and other considerations such as DCB and RSS. See Section 7.1.2.2 for more on routing for virtualization.
- DCB — DCB provides QoS through priority queues, priority flow control, and congestion management. Packets are classified into one of several (up to eight) Traffic Classes (TCs). Each TC is associated with a single unique packet buffer. Packets that reside in a specific packet buffer are then routed to one of a set of Rx queues based on their TC value and other considerations such as RSS and virtualization. See Section 7.7 for details on DCB.
  - DCB is enabled via the *RT Enable* bit



- Receive Side Scaling (RSS) — RSS distributes packet processing between several processor cores by assigning packets into different descriptor queues. RSS assigns to each received packet an RSS index. Packets are routed to one of a set of Rx queues based on their RSS index and other considerations such as DCB and virtualization. See Section 7.1.2.8 for details.
- L2 Ethertype Filters — These filters identify packets by their L2 Ethertype and assigns them to receive queues. Examples of possible uses are LLDP packets, and 802.1X packets. See Section 7.1.2.3 for details. The 82599 incorporates eight Ethertype filters.
- FCoE Redirection Table — FCoE packets that match the L2 filters might be directed to a single legacy Rx queue or multiple queues to ease multi-core processing. See Section 7.1.2.4 for details. See also Section 7.13.3.3 for Large FC receive and direct data placement.
- L3/L4 5-tuple Filters — These filters identify specific L3/L4 flows or sets of L3/L4 flows. Each filter consists of a 5-tuple (protocol, source and destination IP addresses, source and destination TCP/UDP port) and routes packets into one of the Rx queues. The 82599 incorporates 128 such filters. See Section 7.1.2.5 for details.
- Flow Director Filters — These filters are an expansion of the L3/L4 5-tuple filters that provides up to additional 32 K filters. See Section 7.1.2.7 for details.
- TCP SYN Filters — might route TCP packets with their SYN flag set into a separate queue. SYN packets are often used in SYN attacks to load the system with numerous requests for new connections. By filtering such packets to a separate queue, security software can monitor and act on SYN attacks. See Section 7.1.2.6 for details.

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

- Queue by L2 Ethertype filters (if match)
- Queue by FCoE redirection table (relevant for FCoE packets)
- If SYNQF.SYNQFP is zero, then
  - Queue by L3/L4 5-tuple filters (if match)
  - Queue by SYN filter (if match)
- If SYNQF.SYNQFP is one, then
  - Queue by SYN filter (if match)
  - Queue by L3/L4 5-tuple filters (if match)
- Queue by flow director filters
- Define a pool (in case of virtualization)
- Queue by DCB and/or RSS as described in Section 7.1.2.1 and Section 7.1.2.2.

### 7.1.2.1 Queuing in a Non-virtualized Environment

Table 7-1 lists the queuing schemes. Table 7-2 lists the queue indexing. Selecting a scheme is done via the *Multiple Receive Queues Enable* field in the MRQ register.

**Table 7-1 Rx Queuing Schemes Supported (No Virtualization)**

| DCB | RSS | DCB / RSS Queues | Special Filters[1] |
|---|---|---|---|
| No | No | 1 queue<br>Rx queue 0 | Supported |
| No | Yes | 16 RSS queues | Supported |
| Yes | No | 8 TCs x 1 queue<br>4 TCs x 1 queue<br>RSS assign Rx queue 0 of each TC | Supported |
| Yes | Yes | 8 TCs x 16 RSS<br>4 TCs x 16 RSS | Supported |

1. Special filters include: L2 filters, FCoE redirection, SYN filter and L3/L4 5-tuple filters. When possible, it is recommended to assign Rx queues not used by DCB/RSS queues.

**Table 7-2 Queue Indexing Illustration in Non-virtualization Mode**

| Queue Index bits | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RSS | 0 | 0 | 0 | RSS | | | |
| DCB(4) + RSS | TC | | 0 | RSS | | | |
| DCB(8) + RSS | TC | | | RSS | | | |

A received packet is assigned to a queue according to the ordering shown in Figure 7-5):

- DCB and RSS filters and FCoE redirection — Packets that do not meet any of the filtering conditions described in Section 7.1.2 are assigned to one of 128 queues as listed in Table 7-1. The following modes are supported:
  - — No DCB, No RSS and No FCoE redirection — Queue 0 is used for all packets.
  - — RSS only — A set of 16 queues is allocated for RSS. The queue is identified through the RSS index. Note that it is possible to use a subset of these queues.
  - — DCB only — A single queue is allocated per TC to a total of eight queues (if the number of TCs is eight), or to a total of four queues (if the number of TCs is four). The queue is identified through the TC index.
  - — DCB with RSS — A packet is assigned to one of 128 queues (8 TCs x 16 RSS) or one of 64 queues (4 TCs x 16 RSS) through the DCB traffic class of the packet and the RSS index. The TC index is used as the MS bit of the Rx queue index, and the LSBits are defined by the RSS index.
  - — FCoE redirection — Up to eight queues can be allocated for FCoE traffic by the FCoE redirection table defined by FCRETA[n] registers.

When operating in conjunction with DCB, the number of RSS queues can vary per DCB TC. Each TC can be configured to a different number of RSS queues (0/1/2/4 queues). The output of the RSS redirection table is masked accordingly to generate an RSS index of the right width. When configured to less than the maximum number of queues, the respective MS bits of the RSS index are set to zero. The number of RSS queues per TC is configured in the RQTC register.

- Example — Assume a 4 TCs x 16 RSS configuration and that the number of RSS queues for TC=3 is set to 4. The queue numbers for TC=3 are 32, 33, 34, and 35 (decimal).

Figure 7-5 depicts an example of allocation of Rx queues by the various queue filters previously described for the following case:

- DCB and RSS enabled to 4 TCs x 16 RSS queues
- RSS is used at various width per TC
- SYN filter allocated
- Ethertype filters are used
- 5-tuple filters are used

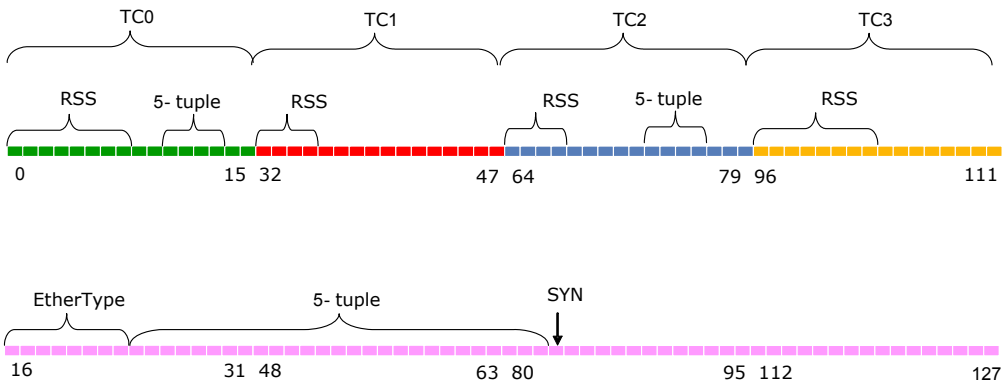



**Figure 7-5 Example of Rx Queue Allocation (Non-Virtualized)**

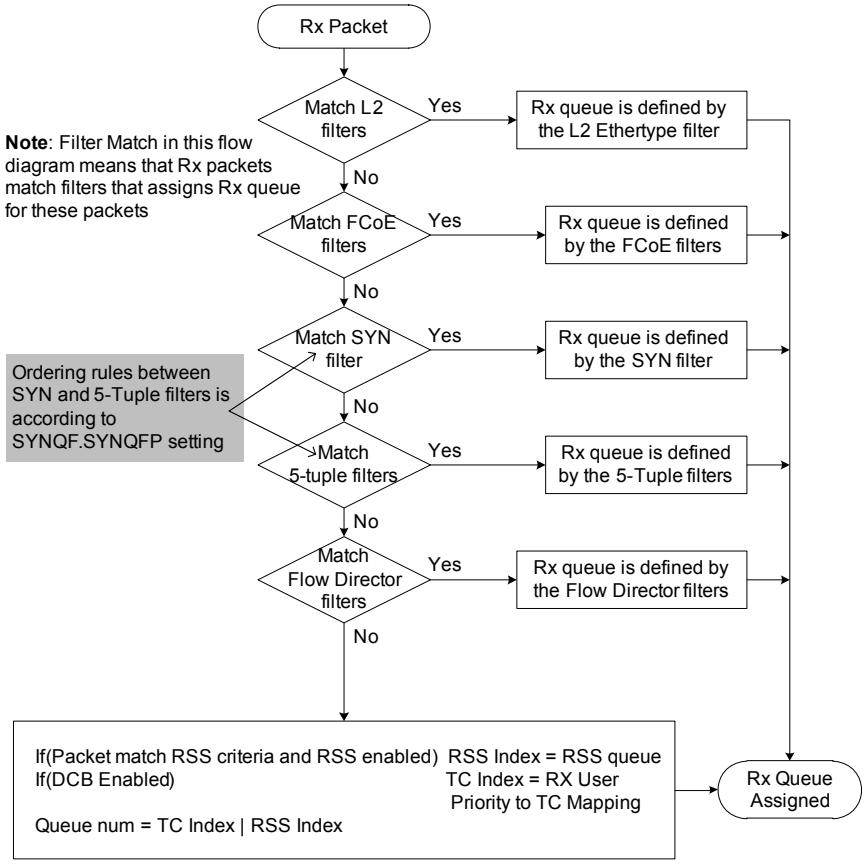


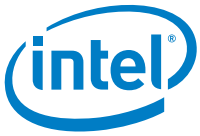

**Figure 7-6 Rx Queuing Flow (Non-Virtualized)**

## 7.1.2.2 Queuing in a Virtualized Environment

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

Incoming packets are associated with pools based on their L2 characteristics as described in Section 7.10.3. This section describes the following stage, where an Rx queue is assigned to each replication of the Rx packet as determined by its pools association.

Table 7-3 lists the queuing schemes supported with virtualization. Table 7-4 lists the queue indexing.



### Table 7-3 Rx Queuing Schemes Supported with Virtualization

| DCB | RSS | DCB / RSS Queues | Special Filters[1] |
|---|---|---|---|
| No | No | 16 pools x 1 queue<br>32 pools x 1 queue<br>64 pools x 1 queue<br>- Rx queue 0 of each pool | Supported |
| No | Yes[2] | 32 pools x 4 RSS<br>64 pools x 2 RSS | Supported |
| Yes | No | 16 pools x 8 TCs<br>32 pools x 4 TCs | Supported |
| Yes | Yes | Not supported | |

1. Special filters include: L2 filters, FCoE redirection, SYN filter and L3/L4 5-tuble filters. When possible, it is recommended to assign Rx queues not used by DCB/RSS queues.
2. RSS might not be useful for IOV mode since the 82599 supports a single RSS table for the entire device.

### Table 7-4 Queue Indexing Illustration in Virtualization Mode

| Queue Index bits | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| VT(64) + RSS | VF Index | | | | | | RSS |
| VT(32) + RSS | VF Index | | | | | RSS | |
| VT(16) + RSS | Not Supported | | | | | | |
| VT(32) + DCB(4) | VF Index | | | | | TC | |
| VT(16) + DCB(8) | VF Index | | | | TC | | |

Selecting a scheme is done in the following manner:

- Non-IOV mode
  - Selected via the *Multiple Receive Queues Enable* field in the MRQC register.
- IOV mode
  - Determine the number of pools: the number must support the value configured by the operating system in the PCIe NumVFs field (see Section 9.4.4.5). Therefore, the number of pools is min of {16, 32, 64} that is still >= NumVFs.
  - Determine DCB mode via the *RT Enable* CSR field.
  - Note that RSS is not supported in IOV mode since there is only a single RSS hash function in the hardware.

A received packet is assigned to an absolute queue index according to the ordering shown in Figure 7-7). It is software responsibility to define a queue that belongs to the matched pool.

- DCB and RSS filters — The supported modes are listed in Table 7-3 and detailed as follows. The associated queue indexes are listed in Table 7-4.
  - No DCB, No RSS — A single queue is allocated per pool with either 32 or 64 pools enabled. In 64 pools setting, queues '2xN'...'2xN+1' are allocated to pool 'N'; In 32 pools setting, queues '4xN'...'4xN+3' are allocated to pool 'N'.
  - RSS only — All 128 queues are allocated to pools. Several configurations are supported: 32 pools with 4 RSS queues each, and 64 pools with 2 queues each. Note that it is possible to use a subset of the RSS queues in each pool. The LSBits of the queue indexes are defined by the RSS index, and the pool index is used as the MS bits.
  - DCB only — All 128 queues are allocated to pools. Several configurations are supported: 16 pools with 8 TCs each, or 32 pools with 4 TCs each. The LSBits of the queue indexes are defined by the TC index, and the pool index is used as the MS bits.

When operating in conjunction with RSS, the number of RSS queues can vary per pool as defined by the PSRTYPE[n].RQPL. Each pool can be configured to a different number of RSS queues (0/1/2/4 queues) up to the maximum possible queues in the selected mode of operation. The output of the RSS redirection table is masked accordingly to generate an RSS index of the right width. When configured to less than the maximum number of queues, the respective MS bits of the RSS index are set to zero.

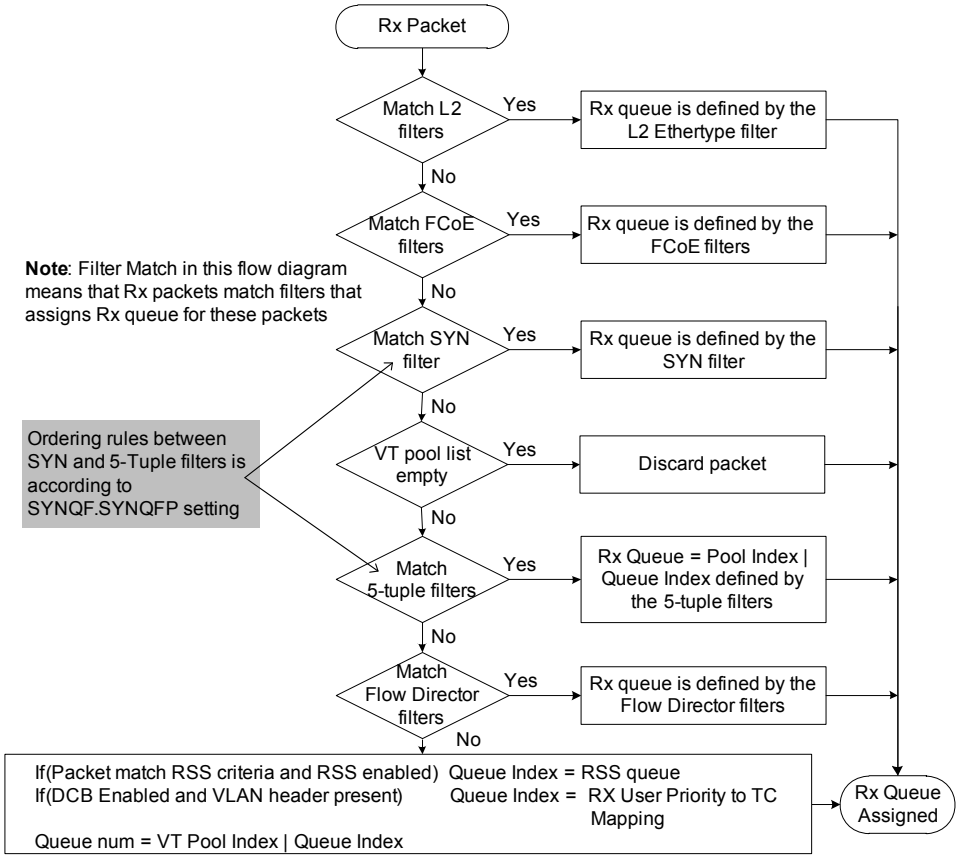



**Figure 7-7 Rx Queuing Flow (Virtualization Case)**

## 7.1.2.3 L2 Ethertype Filters

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

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

The 82599 incorporates eight Ethertype filters defined by a set of two registers per filter: ETQF[n] and ETQS[n].

The L2 packet type is defined by comparing the *Ether-Type* field in the Rx packet with the ETQF[n].EType (regardless of the pool and UP matching). The *Packet Type* field in the Rx descriptor captures the filter number that matched with the L2 Ethertype. See Section 7.1.6.2 for a description of the *Packet Type* field.

The following flow is used by the Ethertype filters:

1. If the *Filter Enable* bit is cleared, the filter is disabled and the following steps are ignored.
2. Receive packet matches any ETQF filters if the *EtherType* field in the packet matches the *EType* field of the filter. Note that the following steps are ignored if the packet does not match the ETQF filters.
3. Packets that match any ETQF filters is a candidate for the host. If the packet also matches the manageability filters, it is directed to the host as well regardless of the MANC2H register setting.
4. If the *FCoE* field is set, the packet is identified as an FCoE packet.
5. If the *1588 Time Stamp* field is set, the packet is identified as an IEEE 1588 packet.
6. If the *Queue Enable* bit is cleared, the filter completed its action on the packet. Else, the filter is also used for queuing purposes as described in the sections that follow.
7. If the *Pool Enable* field is set, the *Pool* field of the filter determines the target pool for the packet. The packet can still be mirrored to other pools as described in Section 7.10.3. See the sections that follow for more details on the use of the *Pool* field.
8. The *RX Queue* field determines the destination queue for the packet. In case of a mirrored packet, only the copy of the packet that is targeted to the pool defined by the *Pool* field in the ETQF register is routed according to the *Rx Queue* field.

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

- The *Filter Enable* bit enables identification of Rx packets by Ethertype according to this filter. If this bit is cleared, the filter is ignored.
- The *EType* field contains the 16-bit Ethertype compared against all L2 type fields in the Rx packet.
- The FCoE bit indicates that the Ethertype defined in the *EType* field is an FCoE EType. Packets that match this filter are identified as FCoE packets.
- The *1588 Time Stamp* bit indicates that the Ethertype defined in the *EType* field is identified as IEEE 1588 EType. Packets that match this filter are time stamped according to the IEEE 1588 specification.
- The *Pool* field defines the target pool for a packet that matches the filter.
  - It applies only in virtualization modes. The pool index is meaningful only if the *Pool Enable* bit is set.
  - If the *Pool Enable* bit is set then the *Queue Enable* bit in the ETQS register must be set as well. In this case, the *Rx Queue* field in the ETQS must be part of the pool number defined in the ETQF.

Setting the ETQS[n] registers is described as follows:

- The *Queue Enable* bit enables routing of the Rx packet that match the filter to Rx queue as defined by the *Rx Queue* field.

- The *Rx Queue* field contains the destination queue (one of 128 queues) for the packet.
- The *Low Latency Interrupt* bit enables LL interrupt assertion by the Rx packet that matches this filter.

Special considerations for virtualization modes:

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

### 7.1.2.4 FCoE Redirection Table

The FCoE redirection table is a mechanism to distribute received FCoE packets into several descriptor queues. Software might assign each queue to a different processor, sharing the load of packet processing among multiple processors. The FCoE redirection table assigns Rx queues to packets that are identified as FCoE in the ETQF[n] registers but not assigned to queues in the ETQS[n] registers.

Figure 7-8 illustrates the computing of the assigned Rx queue index by the FCoE redirection table.

- The Rx packet is parsed extracting the OX_ID or the RX_ID depending on the *Exchange Context* in the *F_CTL* field in the FC header. At zero the RX_ID is used; at one the OX_ID is used.
- The three LSBits of the OX_ID or RX_ID are used as an address to the redirection table (FCRETA[n] register index).
- The FCoE redirection table is enabled by the FCRECTL.ENA bit. If enabled, the content of the selected FCRETA[n] register is the assigned Rx queue index.

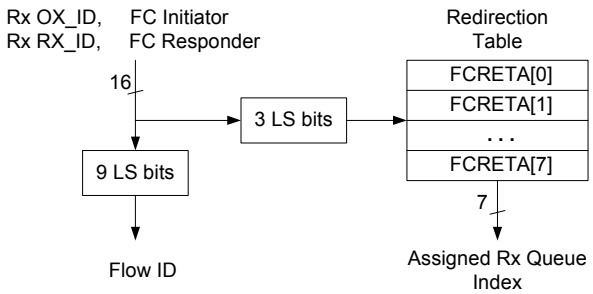



**Figure 7-8 FCoE Redirection Table**

### 7.1.2.5 L3/L4 5-tuple Filters

These filters identify specific L3/L4 flows or sets of L3/L4 flows and routes them to dedicated queues. Each filter consists of a 5-tuple (protocol, source and destination IP addresses, source and destination TCP/UDP/SCTP port) and routes packets into one of the Rx queues.

The 82599 incorporates 128 such filters, used also to initiate Low Latency Interrupts (LLI). The specific filtering rules are:

- Filtering rules for IPv6 packets:
  - If a filter defines at least one of the IP source and destination addresses, then an IPv6 packet always misses such a filter.
  - If a filter masks both the IP source and destination addresses, then an IPv6 packet is compared against the remaining fields of the filter.
- Packets with tunneling (any combination of IPv4 and IPv6) miss the 5-tuple filters.
- Fragmented packets miss the 5-tuple filters.

In a virtualized environment, any 5-tuple filters is associated with a unique pool:

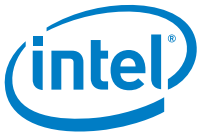
- The packet must first match the L2 filters described in Section 7.10.3.3 and Section 7.10.3.4. The outcome of the L2 filters is a set of pool values associated with the packet. The *Pool* field of the 5-tuple filter is then compared against the set of pools to which the packet is steered. A filter match is considered only to the indicated pool in the filter.

If a packet matches more than one 5-tuple filter, then:

- For queuing decision — The priority field identifies the winning filter and therefore the destination queue.
- For queuing decision — If the packet matches multiple filters with the same priority, the filters with the lower index takes affect.
- For Low Latency Interrupt (LLI) — An LLI is issued if one or more of the matching filters are set for LLI.

The 5-tuple filters are configured via the FTQF, SDPQF, L34TIMIR, DAQF, and SAQF registers, as follows (described by filter):

- Protocol — Identifies the IP protocol, part of the 5-tuple. Enabled by a bit in the mask field. Supported protocol fields are TCP, UDP, SCTP or other (neither TCP nor UDP nor SCTP).
- Source address — Identifies the IP source address, part of the 5-tuple. Enabled by a bit in the mask field. Only IPv4 addresses are supported.
- Destination address — Identifies the IP destination address, part of the 5-tuple. Enabled by a bit in the mask field. Only IPv4 addresses are supported.
- Source port — Identifies the TCP/UDP/SCTP source port, part of the 5-tuple. Enabled by a bit in the mask field.
- Destination port — Identifies the TCP/UDP/SCTP destination port, part of the 5-tuple queue filters. Enabled by a bit in the mask field.
- Queue Enable — Enables the packets routing to queues based on the Rx Queue index of the filter.



- Rx Queue — Determines the Rx queue for packets that match this filter.
- Pool — Applies only in the virtualized case (while *Pool Mask* bit = 0b). This field must match one of the pools enabled for this packet in the L2 filters.
  - In non-virtualized case the *Pool Mask* bit must be set to 1b.
  - In the virtualized case, the pool must be defined (*Pool Mask* = 0b and *Pool* = valid index). The *Rx Queue* field defines the absolute queue index. In case of mirroring or replication, only the copy of the packet destined to the matched pool in the filter is routed according to the *Rx Queue* field.
- Mask — A 5-bit field that masks each of the fields in the 5-tuple (L4 protocol, IP addresses, TCP/UDP ports). The filter is a logical AND of the non-masked 5-tuple fields. If all 5-tuple fields are masked, the filter is not used for queue routing.
- Priority — A 3-bit field that defines one of seven priority levels (001b-111b), with 111b as the highest priority. Software must insure that a packet never matches two or more filters with the same priority value.

**Note:** There are 128 different 5-tuple filter configuration registers sets, with indexes [0] to [127]. The mapping to a specific Rx queue is done by the *Rx Queue* field in the L34TIMIR register, and not by the index of the register set.

### 7.1.2.6 SYN Packet Filters

The 82599 might route TCP packets whose SYN flag is set into a separate queue. SYN packets are used in SYN attacks to load the system with numerous requests for new connections. By filtering such packets to a separate queue, security software can monitor and act on SYN attacks.

The following rules apply:

- A single SYN filter is provided.

The SYN filter is configured via the SYNQF register as follows:

- The *Queue Enable* bit enables SYN filtering capability.
- The *Rx Queue* field contains the destination queue for the packet (one of 128 queues). In case of mirroring (in virtualization mode), only the original copy of the packet is routed according to this filter.

### 7.1.2.7 Flow Director Filters

The flow director filters identify specific flows or sets of flows and routes them to specific queues. The flow director filters are programmed by FDIRCTRL and all other FDIR registers. The 82599 shares the Rx packet buffer for the storage of these filters. Basic rules for the flow director filters are:

- IP packets are candidates for the flow director filters (meaning non-IP packets miss all filters)
- Packets with tunneling (any combination of IPv4 and IPv6) miss all filters
- Fragmented packets miss all filters

- In VT mode, the *Pool* field in FDIRCMD must be valid. If the packet is replicated, only the copy that goes to the pool that matches the *Pool* field is impacted by the filter.

The flow director filters cover the following fields:

- VLAN header
- Source IP and destination IP addresses
- Source port and destination port numbers (for UDP and TCP packets)
- IPv4 / IPv6 and UDP / TCP or SCTP protocol match
- Flexible 2-byte tuple anywhere in the first 64 bytes of the packet
- Target pool number (relevant only for VT mode)

**Note:** IPv6 extended headers are parsed by the 82599, 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 82599 support two types of filtering modes (static setting by the FDIRCTRL.Perfect-Match bit):

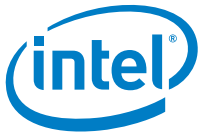
- Perfect match filters — The hardware checks a match between the masked fields of the received packets and the programmed filters. Masked fields should be programmed as zeros in the filter context. The 82599 support up to 8 K - 2 perfect match filters.
- Signature filters — The hardware checks a match between a hash-based signature of the masked fields of the received packet. The 82599 supports up to 32 K - 2 signature filters.
- Notation — The *Perfect Match* fields and *Signature* field are denoted as *Flow ID* fields.

The 82599 supports masking / range for the previously described fields. These masks are defined globally for all filters in the FDIR…M register.

- The following fields can be masked per bit enabling power of two ranges up to complete enable / disable of the fields: IPv4 addresses and L4 port numbers.
- The following fields can be masked per byte enabling lower granularity ranges up to complete enable / disable of the fields: IPv6 addresses. Note that in perfect match filters the destination IPv6 address can only be compared as a whole (with no range support) to the IP6AT.
- The following fields can be either enabled or disabled completely for the match functionality: VLAN ID tag; VLAN Priority + CFI bit; Flexible 2-byte tuple and target pool. Target pool can be enabled by software only when VT is enabled as well.

Flow director filters have the following functionality in virtualization mode:

- Flow director filters are programmed by the registers in the PF described in Section 7.1.2.7.11 and Section 7.1.2.7.12.



#### 7.1.2.7.1 Flow Director Filters Actions

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

- Drop packet or pass to host as defined by the *Drop* bit.
  - Matched packets to a flow director filter is directed to the assigned Rx queue only if the packet does not match the L2 filters for queue assignment nor the SYN filter for queue assignment nor the 5-tuple filters for queue assignment.
  - Packets that match pass filters are directed to the Rx queue defined in the filter context as programmed by the FDIRCMD.Rx-Queue. In a non-VT setting, the *Rx Queue* field defines the absolute queue number. In VT setting, the *Rx Queue* field defines the relative queue number within the pool.
  - Packets that match drop filters are directed to the Rx queue defined per all filters in the FDIRCTRL.DROP-Queue. The 82599 drops these packets if software does not enable the specific Rx queue.
- Trigger low latency interrupt is enabled by the *INT* bit.
  - Matched packets to a flow director filter can generate LLI if the packet does not match the L2 filters for queue assignment nor the SYN filter for queue assignment nor the 5-tuple filters for queue assignment.

#### 7.1.2.7.2 Flow Director Filters Status Reporting

Shared status indications for all packets:

- The 82599 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. Note that  it is required to set the FDIRCTRL.Report-Status bit to enable the FLM status indication as well as any Flow Director error indications in the receive descriptor.
  - 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.
  - For packets that match a flow director filter, the *Flow Director Filter ID* field can be used by software to identify the flow of the Rx packet.

Too long linked list exception (linked list and too long terms are illustrated in Figure 7-9):

- The maximum recommended linked list length is programmed in the FDIRCTRL.Max-Length field
- The length exception is reported in the *FDIRErr* field in the Rx descriptor

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

Collision exception:

- Packets that matches a collided filter report this exception in the *FDIRErr* field in the Rx descriptor.
- Collision events for signature-based filters should be rare. Still it might happen because multiple flows can have the same hash and signature values. Software might leave the setting as is while the collided flows are handled according to the actions of the first programmed flow. On the other hand, software might choose to resolve the collision by programming the collided flows in the 5-tuples filters. Only one flow (out of the collided ones) might remain in the flow director filters. In order to clear the collision indication in the programmed filter, software should remove the filter and then re-program it once again.
- Collision events for a perfect match filter should never happen. A collision error might indicate a programming fault that software might decide to fix.

#### 7.1.2.7.3 Flow Director Filters Block Diagram

The following figure shows a block diagram of the flow director filters. Received flows are identified to buckets by a hash function on the relevant tuples as defined by the FDIR...M registers. Each bucket is organized in a linked list indicated by the hash lookup table. Buckets can have a variable length while the last filter in each bucket is indicated as a last. There is no upper limit for a linked list length during programming; however, a received packet that matches a filter that exceeds the FDIRCTRL.Max-Length are reported to software (see Section 7.1.2.7.5).

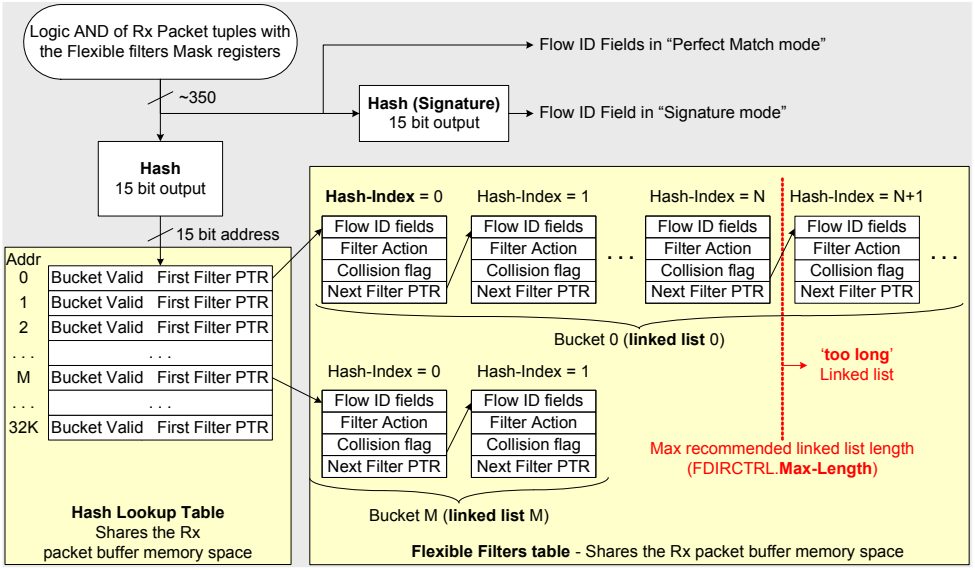



**Figure 7-9 Flow Director Filters Block Diagram**

## 7.1.2.7.4 Rx Packet Buffer Allocation

Flow director filters can consume zero space (when disabled) up to ~256 KB of memory. As shown in Figure 7-9, flow director filters share the same memory with the Rx packet buffer. Setting the *PBALLOC* field in the FDIRCTRL register, the software might enable and allocate memory for the flow director filters. The memory allocated to reception is the remaining part of the Rx packet buffer.

**Table 7-5 Rx Packet Buffer Allocation**

| PBALLOC (2) | Effective Rx Packet Buffer Size (see following note) | Flow Director Filters Memory | Supported Flow Director Filters | | | |
|---|---|---|---|---|---|---|
| | | | Signature | | Perfect Match | |
| | | | Filters | Bucket Hash | Filters | Bucket Hash |
| 00<br>Flow Director is disabled | 512 KB | 0 | 0 | n/a | 0 | n/a |
| 01 | 448 KB | 64 KB | 8 K - 2 | 13 bits | 2 K - 2 | 11 bits |
| 10 | 384 KB | 128 KB | 16 K - 2 | 14 bits | 4 K - 2 | 12 bits |
| 11 | 256 KB | 256 KB | 32 K - 2 | 15 bits | 8 K - 2 | 13 bits |

**Note:** It is the user responsibility to ensure that sufficient buffer space is left for reception. The required buffer space for reception is a function of the number of traffic classes, flow control threshold values and remaining buffer space in between the thresholds. If flow director is enabled (such as PBALLOC > 0), software should set the RXPBSIZE[n] registers according to the total remaining part of the Rx packet buffer for reception.

For example, if PBALLOC equals one and there is only one buffer in the system, software should set RXPBSIZE[0] to 0x70000 (448 K) and RXPBSIZE[1...7] to zero. Another example is if PBALLOC equals two and DCB is enabled with four traffic classes then software might set RXPBSIZE[0...3] to 0x18000 (96 K) and RXPBSIZE[4...7] to zero.

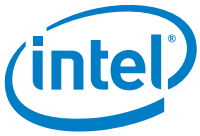
Refer to Section 3.7.7.3.2 through Section 3.7.7.3.5 for recommended setting of the Rx packet buffer sizes and flow control thresholds.

#### 7.1.2.7.5 Flow Director Filtering Reception Flow

- Rx packet is digested by the filter unit which parse the packet extracting the relevant tuples for the filtering functionality.
- The 82599 calculates a 15-bit hash value out of the masked tuples (logic mask of the tuples and the relevant mask registers) using the hash function described in Section 7.1.2.7.15.
- The address in the hash lookup table points to the selected linked list of the flow director filters.
- The 82599 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 82599 checks the linked list until it reaches the last filter in the linked list or until a matched filter is found.
- Case 1: matched filter is found:
  - Increment the FDIRMATCH statistic counter.
  - Process the filter's actions (queue assignment and LLI) according to queue assignment priority. Meaning, the actions defined in this filter takes place only if the packet did not match any L2 filter or SYN filter or 5-tuple filter that assigns an Rx queue to the packet.
  - Rx queue assignment according to the filter context takes place if *Queue-EN* is set. In VT mode, the Rx queue in the filter context defines a relative queue within the pool.
  - LLI is generated if the *INT* bit is set in the filter context.
  - Post the packet to host including the flow director filter match indications as described in Section 7.1.2.7.2.
- Case 2: matched filter is not found:
  - Increment the FDIRMISS statistic counter.
  - Post the packet to host including the flow director filter miss indications as described in Section 7.1.2.7.2.

#### 7.1.2.7.6 Add Filter Flow

The software programs the filters parameters in the registers described in Section 7.1.2.7.12 and Section 7.1.2.7.13 while keeping the FDIRCMD.Filter-Update bit inactive. As a result, the 82599 checks the bucket valid indication in the hash lookup table (that matches the FDIRHASH.Hash) for the presence of an existing linked list.



Following are the two programming flows that handle a presence of an existing linked list or creating a new linked list.

- Case 1: Add a filter to existing linked list:

  The 82599 checks the linked list until it reaches the last filter in the list or until a matched filter is found. Handle the filter programming in one of the following cases:

  — Matched filter is found (equal flow ID) with the same action parameters — The programming is discarded silently. This is a successful case since the programmed flow is treated as requested.

  — Matched filter is found (equal flow ID) with different action parameters — The 82599 keeps the old setting of the filter while setting the *Collision* flag in the filter context and increments the COLL counter in the FDIRFREE register (see Section 7.1.2.7.2 for software handling of collision during packet reception).

  — Matched filter is found (equal flow ID) with different action parameters and the *Collision* flag is already set — The programming is discarded silently. Software gets the same indications as the previous case.

  — Matched filter is not found (no collision) — The 82599 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 82599 looks for an empty space in the flow director filters table:

  — Handle no empty space the same as in Case 1.

  — Good programming case: Add the new filter while indicating it as the last one in the linked list. Then, program the hash lookup table entry by setting the *Valid* flag and the *First Filter PTR* pointing to the new programmed filter.

Additional successful add flow indications:

- Increment the ADD statistic counter in the FDIRUSTAT register.
- Reduce the FREE counter in the FDIRFREE register and then indicate the number of free filters. If the FREE counter crosses the full-thresh value in the FDIRCTRL register, then assert the flow director filter interrupt. Following this interrupt software should read the FDIRFSTAT register and FDIRFREE.FREE field, for checking the interrupt cause.
- Compare the length of the new linked list with MAXLEN in the FDIRLEN register. If the new linked list is longer than MAXLEN, update the FDIRLEN by the new flow.

**Note:** The 82599 also reports the number of collided filters in FDIRFREE.COLL. Software might monitor this field periodically as an indication for the filters efficiency.

#### 7.1.2.7.7 Update Filter Flow

In some applications, it is useful to update the filter parameters, such as the destination Rx queue. Programing filter parameters is described in Section 7.1.2.7.6.

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

- Case 1: Matched filter does not exist in the filter table — Setting the *Filter-Update* bit has no impact and the command is treated as add filter.
- Case 2: Matched filter already exists in the filter table — Setting the *Filter-Update* bit enables filter parameter's update while keeping the collision indication as is.

The Update Filter Flow process requires internal memory space used to store temporary data until the update concludes. Therefore, Update Filter Flow can be used only if the maximum number of allocated flow director filters (as defined by FDIRCTRL.PBALLOC) is not fully used.

For example, if FDIRCTRL.PBALLOC=01b, memory is allocated for 2K-1 perfect filters. In this case, the Update Filter Flow can be used only if not more than 2K-2 filters were programmed.

#### 7.1.2.7.8 Remove Filter Flow

Software programs the filter Hash and Signature / Software-Index in the FDIRHASH register. It then should set the FDIRCMD.CMD field to *Remove Flow*. Software might use a single 64-bit access to the two registers for atomic operation. As a result, the 82599 follows these steps:

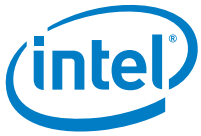
- Check if such a filter exists in the flow director filters table.
- If there is no flow, then increment the FREMOVE counter in the FDIRFSTAT register and skip the next steps.
- If the requested filter is the only filter in the linked list, then invalidate its entry in the hash lookup table by clearing the *Valid* bit.
- Else, if the requested filter is the last filter in the linked list, then invalidate the entry by setting the *Last* flag in the previous filter in the linked list.
- Else, invalidate its entry by programming the Next Filter PTR in the previous filter in the linked list, pointing it to the filter that was linked to the removed filter.

Additional indications for successful filter removal:

- Increment the remove statistic counter in the FDIRUSTAT register.
- Increment the FREE counter in the FDIRFREE register.

#### 7.1.2.7.9 Remove all Flow Director Filters

In some cases there is a need to clear the entire flow director table. It might be useful in some applications that might cause the flow director table becoming too occupied. Then, software might clear the entire table enabling its re-programming with new active flows.



Following are steps required to clear the flow director table:

- Poll the FDIRCMD.CMD until it is zero indicating any previous pending commands to the flow director table is completed (at worst case the FDIRCMD.CMD should be found cleared on the second read cycle). Note that the software must not initiate any additional commands (add / remove / query) before this step starts and until this flow completes.
- Clear the FDIRFREE register (set the *FREE* field to 0x8000 and *COLL* 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 one by hardware.
- Clear the following statistic registers: FDIRUSTAT; FDIRFSTAT; FDIRMATCH; FDIRMISS; FDIRLEN (note that some of these registers are read clear and some are read write).

#### 7.1.2.7.10 Flow Director Filters Initializing Flow

Following a device reset, the flow director is enabled by programming the FDIRCTRL register, as follows:

- Set PBALLOC to non-zero value according to the required buffer allocation to reception and flow director filter (see Section 7.1.2.7.4). All other fields in the register should be valid as well (according to required setting) while the FDIRCTRL register is expected to be programmed by a single cycle. Any further programming of the FDIRCTRL register with non-zero value PBALLOC initializes the flow director table once again.
- Poll the *INIT-Done* flag until it is set to one 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 Mb/s; however, this speed is not expected to be activated unless the 82599 is in a sleep state).

#### 7.1.2.7.11 Query Filter Flow

Software might query specific filter settings and bucket length using the Query command.

- Program the filter Hash and Signature/Software-Index in the FDIRHASH register and set the *CMD* field in the FDIRCMD register to 11b (Query Command). A single 64-bit access can be used for this step.
- As a result, the 82599 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 |
| Found Perfect Match Filter | 1 | 1 | Filter index within the linked list | Filter's parameters | Filter's parameters |

### 7.1.2.7.12 Signature Filter Registers

The signature flow director filter is programmed by setting the FDIRHASH and FDIRCMD registers. These registers are located in consecutive 8-byte aligned addresses. Software should use a 64-bit register to set these two registers in a single atomic operation. Table 7-6 lists the recommended setting.

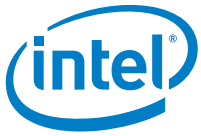
**Table 7-6 Signature Match Filter Parameters**

| | |
|---|---|
| **Filter Bucket Parameters — FDIRHASH** | |
| Hash | 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. The size of this field can be 15 bits, 14 bits or 13 bits as explained in Section 7.1.2.7.4. Non-used upper bits (MS bits) should be set to zero. |
| Valid | Should be set to 1b. |
| **Flow ID — FDIRHASH** | |
| Signature | 16-bit hash function used as the flow matching field. This parameter is also part of the flow director filter ID that can be reported in the Rx descriptor. |
| **FDIRCMD — Programming Command and Filter action** — Set Section 8.2.3.21.22 for all fields descriptions. | |

### 7.1.2.7.13 Perfect Match Filter Registers

Perfect match filters are programmed by the following registers: FDIRSIPv6[n]; FDIRVLAN; FDIRPORT; FDIRIPDA; FDIRIPSA; FDIRHASH; FDIRCMD. Setting the FDIRCMD register, generates the actual programming of the filter. Therefore, write access to this register must be the last cycle after all other registers contain a valid content. Table 7-7 lists the recommended setting.

**Note:** Software filter programming must be an atomic operation. In a multi-core environment, software must ensure that all registers are programmed in a sequence with no possible interference by other cores.



### Table 7-7 Perfect Match Filter Parameters

| | |
|---|---|
| **Filter Bucket Parameters and Software Index — FDIRHASH** | |
| Hash | 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. The size of this field can be 13 bits, 12 bits or 11 bits as explained in Section 7.1.2.7.4. Non-used upper bits (MS bits) should be set to zero. |
| Valid | Should be set to 1b. |
| Software-Index | 15-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.<br>*Note:* The Software-Index is used as the filter identifier. Therefore, it must be within the range of supported filters while any filter must have a single unique Software-Index value. |
| **FDIRCMD — Programming Command and Filter Action See Section 8.2.3.21.22 for All Fields Descriptions** | |
| **Flow ID — Perfect Match Flow ID Parameters are Listed in the Following Registers and Fields** | |
| FDIRSIPv6[0…2].IP6SA | Three MS DWord of the source IPv6. Meaningful for IPv6 flows depending on the FDIRIP6M.SIPM setting. |
| FDIRVLAN.VLAN | VLAN fields are meaningful depending on the FDIRM.VLANID and FDIRM.VLANP setting. |
| FDIRVLAN.FLEX | Flexible 2-byte field at offset FDIRCTRL.Flex-Offset. Meaningful depending on FDIRM.FLEX setting. |
| FDIRPORT.Source | L4 source port. Meaningful for TCP and UDP packets depending on the FDIRTCPM.SportM and FDIRUDPM.SportM setting. |
| FDIRPORT.Destination | L4 destination port. Meaningful for TCP and UDP packets depending on the FDIRTCPM.SportM and FDIRUDPM.SportM setting. |
| FDIRIPDA.IP4DA | IPv4 destination address. Meaningful depending on the FDIRDIP4M.IP-EN setting. |
| FDIRIPSA.IP4SA | IPv4 source address or LS DWord of the source IPv6 address. Meaningful for IPv4 flows depending on the FDIRSIP4M.IP-EN setting and for IPv6 flows depending on the FDIRIP6M.SIPM setting. |

## 7.1.2.7.14 Multiple CPU Cores Considerations

Perfect match filters programming and any query cycles requires access to multiple registers. In order to avoid races between multiple cores, software might need to use one of the following programming methods:

- Use a software-based semaphore between the multiple cores for gaining control over the relevant CSR registers for complete programming or query cycles.
- Manage all programming and queries of the flow director filters by a single core.

Programming signature filters requires only the FDIRHASH and FDIRCMD registers. These two registers are located in 8-byte aligned adjacent addresses. Software could use an 8-byte register for the programming of these registers in a single atomic operation, which avoids the need for any semaphore between multiple cores.

#### 7.1.2.7.15 Flow Director Hash Function

The 82599 supports programmable 16-bit hash functions based on two 32-bit keys, one for the lookup table identifying a bucket of filters and another one for the signature (FDIRHKEY and FDIRSKEY). The hash function is described in the sections that follow. In some cases, a smaller hash value than 16 bits is required. In such cases, the LS bits of the hash value are used.

```
For (i=0 to 350) { if (Ext_K[i]) then Hash[15 : 0] = Hash[15 : 0] XOR Ext_S[15+i : i] }
```

While using the following notations:

'XOR' - Bitwise XOR of two equal length strings

If ( xxx ) - Equals 'true' if xxx = '1' and equals 'false' if xxx = '0'

S[335:0] - The input bit string of the flow director tuples: 42 bytes listed in Table 7-8 AND-logic with the filters masks.

Ext_S[n] - S[14:0] | S[335:0] | S[335:321] // concatenated

K[31:0] - The hash key as defined by the FDIRHKEY or FDIRSKEY registers.

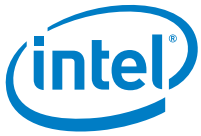
Tmp_K[11*32-1:0] - (Temp Key) equals K[31:0] | K[31:0] ... // concatenated Key 11 times

Ext_K[350:0] - (Extended Key) equals Tmp_K[351:1]

The input bit stream for the hash calculation is listed in the Table 7-8 while byte 0 is the MSByte (first on the wire) of the VLAN, byte 2 is the MSByte of the source IP (IPv6 case) and so on.

**Table 7-8 Input Bit Stream for Hash Calculation**

| Bytes | Field |
|---|---|
| Bytes 0…1 | VLAN tag |
| Bytes 2…17 | Source IP (16 bytes for IPv6; 12 bytes of zero's \| source IP for IPv4) |
| Bytes 18…33 | Destination IP (16 bytes for IPv6; 12 bytes of zero's \| source IP for IPv4) |
| 34...37 | L4 source port number \| L4 destination port number<br>Meaningful for TCP and UDP packets and zero bytes for SCTP packets |
| 38...39 | Flexible bytes |
| 40 | 00b \| pool number (as defined by FDIRCMD.Pool) |
| 41 | 00000b \| IPv6/IPv4 type \| L4 type (as defined by FDIRCMD.IPV6 and FDIRCMD.L4TYPE, respectively) |



### 7.1.2.8 Receive-Side Scaling (RSS)

RSS is a mechanism to distribute received packets into several descriptor queues. Software then assigns each queue to a different processor, therefore sharing the load of packet processing among several processors.

As described in Section 7.1, the 82599 uses RSS as one ingredient in its packet assignment policy (the others are the various filters, DCB and virtualization). The RSS output is an RSS index. The 82599 global assignment uses these bits (or only some of the LSBs) as part of the queue number.

Figure 7-10 shows the process of computing an RSS output:

1. The receive packet is parsed into the header fields used by the hash operation (such as IP addresses, TCP port, etc.)
2. A hash calculation is performed. The 82599 supports a single hash function, as defined by MSFT RSS. The 82599 therefore does not indicate to the device driver which hash function is used. The 32-bit result is fed into the packet receive descriptor.
3. The seven LSBs of the hash result are used as an index into a 128-entry redirection table. Each entry provides a 4-bit RSS output index.

When RSS is enabled, the 82599 provides software with the following information as:

1. Required by Microsoft (MSFT) RSS
2. Provided for device driver assist:
   - A Dword result of the MSFT RSS hash function, to be used by the stack for flow classification, is written into the receive packet descriptor (required by MSFT RSS).
   - A 4-bit RSS *Type* field conveys the hash function used for the specific packet (required by MSFT RSS).

Enabling rules:

- RSS is enabled in the MRQC register.
- RSS enabling cannot be done dynamically while it must be preceded by a software reset.
- RSS status field in the descriptor write-back is enabled when the RXCSUM.PCSD bit is set (fragment checksum is disabled). RSS is therefore mutually exclusive with UDP fragmentation checksum offload.
- Support for RSS is not provided when legacy receive descriptor format is used.

Disabling rules:

- Disabling RSS on the fly is not allowed, and the 82599 must be reset after RSS is disabled.
- When RSS is disabled, packets are assigned an RSS output index = zero.

When multiple request queues are enabled in RSS mode, un-decodable packets are assigned an RSS output index = zero. The 32-bit tag (normally a result of the hash function) equals zero.

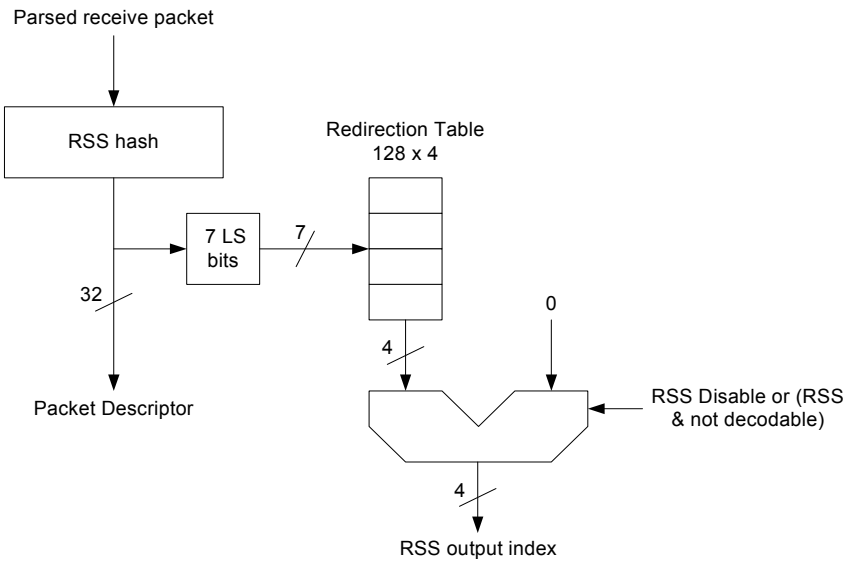



**Figure 7-10 RSS Block Diagram**

### 7.1.2.8.1 RSS Hash Function

This section provides a verification suite used to validate that the hash function is computed according to MSFT nomenclature.

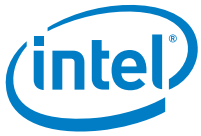
The 82599's hash function follows the MSFT definition. A single hash function is defined with several variations for the following cases:

- TcpIPv4 — The 82599 parses the packet to identify an IPv4 packet containing a TCP segment per the following criteria. If the packet is not an IPv4 packet containing a TCP segment, RSS is not done for the packet.
- IPv4 — parses the packet to identify an IPv4 packet. If the packet is not an IPv4 packet, RSS is not done for the packet.
- TcpIPv6 — parses the packet to identify an IPv6 packet containing a TCP segment per the following criteria. If the packet is not an IPv6 packet containing a TCP segment, RSS is not done for the packet.
- IPv6 — The 82599 parses the packet to identify an IPv6 packet. If the packet is not an IPv6 packet, RSS is not done for the packet.

**Note:** Tunneled IP to IP packets are considered for the RSS functionality as IP packets. The RSS logic ignores the L4 header while using the outer (first) IP header for the RSS hash.

The following additional cases are not part of the MSFT RSS specification:

- UdpIPV4 — The 82599 parses the packet to identify a packet with UDP over IPv4.
- UdpIPV6 — The 82599 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 82599, 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) register enable each of the above hash function variations (several might be set at a given time). If several functions are enabled at the same time, priority is defined as follows (skip functions that are not enabled):

- IPv4 packet
  - Try using the TcpIPv4 function
  - Try using UdpIPv4 function
  - Try using the IPv4 function
- IPv6 packet
  - Try using the TcpIPv6 function.
  - Try using UdpIPv6 function.
  - Try using the IPv6 function

The following combinations are currently supported:

- Any combination of IPv4, TcpIPv4, and UdpIPv4.

And/Or:

- Any combination of either IPv6, TcpIPv6, and UdpIPv6.

When a packet cannot be parsed by the previous rules, it is assigned an RSS output index = zero. The 32-bit tag (normally a result of the hash function) equals zero.

The 32-bit result of the hash computation is written into the packet descriptor and also provides an index into the redirection table.

The following notation is used to describe the following hash functions:

- Ordering is little endian in both bytes and bits. For example, the IP address 161.142.100.80 translates into 0xa18e6450 in the signature.
- A " ^ " denotes bit-wise XOR operation of same-width vectors.
- @x-y denotes bytes x through y (including both of them) of the incoming packet, where byte 0 is the first byte of the IP header. In other words, we consider all byte-offsets as offsets into a packet where the framing layer header has been stripped out. Therefore, the source IPv4 address is referred to as @12-15, while the destination v4 address is referred to as @16-19.

- @x-y, @v-w denotes concatenation of bytes x-y, followed by bytes v-w, preserving the order in which they occurred in the packet.

All hash function variations (IPv4 and IPv6) follow the same general structure. Specific details for each variation are described in the following section. The hash uses a random secret key of length 320 bits (40 bytes); the key is stored in the RSS Random Key Register (RSSRK).

The algorithm works by examining each bit of the hash input from left to right. Our nomenclature defines left and right for a byte-array as follows: Given an array K with k bytes, our nomenclature assumes that the array is laid out as follows:

- K[0] K[1] K[2] … K[k-1]

K[0] is the left-most byte, and the MSB of K[0] is the left-most bit. K[k-1] is the right-most byte, and the LSB of K[k-1] is the right-most bit.

ComputeHash(input[], N)

```
For hash-input input[] of length N bytes (8N bits) and a random secret key
K of 320 bits
Result = 0;
For each bit b in input[] {
if (b == 1) then Result ^= (left-most 32 bits of K);
shift K left 1 bit position;
}
return Result;
```

#### 7.1.2.8.1.1 Pseudo-Code Examples

The following four pseudo-code examples are intended to help clarify exactly how the hash is to be performed in four cases, IPv4 with and without ability to parse the TCP header, and IPv6 with and without a TCP header.

Hash for IPv4 with TCP

Concatenate SourceAddress, DestinationAddress, SourcePort, DestinationPort into one single byte-array, preserving the order in which they occurred in the packet: Input[12] = @12-15, @16-19, @20-21, @22-23.

```
Result = ComputeHash(Input, 12);
```

Hash for IPv4 with UDP

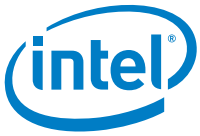
Concatenate SourceAddress, DestinationAddress, SourcePort, DestinationPort into one single byte-array, preserving the order in which they occurred in the packet: Input[12] = @12-15, @16-19, @20-21, @22-23.

```
Result = ComputeHash(Input, 12);
```

Hash for IPv4 without TCP

Concatenate SourceAddress and DestinationAddress into one single byte-array

```
Input[8] = @12-15, @16-19
Result = ComputeHash(Input, 8)
```



#### Hash for IPv6 with TCP

Similar to above:

```
Input[36] = @8-23, @24-39, @40-41, @42-43
Result = ComputeHash(Input, 36)
```

#### Hash for IPv6 with UDP

Similar to above:

```
Input[36] = @8-23, @24-39, @40-41, @42-43
Result = ComputeHash(Input, 36)
```

#### Hash for IPv6 without TCP

```
Input[32] = @8-23, @24-39
Result = ComputeHash(Input, 32)
```

### 7.1.2.8.2 Redirection Table

The redirection table is a 128-entry structure, indexed by the seven LSBs of the hash function output.

System software might update the redirection table during run time. Such updates of the table are not synchronized with the arrival time of received packets. Therefore, it is not guaranteed that a table update takes effect on a specific packet boundary.

### 7.1.2.8.3 RSS Verification Suite

Assume that the random key byte-stream is:

```
0x6d, 0x5a, 0x56, 0xda, 0x25, 0x5b, 0x0e, 0xc2,
0x41, 0x67, 0x25, 0x3d, 0x43, 0xa3, 0x8f, 0xb0,
0xd0, 0xca, 0x2b, 0xcb, 0xae, 0x7b, 0x30, 0xb4,
0x77, 0xcb, 0x2d, 0xa3, 0x80, 0x30, 0xf2, 0x0c,
0x6a, 0x42, 0xb7, 0x3b, 0xbe, 0xac, 0x01, 0xfa
```

**Table 7-9 IPv4**

| Destination Address/Port | Source Address/Port | IPv4 only | IPv4 with TCP |
|---|---|---|---|
| 161.142.100.80 :1766 | 66.9.149.187 :2794 | 0x323e8fc2 | 0x51ccc178 |
| 65.69.140.83 :4739 | 199.92.111.2 :14230 | 0xd718262a | 0xc626b0ea |
| 12.22.207.184 :38024 | 24.19.198.95 :12898 | 0xd2d0a5de | 0x5c2b394a |
| 209.142.163.6 :2217 | 38.27.205.30 :48228 | 0x82989176 | 0xafc7327f |
| 202.188.127.2 :1303 | 153.39.163.191 :44251 | 0x5d1809c5 | 0x10e828a2 |

The IPv6 address tuples are only for verification purposes, and may not make sense as a tuple.

**Table 7-10 IPv6**

| Destination Address/Port | Source Address/Port | IPv6 only | IPv6 with TCP |
|---|---|---|---|
| 3ffe:2501:200:3::1 (1766) | 3ffe:2501:200:1fff::7 (2794) | 0x2cc18cd5 | 0x40207d3d |
| ff02::1 (4739) | 3ffe:501:8::260:97ff:fe40:efab (14230) | 0x0f0c461c | 0xdde51bbf |
| fe80::200:f8ff:fe21:67cf (38024) | 3ffe:1900:4545:3:200:f8ff:fe21:67cf (44251) | 0x4b61e985 | 0x02d1feef |

## 7.1.3 MAC Layer Offloads

### 7.1.3.1 CRC Strip

The 82599 potentially strips the L2 CRC on incoming packets.

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

The policy for CRC strip is as follows:

- When RSC is enabled on any queue, the global CRC strip bit should be set (HLREG0.RXCRCSTRP = 1).
- When either LinkSec or IPsec are enabled, the global CRC strip bit should be set (HLREG0.RXCRCSTRP= 1b), since the payload of the packet changes and the CRC value is stale due to it.

## 7.1.4 Receive Data Storage in System Memory

The 82599 posts receive packets into data buffers in system memory.

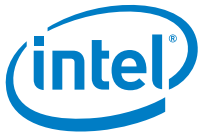
The following controls are provided for the data buffers:

- The SRRCTL[n].BSIZEPACKET field defines the data buffer size. See section Section 7.1.2 for packet filtering by size.
- The SRRCTL.BSIZEHEADER field defines the size of the buffers allocated to headers (advanced descriptors only).
- Each queue is provided with a separate SRRCTL register.

Receive memory buffer addresses are word (2 x byte) aligned (both data and headers).

## 7.1.5 Legacy Receive Descriptor Format

A receive descriptor is a data structure that contains the receive data buffer address and fields for hardware to store packet information. Upon receipt of a packet for this device, hardware stores the packet data into the indicated buffer and writes the length, status and errors to the receive descriptor. If SRRCTL[n].DESCTYPE = zero, the 82599 uses the Legacy Rx descriptor as listed in Table 7-11. The shaded areas indicate fields that are modified by hardware upon packet reception (so-called descriptor write-back).



Legacy descriptors should not be used when advanced features are enabled: SCTP, Virtualization, DCB, LinkSec, IPSec, FCoE or RSC. Packets that match these cases might be dropped from queues that use legacy receive descriptors.

Refer to Table 7-11 and the field descriptions that follow.

### Table 7-11 Legacy Receive Descriptor (RDESC) Layout

| | 63 48 | 47 40 | 39 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 7.1.13.

#### Status Field (8-bit offset 32, 2nd line)

Status information indicates whether the descriptor has been used and whether the referenced buffer is the last one for the packet. Error status information is listed in Table 7-13.

### Table 7-12 Receive Status (RDESC.STATUS) Layout

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| PIF | IPCS | L4CS | UDPCS | VP | Reserved | EOP | DD |

#### *EOP (End of Packet) and DD (Descriptor Done)*

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* fieldis 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)*

The VP field indicates whether the incoming packet's type is a VLAN (802.1q). It is set if the packet type matches VLNCTRL.VET. Furthermore, if the RXDCTL.VME bit is set then active VP bit also indicates that VLAN has been stripped in the 802.1q packet. For a further description of 802.1q VLANs, see Section 7.4.

*IPCS (Ipv4 Checksum), L4CS (L4 Checksum), UDPCS (UDP Checksum)*

These bits are described in the following table. In I/O mode: switched packets from a local VM that do not use the Tx IP checksum offload by hardware have the IPCS equal to zero; switched packets from a local VM that do not use the Tx L4 checksum offload by hardware have the L4CS and UDPCS equal to zero.

| L4CS | UDPCS | IPCS | Functionality |
|---|---|---|---|
| 0 | 0 | 0 | Hardware does not provide checksum offload. |
| 0 | 0 | 1 | Hardware provides IPv4 checksum offload. Pass/fail indication is provided in the *Error* field – IPE. |
| 1 | 0 | 1 / 0 | Hardware provides IPv4 checksum offload if IPCS is active along with TCP checksum offload. Pass/fail indication is provided in the *Error* field – IPE and TCPE |
| 1 | 1 | 1 / 0 | Hardware provides IPv4 checksum offload if IPCS is active along with UDP checksum offload. Pass/fail indication is provided in the *Error* field – IPE and TCPE |

See Section 7.1.11 for a description of supported packet types for receive checksum offloading. IPv6 packets do not have the *IPCS* bit set, but might have the *L4CS* bit and *UDPCS* bit set if the 82599 recognizes the transport header.

*PIF (Non Unicast Address)*

The *PIF* bit is set on packets with a non-unicast destination Ethernet MAC address — multicast and broadcast.

Error Field (8-bit offset 40, 2nd line)

Table 7-13 and the following text describes the possible errors reported by the hardware.

**Table 7-13 Receive Errors (RDESC.ERRORS) Layout**

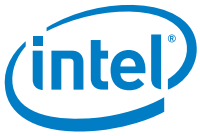
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IPE | TCPE | Reserved | Reserved | Reserved | Reserved | Reserved | RXE |

*IPE (Ipv4 Checksum Error)*

The IP checksum error is valid only when the *IPCS* bit in the *Status* field is set (indicating that the hardware validated the IP checksum). This bit is meaningful only on the last descriptor of a packet while the *EOP* bit is set as well. Packets with IP error are posted to host memory regardless of the store bad packet setting (FCTRL.SBP).

*TCPE (TCP/UDP Checksum Error)*

The TCP/UDP checksum error is valid only when the *L4CS* bit in the *Status* field is set (indicating that the hardware validated the L4 checksum). This bit is meaningful only on the last descriptor of a packet while the *EOP* bit is set as well. Packets with a TCP/UDP error are posted to host memory regardless of the store bad packet setting (FCTRL.SBP).



*RXE*

The RXE error bit is an indication for any MAC error. It is a logic or function of the following errors:

- CRC or symbol error might be a result of receiving a /V/ symbol on the TBI interface, /FE/ symbol on the GMII/XGMII interface, RX_ER assertion on GMII interface, bad EOP or loss of sync during packet reception.
- Undersize frames shorter than 64 bytes.
- Oversize frames larger than the MFS definition in the MAXFRS register.
- Length error in 802.3 packet format. Packets with an RXE error are posted to host memory only when store bad packet bit (FCTRL.SBP) is set.

VLAN Tag Field (16-bit offset 48, 2nd line)

If the RXDCTL.VME is set and the received packet type is 802.1q (as defined by VLNCTRL.VET) then the VLAN header is stripped from the packet data storage. In this case the 16 bits of the VLAN tag, priority tag and CFI from the received packet are posted to the *VLAN Tag* field in the receive descriptor. Otherwise, the *VLAN Tag* field contains 0x0000.

**Table 7-14 VLAN Tag Field Layout (for 802.1q Packet)**

| 15 13 | 12 | 11 0 |
|---|---|---|
| PRI | CFI | VLAN |

Priority and CFI are part of 803.1Q specifications. The VLAN field is provided in network order.

# 7.1.6 Advanced Receive Descriptors

## 7.1.6.1 Advanced Receive Descriptors — Read Format

Table 7-15 lists the advanced receive descriptor programming by the software. The SRRCTL[n]. DESCTYPE should be set to a value other than 000 when using the advanced descriptor format.

**Table 7-15 Descriptor Read Format**

| | 63 1 | 0 |
|---|---|---|
| 0 | Packet Buffer Address [63:1] | A0 |
| 8 | Header Buffer Address [63:1] | DD |

Packet Buffer Address (64)

This is the physical address of the packet buffer. The lowest bit is A0 (LSB of the address).Header Buffer Address (64)

The physical address of the header buffer with the lowest bit being Descriptor Done (DD). When a packet spans in multiple descriptors, the header buffer address is used only on the first descriptor. During the programming phase, software must set the *DD* bit to zero (see the description of the DD bit in this section). This means that header buffer addresses are always word aligned.

When a packet spans in more than one descriptor, the header buffer address is not used for the second, third, etc. descriptors; only the packet buffer address is used in this case.

**Note:** The 82599 does not support null descriptors meaning packet or header addresses are zero.

### 7.1.6.2 Advanced Receive Descriptors — Write-Back Format

When the 82599 writes back the descriptors, it uses the format listed in Table 7-16. The SRRCTL[n]. DESCTYPE should be set to a value other than 000 when using the advanced descriptor format.

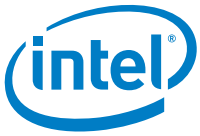
**Table 7-16 Descriptor Write-Back Format**

| | 63 48 | 47 32 | 31 | 30 21 | 20 17 | 16 4 | 3 0 |
|---|---|---|---|---|---|---|---|
| 0 | RSS Hash / Fragment Checksum / RTT / FCoE_PARAM / Flow Director Filters ID | | SPH | HDR_LEN | RSCCNT | Packet Type | RSS Type |
| 8 | VLAN Tag | PKT_LEN | Extended Error | | Extended Status / NEXTP | | |
| | 63 48 | 47 32 | 31 20 | | 19 0 | | |

#### RSS Type (4-bit offset 0, 1st line)

The 82599 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 |



| RSS Type | Description |
|---|---|
| 0x9 – 0xE | Reserved |
| 0xF | Packet reports flow director filters status |

### Packet Type (13-bit at offset 4, 1st line)

The Packet Type field reports the packet type identified by the hardware as follows. Note that some of the fields in the receive descriptor are valid for specific packet types. For example, the *FCOE_PARAM* field (multiplexed with the RSS) is valid only for FCoE packets.

| Bit Index | Bit 11 = 0 | Bit 11 = 1 (L2 packet) |
|---|---|---|
| 0 | IPV4 — IPv4 header present | Ethertype — ETQF register index that matches the packet. Special types are defined for 802.1x, 1588, and FCoE. |
| 1 | IPV4E — IPv4 with extensions | |
| 2 | IPV6 — IPv6 header present | |
| 3 | IPV6E — IPv6 with extensions | Reserved for extension of the *EtherType* field. |
| 4 | TCP — TCP header present | Reserved for extension of the *EtherType* field. |
| 5 | UDP — UDP header present | Reserved |
| 6 | SCTP — SCTP header | Reserved |
| 7 | NFS — NFS header | Reserved |
| 8 | IPSec ESP — IPSec encapsulation | Reserved |
| 9 | IPSec AH — IPSec encapsulation | Reserved |
| 10 | LinkSec — LinkSec encapsulation | LinkSec — LinkSec encapsulation |
| 11 | 0b = non L2 packet | 1b = L2 packet |
| 12 | Reserved | Reserved |

**Note:** UDP, TCP and IPv6 indications are not set in an IPv4 fragmented packet.

In IOV mode, packets might be received from other local VMs. the 82599 does not check the L5 header for these packets and does not report NFS header. If such packets carry IP tunneling (IPv4 — IPv6), they are reported as IPV4E. The packets received from local VM are indicated by the *LB* bit in the status field.

### RSC Packet Count- RSCCNT (4-bit offset 17, 1st line)

The *RSCCNT* field is valid only for RSC descriptors while in non-RSC it equals zero. RSCCNT minus one indicates the number of coalesced packets that start in this descriptor. RSCCNT might count up to 14 packets. Once 14 packets are coalesced in a single buffer, RSC is closed enabling accurate coalesced packet count. If the *RSCCNTBP* bit in RDRXCTL is set, coalescing might proceed beyond the 14 packets per buffer while RSCCNT stops incrementing beyond 0xF.

**Note:** Software can identify RSC descriptors by checking the *RSCCNT* field for non-zero value.

### HDR_LEN (10-bit offset 21, 1st line)

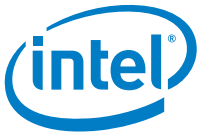
The HDR_LEN reflects the size of the packet header in byte units (if the header is decoded by the hardware). This field is meaningful only in the first descriptor of a packet and should be ignored in any subsequent descriptors. Header split is explained in Section 7.1.10 while the packet types for this functionality are enabled by the PSRTYPE[n] registers.

### Split Header — SPH (1-bit offset 31, 1st line)

When set to 1b, indicates that the hardware has found the length of the header. If set to 0b, the header buffer may be used only in split always mode. The header buffer length as well as split header support is indicated in the following table. 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 first descriptor of a packet. See additional details on SPH, PKT_LEN and HDR_LEN as a function of split modes in Table 7-20.

| Packet Type | Header Length (includes all fields up to the field specified) | Header Split |
|---|---|---|
| Unrecognized Ethertype only with / without SNAP and with / without VLAN or packets that match the L2 filters (MTQF) other than FCoE with / without VLAN. | VLAN header(s) if present<br>Else, *EtherType* field | No |
| FCoE packet without ESP option header | FC header including FC options | N/A (1) |
| FCoE packet with ESP option header | FC header excluding FC options | N/A (1) |
| IPv4 only or fragmented IPv4 with any payload including IPv4-IPv6 tunneling | IPv4 header | Enabled |
| Non-fragmented IPv4, TCP / UDP / SCTP | L4 header | Enabled |
| IPv4-IPv6, only or fragmented IPv4-IPv6 at IPv6 header with any payload | IPv6 header (up to the fragment extension header if exist) | Enabled |
| IPv4-IPv6,TCP / UDP / SCTP | L4 header | Enabled |
| IPv4 / IPv6 / IPv4-IPv6, TCP / UDP, NFS | L5 header | Enabled |

*Notes:* (1) Header split is not permitted in queues that might receive FCoE packets.



### RSS Hash or FCOE_PARAM or Flow Director Filters ID (32-bit offset 32, 1st line) / Fragment Checksum (16-bit offset 48, 1st line)

This field has multiplexed functionality according to the received packet type (reported on the *Packet Type* field in this descriptor) and device setting.

#### *FCoE_PARAM*

For FCoE packets that matches a valid DDP context, this field holds the *PARAM* field in the DDP context after processing the received packet. If the *Relative Offset Present* bit in the F_CTL was set in the data frames, the *PARAM* field indicates the size in bytes of the entire exchange inclusive the frame reported by this descriptor.

#### *Fragment Checksum*

For non-FCoE packets, this field might hold the UDP fragment checksum (described in Section 7.1.13) if both the RXCSUM.PCSD bit is cleared and RXCSUM. The *IPPCSE* bit is also set. This field is meaningful only for UDP packets when the *UDPV* bit in the Extended Status word is set.

#### RSS Hash / Flow Director Filters ID

For  non-FCoE packets, this field might hold the RSS hash value or flow director filters ID if the RXCSUM.PCSD bit is set. Furthermore, if the FDIRCTRL.Report-Status bit is set, then the flow director filters ID is reported; otherwise, the RSS hash is reported.

#### *RSS Hash*

The RSS hash value is required for RSS functionality as described in Section 7.1.2.8.

#### *Flow Director Filters ID*

The flow director filters ID is reported only when the received packet matches a flow directory filter (see Section 7.1.2.7).

The flow director filter ID field has a different structure for signature-based filters and perfect match filters as follows:

| Filter Type | 31 | 30 29 | 28 16 | 15 13 | 12 0 |
|---|---|---|---|---|---|
| Hash-based Flow Director Filter ID | Rsv | Bucket Hash | | Signature | |
| | | | | | |
| Perfect Match Flow Director Filter ID | Rsv | | Hash | Rsv | SW-Index |

#### *Bucket Hash*

A hash value that identifies a flow director bucket. When the flow director table is smaller than 32 K filters the bucket hash is smaller than 15 bits. In this case the upper bit(s) are set to zero.

#### *Signature*

A hash value used to identify flow within a bucket.

#### *SW-Index*

The SW-Index that is taken from the filter context, programmed by software. It is meaningful only when the *FLM* bit in the Extended Status is set as well.

#### *Rsv*

Reserved.

### Extended Status / NEXTP (20-bit offset 0, 2nd line)

Status information indicates whether the descriptor has been used and whether the referenced buffer is the last one for the packet. Table 7-17 lists the extended status word in the last descriptor of a packet (*EOP* bit is set). Table 7-18 lists the extended status word in any descriptor but the last one of a packet (*EOP* bit is cleared).

**Table 7-17 Receive Status (RDESC.STATUS) Layout of *Last* Descriptor**

| 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| Rsv | LB | SECP | TS | Rsv | | | | LLINT | UDPV |

| VEXT | Rsv | PIF | IPCS | L4I | UDPCS | VP | FLM | EOP | DD |
|---|---|---|---|---|---|---|---|---|---|
| | | | FCEOFs | FCSTAT | | | | | |
| **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |

**Table 7-18 Receive Status (RDESC.STATUS) Layout of *Non-Last* Descriptor**

| 19 4 | 3 : 2 | 1 | 0 |
|---|---|---|---|
| Next Descriptor Pointer — NEXTP | Rsv | EOP = 0b | DD |

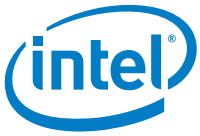
Rsv (8), Rsv (15:12), Rsv(19) — Reserved at zero.

FLM(2) — Flow director filter match indication is set for packets that match these filters.

VP(3), PIF (7) — These bits are described in the legacy descriptor format in Section 7.1.5.

EOP (1) and DD (0) — *End of Packet* and *Done* bits are listed in the following table:

| DD | EOP | Description |
|---|---|---|
| 0 | 0 | Software setting of the descriptor when it hands it to hardware. |
| 0 | 1 | Reserved (invalid option) |
| 1 | 0 | A completion status indication for a non last descriptor of a packet (or multiple packets in the case of RSC) that spans across multiple descriptors. In a single packet case the *DD* bit indicates that the hardware is done with the descriptor and its buffers. In the case of RSC, the *DD* bit indicates that the hardware is done with the descriptor but might still use its buffers (for the coalesced header) until the last descriptor of the RSC completes.<br>Only the *Length* fields are valid on this descriptor. In the RSC case, the next descriptor pointer is valid as well. |
| 1 | 1 | A completion status indication of the entire packet (or the multiple packets in the case of RSC) and software might take ownership of its descriptors.<br>All fields in the descriptor are valid (reported by the hardware). |



UDPCS(4), L4I (5) / FCSTAT (5:4) — This field has multiplexed functionality for FCoE and non-FCoE packets. Hardware identifies FCoE packets in the filter unit and indicates it in the *Packet Type* field in the Rx descriptor. For non-FCoE packets this field is UDPCS and L4I. The UDPCS (UDP checksum) is set when hardware provides UDP checksum offload. The L4I (L4 Integrity) is set when hardware provides any L4 offload as: UDP checksum, TCP checksum or SCTP CRC offload. For FCoE packets, this field represents the FCSTAT (FCoE Status) as follows:

| FCSTAT | Meaning |
|---|---|
| 00 | No match to any active FC context |
| 01 | FCoE frame matches an active FC context with no DDP. The entire frame is posted to the receive buffer indicated by this descriptor. |
| 10 | FCP_RSP frame received that invalidates an FC read context or last data packet in a sequence with sequence initiative set that invalidates an FC write context. |
| 11 | FCoE frame matches an active FC context and found liable for DDP by the filter unit. The packet's data was posted directly to the user buffers if no errors were found by the DMA unit as reported in the *FCERR* field. If any error is found by the DMA unit the entire packet is posted to the legacy queues. |

IPCS(6), FCEOFs (6) — This bit has multiplexed functionality for FCoE and non-FCoE packets. The hardware identifies FCoE packets in the filter unit and indicates it in the *Packet Type* field in the Rx descriptor. For non-FCoE packets it is IPCS as described in Legacy Rx descriptor (in Section 7.1.5). For FCoE packets, this bit and the *FCEOFe* bit in the *Extended Error* field indicates the received EOF code as follows:

| FCEOFe | FCEOFs | Description and Digested meaning and Device Behavior |
|---|---|---|
| 0 | 0 | EOFn. Nominal operation, DDP is enabled. |
| 0 | 1 | EOFt. Nominal operation (end of sequence), DDP is enabled. |
| 1 | 0 | Unexpected EOFn-EOFt or SOFi-SOFn. No DDP while filter context is updated by the packet. |
| 1 | 1 | EOFa, EOFni or un-recognized EOF / SOF. No DDP while filter context is invalidated. |

VEXT (9) — Outer-VLAN is found on a double VLAN packet. This bit is valid only when CTRL_EXT.EXTENDED_VLAN is set. See more details in Section 7.4.5.

UDPV (10) — The *UDP Checksum Valid* bit indicates that the incoming packet contains a valid (non-zero value) checksum field in an incoming fragmented (non-tunneled) UDP IPv4 packet. It means that the *Fragment Checksum* field in the receive descriptor contains the UDP checksum as described in Section 7.1.13. When this field is cleared in the first fragment that contains the UDP header, it means that the packet does not contain a valid UDP checksum and the checksum field in the Rx descriptor should be ignored. This field is always cleared in incoming fragments that do not contain the UDP header.

LLINT (11) — The *Low Latency Interrupt* bit indicates that the packet caused an immediate interrupt via the low latency interrupt mechanism.

TS (16) — The *Time Stamp* bit is set when the device recognized a time sync packet. In such a case the hardware captures its arrival time and stores it in the Time Stamp register. For more details see Section 7.9.

SECP (17) — Security processing bit indicates that the hardware identified the security encapsulation and processed it as configured.

LinkSec processing — This bit is set each time LinkSec processing is enabled regardless if a matched SA was found.

IPsec processing — This bit is set only if a matched SA was found. Note that hardware does not process packets with an IPv4 option or IPv6 extension header and the SECP bit is not set. This bit is not set for IPv4 packets shorter than 70 bytes, IPv6 ESP packets shorter than 90 bytes, or IPv6 AH packets shorter than 94 bytes (all excluding CRC). Note that these packet sizes are never expected and set the length error indication in the SECERR field.

LB (18) — This bit provides a loopback status indication which means that this packet is sent by a local VM (VM to VM switch indication).

NEXTP (19:4) — Large receive might be composed of multiple packets and packets might span in multiple buffers (descriptors). These buffers are not guaranteed to be consecutive while the *NEXTP* field is a pointer to the next descriptor that belongs to the same RSC. The *NEXTP* field is defined in descriptor unit (the same as the head and tail registers). The *NEXTP* field is valid for any descriptor of a large receive (the *EOP* bit is not set) except the last one. It is valid even in consecutive descriptors of the same packet. In the last descriptor (on which the *EOP* bit is set), *NEXTP* is not indicated but rather all other status fields previously described in this section.

Extended Error (12-bit offset 21, 2nd line)

Table 7-19 and the following text describe the possible errors reported by hardware.

t

**Table 7-19 Receive Errors (RDESC.ERRORS) Layout**

| 11 | 10 | 9 | 8:7 | 6:4 | 3 | 2:0 |
|---|---|---|---|---|---|---|
| IPE | L4E | RXE | Rsv | Rsv | HBO | Rsv |
| FCEOFe | | | SECERR | | | FCERR / FDIRERR |

FCERR (2:0) — Defines error cases for FCoE packets. Note that hardware indicates FCoE packet recognition on the Packet Type field in the Rx descriptor. Packets with FCERR are posted to host memory regardless of the store bad packet setting in the Filter Control register.

| FCERR Code | Meaning |
|---|---|
| 000 | No exception errors found |
| 001 | Bad FC CRC. Hardware does not check any other FC fields in the packet. |
| 010 | One of the following error indications found by the filter unit (hardware auto-invalidates a matched DDP filter context if exists):<br>1. Received non-zero abort sequence condition in the *F_CTL* field in FC read packet.<br>2. Received EOFa or EOFni or any un-recognized EOF or SOF flags. |
| 011 | The DMA unit gets FCoE packets that match a DDP context while it missed the packet that was marked as first by the filter unit. Filter context parameters might be updated while DMA context parameters are left intact (see error code 101b). |

| FCERR Code | Meaning |
|---|---|
| 100 | One of the following cases:<br>1. Unsupported FCoE version. FCSTAT equals to 00b.<br>2. Out of order reception (SEQ_CNT does not match expected value) of a packet that matches an active DDP context. The filter unit might set the FCSTAT to 01b, 10b or 11b. |
| 101 | No DMA resources due to one of the following cases listed while the hardware auto-invalidates the DDP DMA context. Software should invalidate the filter context before it can reuse it.<br>(1) Last buffer exhausted (no space in the user buffers).<br>(2) Legacy receive queue is not enabled or no legacy receive descriptor.<br>(3) Some cases of a missed packet as described in FCERR code 011b.<br>This code should be ignored when FCSTAT equals 00b (meaning no context match). |
| 110 | Filter context valid and DMA context invalid. Indicates that some packet(s) were lost by the DMA context due to lack of legacy receive descriptors or were missed by the Rx packet buffer.<br>*Note:* The software might ignore this error when FCSTAT equals 00b. |
| 111 | Reserved |

FDIRERR (2:0) — This field is relevant for non-FCoE packets when the flow director filters are enabled.

FDIRErr(0) - Length — If the flow director filter matches the *Length* bit, this indicates that the distance of the matched filter from the hash table exceeds the FDIRCTRL.Max-Length. If there is no matched filter, the *Length* bit is set if the flow director linked list of the matched hash value exceeds the FDIRCTRL.Max-Length.

FDIRErr(1) - Drop — The *Drop* bit indicates that a received packet matched a flow director filter with a drop action. In the case of perfect mode filtering, it is expected to find the drop indication only when the linked list in the flow director bucket exceeds the permitted Max-Length. In this case, the packet is not dropped. Instead, it is posted to the Rx queue (indicated in the filter context) for software handling of the Max-Length exception. In the case of hash mode filtering, it is expected that the drop queue is always a valid queue so all packets that match the drop filter are visible to software.

FDIRErr(2) - Coll — A matched flow director filter with a collision indication was found. The collision indicates that software attempted to step over this filter with a different action that was already programmed.

HBO (3) — The *Header Buffer Overflow* bit is set if the packet header (calculated by hardware) is bigger than the header buffer (defined by PSRCTL.BSIZEHEADER). *HBO* reporting might be used by software to allocate bigger buffers for the headers. It is meaningful only if the *SPH* bit in the receive descriptor is set as well. The HDR_LEN field is valid even when the *HBO* bit is set. Packets with HBO error are posted to host memory regardless of the store bad packet setting (FCTRL.SBP). Packet DMA to its buffers when the *HBO* bit is set, depends on the device settings as follows:

| SRRCTL.DESCTYPE | DMA Functionality |
|---|---|
| Header Split (010b) | The header is posted with the rest of the packet data to the packet buffer. |
| Always Split Mode (101b) | The header buffer is used as part of the data buffers and contains the first PSRCTL.BSIZEHEADER bytes of the packet. |

Rsv (5:4) — Reserved at zero.

SECERR (8:7) — Security error indication for LinkSec or IPsec. This field is meaningful only if the SECP bit in the extended status is set.

| SECERR | LinkSec Error Reporting | IPsec Error Reporting |
|---|---|---|
| 00 | No errors found or no security processing | No errors found while an active SA found or no security processing. |
| 01 | No SA match | Invalid IPsec Protocol: IPsec protocol field (*ESP* or *AH*) in the received IP header does not match expected one stored in the SA table. |
| 10 | Replay error | Length error: ESP packet is not 4-bytes aligned or AH/ESP header is truncated (for example, a 28-byte IPv4 packet with IPv4 header + ESP header that contains only SPI and SN) or *AH Length* field in the AH header is different than 0x07 for IPv4 or 0x08 for IPv6 or the entire packet size excluding CRC is shorter than 70 bytes for IPv4 or 90 bytes for IPv6 ESP or 94 bytes for IPv6 AH. |
| 11 | Authentication failed: Bad signature | Authentication failed: Bad signature. |

RXE (9) — RXE is described in the legacy descriptor format in Section 7.1.5.

L4E (10) — L4 integrity error is valid only when the *L4I* bit in the *Status* field is set. It is active if L4 processing fails (TCP checksum or UDP checksum or SCTP CRC). Packets with L4 integrity error are posted to host memory regardless of the store bad packet setting (FCTRL.SBP).

FCEOFe(11) / IPE(11) — This bit has multiplexed functionality. FCoE packets are indicated as such in the Packet Type field in the Rx descriptor.

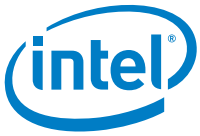
| Non-FCoE Packet | FCoE Packet |
|---|---|
| **IPE** (IPv4 checksum error) is described in Section 7.1.5. | FC EOF Exception (**FCEOFe**). This bit indicates unexpected EOF or SOF flags. The specific error is defined by the FCEOF bit in the extended status previously described. |

PKT_LEN (16-bit offset 32, 2nd line)

PKT_LEN holds the number of bytes posted to the packet buffer. The length covers the data written to a receive buffer including CRC bytes (if any). Software must read multiple descriptors to determine the complete length for packets that span multiple receive buffers. If SRRCTL.DESCTYPE = 2 (advanced descriptor header splitting) and the buffer is not split because the header is bigger than the allocated header buffer, this field reflects the size of the data written to the data buffer (header + data).

VLAN Tag (16-bit offset 48, 2nd line)

This field is described in the legacy descriptor format in Section 7.1.5.



## 7.1.7 Receive Descriptor Fetching

### 7.1.7.1 Fetch On Demand

The 82599 implements a fetch-by-demand mechanism for descriptor fetch. Descriptors are not fetched in advance, but rather fetched after a packet is received. Such a strategy eliminates the need to store descriptors on-die for each and every descriptor queue in anticipation for packet arrival.

## 7.1.8 Receive Descriptor Write-Back

The 82599 writes back the receive descriptor immediately following the packet write into system memory. It is therefore possible for a single descriptor to be written at a time into memory. However, if aggregation occurs during descriptor fetch (see Section 7.1.7), then the descriptors fetched in the aggregated operation are written back in a single write-back operation. In Receive Coalescing (RSC), all the descriptors except the last one are written back when they are completed. This does not have to be on packet boundaries but rather when the next descriptor of the same RSC is fetched. See Section 7.11.5.1 for more on RSC.

**Note:** Software can determine if a packet has been received by checking the receive descriptor *DD* bit in memory or by checking the value of the receive head pointer in the RDH/RDL registers. Checking through *DD* bits 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 82599.

## 7.1.9 Receive Descriptor Queue Structure

Figure 7-11 shows the structure of each of the receive descriptor rings. Note that each ring uses a contiguous memory space.

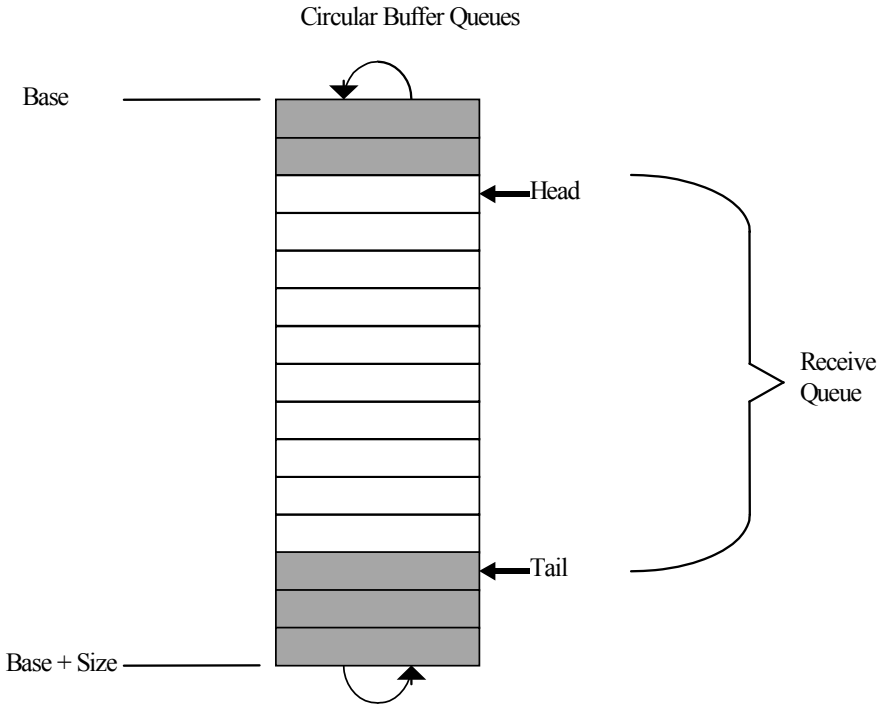



**Figure 7-11 Receive Descriptor Ring Structure**

Software inserts receive descriptors by advancing the tail pointer(s) to refer to the address of the entry just beyond the last valid descriptor. This is accomplished by writing the descriptor tail register(s) with the offset of the entry beyond the last valid descriptor. The 82599 adjusts its internal tail pointer(s) accordingly. As packets arrive, they are stored in memory and the internal head pointer(s) is incremented by the 82599.

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 82599 stops storing packets in system memory until software advances the tail pointer(s), making more receive buffers available.

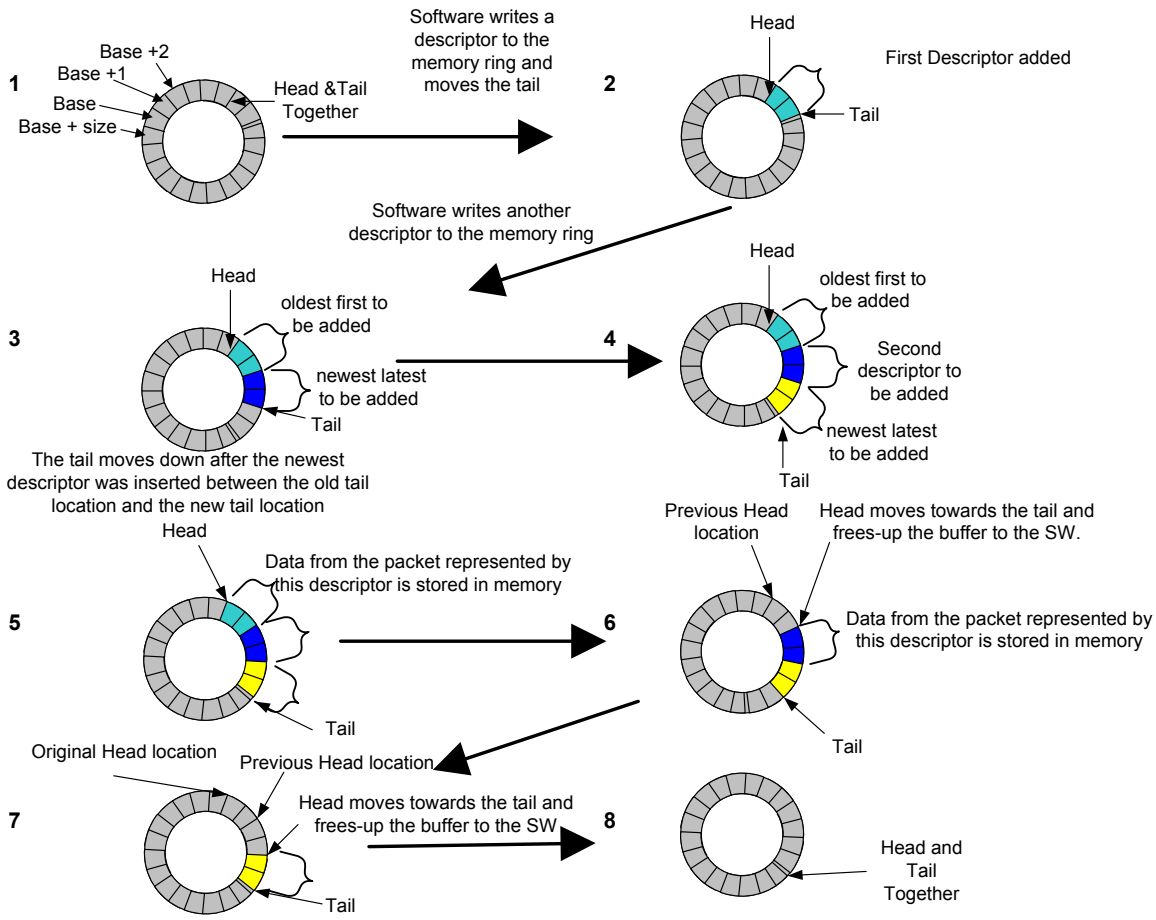



**Figure 7-12 Descriptors and Memory Rings**

The 82599 writes back used descriptors just prior to advancing the head pointer(s). Head and tail pointers wrap back to base when the number of descriptors corresponding to the size of the descriptor ring have been processed.

The receive descriptor head and tail pointers reference to 16-byte blocks of memory. Shaded boxes in Figure 7-12 represent descriptors that have stored incoming packets but have not yet been recognized by software. Software can determine if a receive buffer is valid by reading descriptors in memory rather than by I/O reads. Any descriptor with a *DD* bit set has been used by the hardware, and is ready to be processed by software.

**Note:** The head pointer points to the next descriptor that is to be written back. At the completion of the descriptor write-back operation, this pointer is incremented by the number of descriptors written back. Hardware owns all descriptors between [head... tail]. Any descriptor not in this range is owned by software.

The receive descriptor rings are described by the following registers:

- Receive Descriptor Base Address registers (RDBA) — This register indicates the start of the descriptor ring buffer; this 64-bit address is aligned on a 16-byte boundary and is stored in two consecutive 32-bit registers. Hardware ignores the lower 4 bits.
- Receive Descriptor Length registers (RDLEN) — This register determines the number of bytes allocated to the circular buffer. This value must be a multiple of 128 (the maximum cache line size). Since each descriptor is 16 bytes in length, the total number of receive descriptors is always a multiple of 8.
- Receive Descriptor Head registers (RDH) — This register holds a value that is an offset from the base, and indicates the in-progress descriptor. There can be up to 64K-8 descriptors in the circular buffer. Hardware maintains a shadow copy that includes those descriptors completed but not yet stored in memory.

  Software can determine if a packet has been received by either of two methods: reading the *DD* bit in the receive descriptor field or by performing a PIO read of the Receive Descriptor Head register. Checking the descriptor *DD* bit in memory eliminates a potential race condition. All descriptor data is written to the I/O bus prior to incrementing the head register, but a read of the head register could pass the data write in systems performing I/O write buffering. Updates to receive descriptors use the same I/O write path and follow all data writes. Consequently, they are not subject to the race.
- Receive Descriptor Tail registers (RDT) — This register holds a value that is an offset from the base, and identifies the location beyond the last descriptor hardware can process. This is the location where software writes the first new descriptor.

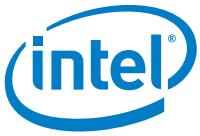
If software statically allocates buffers, and uses a memory read to check for completed descriptors, it simply has to zero the status byte in the descriptor to make it ready for re-use by hardware. This is not a hardware requirement, but is necessary for performing an in-memory scan. This is relevant only to legacy descriptors.

All the registers controlling the descriptor rings behavior should be set before receive is enabled, apart from the tail registers which are used during the regular flow of data.

#### 7.1.9.1 Low Receive Descriptors Threshold

As previously described, the size of the receive queues is measured by the number of receive descriptors. During run time, software processes descriptors and upon completion of descriptors, increments the Receive Descriptor Tail registers. At the same time, the hardware may post new received packets incrementing the Receive Descriptor Head registers for each used descriptor.

The number of usable (free) descriptors for the hardware is the distance between the Tail and Head registers. When the tail reaches the head, there are no free descriptors and further packets might be either dropped or block the receive FIFO. In order to avoid this situation, the 82599 might generate a low latency interrupt (associated to the relevant Rx queue) once there are less equal free descriptors than specified by a low level threshold. The threshold is defined in 64 descriptors granularity per queue in the SRRCTL[n].RDMTS field.



## 7.1.10 Header Splitting

**Note:** Header Splitting mode might cause unpredictable behavior and should not be used with the 82599. For more information, see the product specification update errata on this subject.

### 7.1.10.1 Purpose

This feature consists of splitting a packet header to a different memory space. This helps the host to fetch headers only for processing: headers are posted through a regular snoop transaction in order to be processed by the host CPU. It is recommended to perform this transaction with DCA enabled (see Section 7.5).

The packet (header + payload) is stored in memory. Later, an IOAT transaction moves the payload from the driver space to the application memory.

The 82599'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. When using a split always descriptor type, the header buffer size (BSIZEHEADER) should be set to four which equals to 256 bytes.

The 82599 uses packet splitting when the SRRCTL[n].DESCTYPE is greater than one.

## 7.1.10.2 Description

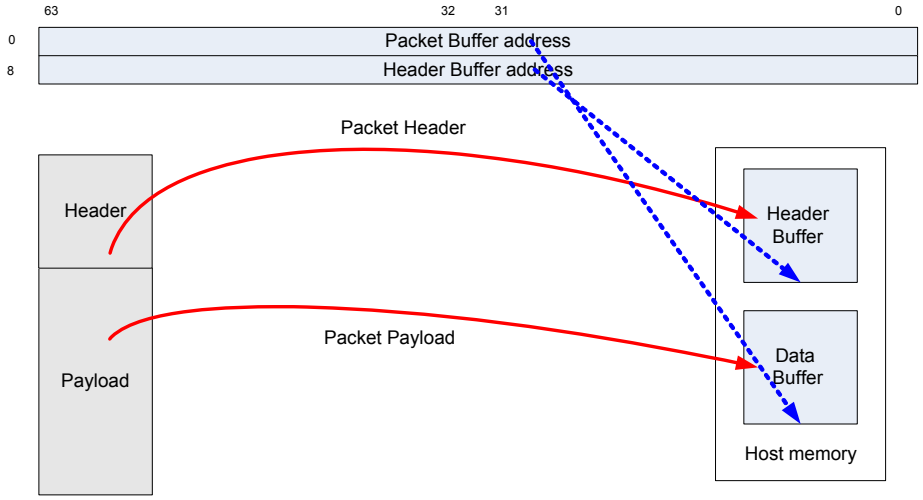



**Figure 7-13 Header Splitting Diagram**

The physical address of each buffer is written in the *Buffer Addresses* fields:

- The packet buffer address includes the address of the buffer assigned to the packet data.
- The header buffer address includes the address of the buffer that contains the header information. The receive DMA module stores the header portion of the received packets into this buffer.

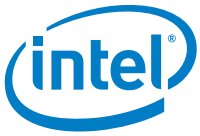
The sizes of these buffers are statically defined in the SRRCTL[n] registers:

- The *BSIZEPACKET* field defines the size of the buffer for the received packet.
- The *BSIZEHEADER* field defines the size of the buffer for the received header. If header split is enabled, this field must be configured to a non-zero value. The 82599 only writes the header portion into the header buffer. The header size is determined by the options enabled in the PSRTYPE registers.

When header split is selected, the packet is split only on selected types of packets. A bit exists for each option in PSRTYPE[n] registers, so several options can be used in conjunction. If one or more bits are set, the splitting is performed for the corresponding packet type. See Section 8.2.3.7.4 for details on the possible header types supported. In virtualization mode, a separate PSRTYPE register is provided per pool up to the number of pools enabled. In non-virtualization mode, only PSRTYPE[0] is used.

Rules regarding header split:

- Packets that have headers bigger than 1023 bytes are not split.
- The header of a fragmented IPv6 packet is defined until the fragmented extension header.
- Header split must not be used in a queue used for a FCoE large receive.
- An IP packet with more than a single IP header (such as any combination of IPv4 and IPv6 tunneling) is not split.



- Packet header cannot span across buffers, therefore, the size of the header buffer must be larger than any expected header size. In case of header split mode (SRRCTL.DESCTYPE = 010b), a packet with a header larger than the header buffer is not split.
- If an IPsec header is present in the receive packet, the following rules apply:
  - IPsec packets handled in the 82599 always include IPsec header in a split done at IP boundary.
  - IPsec packets handled in software must never do header split.

Table 7-20 lists the behavior of the 82599 in the different modes.

### Table 7-20 Behavior in Header Split Modes

| DESCTYPE | Condition | SPH | HBO | PKT_LEN | HDR_LEN | Header and Payload DMA |
|---|---|---|---|---|---|---|
| Split | 1. Header can't be decoded | 0 | 0 | Min(packet length, buffer size) | 0x0 | Header + Payload --> Packet Buffer |
| | 2. Header <= BSIZEHEADER | 1 | 0 | Min (payload length, buffer size)[3] | Header size | Header --> Header Buffer<br>Payload --> Packet Buffer |
| | 3. Header > BSIZEHEADER | 1 | 1 | Min (packet length, buffer size) | Header size[5] | Header + Payload --> Packet Buffer |
| Split – always use header buffer | 1. Header can't be decoded and packet length <= BSIZEHEADER | 0 | 0 | 0x0 | Packet length | Header + Payload --> Header Buffer |
| | 2. Header can’t be decoded and packet length > BSIZEHEADER | 0 | 0 | Min (packet length – BSIZEHEADER, data buffer size) | Undefined | Header + Payload --> Header + Packet Buffers[4] |
| | 3. Header <= BSIZEHEADER | 1 | 0 | Min (payload length, data buffer size) | Header Size | Header --> Header Buffer<br>Payload --> Packet Buffer |
| | 4. Header > BSIZEHEADER | 1 | 1 | Min (packet length – BSIZEHEADER, data buffer size) | Header Size[5] | Header + Payload --> Header + Packet Buffer |

*Notes:*

1. Partial means up to BSIZEHEADER.
2. HBO is set to 1b if the header size is bigger than BSIZEHEADER and zero otherwise.
3. In a header only packet (such as TCP ACK packet), the PKT_LEN is zero.
4. If the packet spans more than one descriptor, only the header buffer of the first descriptor is used.
5. HDR_LEN doesn't reflect the actual data size stored in the header buffer. It reflects the header size determined by the parser.

## 7.1.11 Receive Checksum Offloading

The 82599 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 82599. The 82599 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 82599 calculates the TCP or UDP checksum and indicates a pass/fail condition to software via the *TCP/UDP Checksum Error* bit (RDESC.TCPE). These error bits are valid when the respective status bits indicate the checksum was calculated for the packet (RDESC.IPCS and RDESC.L4CS, respectively).

Similarly, if RFCTL.Ipv6_DIS and RFCTL.IP6Xsum_DIS are cleared to zero, the 82599 calculates the TCP or UDP checksum for IPv6 packets. It then indicates a pass/fail condition in the *TCP/UDP Checksum Error* bit (RDESC.TCPE).

Supported frame types:

- Ethernet II
- Ethernet SNAP

**Table 7-21 Supported Receive Checksum Capabilities**

| Packet Type | Hardware IP Checksum Calculation | Hardware TCP/UDP Checksum Calculation |
|---|---|---|
| IP header's protocol field contains a protocol # other than TCP or UDP. | Yes | No |
| IPv4 + TCP/UDP packets. | Yes | Yes |
| IPv6 + TCP/UDP packets. | No (N/A) | Yes |
| IPv4 packet has IP options (IP header is longer than 20 bytes). | Yes | Yes |
| IPv6 packet with next header options:<br>• Hop-by-hop options.<br>• Destinations options (without home address).<br>• Destinations options (with home address).<br>• Routing (with segment left 0).<br>• Routing (with segment left > 0).<br>• Fragment. | <br>No (N/A)<br>No (N/A)<br>No (N/A)<br>No (N/A)<br>No (N/A)<br>No (N/A) | <br>Yes<br>Yes<br>No<br>Yes<br>No<br>No |
| Packet has TCP or UDP options. | Yes | Yes |
| Ipv4 tunnels:<br>• IPv4 packet in an IPv4 tunnel.<br>• IPv6 packet in an IPv4 tunnel. | <br>No<br>Yes (IPv4) | <br>No<br>No |
| IPv6 tunnels:<br>• IPv4 packet in an IPv6 tunnel.<br>• IPv6 packet in an IPv6 tunnel. | <br>No<br>No | <br>No<br>No |
| Packet is an IPv4 fragment. | Yes | UDP checksum assist |

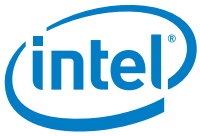


**Table 7-21 Supported Receive Checksum Capabilities (Continued)**

| Packet Type | Hardware IP Checksum Calculation | Hardware TCP/UDP Checksum Calculation |
|---|---|---|
| Packet is greater than 1522 bytes. | Yes | Yes |
| Packet has 802.3ac tag. | Yes | Yes |

The previous table lists general details about what packets are processed. In more detail, the packets are passed through a series of filters to determine if a receive checksum is calculated.

### Ethernet MAC Address Filter

This filter checks the MAC destination address to be sure it is valid (that is IA match, broadcast, multicast, etc.). The receive configuration settings determine which Ethernet MAC addresses are accepted. See the various receive control configuration registers such as FCTRL, MCSTCTRL (RTCL.UPE, MCSTCTRL.MPE, FCTRL.BAM), MTA, RAL, and RAH for details.

### SNAP/VLAN Filter

This filter checks the next headers looking for an IP header. It is capable of decoding Ethernet II, Ethernet SNAP, and IEEE 802.3ac headers. It skips past any of these intermediate headers and looks for the IP header. The receive configuration settings determine which next headers are accepted. See the various receive control configuration registers such as VLNCTRL.VFE, VLNCTRL.VET, and VFTA for more details.

### IPv4 Filter

This filter checks for valid IPv4 headers. The version field is checked for a correct value (4).

IPv4 headers are accepted if they are any size greater than or equal to five (Dwords). If the IPv4 header is properly decoded, the IP checksum is checked for validity.

### IPv6 Filter

This filter checks for valid IPv6 headers, which are a fixed size and have no checksum. The IPv6 extension headers accepted are: Hop-by-hop, destination options, and routing. The maximum extension header size supported is 256 bytes. The maximum total header size supported is 1 KB.

### IPv6 Extension Headers

IPv4 and TCP provide header lengths, which enable hardware to easily navigate through these headers on packet reception for calculating checksum and CRC, etc. For receiving IPv6 packets, however, there is no IP header length to help hardware find the packet's ULP (such as TCP or UDP) header. One or more IPv6 extension headers might exist in a packet between the basic IPv6 header and the ULP header. Hardware must skip over these extension headers to calculate the TCP or UDP checksum for received packets.

The IPv6 header length without extensions is 40 bytes. The IPv6 field *Next Header Type* indicates what type of header follows the IPv6 header at offset 40. It might be an upper layer protocol header such as TCP or UDP (next header type of 6 or 17, respectively), or it might indicate that an extension header follows. The final extension header indicates with it's *Next Header Type* field the type of ULP header for the packet.

IPv6 extension headers have a specified order. However, destinations must be able to process these headers in any order. Also, IPv6 (or IPv4) might be tunneled using IPv6, and thus another IPv6 (or IPv4) header and potentially its extension headers might be found after the extension headers.

The IPv4 next header type is at byte offset 9. In IPv6, the first next header type is at byte offset 6.

All IPv6 extension headers have the next header type in their first eight bits. Most have the length in the second eight bits (Offset Byte[1]) as follows:

### Table 7-22 Typical IPv6 Extended Header Format (Traditional Representation)

| 1<br>0 1 2 3 4 5 6 7 8 9 0 | 2<br>0 1 2 3 4 5 6 7 8 9 0 | 3<br>0 1 2 3 4 5 6 7 8 9 0 |
|---|---|---|
| Next Header Type | Length | |
| | | |

Table 7-23 lists the encoding of the *Next Header Type* field and information on determining each header type's length. Other IPv6 extension headers - not indicated in Table 7-23 - are not recognized by the 82599. Any processing of packet content that follows such extension headers is not supported.

### Table 7-23 Header Type Encoding and Lengths

| Header | Next Header Type | Header Length (units are bytes unless otherwise specified) |
|---|---|---|
| IPv6 | 6 | Always 40 bytes |
| IPv4 | 4 | Offset bits[7:4]<br>Unit = 4 bytes |
| TCP | 6 | Offset Byte[12].Bits[7:4]<br>Unit = 4 bytes |
| UDP | 17 | Always 8 bytes |
| Hop-by-Hop Options | 0 - Note 1 | 8+Offset Byte[1] |
| Destination Options | 60 | 8+Offset Byte (note 1) |
| Routing | 43 | 8+Offset Byte (note 1) |
| Fragment | 44 | Always 8 bytes |
| Authentication | 51 | Note 3 |

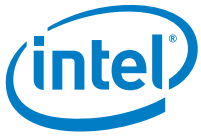


**Table 7-23 Header Type Encoding and Lengths (Continued)**

| Header | Next Header Type | Header Length (units are bytes unless otherwise specified) |
|---|---|---|
| Encapsulating Security Payload | 50 | Note 3 |
| No Next Header | 59 | Note 2 |

*Notes:*

1. Hop-by-hop options header is only found in the first next header type of an IPv6 header.
2. When no next header type is found, the rest of the packet should not be processed.
3. Encapsulated security payload packet handled by software — The 82599 cannot offload packets with this header type.

### UDP/TCP Filter

This filter checks for a valid UDP or TCP header. The prototype next header values are 0x11 and 0x06, respectively.

## 7.1.12 SCTP Receive Offload

If a receive packet is identified as SCTP, the 82599 checks the CRC32 checksum of this packet and identifies this packet as SCTP. Software is notified of the CRC check via the L4I and L4E bits in the *Extended Status* field and *Extended Error* field in the Rx descriptor. The detection of an SCTP packet is indicated via the *SCTP* bit in the *Packet Type* field of the Rx descriptor. SCTP CRC uses the CRC32c polynomial as follows (0x11EDC6F41):

$$X_{32}+X_{28}+X_{27}+X_{26}+X_{25}+X_{23}+X_{22}+X_{20}+X_{19}+X_{18}+X_{14}+X_{13}+X_{11}+X_{10}+X_{9}+X_{8}+X_{6}+X_{0}$$

The checker assumes the following SCTP packet format.

**Table 7-24 SCTP Header**

| 0 1 2 3 4 5 6 7 | 1<br>8 9 0 1 2 3 4 5 | 2<br>6 7 8 9 0 1 2 3 | 3<br>4 5 6 7 8 9 0 1 |
|---|---|---|---|
| Source Port | | Destination Port | |
| Verification Tag | | | |
| CRC Checksum (CRC32c) | | | |
| Chunks 1..n | | | |

## 7.1.13 Receive UDP Fragmentation Checksum

The 82599 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 | 0 | 0 | Depends on transport<br>UDP: 1 / 1<br>TCP: 0 / 1 |
| Non fragmented IPv4 packet | | | |
| Fragmented IPv4 with protocol = UDP, first fragment (UDP protocol present) | The unadjusted 1's complement checksum of the IP payload | 1 if the UDP header checksum is valid (not 0) | 1 / 0 |
| Fragmented IPv4, when not first fragment | The unadjusted 1's complement checksum of the IP payload | 0 | 1 / 0 |

When the driver computes the 16-bit ones complement sum on the incoming packets of the UDP fragments, it should expect a value of 0xFFFF. Refer to Section 7.1.2.8.3 for supported packet formats.

# 7.2 Transmit Functionality

## 7.2.1 Packet Transmission

Transmit packets are made up of data buffers in host memory that are indicated to hardware by pointer and length pairs. These pointer and length pairs are named as transmit descriptors that are stored in host memory as well.

Software prepares memory structures for transmission by assembling a list of descriptors. It then indicates this list to hardware for updating the on-chip transmit tail pointer. Hardware transmits the packet only after it has completely fetched all packet data from host memory and deposited it into the on-chip transmit FIFO. This store and forward scheme enables hardware-based offloads such as TCP or UDP checksum computation, and many other ones detailed in this document while avoiding any potential PCIe under-runs.

### 7.2.1.1 Transmit Storage in System Memory

A packet (or multiple packets in transmit segmentation) can be composed of one or multiple buffers. Each buffer is indicated by a descriptor. Descriptors of a single packet are consecutive, while the first one points to the first buffer and the last one points to the last buffer (see Figure 7-14). 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 7.2.3.3 for Tx Ring details and section Section 7.2.3.5.1 for WTHRESH details). For best performance it is recommended to minimize the number of buffers as possible.

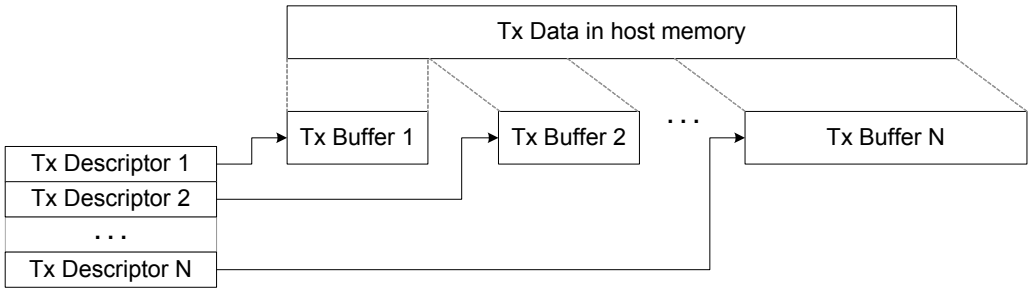



**Figure 7-14 Tx Packet in Host Memory**

### 7.2.1.2 Transmit Path in the 82599

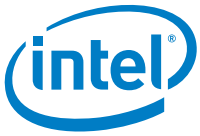
The transmit path in the 82599 consists of the following stages:

- Descriptor plane
  - The 82599 maintains a set of 128 on-die descriptor queues. Each queue is associated with a single descriptor ring in system memory. See Section 7.2.3.3 for more details on the Tx descriptor rings. Each on-die descriptor queue contains up to 40 descriptors.
  - A fetch mechanism loads Tx descriptors from the descriptor rings in system memory to the respective descriptor queues in the 82599. A descriptor fetch arbiter determines the order in which descriptors are fetched into the various on-die descriptor queues. See Section 7.2.3.4 for more details on the fetch mechanism.
  - An arbitration scheme determines the order in which descriptors are processed and requests are generated for data reads. These requests load packet data from system memory into a set of Tx packet buffers. The arbitration mechanism varies with configuration and is shown in Figure 7-17.
  - Once a packet has been fetched into a packet buffer, status is (optionally) written back into system memory. See Section 7.2.3.5 for more details.
- Packet plane (data plane)
  - Packet data is stored in up to eight packet buffers. The number and size of packet buffers vary with the mode of operation and is described in Section 7.2.1.2.2.
  - If more than a single packet buffer is enabled, an arbitration scheme determines the order in which packets are taken out of the packet buffers and sent to the MAC for transmission.The arbitration mechanism is shown in Figure 7-17.

#### 7.2.1.2.1 Tx Queues Assignment

The 82599 supports a total of 128 queues per LAN port. Each Tx queue is associated with a packet buffer and the association varies with the operational mode. The following mechanisms impact the association of the Tx queues. These are described briefly in this section, and in full details in separate sections:

- Virtualization (VT) - In a virtualized environment, DMA resources are shared between more than one software entity (operating system and/or device driver). This is done through allocation of transmit descriptor queues to virtual partitions (VMM, IOVM, VMs, or VFs). Allocation of queues to virtual partitions is done in sets of queues of the same size, called queue pools, or **pools**. A pool is associated with a single virtual partition. Different queues in a pool can be associated with different packet buffers. For example, in a DCB system, each of the queues in a pool might belong to a different TC and therefore to a different packet buffer. The PFVFTE register contains a bit per VF. When the bit is set to 0b, packet transmission from the VF is disabled. When set to 1b, packet transmission from the VF is enabled.
- DCB — DCB provides QoS through priority queues, priority flow control, and congestion management. Queues are classified into one of several (up to eight) Traffic Classes (TCs). Each TC is associated with a single unique packet buffer.



- Transmit fanout — A single descriptor queue might be enough for a given functionality. For example, in a VT system, a single Tx queue can be allocated per VM. However, it is often the case that the data rate achieved through a single buffer is limited. This is especially true with 10 GbE, and traffic needs to be divided into several Tx queues in order to reach the desired data rate. Therefore, multiple queues might be provided for the same functionality.

Table 7-25 lists the queuing schemes. Selection of a scheme is done via the MTQC register.

### Table 7-25 Tx Queuing Schemes

| VT | DCB | Queues Allocation | Packet Buffers allocation |
|---|---|---|---|
| No | No | A single set of 64 queues is assigned to a single packet buffer. Queues 64…127 should not be used. | A single packet buffer for all traffic |
| No | Yes | Eight TCs mode – allocation of 32-32-16-16-8-8-8-8 queues for TC0-TC1-…- TC7, respectively.<br>Four TCs mode — allocation of 64-32-16-16 queues for TC0-TC1-…- TC3, respectively. | A separate packet buffer is allocated to each TC (total of four or eight). |
| Yes | No | 32 pools x 4 queues, or<br>64 pools x 2 queues | A single packet buffer for all traffic. |
| Yes | Yes | 16 pools x 8 TCs, or<br>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.

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.

Allocating descriptor queues to VFs uses a consistent indexing over the different Tx queuing schemes. The most significant bits of the queue index represent the VF index, and the least significant bits are either the TC index or are used by software to dispatch traffic according to a Transmit Side Scaling (TSS) algorithm — similar to RSS in the Rx path.

The Tx queue numbers associated with the TCs are listed in the following tables: Table 7-26 and Table 7-27.

### Table 7-26 Tx Queues Indexing When VT-on

| VT mode | Allocation of queue index bits according to | | | | | | |
|---|---|---|---|---|---|---|---|
| | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **64 VFs + TSS** | VF (63..0) | | | | | | TSS |
| **32 VFs + TSS *or* 4 TCs** | VF (31 ..0) | | | | | TSS / TC | |
| **16 VFs + 8 TCs** | VF (15 ..0) | | | | TC | | |

**Table 7-27 Tx Queues Indexing When VT-off and DCB-on**

| TC mode | TCn | # of Qs | Queues attached to TCn |
|---|---|---|---|
| 4 TCs | TC0 | 64 | 0xxxxxx |
| | TC1 | 32 | 10xxxxx |
| | TC2 | 16 | 110xxxx |
| | TC3 | 16 | 111xxxx |
| 8 TCs | TC0 | 32 | 00xxxxx |
| | TC1 | 32 | 01xxxxx |
| | TC2 | 16 | 100xxxx |
| | TC3 | 16 | 101xxxx |
| | TC4 | 8 | 1100xxx |
| | TC5 | 8 | 1101xxx |
| | TC6 | 8 | 1110xxx |
| | TC7 | 8 | 1111xxx |

*Note:* "x" refers to both 0 or 1, and is used by software to dispatch Tx flows via TSS algorithm.

### 7.2.1.2.2 Tx Packet Buffers

As previously described, the following modes exist for the 82599 packet buffers:

- A single 160 KB packet buffer that serves all Tx descriptor queues, leading to one single (or no) TC enabled, TC0
- Four 40 KB packet buffers, one per enabled TC, leading to four TCs, TC0 to TC3
- Eight 20 KB packet buffers, one per enabled TC, leading to eight TCs, TC0 to TC7

The size of the Tx packet buffer(s) is programmed via the TXPBSIZE registers, one register per TC. Null-sized packet buffer corresponds to a disabled TC.

**Note:** Setting the packet buffers' size leads to a different partition of a shared internal memory and must be done during boot, prior to communicating, and followed by a software reset.

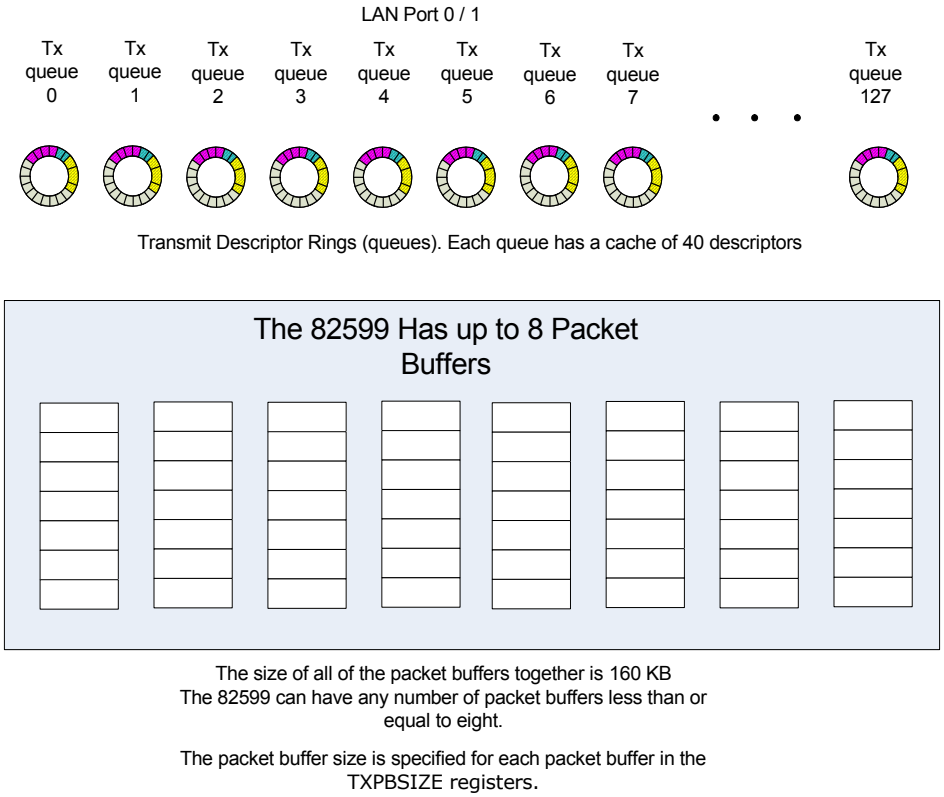



**Figure 7-15 Tx Arbitration Schemes**

## 7.2.1.2.3 Tx Arbitration Schemes

There are basically four Tx arbitration schemes, one per each combination of the DCB and Virtualization (VT) enabled/disabled modes. They are configured via the MTQC.MTQE register field.

DCB-on/VT-on

When both DCB and virtualization are enabled, queues are allocated to the packet buffers in a fixed manner, the same number of queues per each TC. Two DCB modes are supported, four TCs or eight TCs mode, according to coherent configuration made in registers TXPBSIZE and MTQC.

Descriptor Plane Arbiters and Schedulers:

- Transmit Rate-Scheduler — Once a frame has been fetched out from a transmit rate-limited queue, the next time another frame could be fetched from that queue is regulated by the transmit 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 rate limitation as previously described. Weights or credits allocated to each queue are configured via the RTTDT1C register. Bandwidth unused by one queue is reallocated to the other queues within the TC, proportionally to their relative bandwidth shares. TC bandwidth limitation is distributed across all the queues attached to the TC, proportionally to their relative bandwidth shares. Details on weighted round-robin arbiter between the queues can be found in Section 7.7.2.3. It is assumed traffic is dispatched across the queues attached to a same TC in a straightforward manner, according to the VF to which it belongs.
- TC Weighted Strict Priority Arbiter — Descriptors are fetched out from queues attached to different TCs in a frame-by-frame weighted strict-priority manner. Bandwidth unused by one TC is reallocated to the others, proportionally to their relative bandwidth shares. Link bandwidth limitation is distributed across all the TCs, proportionally to their relative bandwidth shares. Details on weighted strict-priority arbiter between the TCs can be found at Section 7.7.2.3. It is assumed (each) driver dispatches traffic across the TCs according to the 802.1p User Priority field inserted by the operating system and according to a user priority-to-TC Tx mapping table.

Packet Plane Arbiters:

- TC Weighted Strict Priority Arbiter — Packets are fetched out from the different packet buffers in a frame-by-frame weighted strict-priority manner. Weights or credits allocated to each TC (such as to each packet buffer) are configured via RTTPT2C registers, with the same allocation done at the descriptor plane. Bandwidth unused by one TC and link bandwidth limitation is distributed over the TCs as in the descriptor plane. Details on weighted strict-priority arbiter between the TCs can be found in Section 7.7.2.3.
- Priority Flow Control Packets are inserted with strict priority over any other packets.
- Manageability Packets are inserted with strict priority over data packets from the same TC, with respect to the bandwidth allocated to the concerned TC. TCs that belong to manageability packets are controlled by MNGTXMAP.MAP.

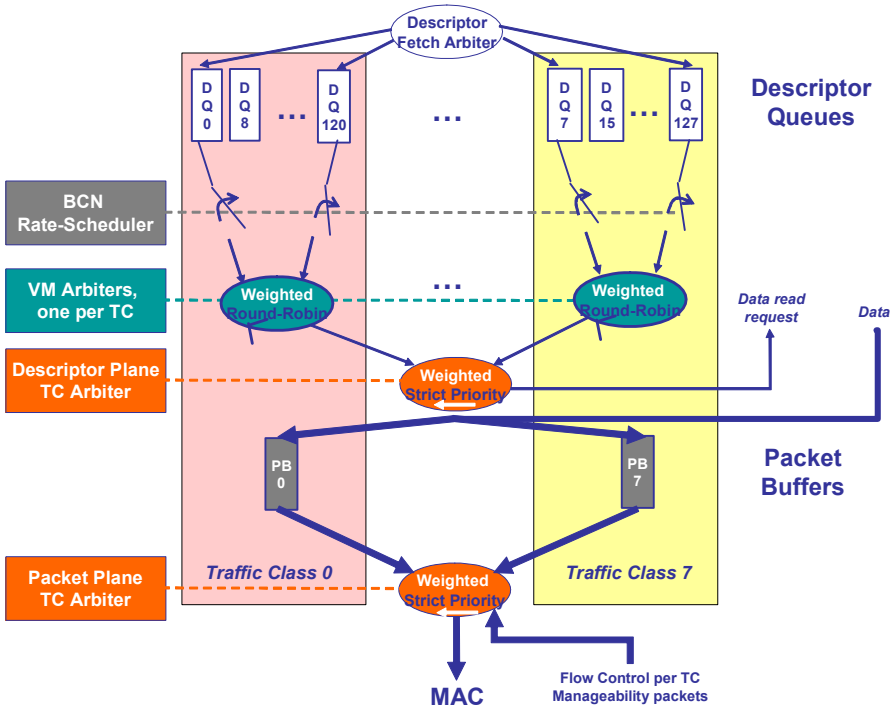



**Figure 7-16 Transmit Architecture DCB-on/VT-on — Eight TCs Mode**

**Note:** Replication of TC arbiters before and after the packet buffers is required to provide arbitration whether PCI bandwidth is smaller or greater than the link bandwidth, respectively.

### DCB-on/VT-off

When DCB is enabled and virtualization disabled, queues are allocated to the packet buffers in a fixed manner according to the number of TCs. Two DCB modes are supported, four TCs or eight TCs mode, according to coherent configuration made in registers TXPBSIZE and MTQC. In Figure 7-17, eight TCs mode is shown.

- The unique difference with the DCB-on/VT-on arbitration scheme previously described is that the VM weighted round-robin arbiters are degenerated into simple frame-by-frame round-robin arbiters across the queues attached to the same TC. It is assumed driver dispatches traffic across the queues attached to a same TC according to hashing performed on MAC destination addresses. This is aimed to minimize crosstalk between transmit rate-limited and non-rate-limited flows.

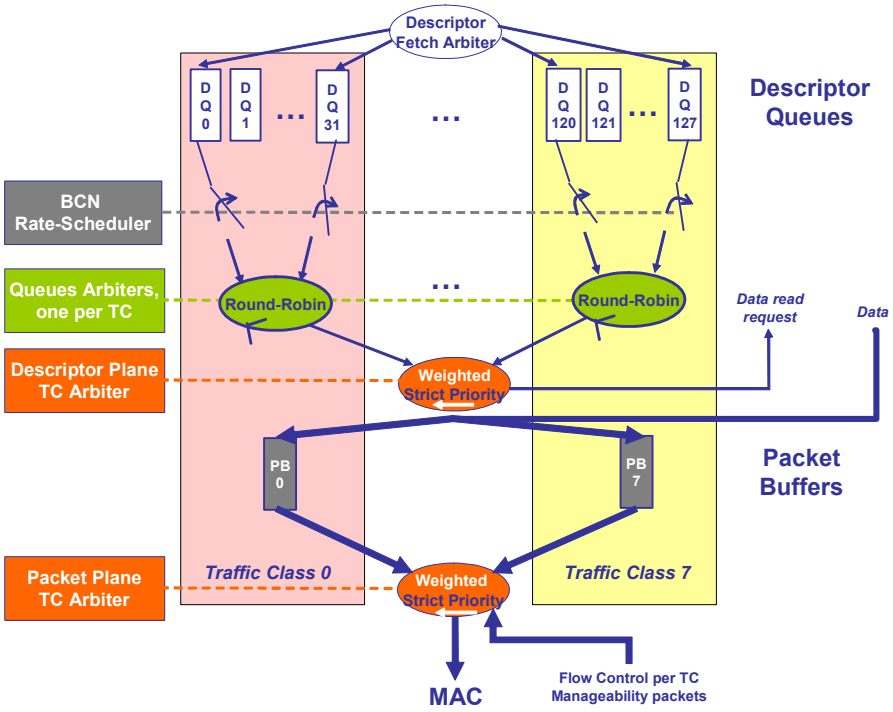



**Figure 7-17 Transmit Architecture DCB-on/VT-off — Eight TCs Mode**

### DCB-off/VT-on

When DCB is disabled and virtualization enabled, all the 128 queues are allocated to a single packet buffer PB(0). Queues are grouped into 32 or 64 pools of 4 or 2 queues, respectively. The number of queue pools corresponds to the number of VFs exposed. Queues are attached to pools according to consecutive indexes

— For the 32 pools case, queues 0, 1, 2, 3 are attached to VF0, queues 4, 5, 6, 7 are attached to VF1, and so forth up to VF31.

— For the 64 pools case, queues 0 and 1 are attached to VF0, queues 2 and 3 are attached to VF1, and so forth up to VF63.

### Descriptor Plane Arbiters:

- Descriptor Queues Round Robin Arbiter — Descriptors are fetched out from the internal descriptor queues attached to the same pool in a frame-by-frame round-robin manner. It is assumed driver dispatches traffic across the queues of a same pool according to some Transmit Side Scaling (TSS) algorithm similarly to what is done by hardware in the Rx path with RSS.
- VM Weighted Round Robin Arbiter — Descriptors are fetched out from queues attached to different pools in a frame-by-frame weighted round-robin manner. Weights or credits allocated to a pool are those allocated for the lowest queue of the pool via the RTTDT1C register. Bandwidth unused by one pool is reallocated to the others proportionally to their relative bandwidth shares. Link bandwidth limitation is distributed across all the pools, proportionally to their relative bandwidth shares.

Details on weighted round-robin arbiter between the pools can be found in Section 7.7.2.3.

Packet Plane Arbiter:

- Manageability packets are inserted with strict priority over data packets.

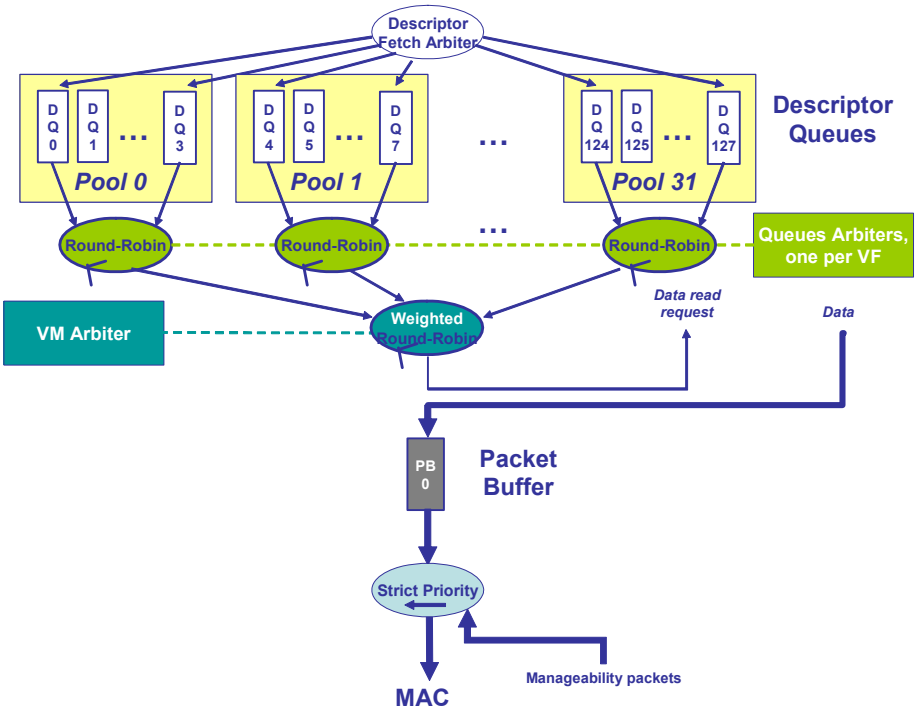



**Figure 7-18 Transmit Architecture DCB-off/VT-on — 32 VFs**

When both DCB and virtualization features are disabled, a single set of up to 64 queues is allocated to a single packet buffer PB(0).

Descriptor Plane Arbiter:

- Descriptor Queues Round Robin Arbiter — Descriptors are fetched out from the internal descriptor queues in a frame-by-frame round-robin manner. It is assumed driver dispatches traffic across the queues according to some TSS algorithm similarly to what is done by hardware in the Rx path with RSS.

Packet Plane Arbiter:

- Manageability packets are inserted with strict priority over data packets.

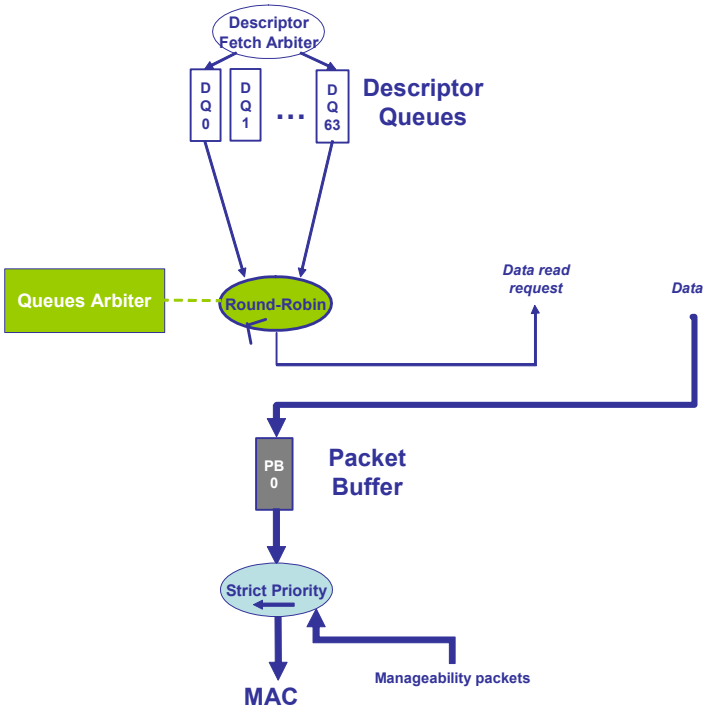



**Figure 7-19 Transmit Architecture DCB-off/VT-off**

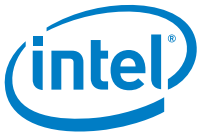
## 7.2.2 Transmit Contexts

The 82599 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 82599 supports two contexts for each of its 128 transmit queues. The *IDX* bit contains an index to one of these two contexts. Each advanced data descriptor that uses any of the advanced offloading features must refer to a context by the *IDX* field.

Contexts can be initialized with a transmit context descriptor and then used for a series of related transmit data descriptors. Software can use these contexts as long lived ones, while one of the two contexts is used for checksum offload and the other one for transmit segmentation detailed in the following sections. The contexts should be modified when new offload parameters are required.



# 7.2.3 Transmit Descriptors

## 7.2.3.1 Introduction

The 82599 supports legacy descriptors and advanced descriptors.

Legacy descriptors are intended to support legacy drivers, in order to enable fast platform power up and to facilitate debug. The legacy descriptors are recognized as such based on *DEXT* bit (see the sections that follow). Legacy descriptors are not supported together with DCB, virtualization, Rate Scheduler, MACSec, and IPsec. These modes are recognized by a dedicated enable bit for each.

In addition, the 82599 supports two types of advanced transmit descriptors:

1. Advanced transmit context descriptor, DTYP = 0010b
2. Advanced transmit data descriptor, DTYP = 0011b

**Note:** DTYP = 0000b and 0001b are reserved values.

The transmit data descriptor (both legacy and advanced) points to a block of packet data to be transmitted. The advanced transmit context descriptor does not point to packet data. It contains control/context information that is loaded into on-chip registers that affect the processing of packets for transmission. The following sections describe the descriptor formats.

## 7.2.3.2 Transmit Descriptors Formats

### 7.2.3.2.1 Notations

This section defines the structure of descriptors that contain fields carried over the network. At the moment, the only relevant field is the *VLAN Tag* field.

The rule for VLAN tag is to use network ordering (also called big endian). It appears in the following manner in the descriptor:

**Table 7-28 VLAN Tag**

| Byte address N + 1 -> first byte on the wire<br>Bit 7 — first on the wire <- Bit 0 | | | Byte address N -> second byte on the wire<br>Bit 7 -> last on the wire — Bit 0 |
|---|---|---|---|
| PRI (3 bits) | CFI | VID (4 bits) | VID (8 bits) |

### 7.2.3.2.2 Legacy Transmit Descriptor Format

To select legacy mode operation, bit 29 (TDESC.DEXT) should be set to 0b. In this case, the descriptor format is defined as listed in Table 7-29. Address and length must be supplied by software on all descriptors. Bits in the command byte are optional, as are the *CSO*, and *CSS* fields.

### Table 7-29 Transmit Descriptor (TDESC) Layout — Legacy Mode

| | 63 48 | 47 40 | 39 36 | 35 32 | 31 24 | 23 16 | 15 0 |
|---|---|---|---|---|---|---|---|
| 0 | Buffer Address [63:0] | | | | | | |
| 8 | VLAN | CSS | Rsvd | STA | CMD | CSO | Length |

### Table 7-30 Transmit Descriptor Write-Back Format — Legacy Mode

| | 63 48 | 47 40 | 39 36 | 35 32 | 31 24 | 23 16 | 15 0 |
|---|---|---|---|---|---|---|---|
| 0 | Reserved | | | | Reserved | | |
| 8 | VLAN | CSS | Rsvd | STA | CMD | CSO | Length |

#### Buffer Address (64) and Length (16)

The buffer address is a byte-aligned address. Length (TDESC.LENGTH) specifies the length in bytes to be fetched from the buffer address provided. The maximum length associated with a single descriptor is 15.5 KB while the total frame size must meet the maximum supported frame size. There is no limitation for the minimum buffer size.

**Note:** Descriptors with zero length (null descriptors) transfer no data. Null descriptors might appear only between packets and must have their *EOP* bits set.

#### Checksum Offset and Start — CSO (8) and CSS (8)

A *Checksum Offset* (TDESC.CSO) field indicates where, relative to the start of the packet, to insert a TCP checksum if this mode is enabled. A *Checksum Start* (TDESC.CSS) field indicates where to begin computing the checksum. Note that *CSO* and *CSS* are meaningful only in the first descriptor of a packet.

Both *CSO* and *CSS* are in units of bytes. These must both be in the range of data provided to the device in the descriptor. This means for short packets that are padded by software, *CSO* and *CSS* must be in the range of the unpadded data length, not the eventual padded length (64 bytes). The allowed ranges for *CSO* and *CSS* are:

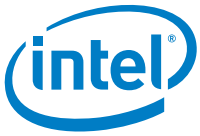
```
14 ≤ CSS ≤ unpadded packet length minus 1
CSS + 2 ≤ CSO ≤ unpadded packet length minus 4
```

For the 802.1Q header, the offset values depend on the VLAN insertion enable bit — the *VLE* bit. If they are not set (VLAN tagging included in the packet buffers), the offset values should include the VLAN tagging. If these bits are set (VLAN tagging is taken from the packet descriptor), the offset values should exclude the VLAN tagging.

Hardware does not add the 802.1q Ethertype or the VLAN field following the 802.1Q Ethertype to the checksum. So for VLAN packets, software can compute the values to back out only on the encapsulated packet rather than on the added fields.

**Note:** UDP checksum calculation is not supported by the legacy descriptor because the legacy descriptor does not support the translation of a checksum result of 0x0000 to 0xFFFF needed to differentiate between an UDP packet with a checksum of zero and an UDP packet without checksum.

Because the *CSO* field is eight bits wide, it puts a limit on the location of the checksum to 255 bytes from the beginning of the packet.



**Note:** CSO must be larger than CSS.

Software must compute an offsetting entry to back out the bytes of the header that should not be included in the TCP checksum and store it in the position where the hardware computed checksum is to be inserted.

Hardware adds the checksum at the byte offset indicated by the *CSO* field. Checksum calculations are for the entire packet starting at the byte indicated by the *CSS* field. The byte offset is counted from the first byte of the packet fetched from host memory.

Command Byte — CMD (8)

The CMD byte stores the applicable command and has the fields listed in Table 7-31.

**Table 7-31 Transmit Command (TDESC.CMD) Layout**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RSV | VLE | DEXT | RSV | RS | IC | IFCS | EOP |

- RSV (bit 7) — Reserved
- VLE (bit 6) — VLAN Packet Enable

  When set to 1b, VLE indicates that the packet is a VLAN packet and hardware adds the VLAN header to the Tx packet. The VLAN Ethertype is taken from DMATXCTL.VT and the 802.1q VLAN tag is taken from the *VLAN* field in the Tx descriptor. See Section 7.4.5 for details about double VLAN.

**Table 7-32 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 VMVIR.VLANA = 00b (use descriptor command) for the queue.

- DEXT (bit 5) — Descriptor extension (zero for legacy mode)
- RSV (bit 4) — Reserved
- RS (bit 3) — Report Status - RS signals hardware to report the DMA completion status indication as well as triggering ITR. Hardware indicates a DMA completion by setting the *DD* bit in the Tx descriptor when TDWBAL[n].Head_WB_En = 0b or by Head Write-back if Head_WB_En = 1b (see Section 7.2.3.5.2). The *RS* bit is permitted only on descriptors that has the *EOP* bit set (last descriptor of a packet).

**Note:** Software should not set the *RS* bit when TXDCTL.WTHRESH is greater than zero. Instead, the hardware reports the DMA completion according to the WTHRESH rules (explained in Section 7.2.3.5.1). This note is relevant only for descriptor write back while in head write-back mode. WTRESH must also be set to zero.

When TXDCTL.WTHRESH = zero, software must set the *RS* bit on the last descriptor of every packet.

There are some exceptions for descriptor completion indication in head write-back mode. For more details see Section 7.2.3.5.2.

- IC (bit 2) — Insert Checksum - Hardware inserts a checksum at the offset indicated by the *CSO* field if the *Insert Checksum* bit (IC) is set.
- IFCS (bit 1) — Insert FCS - When set, hardware appends the MAC FCS at the end of the packet. When cleared, software should calculate the FCS for proper CRC check. There are several cases in which software must set IFCS as follows:
  - Transmitting a short packet while padding is enabled by the HLREG0.TXPADEN bit.
  - Checksum offload is enabled by the *IC* bit in the TDESC.CMD.
  - VLAN header insertion enabled by the *VLE* bit in the TDESC.CMD or by the PFVMVIR registers.
  - TSO or TCP/IP checksum offload using a context descriptor.
  - LinkSec offload is requested.

Note that TSO, Transmit Rate Scheduler, and LinkSec 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 7.2.3.2.1 for more details.

**Table 7-33 VLAN Field (TDESC.VLAN) Layout**

| 15 13 | 12 | 11 0 |
|---|---|---|
| PRI | CFI | VLAN |

### 7.2.3.2.3 Advanced Transmit Context Descriptor

**Table 7-34 Transmit Context Descriptor (TDESC) Layout — (Type = 0010)**

| | 63 | 48 | 47 | 42 | 41 | 32 | 31 | 16 | 15 | 9 | 8 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | RSV | | FCoEF | | IPsec SA Index | | VLAN | | MACLEN | | IPLEN/HEADLEN | |
| 8 | MSS | | L4LEN | | RSV | IDX | BCNTLEN | DEXT RSV | DTYP | | TUCMD | IPsec ESP_LEN |

| | 63 | 48 | 47 | 40 | 39 | 37 | 36 | 35 | 30 | 29 | 28 24 | 23 | 20 | 19 | 9 | 8 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

#### IPLEN/HEADLEN (9)

- IPLEN — for IP packets:

  This field holds the value of the IP header length for the IP checksum offload feature. If an offload is requested, IPLEN must be greater than or equal to 20, and less than or equal to 511. For IP tunnel packets (IPv4-IPv6) IPLEN must be defined as the length of the two IP headers. The hardware is able to offload the L4 checksum calculation while software should provide the IPv4 checksum. For IPsec packets, it is the sum of IP header length plus IPsec header length.

- HEADLEN — for FCoE packets:

  This field indicates the size (in bytes) of the FCoE frame header. The frame header includes the MAC header, optional VLAN and FCoE header(s) as shown in Figure 7-47. HEADLEN does not include the LinkSec header if it exists. HEADLEN is meaningful only if transmit FCoE offload is enabled by setting the *FCoE* bit in the TUCMD field. HEADLEN that matches Figure 7-47 equals 56 or 64 for packets without FC extended headers or packets with VFT header respectively. The 82599 supports FC extended headers only for single send. Segmentation offload can be used only when extended headers are not present.

#### MACLEN (7)

- For nonFCoE packets:

  This field indicates the length of the MAC header. When an offload is requested, the TSE bit (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. Table 7-35 lists the value of MACLEN in the different cases.

**Table 7-35 MACLEN Values**

| SNAP | Regular VLAN | Extended VLAN | MACLEN |
|---|---|---|---|
| No | By hardware or no | No | 14 |
| No | By hardware or no | Yes | 18 |
| No | By software | No | 18 |
| No | By software | Yes | 22 |
| Yes | By hardware or no | No | 22 |

**Table 7-35 MACLEN Values (Continued)**

| SNAP | Regular VLAN | Extended VLAN | MACLEN |
|---|---|---|---|
| Yes | By hardware or no | Yes | 26 |
| Yes | By software | No | 26 |
| Yes | By software | Yes | 30 |

- For FCoE packets:

  This field is a byte offset to the last Dword of the FCoE header (supplied by the driver) that includes the SOF flag. The FC frame header starts four bytes after the MACLEN as shown in Figure 7-47. The MACLEN that matches Figure 7-47 equals 28.

VLAN (16)

This field contains the 802.1Q VLAN tag to be inserted in the packet during transmission. This VLAN tag is inserted when a packet using this context has its DCMD.VLE bit is set. This field should include the entire 16-bit VLAN field including CFI and priority fields as listed in Table 7-33.

Note that the *VLAN* field is provided in network order. See Section 7.2.3.2.1.

Ipsec SA IDX (10) – IPsec SA Index. If an IPsec offload is requested for the packet (IPSEC bit is set in the advanced Tx data descriptor), indicates the index in the SA table where the IPsec key and SALT are stored for that flow.

FCoEF (6) — see the following:

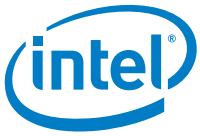
- EOF (bits 1:0) — End of frame delimiter index.
- ORIE (bit 4) — Orientation relative to the last frame in an FC sequence.

The *EOF* and *ORIE* fields define the *EOF* that is inserted by hardware. In a single packet send, the *EOF* field is defined completely by the *EOF* setting while in TSO mode, the *EOF* field is defined by the *EOF* and the *ORIE* bits as listed in Table 7-36. The values EOF0...EOF3 are taken from the TEOFF register.

**Table 7-36 EOF Codes in TSO**

| EOF bits (1:0) | ORIE bit (4) | E-EOF code in a single packet send | FCoE Large send | | |
|---|---|---|---|---|---|
| | | | E-EOF code in the last frame | E-EOF code in other frames | TSO that ends up in a single packet |
| 00 (EOFn) | 0 (not a sequence end) | EOF0 (EOFn) | EOF0 (EOFn) | EOF0 (EOFn) | EOF0 (EOFn) |
| 00 (EOFn) | 1 (sequence end) | EOF0 (EOFn) | EOF1 (EOFt) | EOF0 (EOFn) | EOF1 (EOFt) |
| 01 (EOFt) | 1 (don't care) | EOF1 (EOFt) | n/a | n/a | EOF1 (EOFt) |
| 10 (EOFni) | 1 (don't care) | EOF2 (EOFni) | n/a | n/a | EOF2 (EOFni) |
| 11 (EOFa) | 1 (don't care) | EOF3 (EOFa) | n/a | n/a | EOF3 (EOFa) |

- SOF (bit 2) — Start of frame delimiter index.
- ORIS (bit 5) — Orientation relative to the first frame in an FC sequence.



In a single packet send, SOF is taken from the data buffer. In TSO, hardware places the SOF in the transmitted packet replacing the data buffer content. The *SOF* and *ORIS* bits in the context descriptor define the SOF that is placed by the hardware as listed in Table 7-37. The values SOF0...SOF3 are taken from the SOFF register.

**Table 7-37 SOF Codes**

| SOF bit (2) | ORIS bit (5) | SOF code in the first frame | SOF code in other frames | SOF code in a single packet |
|---|---|---|---|---|
| 1 (Class 3) | 1 (sequence start) | SOF1 (SOFi3) | SOF3 (SOFn3) | SOF1 (SOFi3) |
| 1 (Class 3) | 0 (not a sequence start) | SOF3 (SOFn3) | SOF3 (SOFn3) | SOF3 (SOFn3) |
| 0 (Class 2) | 1 (sequence start) | SOF0 (SOFi2) | SOF2 (SOFn2) | SOF0 (SOFi2) |
| 0 (Class 2) | 0 (not a sequence start) | SOF2 (SOFn2) | SOF2 (SOFn2) | SOF2 (SOFn2) |

- PARINC (bit 3) — When this bit is set, hardware relates to the *PARAM* field in the FC header as relative offset. In this case, hardware increments the *PARAM* field in TSO by an MSS value on each transmitted packet of the TSO. Software should set the *PARINC* bit when it sets the *Relative Offset Present* bit in the F_CTL.

RSV(16)

Reserved

IPS_ESP_LEN(9) - Size of the ESP trailer and ESP ICV appended by software. Meaningful only if the IPSEC_TYPE bit is set in the TUCMD field and to single send packets for which the IPSEC bit is set in their advanced Tx data descriptor.

TUCMD (11)

- RSV (bit 10-7) — Reserved
- FCoE (bit 6) — This bit defines the context descriptor and the associated data descriptors as FCoE frame type. See Section 7.13.2 for a description of the offload provided by the hardware while transmitting a single frame and TSO.
- Encryption (bit 5) — ESP encryption offload is required. Meaningful only to packets for which the IPSEC bit is set in their advanced Tx data descriptor.
- IPSEC_TYPE (bit 4) — Set for ESP. Cleared for AH. Meaningful only to packets for which the IPSEC bit is set in their advanced Tx data descriptor.
- L4T (bit 3:2) — L4 Packet TYPE (00: UDP; 01: TCP; 10: SCTP; 11: RSV)
- IPV4(bit 1) — IP Packet Type: When 1b, IPv4; when 0b, IPv6
- SNAP (bit 0) — SNAP indication

DTYP (4)

This field is always 0010b for this type of descriptor.

RSV(1)

Reserved

DEXT (1) — Descriptor extension (one for advanced mode)

BCNTLEN(6) — For rate limited queues this field must be set to 0x3F.

### 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 82599 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 255. 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)  + IPLEN + L4LEN + MSS + [PADLEN + 18] (if ESP
packet)
```

PADLEN ranges from zero to three in Tx and is the content of the ESP Padding Length field that is computed when offloading ESP in cipher blocks of 16 bytes (AES-128) with respect to the following alignment formula:

```
[L4LEN + MSS + PADLEN + 2] modulo(4) = 0
```

For a single send the IPS_ESP_LEN equals to PADLEN + 18.

**Note:** The headers lengths must meet the following: MACLEN + IPLEN + L4LEN <= 512

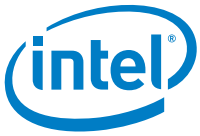
### FCoE Segmentation

The total length of each frame (or segment) excluding Ethernet CRC equals to:

```
MACLEN + 4(if VLE set)  + 8 (FC CRC + EOF)
```

**Note:** For FCoE packets, the maximum segment size defines the FC payload size in all packets but the last one, which can be smaller.

The context descriptor requires valid data only in the fields used by the specific offload options. The following table lists the required valid fields according to the different offload options.



### Table 7-38 Valid Fields by Offload Option

| | Context Fields -> | FCoE | FCoEF | VLAN | MACLEN | IPLEN/HEADLEN | L4LEN | SNAP | IPV4 | L4T | Encryption | IPSECTYPE | SAIDX | ESP_LEN | MSS | BCNTLEN | CC (data descriptor) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Required Offload | VLAN insertion | | | yes | | | | | | | | | | | | yes | |
| | IPv4 XSUM | n/a | n/a | | yes | yes | | | 1 | | | | | | | yes | 0 |
| | L4 XSUM | n/a | n/a | | yes | yes | | | | yes | | | | | | yes | 0 |
| | TCP/UDP Seg | n/a | n/a | | yes | yes | yes | yes | yes | yes | | | | | yes | yes | 0 |
| | FCoE CRC | yes | yes | | yes | yes | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | | yes | 1 |
| | FCoE Seg | yes | yes | | yes | yes | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | yes | yes | 1 |
| | IPSec ESP | n/a | n/a | | yes | yes | | | yes | | yes | yes | yes | yes | | yes | 1 |
| | IPSec AH | n/a | n/a | | yes | yes | | | yes | | yes | yes | yes | n/a | | yes | 1 |
| | Tx switch | n/a | n/a | | yes | yes | yes | | yes | yes | n/a | n/a | n/a | n/a | | yes | 1 |

**Note:** All fields that are not used in the context descriptor must be set to zero.

## 7.2.3.2.4 Advanced Transmit Data Descriptor

### Table 7-39 Advanced Transmit Data Descriptor Read Format

| 0 | Address[63:0] | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 8 | PAYLEN | POPTS | CC | IDX | STA | DCMD | DTYP | MAC | RSV | DTALEN |
| | **63 46** | **45 40** | **39** | **38 36** | **35 32** | **31 24** | **23 20** | **19 18** | **17 16** | **15 0** |

### Table 7-40 Advanced Transmit Data Descriptor Write-back Format

| 0 | RSV | | |
|---|---|---|---|
| 8 | RSV | STA | RSV |
| | **63 36** | **35 32** | **31 0** |

#### General Rule for all Fields

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.

#### 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 15.5 KB with no limitations on the minimum size. Refer to the comment on descriptors with zero length described in the sections that follow.

RSV(2)

Reserved

MAC (2)

See the following. This field is meaningful on the first descriptor of the packet(s).

- ILSec (bit 0) — Apply LinkSec on packet. When set, hardware includes the LinkSec header (SecTAG) and LinkSec header digest (signature). The LinkSec processing is defined by the *Enable Tx LinkSec* field in the LSECTXCTRL register. The *ILSec* bit in the packet descriptor should not be set if LinkSec processing is not enabled by the *Enable Tx LinkSec* field. If the *ILSec* bit is set erroneously while the *Enable Tx LinkSec* field is set to 00b, then the packet is dropped.
- 1588 (bit 1) — IEEE1588 time stamp packet.

DTYP (4)

0011b for advanced data descriptor. DTYP should be valid in all descriptors of the packet(s).

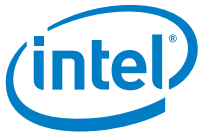
DCMD (8)

See the following:

- TSE (bit 7) — Transmit Segmentation Enable: This bit indicates a TCP or FCoE segmentation request. When *TSE* is set in the first descriptor of a TCP or FCoE packet, hardware must use the corresponding context descriptor in order to perform segmentation.

**Note:** It is recommended that HLREG0.TXPADEN be enabled when TSE is used since the last frame can be shorter than 60 bytes — resulting in a bad frame.

- VLE (bit 6) — VLAN Packet Enable: This bit indicates that the packet is a VLAN packet (hardware must add the VLAN Ethertype and an 802.1q VLAN tag to the packet).
- DEXT (bit 5) — Descriptor Extension: This bit must be one to indicate advanced descriptor format (as opposed to legacy).
- Rsv (bit 4) — Reserved
- RS (bit 3) — Report Status: See the description in the legacy transmit descriptor in Section 7.2.3.2.2.
- Rsv (bit 2) — Reserved
- IFCS (bit 1) — Insert FCS: When this bit is set, the hardware appends the MAC FCS at the end of the packet. When cleared, software should calculate the FCS for proper CRC check. There are several cases in which software must set IFCS as follows:
  - — Transmitting a short packet while padding is enabled by the HLREG0.TXPADEN bit.
  - — Checksum offload is enabled by the either *IC, TXSM* or *IXSM* bits in the TDESC.DCMD.



— VLAN header insertion enabled by the *VLE* bit in the TDESC.DCMD.

— FC CRC (FCoE) offload is enabled by the *FCoE* bit in the transmit context descriptor.

— TCP or FCoE segmentation offload enabled by the *TSE* bit in the TDESC.DCMD.

- EOP (bit 0) — End of Packet: A packet might be composed of multiple buffers (each of them is indicated by its own descriptor). When EOP is set, it indicates the last descriptor making up the packet. In transmit segmentation (explained later on in this section) the EOP flag indicates the last descriptor of the last packet of the segmented transmission.

**Note:** *TSE*, *VLE* and *IFCS* fields should be set in the first descriptor of the packet(s). The *RS* bit can be set only on the last descriptor of the packet. The *EOP* bit is valid in all descriptors. The *DEXT* bit must be set to 1b for all descriptors.

Descriptors with zero length, transfer no data. If the *RS* bit in the command byte is set, then the *DD* field in the status word is not written when hardware processes them.

STA (4)

- Rsv (bit 3:1) — Reserved
- DD (bit 0) — Descriptor Done: The *DD* bit provides a status indication that the DMA of the buffer has completed. Software might re-use descriptors with the *DD* bit set, and any other descriptors processed by hardware before this one. In TSO, the buffers that include the TSO header are used multiple times during transmission and special considerations should be made as described in Section 7.2.4.2.2.

IDX (3)

This field holds the index into the hardware context table to indicate which of the two per-queue contexts should be used for this request. If no offload is required and the *CC* bit is cleared, this field is not relevant and no context needs to be initiated before the packet is sent. See Table 7-38 for details of which packets requires a context reference. This field is relevant only on the first descriptor of the packet(s).

CC (1)

Check Context bit — When set, a Tx context descriptor indicated by IDX index should be used for this packet(s). The *CC* bit should be set in the following cases:

1. Non-zero *BCNTLEN* field is required (defined in the context descriptor).
2. Any FCoE offload is required.
3. Tx switching is enabled.

POPTS (6)

This field is relevant only on the first descriptor of the packet(s).

- Rsv (bits 5:3) — Reserved
- IPSEC (bit 2) — Ipsec offload request
- TXSM (bit 1) — Insert TCP/UDP Checksum: When set to 1b, the L4 checksum must be inserted. In this case, TUCMD.LP4 indicates whether the checksum is TCP or UDP or SCTP. When DCMD.TSE is set, TXSM must be set to as well. If this bit is set, the packet should at least contain an L4 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 as well. If this bit is set, the packet should at least contain an IP header.

PAYLEN (18)

PAYLEN indicates the size (in byte units) of the data buffer(s) in host memory for transmission. In a single-send packet, PAYLEN defines the entire packet size fetched from host memory. It does not include the fields that hardware adds such as: optional VLAN tagging, the FCoE trailer containing the FC CRC and EOF (for FCoE packets), Ethernet CRC or Ethernet padding.

When LinkSec offload is enabled, the PAYLEN field does not include the LinkSec encapsulation. When IPsec offload is enabled, the PAYLEN field does not include the ESP trailer added by hardware. In TSO (regardless if it is transmitted on a single or multiple packets), the PAYLEN defines the protocol payload size fetched from host memory. In TCP or UDP segmentation offload, PAYLEN defines the TCP/UDP payload size. In FCoE TSO offload, the PAYLEN field defines the FC payload size. It includes the FC option headers (if present) and the FC data payload but excludes the FCoE trailer containing the FC CRC and EOF.

This field is relevant only on the first descriptor of the packet(s). The minimum transmitted packet size excluding VLAN padding and CRC bytes is 17 and the PAYLEN size should meet this limitation. On a single-packet send, the maximum size of the PAYLEN is dictated by the maximum allowed packet size which is 15.5 KB. On TSO, the maximum PAYLEN can be up to $2^{18}$-1.

### 7.2.3.3 Transmit Descriptor Ring

The transmit descriptor ring structure (shown in Figure 7-20) 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 40 descriptors per queue to hold the descriptors that were fetched from the software ring.

Descriptors handled to hardware should not be manipulated by software until hardware completes its processing. It is indicated by advancing the head pointer beyond these descriptors.

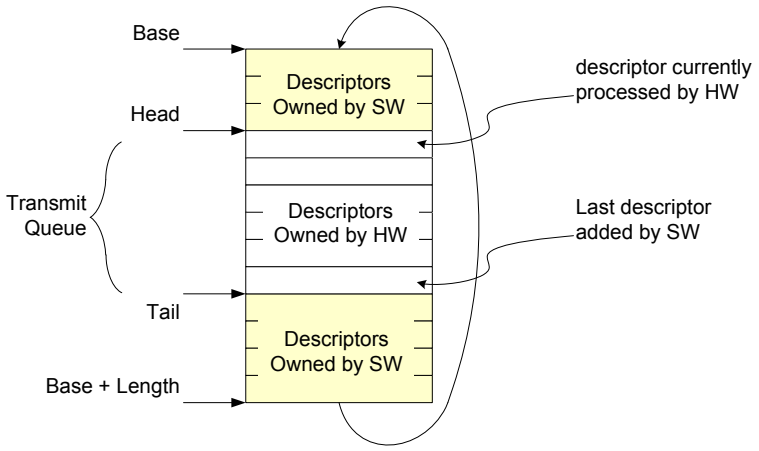



**Figure 7-20 Transmit Descriptor Ring Structure**

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 7.2.3.5.2.
- When head write back is not enabled (TDWBAL[n].Head_WB_En = 0b), software might track the *DD* bits in the descriptor ring. Descriptor write back is controlled by the *RS* bit and the WTHRESH setting as well as interrupt assertion.
- Issue an interrupt. An interrupt condition is generated each time a packet was transmitted or received and a descriptor was write back or transmit queue goes empty (EICR.RTxQ[0-19]). This interrupt can either be enabled or masked.

All of the registers controlling the descriptor rings behavior should be set before transmit is enabled.

### 7.2.3.4 Transmit Descriptor Fetching

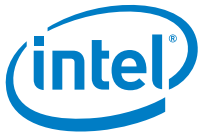
The 82599 fetches new descriptors as required for packet transmission depending on its on-die descriptor buffer state:

Fetch — The on-chip descriptor buffer is empty or contains less descriptors than a complete packet.

- A fetch starts as soon as any descriptors are made available (host writes to the tail pointer).
- A request is issued for any available descriptors up to the size of the on-die buffer.
- Once the sum of on-die descriptors and requested descriptors is more than required for a single packet, the buffer transitions to the pre-fetch state.
- If several on-chip descriptor queues are empty simultaneously, queues are served in round robin arbitration except the queues indicated as strict priority which are served first.

Pre-Fetch — The on-chip descriptor buffer becomes almost empty while there are enough descriptors in the host memory.

- The on-chip descriptor buffer is defined as almost empty if it contains less descriptors then the threshold defined by TXDCTL[n].PTHRESH
- The transmit descriptor contains enough descriptors if it includes more ready descriptors than the threshold defined by TXDCTL[n].HTHRESH
- In pre-fetch mode descriptors are fetched only after there are no other DMA activity of greater priority as: transmit descriptor fetch; status write-backs or packet data transfers)
- A request is issued for any available descriptors up to the capacity of the on-die buffer.



- If several on-chip descriptor queues are in this situation simultaneously, queues are served in round robin arbitration except the queues indicated as strict priority which are served first.

Idle — Requests are not issued. This is the state reached when none of the previous states apply.

**Note:** Software must update the Tail register on packet boundaries. That is, the last valid descriptor might not be a context descriptor and must have the *EOP* bit set.

#### 7.2.3.4.1 Transmit Descriptor Fetch and Write-back Settings

This section describes the settings of transmit descriptor thresholds. It relates to fetch thresholds described above as well as the write-back threshold (WTHRESH) when operating in descriptor write-back mode which is described in Section 7.2.3.5.1.

- Transmit descriptor fetch setting is programmed in the TXDCTL[n] register per queue. The default settings of PTHRESH, HTHRESH and WTHRESH are zero's.
- In order to reduce transmission latency, it is recommended to set the PTHRESH value as high as possible while the HTHRESH and WTHRESH as low as possible (down to zero).
- In order to minimize PCIe overhead the PTHRESH should be set as low as possible while HTHRESH and WTHRESH should be set as high as possible.
- The sum of PTHRESH plus WTHRESH must not be greater than the on-chip descriptor buffer size
- Some practical rules
  - CPU cache line optimization: Assume 'N' equals the CPU cache line divided by 16 (descriptor size). Then, in order to align descriptors pre-fetch to CPU cache line (in most cases), it is advised to set PTHRESH to the on-chip descriptor buffer size minus 'N' and HTHRESH to 'N'. In order to align descriptor write back to the CPU cache line it is advised to set WTHRESH to either 'N' or even 2 times 'N'. Note that partial cache line writes might significantly degrade performance. Therefore, it is highly recommended to follow this advice.
  - Minimizing PCIe overhead: As an example, setting PTHRESH to the on-chip descriptor buffer size minus 16 and HTHRESH to 16 minimizes the PCIe request and header overhead to ~20% of the bandwidth required for the descriptor fetch.
  - Minimizing transmission latency from tail update: Setting PTHRESH to the on-chip descriptor buffer size minus 'N' ('N' previously defined) while HTHRESH and WTHRESH to zero.
  - Threshold settings in DCB mode: Note that only values of PTHRESH equals on-chip descriptor buffer size minus 8 and HTHRESH equals 4 were thoroughly tested.

**Note:** As previously described, device setting is a trade off between overhead (translated to performance) and latencies. It is expected that some level of optimization is done at software driver development phase. Customers who want better performance might need to adjust the threshold values according to the previous guidelines while optimizing to specific platform and targets.

### 7.2.3.5 Transmit Write Back

The 82599 periodically updates software on its progress in processing transmit buffers. Two methods are described for doing so:

- Updating by writing back into the Tx descriptor
- Update by writing to the head pointer in system memory

#### 7.2.3.5.1 Tx Descriptor Write Back

When the TXDCTL[n].WTHRESH equals zero, descriptors are written back for those descriptors with the *RS* bit set. When the TXDCTL[n].WTHRESH value is greater than zero, descriptors are accumulated until the number of accumulated descriptors equals the TXDCTL[n].WTHRESH value, then these descriptors are written back. Accumulated descriptor write back enables better use of the PCIe bus and memory bandwidth.

Any descriptor write back includes the full 16 bytes of the descriptor.

Descriptors are written back in one of three cases:

- TXDCTL[n].WTHRESH = 0 and a descriptor that has *RS* set is ready to be written back.
- TXDCTL[n].WTHRESH > 0 and TXDCTL[n].WTHRESH descriptors have accumulated.
- TXDCTL[n].WTHRESH > 0 and the corresponding EITR counter has reached zero. The timer expiration flushes any accumulated descriptors and sets an interrupt event (TXDW).

An additional mode in which transmit descriptors are not written back at all and the head pointer of the descriptor ring is written instead is described in the following section.

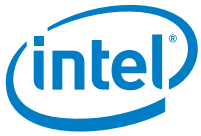
#### 7.2.3.5.2 Tx Head Pointer Write Back

In legacy hardware, transmit requests are completed by writing the *DD* bit to the transmit descriptor ring. This causes cache thrash since both the driver and hardware are writing to the descriptor ring in host memory. Instead of writing the *DD* bits to signal that a transmit request is complete, hardware can write the contents of the descriptor queue head to host memory. The driver reads that memory location to determine which transmit requests are complete. In order to improve the performance of this feature, the driver needs to program DCA registers to configure which CPU will be processing each TX queue.

The head pointer is reflected in a memory location that is allocated by software for each queue.

Rules for head pointer write back:

- Head write back occurs if TDWBAL[n].Head_WB_En is set for this queue, and the *RS* bit is set in the Tx descriptor, following its corresponding data upload into packet buffer.
  - If the head write-back feature is enabled, software must set WTHRESH to 0x0 while only descriptors with the RS *bit* set, generate header write back.



— Note that the head pointer write back does not hold transmission. Instead, if packets with the *RS* bit are transmitted fast enough, it might happen that the header pointer write back is not updated for each and every packet. In addition, it might happen that the head pointer write back might be updated up to descriptors that do not have the *RS* bit set. In such cases, hardware might report a completion of a descriptor that might not be the last descriptor in a TSO or even the last descriptor in a single packet.

The driver has control of this feature per queue through the TDWBAL and TDWBAH registers.

The low register's LSB hold the control bits.

- The Head_WB_EN bit enables activation of tail write back. In this case, no descriptor write back is executed.
- The 30 upper bits of this register hold the lowest 32 bits of the head write-back address, assuming that the two last bits are zero.

The high register holds the high part of the 64-bit address.

**Note:** Hardware writes a full Dword when writing this value, so software should reserve enough space for each head value and make sure the TDBAL value is Dword-aligned.

## 7.2.4 TCP and UDP Segmentation

Hardware TCP segmentation is one of the offloading options supported by the Windows* and Linux* TCP/IP stack. This is often referred to as Large Send offloading or TSO. This feature enables the TCP/IP stack to pass to the network device driver a message to be transmitted that is bigger than the Maximum Transmission Unit (MTU) of the medium. It is then the responsibility of the device driver and hardware to divide the TCP message into MTU size frames that have appropriate layer 2 (Ethernet), 3 (IP), and 4 (TCP) headers. These headers must include sequence number, checksum fields, options and flag values as required. Note that some of these values (such as the checksum values) are unique for each packet of the TCP message, and other fields such as the source IP address is constant for all packets associated with the TCP message.

Similar to TCP segmentation, the 82599 also provides a capability to offload UDP segmentation. Note that current UDP segmentation offload is not supported by any standard OS.

**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 82599 saves significant CPU cycles. The device driver shares the additional tasks to support these options with the 82599.

#### 7.2.4.1 Assumptions and Restrictions

The following assumptions apply to the TCP / UDP segmentation implementation in the 82599:

- To limit the internal cache dimensions, software is required to spread the header onto a maximum four descriptors, while still allowed to mix header and data in the last header buffer. This limitation stands for up to Layer 4 header included, and for IPv4 or IPv6 independently.
- The maximum size of a single TSO can be as large as defined by the *PAYLEN* field in the Tx data descriptor (such as up to 256 KB).
- The *RS* bit operation is not changed. Interrupts are set after data in the buffers pointed to by individual descriptors is transferred (DMA'ed) to hardware.
- SNAP packets are supported for segmentation with the following restriction. The location of the 802.3 length field in 802.3+SNAP packets is at MACLEN minus eight bytes (MACLEN is indicated in the context descriptor).
- IP tunneled packets are not supported for offloading under TSO operation.
- Software must enable the Ethernet CRC offload in the HLREG0.TXCRCEN register since CRC must be inserted by hardware after the checksum has been calculated.
- Software must initialize the appropriate checksum fields in the packet's header.

#### 7.2.4.2 Transmission Process

The transmission process involves the following:

- The protocol stack receives from an application a block of data that is to be transmitted.
- The protocol stack calculates the number of packets required to transmit this block based on the MTU size of the media and required packet headers.
- The stack interfaces with the device driver and passes the block down with the appropriate header information: Ethernet, IP, optional IPSec and TCP / UDP headers.
- The stack interfaces with the device driver and commands the driver to send the individual packet. The device driver sets up the interface to the hardware (via descriptors) for the TCP / UDP segmentation.
- The hardware transfers (DMA's) the packet data and performs the Ethernet packet segmentation and transmission based on offset and payload length parameters in the TCP/IP or UDP/IP context descriptor including:
  - Packet encapsulation
  - Header generation and field updates including IPv4/IPv6 and TCP/UDP checksum generation.
- The driver returns ownership of the block of data to the NOS when the hardware has completed the DMA transfer of the entire data block.

#### 7.2.4.2.1 TCP and UDP Segmentation Data Fetch Control

To perform TCP / UDP segmentation in the 82599, 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 82599 enables interleaving between different TSO requests at an Ethernet packet level. In other words, the 82599 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 82599 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.

#### 7.2.4.2.2 TCP and UDP Segmentation Write-back Modes

TCP / UDP segmentation mode uses the buffers that contain the header of the packet multiple times (once for each transmitted segment). Software should guarantee that the header buffers are available throughout the entire TSO transmission. Therefore, software should not re-use any descriptors of the TSO header during the TSO transmission.

### 7.2.4.3 TCP and UDP Segmentation Performance

Performance improvements for a hardware implementation of TCP / UDP segmentation offload include:

- The stack does not need to partition the block to fit the MTU size, saving CPU cycles.
- The stack only computes one Ethernet, IP, and TCP / UDP header per segment, saving CPU cycles.
- The stack interfaces with the device driver only once per block transfer, instead of once per frame.
- Larger PCI bursts are used, which improves bus efficiency (such as lowering transaction overhead).
- Interrupts are easily reduced to one per TCP / UDP message instead of one per packet.
- Fewer I/O accesses are required to command the hardware.

### 7.2.4.4 Packet Format

Typical TCP/IP transmit window size is 8760 bytes (about six full size frames). Today the average size on corporate Intranets is 12-14 KB, and normally the maximum window size allowed is 64 KB (unless Windows Scaling — RFC 1323 is specified). A TCP / UDP message can be as large as 256 KB and is generally fragmented across multiple pages in host memory. The 82599 partitions the data packet into standard Ethernet frames prior to transmission. The 82599 supports calculating the Ethernet, IP, TCP, and even UDP headers, include checksum, on a frame-by-frame basis.

**Table 7-41 TCP/IP and UDP/IP Packet Format Sent by Host**

| Pseudo Header | | | Data |
|---|---|---|---|
| Ethernet | IPv4/IPv6 | TCP/UDP | DATA (full TCP message) |

**Table 7-42 Packets Format Sent by Device**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Pseudo Header (updated) | Data (first MSS) | FCS | ... | Pseudo Header (updated) | Data (Next MSS) | FCS | ... |

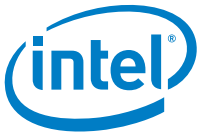
Frame formats supported by the 82599 include:

- Ethernet 802.3
- IEEE 802.1Q VLAN (Ethernet 802.3ac)
- Ethernet Type 2
- Ethernet SNAP
- IPv4 headers with options
- IPv4 headers without options with one AH/ESP IPsec header
- IPv6 headers with extensions
- TCP with options
- UDP with options

VLAN tag insertion is handled by hardware.

**Note:** UDP (unlike TCP) is not a reliable protocol and fragmentation is not supported at the UDP level. UDP messages that are larger than the MTU size of the given network medium are normally fragmented at the IP layer. This is different from TCP, where large TCP messages can be fragmented at either the IP or TCP layers depending on the software implementation.

The 82599 has the ability to segment UDP traffic (in addition to TCP traffic); however, because UDP packets are generally fragmented at the IP layer, the 82599's segmentation capability might not be used in practice for UDP.



### 7.2.4.5 TCP and UDP Segmentation Indication

Software indicates a TCP / UDP segmentation transmission context to the hardware by setting up a TCP/IP or UDP/IP context transmit descriptor (see Section 7.2.3). The purpose of this descriptor is to provide information to the hardware to be used during the TCP / UDP segmentation offload process.

Setting the *TSE* bit in the DCMD field to one (in the data descriptor) indicates that this descriptor refers to the segmentation context (as opposed to the normal checksum offloading context). This causes the checksum offloading, packet length, header length, and maximum segment size parameters to be loaded from the descriptor into the device.

The TCP / UDP segmentation prototype header is taken from the packet data itself. Software must identity the type of packet that is being sent (IPv4/IPv6, TCP/UDP, other), calculate appropriate checksum off loading values for the desired checksums, and then calculate the length of the header that is prepended. The header can be up to 240 bytes in length.

Once the TCP / UDP segmentation context has been set, the next descriptor provides the initial data to transfer. This first descriptor(s) must point to a packet of the type indicated. Furthermore, the data it points to might need to be modified by software as it serves as the prototype header for all packets within the TCP / UDP segmentation context. The following sections describe the supported packet types and the various updates that are performed by hardware. This should be used as a guide to determine what must be modified in the original packet header to make it a suitable prototype header.

The following summarizes the fields considered by the driver for modification in constructing the prototype header.

#### IP Header

For IPv4 headers:

- Identification field should be set as appropriate for first packet of send (if not already).
- Header checksum should be zeroed out unless some adjustment is needed by the driver.

#### TCP Header

- Sequence number should be set as appropriate for first packet of send (if not already).
- PSH, and FIN flags should be set as appropriate for LAST packet of send.
- TCP checksum should be set to the partial pseudo-header checksum as follows (there is a more detailed discussion of this in Section 7.2.4.6:

**Table 7-43 TCP Partial Pseudo-header Checksum for IPv4**

| | | |
|---|---|---|
| IP Source Address | | |
| IP Destination Address | | |
| Zero | Layer 4 Protocol ID | Zero |

**Table 7-44 TCP Partial Pseudo-header Checksum for IPv6**

| | |
|---|---|
| IPv6 Source Address | |
| IPv6 Final Destination Address | |
| Zero | |
| Zero | Next Header |

UDP Header

- Checksum should be set as in TCP header, as previously explained.

The following sections describe the updating process performed by the hardware for each frame sent using the TCP segmentation capability.

### 7.2.4.6 Transmit Checksum Offloading with TCP and UDP Segmentation

The 82599 supports checksum offloading as a component of the TCP / UDP segmentation off-load feature and as stand-alone capability. Section 7.2.5 describes the interface for controlling the checksum off-loading feature. This section describes the feature as it relates to TCP / UDP segmentation.

The 82599 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.

**Table 7-45 Supported Transmit Checksum Capabilities**

| Packet Type | HW IP Checksum Calculation | HW TCP / UDP Checksum Calculation |
|---|---|---|
| IPv4 packets | Yes | Yes |
| IPv6 packets<br>(no IP checksum in IPpv6) | NA | Yes |
| Packet has 802.3ac tag | Yes | Yes |
| Packet has IP options<br>(IP header is longer than 20 bytes) | Yes | Yes |

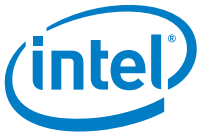


**Table 7-45 Supported Transmit Checksum Capabilities (Continued)**

| Packet Type | HW IP Checksum Calculation | HW TCP / UDP Checksum Calculation |
|---|---|---|
| Packet has TCP options | Yes | Yes |
| IP header's protocol field contains a protocol # other than TCP or UDP | Yes | No |

## 7.2.4.7 IP/TCP / UDP Header Updating

IP/TCP and IP/UDP header is updated for each outgoing frame based on the header prototype that hardware DMA's from the first descriptor(s). The checksum fields and other header information are later updated on a frame-by-frame basis. The updating process is performed concurrently with the packet data fetch.

The following sections define what fields are modified by hardware during the TCP / UDP segmentation process by the 82599.

### 7.2.4.7.1 TCP/IP/UDP Header for the First Frame

The hardware makes the following changes to the headers of the first packet that is derived from each TCP segmentation context.

MAC Header (for SNAP)

- Type/Len field = MSS + MACLEN + IPLEN + L4LEN — 14

IPv4 Header

- IP Total Length = MSS + L4LEN + IPLEN
- Calculates the IP Checksum

IPv6 Header

- Payload Length = MSS + L4LEN + IPV6_HDR_extension[1]

TCP Header

- Sequence Number: The value is the sequence number of the first TCP byte in this frame.
- The flag values of the first frame are set by logic AND function between the flag word in the pseudo header and the DTXTCPFLGL.TCP_flg_first_seg. The default values of the DTXTCPFLGL.TCP_flg_first_seg are set. The flags in a TSO that ends up as a single segment are taken from the in the pseudo header in the Tx data buffers as is.
- Calculates the TCP checksum.

UDP Header

- Calculates the UDP checksum.

1. IPV6_HDR_extension is calculated as IPLEN — 40 bytes.

#### 7.2.4.7.2 TCP/IP Header for the Subsequent Frames

The hardware makes the following changes to the headers for subsequent packets that are derived as part of a TCP segmentation context:

Number of bytes left for transmission = PAYLEN — (N * MSS). Where N is the number of frames that have been transmitted.

MAC Header (for SNAP packets)

Type/Len field = MSS + MACLEN + IPLEN + L4LEN — 14

IPv4 Header

- IP Identification: incremented from last value (wrap around)
- IP Total Length = MSS + L4LEN + IPLEN
- Calculate the IP Checksum

IPv6 Header

- Payload Length = MSS + L4LEN + IPV6_HDR_extension[1]

TCP Header

- Sequence Number update: Add previous TCP payload size to the previous sequence number value. This is equivalent to adding the MSS to the previous sequence number.
- The flag values of the subsequent frames are set by logic AND function between the flag word in the pseudo header with the DTXTCPFLGL.TCP_Flg_mid_seg. The default values of the DTXTCPFLGL.TCP_Flg_mid_seg are set.
- Calculate the TCP checksum

UDP Header

- Calculates the UDP checksum.

#### 7.2.4.7.3 TCP/IP Header for the Last Frame

Hardware makes the following changes to the headers for the last frame of a TCP segmentation context:

Last frame payload bytes = PAYLEN — (N * MSS).

MAC Header (for SNAP packets)

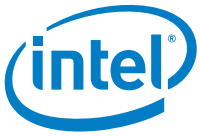
- Type/Len field = Last frame payload bytes + MACLEN + IPLEN + L4LEN — 14

IPv4 Header

- IP Total length = last frame payload bytes + L4LEN + IPLEN
- IP identification: incremented from last value (wrap around based on 16-bit width)
- Calculate the IP checksum

IPv6 Header

- Payload length = last frame payload bytes + L4LEN + IPV6_HDR_extension[1]



TCP Header

- Sequence number update: Add previous TCP payload size to the previous sequence number value. This is equivalent to adding the MSS to the previous sequence number.
- The flag values of the last frames are set by logic AND function between the flag word in the pseudo header and the DTXTCPFLGH.TCP_Flg_lst_seg. The default values of the DTXTCPFLGH.TCP_Flg_lst_seg are set. The flags in a TSO that ends up as a single segment are taken from the in the pseudo header in the Tx data buffers as is.
- Calculate the TCP checksum

UDP Header

- Calculates the UDP checksum.

## 7.2.5 Transmit Checksum Offloading in Non-segmentation Mode

The previous section on TCP / UDP segmentation offload describes the IP/TCP/UDP checksum offloading mechanism used in conjunction with segmentation. The same underlying mechanism can also be applied as a stand-alone checksum offloading. The main difference in a single packet send is that only the checksum fields in the IP/TCP/UDP headers are calculated and updated by hardware.

Before taking advantage of the 82599's enhanced checksum offload capability, a checksum context must be initialized. For a single packet send, DCMD.TSE should be set to zero (in the data descriptor). For additional details on contexts, refer to Section 7.2.3.3.

Enabling checksum offload, software must also enable Ethernet CRC offload by the HLREG0.TXCRCEN since CRC must be inserted by hardware after the checksum has been calculated.

As mentioned in Section 7.2.3, transmit descriptors, it is not necessary to set a new context for each new packet. In many cases, the same checksum context can be used for a majority of the packet stream. In this case, some performance can be gained by only changing the context on an as needed basis or electing to use the off-load feature only for a particular traffic type, thereby avoiding all context descriptors except for the initial one.

Each checksum operates independently. Insertion of the IP and TCP / UDP checksum for each packet are enabled through the transmit data descriptor POPTS.TXSM and POPTS.IXSM fields, respectively.

#### 7.2.5.1 IP Checksum

Three fields in the transmit context descriptor set the context of the IP checksum offloading feature:

- TUCMD.IPV4
- IPLEN
- MACLEN

TUCMD.IPV4=1 specifies that the packet type for this context is IPv4, and that the IP header checksum should be inserted. TUCMD.IPV4=0 indicates that the packet type is IPv6 (or some other protocol) and that the IP header checksum should not be inserted.

MACLEN specifies the byte offset from the start of the DMA'ed data to the first byte to be included in the checksum, the start of the IP header. The minimal allowed value for this field is 14. Note that the maximum value for this field is 127. This is adequate for typical applications.

**Note:** The MACLEN+IPLEN value must be less than the total DMA length for a packet. If this is not the case, the results are unpredictable.

IPLEN specifies the IP header length. Maximum allowed value for this field is 511 bytes.

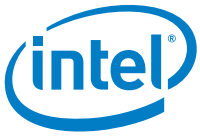
MACLEN+IPLEN specify where the IP checksum should stop. The sum of MACLEN+IPLEN must be smaller equals to the first 638 (127+511) bytes of the packet and obviously must be smaller or equal to the total length of a given packet. If this is not the case, the result is unpredictable.

**Note:** For IPsec packets offloaded by hardware in Tx, it is assumed that IPLEN provided by software in the Tx context descriptor is the sum of the IP header length and the IPsec header length. Thus, for the IPv4 header checksum offload, hardware could no longer rely on the *IPLEN* field provided by software in the Tx context descriptor, but should rely on the fact that no IPv4 options is present in the packet. Consequently, for IPsec offload packets, hardware computes IP header checksum over a fixed amount of 20 bytes.

For IP tunnel packets (IPv4-IPv6), IPLEN must be defined as the length of the two IP headers. Hardware is able to offload the L4 checksum calculation while software should provide the IPv4 checksum.

The 16-bit IPv4 header checksum is placed at the two bytes starting at MACLEN+10.

As mentioned in Section 7.2.3.2.3, transmit contexts, it is not necessary to set a new context for each new packet. In many cases, the same checksum context can be used for a majority of the packet stream. In this case, some performance can be gained by only changing the context on an as needed basis or electing to use the off-load feature only for a particular traffic type, thereby avoiding all context descriptors except for the initial one.



### 7.2.5.2 TCP and UDP Checksum

Three fields in the transmit context descriptor set the context of the TCP / UDP checksum offloading feature:

- MACLEN
- IPLEN
- TUCMD.L4T

TUCMD.L4T=01b specifies that the packet type is TCP, and that the 16-bit TCP header checksum should be inserted at byte offset MACLEN+IPLEN+16. TUCMD.L4T=00b indicates that the packet is UDP and that the 16-bit checksum should be inserted starting at byte offset MACLEN+IPLEN+6.

MACLEN+IPLEN specifies the byte offset from the start of the DMA'ed data to the first byte to be included in the checksum, the start of the UDP/TCP header. See MACLEN table in Section 7.2.3.2.3 for its relevant values.

**Note:** The MACLEN+IPLEN+L4LEN value must be less than the total DMA length for a packet. If this is not the case, the results are unpredictable.

The TCP/UDP checksum always continues to the last byte of the DMA data.

**Note:** For non-TSO, software still needs to calculate a full checksum for the TCP/UDP pseudo-header. This checksum of the pseudo-header should be placed in the packet data buffer at the appropriate offset for the checksum calculation.

### 7.2.5.3 SCTP Transmit Offload

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.

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.

Software must initialize the SCTP CRC field to zero (0x00000000).

### 7.2.5.4 Checksum Supported per Packet Types

The following table lists which checksums are supported per packet type.

**Note:** TSO is not supported for packet types for which IP checksum and TCP / UDP checksum cannot be calculated.

# 7.3 Interrupts

The 82599 supports the following interrupt modes. Mapping of interrupts causes is different in each of these modes as described in this section.

- PCI legacy interrupts or MSI or MSI-X and only a single vector is allocated — selected when GPIE.Multiple_MSIX is set to 0b.
- MSI-X with multiple MSI-X vectors in non-IOV mode — selected when GPIE.Multiple_MSIX is set to 1b and GPIE.VT_Mode is set to 00b.
- MSI-X in IOV mode — selected when GPIE.Multiple_MSIX is set (as previously stated) and GPIE.VT_Mode DOES NOT equal 00b.

The following sections describe the interrupt registers and device functionality at all operation modes.

## 7.3.1 Interrupt Registers

### Physical Function (PF) Registers

The PF interrupt logic consists of the registers listed in the Table 7-46 followed by their description:

**Table 7-46 PF Interrupt Registers**

| Acronym | Complete Name |
|---|---|
| EICR | Extended Interrupt Cause register |
| EICS | Extended Interrupt Cause Set register (enables software to initiate interrupts) |
| EIMS | Extended Interrupt Mask Set/Read register |
| EIMC | Extended Interrupt Mask Clear register |
| EIAC | Extended Interrupt Auto Clear register (following interrupt assertion) |
| EIAM | Extended Interrupt Auto Mask register (auto set/clear of the EIMS) |
| EITR | Extended Interrupt Throttling register [throttling and Low Latency Interrupt (LLI) setting] |
| IVAR | Interrupt Vector Allocation Registers (described in Section 7.3.4) |
| IVAR_MISC | Miscellaneous Interrupt Vector Allocation Register (described in Section 7.3.4) |

These registers are extended to 64 bits by an additional set of two registers. EICR has an additional two registers EICR(1)... EICR(2) and so on for the EICS, EIMS, EIMC, EIAM and EITR registers. The EIAC register is not extended to 64 bits as this extended interrupt causes are always auto cleared. Any reference to EICR... EIAM registers as well as any global interrupt settings in the GPIE register relates to their extended size of 64 bits.

The legacy EICR[15:0] mirror the content of EICR(1)[15:0]. In the same manner the lower 16 bits of EICS, EIMS, EIMC, EIAC, EIAM mirror the lower 16 bits of EICS(1), EIMS(1), EIMC(1), EIAM(1). For more details on the use of these registers in the various interrupt modes (Legacy, MSI, MSI-X) see Section 7.3.4.

Virtual Function (VF) Registers

The VF interrupt logic has the same set of interrupt registers while each of them has three entries for three interrupt causes. The names and functionality of these registers are the same as those of the PF with a prefix of VT as follows: VFEICR, VFEICS, VFEIMS, VFEIMC, VFEIAM, VFEITR. The VFEIAC registers are not supported since interrupt causes are always auto cleared. Although each VF can generate up to three interrupts, only the first two registers are capable of interrupt throttling and are associated to VFEITR registers (see Section 7.3.4.3.2 for its proper usage). Each VF also has the mapping registers VFIVAR and VFIVAR_MISC. Note that any global interrupt setting by the GPIE register affect both interrupt settings of the PF as well as the VFs.

### 7.3.1.1 Extended Interrupt Cause (EICR) Registers

This register records the interrupt causes to provide software information on the interrupt source. Each time an interrupt cause happens, the corresponding interrupt bit is set in the EICR registers. An interrupt is generated each time one of the bits in these registers is set, and the corresponding interrupt is enabled via the EIMS registers. The possible interrupt causes are as follows:

- Each *RTxQ* bit represents the following events: Tx or Rx descriptor write back; Rx queue full and Rx descriptor queue minimum threshold.
  - The *RTxQ* interrupts can be throttled by ITR or LLI as configured in the EITR register (LLI does not impact Tx). Following interrupt assertion, software cannot distinguish between ITR or LLI events.
  - Mapping the Tx and Rx queues to EICR is done by the IVAR registers as described in Section 7.3.4. Each bit might represent an event on a single Tx or Rx queue or could represent multiple queues according to the IVAR setting. In the later case, software might not be able to distinguish between the interrupt causes other than checking all associated Tx and Rx queues.
  - The Multiple_MSIX = 1b setting is useful when multiple MSI-X vectors are assigned to the device. When the GPIE.Multiple_MSIX bit is set, the *RTxQ* bits are associated with dedicated MSI-X vectors. Bit 0 is Tx / Rx interrupt associated with MSI-X vector 0 and bit 15 is Tx / Rx interrupt associated with MSI-X vector 15.
- Bits 29:16 in the EICR are named in the EAS as the “other” interrupt causes. Please refer to the EICR register definition for the exact interrupt causes included in this group. All these causes are mapped to the same interrupt even in Multiple_MSIX mode. In Multiple_MSIX mode the “other” interrupt causes are mapped to a specific MSI-X vector by the INT_Alloc[1] in the IVAR_MISC register.
- Bit 30 in the EICR register is the “TCP Timer” interrupt usually used to wake the SW driver periodically according to the TCPTIMER setting. In Multiple_MSIX mode the “TCP Timer” interrupt is mapped to a specific MSI-X vector by the INT_Alloc[0] in the IVAR_MISC register.

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 (detailed in Section 7.3.1.4).

### 7.3.1.2 Extended Interrupt Cause Set (EICS) Register

This register enables software to initiate a hardware interrupt. Setting any bit on the EICS sets its corresponding bit in the EICR register while bits written to 0b have no impact. It then causes an interrupt assertion if enabled by the EIMS register. Setting any bit generates either LLI or throttled interrupt depending on the GPIE.EIMEN setting: When the *EIMEN* bit is set, then setting the EICS register causes an LLI interrupt; When the *EIMEN* bit is cleared, then setting the EICS register causes an interrupt after the corresponding interrupt throttling timer expires.

**Note:** The *EIMEN* bit can be set high only when working in auto-mask mode (*EIAM* bit of the associated interrupt is set).

#### 7.3.1.2.1 EICS Affect on RSC Functionality

Setting *EICS* bits causes interrupt assertion (if enabled). EICS settings have the same impact on RSC functionality as nominal operation:

- In ITR mode (GPIE.EIMEN = 0b), setting the *EICS* bits impact the RSC completion and interrupt assertion the same as any Rx packet. The functionality depends on the EICS setting schedule relative to the ITR intervals as described in Section 7.3.2.1.1.
- In LLI mode (GPIE.EIMEN = 1b), setting the *EICS* bits impact the RSC completion and interrupt assertion the same as any LLI Rx packet. Device behavior is described in Section 7.3.2.2.3 starting with the 2nd step.

### 7.3.1.3 Extended Interrupt Mask Set and Read (EIMS) Register, and Extended Interrupt Mask Clear (EIMC) Register

The Extended Interrupt Mask Set and Read (EIMS) register enables the interrupts in the EICR. When set to 1b, each bit in the EIMS register, enables its corresponding bit in the EICR. Software might enable each interrupt by setting bits in the EIMS register to 1b. Reading EIMS returns its value. Software might clear any bit in the EIMS register by setting its corresponding bit in the Extended Interrupt Mask Clear (EIMC) register. Reading the EIMC register does not return any meaningful data.

This independent mechanism of setting and clearing bits in the EIMS register saves the need for read modify write and also enables simple programming in multi-thread, multi-CPU core systems.

**Note:** The EICR register stores the interrupt events regardless of the state of the EIMS register.

### 7.3.1.4 Extended Interrupt Auto Clear Enable (EIAC) Register

Each bit in this register enables auto clearing of its corresponding bit in EICR following interrupt assertion. It is useful for Tx and Rx interrupt causes that have dedicated MSI-X vectors. When the Tx and Rx interrupt causes share an interrupt with the other or a timer interrupt, the relevant EIAC bits should not be set. Bits in the EICR register that are not enabled by auto clear, must be cleared by either writing a 1b to clear or a read to clear.

Note that there are no EIAC(1)...EIAC(2) registers. The hardware setting for interrupts 16...63 is always auto clear.

**Note:** Bits 29:16 should never be set to auto clear since they share the same MSI-X vector.

Writing to the EIAC register changes the setting of the entire register. In IOV mode, some of the bits in this register might affect VF functionality (VF-56...VF-63). It is recommended that software set the register in PF before VF's are enabled. Otherwise, a software semaphore might be required between the VF and the PF to avoid setting corruption.

### 7.3.1.5 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.

## 7.3.2 Interrupt Moderation

Interrupt rates can be tuned by the EITR register for reduced CPU utilization while minimizing CPU latency. In MSI or legacy interrupt modes, only EITR register 0 can be used. In MSI-X, non-IOV mode, the 82599 includes 64 EITR registers 0...63 that are mapped to MSI-X vectors 0...63, respectively. In IOV mode, there are an additional 65 EITR registers that are mapped to the MSI-X vectors of the virtual functions. The mapping of MSI-X vectors to EITR registers are described in Section 7.3.1.1.

The EITR registers include two types of throttling mechanisms: ITR and LLI. Both are described in the sections that follow.

### 7.3.2.1 Time-based Interrupt Throttling — ITR

Time-based interrupt throttling is useful to limit the maximum interrupt rate regardless of network traffic conditions. The ITR logic is targeted for Rx/Tx interrupts only. It is assumed that the timer, other and mail box (IOV mode) interrupts are not moderated. In non-IOV mode, all 64 interrupts can be associated with ITR logic. In IOV mode, the ITR logic is shared between the PF and VFs as shown in Figure 7-21. 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 µs units (at 1 Gb/s or 10 Gb/s link). When the ITR Interval equals zero, interrupt throttling is disabled and any event causes an immediate interrupt. The field is composed of nine bits enabling a range of 2 µs up to 1024 µs. These ITR interval times correspond to interrupt rates in the range of 500 K INT/sec to 980 INT/sec. When operating at 100 Mb/s link, the ITR interval is specified in 20 µs units.
  - Due to internal synchronization issues, the ITR interval can be shortened by up to 1 µs at
    10 Gb/s or 1 Gb/s link and up to 10 µs at 100 Mb/s link when it is triggered by packet write back or interrupt enablement or the last interrupt was LLI.
- ITR Counter partially exposed in the EITR registers — Down counter that is loaded by the ITR interval each time the associated interrupt is asserted.
  - The counter is decremented by one each 2 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 20 µs.
  - If an event happens before the counter is zero, it sets the EICR. The interrupt can be asserted only when the ITR time expires (counter is zero).
  - Else (no events during the entire ITR interval), the EICR register is not set and the interrupt is not asserted on ITR expiration. The next event sets the EICR bit and generates an immediate interrupt. See Section 7.3.2.1.1 for interrupt assertion when RSC is enabled.
  - Once the interrupt is asserted, the ITR counter is loaded by the ITR interval and the entire cycle re-starts. The next interrupt can be generated only after the ITR counter expires once again.

#### 7.3.2.1.1 ITR Affect on RSC Functionality

Interrupt assertion is one of the causes for RSC completion (see Section 7.11.6). When RSC is enabled on specific Rx queues, the associated ITR interval with these queues must be enabled and must be larger (in time units) than RSC delay. The ITR is divided to the two time intervals that are defined by the ITR interval and RSC delay. RSC completion is triggered after the first interval completes and the interrupt is asserted when the second interval completes.

The RSC_DELAY field is defined in the GPIE registers. *RSC Delay* can have one of the following eight values: 4 µs, 8 µs, 12 µs... 32 µs.

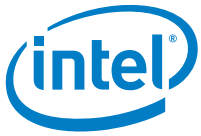
- The first ITR interval equals ITR interval minus RSC delay. The internal ITR counter starts at ITR interval value and counts down until it reaches the RSC delay value. Therefore, the ITR interval must be set to a larger value than the RSC delay.
- The second ITR interval equals RSC delay. The internal ITR counter continues to count down until it reaches zero.
- RSC completion can take some time (usually in the range of a few micro seconds). This time is composed by completing triggering latency and completing process latency. These delays should be considered when tuning the RSC delay. The clock frequency of the RSC completion logic depends on the link speed. As a result, the completion delay can as high as ~0.8 µs at 10 Gb/s link and ~8 µs at 1 Gb/s link. The RSC completion logic might take additional ~50 ns at 10 Gb/s link and ~0.5 µs at 1 Gb/s link per RSC. In addition, there is the PCIe bus arbitration latency as well as system propagation latencies from the device up to host memory.
- Recommended RSC delay numbers are: 8 µs at 10 Gb/s link and 28 µs at 1 Gb/s link.
- RSC is not recommended when operating at 100 Mb/s link.

Following are cases of packet reception with respect to the ITR intervals:

- Packets are received and posted (including their status) to the Rx queue in the first ITR interval. In this case, RSC completion is triggered at the end of the first ITR interval and the interrupt is asserted at the second interval expiration.
- a packet (and its status) is received and posted to the Rx queue only after the first ITR interval has expired (either on the second interval or after the entire ITR interval has expired). In this case, RSC completion is triggered almost instantly (other than internal logic latencies). The interrupt is asserted at RSC delay time after the non-coalesced Rx status is queued to be posted to the host.
- Due to internal synchronization issues, the RSC delay can be shorten by up to 1 µs when it is triggered by packet write back.

### 7.3.2.2 LLI

LLI provides low latency service for specific packet types, bypassing the ITR latency. LLIs are bound by a credit-based throttling mechanism that limits the maximum rate of low latency events that require a fast CPU response. Low latency events are triggered by the write back of the LLI packets. It then generates an immediate interrupt if LLI credits are not exhausted. See more details on the credit mechanism in the Section 7.3.2.2.2. Note also that in the case of RSC, the interrupt is not immediate as described in Section 7.3.2.2.3.



#### 7.3.2.2.1 LLI Filters and Other Cases

Following is a list of all Rx packets that are defined as low latency events (LLI packets):

- LLI by 5-tuple / TCP flags / frame size — The 82599 supports a set of 128 filters that initiate LLI by a 5-tuple value and frame size. An LLI is issued if any of the filters are set for LLI matches against the enabled fields of 5-tuple, TCP flags, and frame size. Configuration is done via the FTQF, SDPQF, L34TIMIR, DAQF, and SAQF registers as follows per filter (more details about these filters can be found in Section 7.1.2.5). Note that if a packet matches multiple 5-tuple filters an LLI is initiated if it is enabled by any of the matched filters:
- 5-tuple fields (protocol, IP address, port) and mask options for these fields
- Pool and pool mask
- SizeThresh — A frame with a length below this threshold triggers an interrupt. Unlike other fields, the *SizeThresh* field is shared by all filters (like there is a single copy of it). Matching the frame size is enabled by the *Size_BP* bit.
- *Size_BP* bit, when set to 0b, equates to a match that is performed against the frame size.
- LLI field — When set, an LLI is issued for packets that match the filter.
- LLI by Ethertype — The 82599 supports eight Ethertype filters. Any filter has an LLI action defined by the LLI field in the ETQS registers.
- LLI by VLAN priority — The 82599 supports VLAN priority filtering as defined in the IMIRVP register. Packets with VLAN header that have higher priority tagging than the one defined by IMIRVP register generates an LLI.
- LLI by FCoE — FCoE FCP_RSP packets can trigger LLI as defined in the FCRXCTRL.RSCINT bit. The 82599 identifies FCoE packets by the Ethertype filters defined by the ETQF registers. FCP_RSP packets recognition is explained in Section 7.13.3.3.10.

The 82599 might initiate an LLI when the receive descriptor ring is almost empty (Rx descriptors below a specific threshold). The threshold is defined by SRRCTL[n].RDMTS per Rx queue. This mechanism can protect against memory resources being used up during reception of a long burst of short packets.

#### 7.3.2.2.2 LLI Parameters

LLI generation is based on the following parameters:

- LLI Moderation bit in the EITR registers — When the LLI Moderation bit is cleared, any low latency event generates an immediate interrupt. When set, LLI moderation is based on the LLI credit and LLI interval.
- LLI Credit field in the EITR register — LLI packets might generate immediate interrupts as long as the LLI credits counter is greater than one (positive credit).
  - The credit counter is incremented by one on each LL interval with a maximum ceiling of 31 credits. It then stops incrementing.

  - If an LLI packet is received and the credit counter is greater than zero, an immediate interrupt is triggered internally. The interrupt is asserted externally when an interrupt is enabled (EIMS setting) and PCI credits are available. Once the interrupt is asserted, the credit counter is decremented by one Note that the counter never goes below zero.
  - LLI assertion might be delayed due to: interrupt enablement, lack of LLI credits or lack of PCI credits. Each time the interrupt is asserted, the LLI credit is decremented by one regardless of the number of received LLI packets and regardless if the ITR timer expires in the mean time.
  - If an LLI packet is received and the credit counter is zero (no credits), an interrupt can be asserted only on the next LL interval or when the ITR timer expires, whichever comes first.
  - The LLI credit counter is not affected by the ITR timer. Conversely, LLI assertion initializes the ITR timer to its timer interval.
  - Note that during nominal operation software may not need to access the LL credit field.
- LL interval is defined in units of 4 ms (at 1 Gb/s or 10 Gb/s link) in the GPIE register. At 100 Mb/s link speed, the LL interval is defined in units of 40 ms. This parameter defines the clock that increments the LLI credit counter. The maximum rate of the LLI interrupts per second is bound by the LL interval, which equals to 1/LL Interval. When LLI moderation is enabled, the ITR interval of the same interrupt must be greater than the LL interval.

#### 7.3.2.2.3 LLIs Affect on RSC Functionality

LLI packet reception requires instant CPU processing. Software might be able to access a specific descriptor only if all its proceeding descriptors complete. If RSC's are enabled, some of the preceding descriptors might be incomplete at the time that the LLI packet is received. Hardware overcomes this problem by:

- Following LLI packet completion, all RSC's on the same queue are completed as well.
- Then, the associated interrupt is asserted.
- Concurrently, hardware triggers RSC completion in all Rx queues associated with the same interrupt.
- Most likely these RSC(s) are completed to host memory after the interrupt is already asserted. In this case, it is guaranteed that an additional interrupt is asserted when the ITR expires.

## 7.3.3 TCP Timer Interrupt

### 7.3.3.1 Introduction

In order to implement TCP timers for IOAT, software needs to take action periodically (every 10 ms). Today, the driver must rely on software-based timers, whose granularity can change from platform to platform. This software timer generates a software NIC interrupt, which then enables the driver to perform timer functions, avoiding cache thrash and enabling parallelism. The timer interval is system-specific.

It would be more accurate and more efficient for this periodic timer to be implemented in hardware. The driver would program a timeout value (usual value of 10 ms), and each time the timer expires, hardware sets a specific bit in the EICR register. When an interrupt occurs (due to normal interrupt moderation schemes), software reads the EICR register and discovers that it needs to process timer events.

The timeout should be programmable by the driver, and the driver should be able to disable the timer interrupt if it is not needed.

### 7.3.3.2 Description

A stand-alone, down-counter is implemented. An interrupt is issued each time the value of the counter is zero.

Software is responsible for setting an initial value for the timer in the *Duration* field. Kick-starting is done by writing a 1b to the *KickStart* bit.

Following kick starting, an internal counter is set to the value defined by the *Duration* field. Then the counter is decreased by one each ms. When the counter reaches zero, an interrupt is issued. The counter re-starts counting from its initial value if the *Loop* field is set.

## 7.3.4 Mapping of Interrupt Causes

The following sections describe legacy, MSI and MSI-X interrupt modes.

### 7.3.4.1 Legacy and MSI Interrupt Modes

In legacy and MSI modes, an interrupt cause is reflected by setting one of the bits in the EICR register, where each bit reflects one or more causes. All interrupt causes are mapped to a single interrupt signal: either legacy INTA/B or MSI. This section describes the mapping of interrupt causes (that is a specific Rx or Tx queue event or any other event) to bits in the EICR.

The TCP timer and all other interrupt causes are mapped directly to EICR[30:16]. Note that the IVAR_MISC register is not used in legacy and MSI modes.

Mapping the Tx and Rx queues to interrupt bits in the EICR register is programmed in the IVAR registers as shown in Figure 7-21. 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 this interrupt cause.

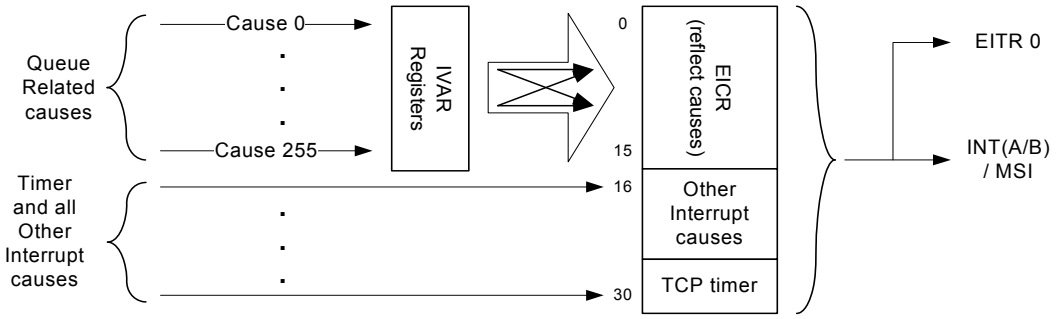



**Figure 7-21 Cause Mapping in Legacy and MSI Modes**

Mapping between the Tx and Rx queue to the IVAR registers is hardwired as shown in the Figure 7-22 below:

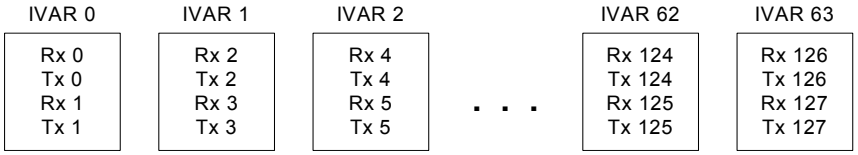
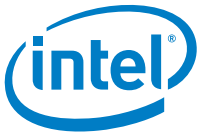



**Figure 7-22 Rx and Tx Queue Mapping to IVAR Registers**

## 7.3.4.2 MSI-X Mode in Non-IOV Mode

MSI-X defines a separate optional extension to basic MSI functionality. The number of requested MSI-X vectors is loaded from the MSI_X_N fields in the EEPROM up to maximum of 64 MSI-X vectors.

- Hardware indicates the number of requested MSI-X vectors in the table size in the MSI-X capability structure in the configuration space. This parameter is loaded from the *MSI_X _N* field in the EEPROM. The operating system might allocate any number of MSI-X vectors to the device from a minimum of one up to the requested number of MSI-X vectors.
- Enables interrupts causes allocation to the assigned MSI-X vectors. Interrupt allocation is programmed by the IVAR registers and are described in this section.
- Each vector can use an independent address and data value as programmed directly by the operating system in the MSI-X vector table.
- Each MSI-X vector is associated to an EITR register with the same index (MSI-X 0 to EITR[0], MSI-X 1 to EITR[1],...).



For more information on MSI-X, refer to the PCI Local Bus Specification, Revision 3.0.

MSI-X vectors can be used for several purposes:

1. Dedicated MSI-X vectors per interrupt cause (avoids the need to read the interrupt cause register).
2. Load balancing by MSI-X vectors assignment to different CPUs.
3. Optimized interrupt moderation schemes per MSI-X vector using the EITR registers.

The MSI-X vectors are used for Tx and Rx interrupt causes as well as the other and timer interrupt causes. The remainder of this section describes the mapping of interrupt causes (such as a specific Rx or Tx queue event or any other event) to the interrupts registers and the MSI-X vectors.

The TCP timer and other events are reflected in EICR[30:16] the same as the legacy and MSI mode. It is then mapped to the MSI-X vectors by the IVAR_MISC register as shown in Figure 7-23. The IVAR_MISC register includes two entries for the timer interrupt and an additional entry for all the other causes. The structure of each entry is as follows:

INT_Alloc - Defines the MSI-X vector (0...63) assigned to this interrupt cause.

INT_Alloc_val - Valid bit for the this interrupt cause.

The Tx and Rx queues are associated to the IVAR0...IVAR63 the same as legacy and MSI mode shown in Figure 7-22. The Tx and Rx queues are mapped by the IVAR registers to EICR(1),...EICR(2) registers and MSI-X vectors 0...63 illustrated in Figure 7-23. The IVAR entries have the same structure as the IVAR_MISC register previously shown. Each bit in EICR(1...2) registers is associated to MSI-X vector 0...63 as follows:

- EICR(i).bit_num is associated to MSI-X vector (n x 32 + bit_num).
- The legacy EICR[15:0] mirror the content of EICR(1)[15:0]. In the same manner the lower 16 bits of EICS, EIMS, EIMC, EIAC, EIAM mirror the lower 16 bits of EICS(1), EIMS(1), EIMC(1), EIAC(1), EIAM(1). The use of these registers depends on the number of assigned MSI-X interrupts as follows:
- 16 Tx and Rx Interrupts - When using up to 16 Tx and Rx interrupts, software might access the Tx and Rx interrupt bits in the legacy EICR, EICS,... registers.
- More than 16 Tx and Rx Interrupts - When using more then 16 Tx and Rx interrupts, software must use EICS(1)...EICS(2), EIMS(1)...EIMS(2),... In the later case, software should avoid modifying the lower 16 bits in the SEIC, EICS... registers when it accesses the higher bits of these registers as follows:
  - EICR, EICS, EIMS and EIMC — When software programs the higher 16 bits of these registers, it should set their lower 16 bits to zero’s keeping the EICR(1), EICS(1), EIMS(1) and EIMC(1) unaffected.
  - EIAM — When software programs the higher 16 bits, it should keep the lower 16 bits at their previous setting so the EIAM(1) is unaffected.
  - EIAC — When software programs the higher 16 bits, it should set the lower 16 bits to one’s.

Single MSI-X vector - If the operating system allocates only a single MSI-X vector, the driver might use the non-MSI-X mapping method (setting the GPIE.Multiple_MSIX to 0b). In this case, the INT_Alloc field in the IVAR registers might define one of the lower 16 bits in the EICR register while using MSI-X vector 0. The IVAR_MISC should be programmed to MSI-X vector 0.

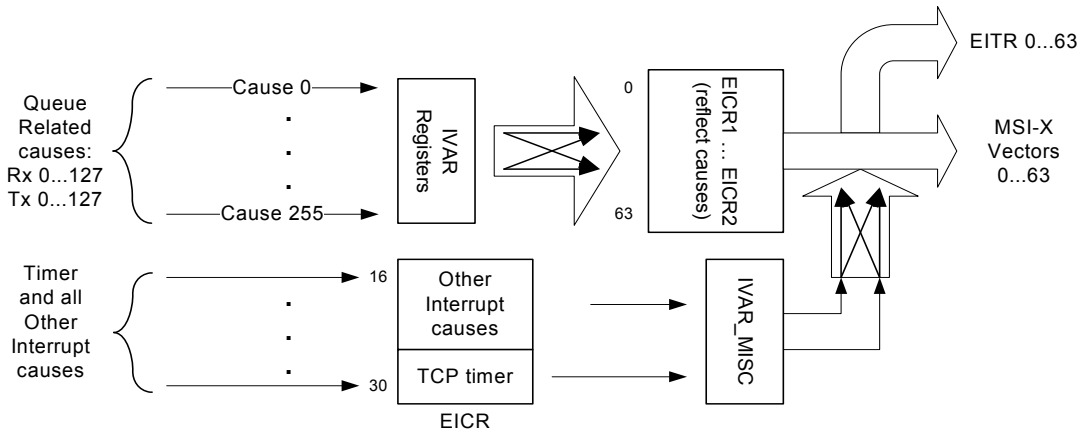



**Figure 7-23 Cause Mapping in MSI-X Mode (non-IOV)**

## 7.3.4.3 MSI-X Interrupts In IOV Mode

In IOV mode, interrupts must be implemented by MSI-X vectors. The 82599 supports up to 64 virtual functions VF(0...63). Each VF can generate up to three MSI-X vectors. The number of requested MSI-X vectors per VF is loaded from the *MSI-X Table* field in the EEPROM. It is reflected in the *Table Size* field in the PCIe MSI-X capability structure of the VF's. In addition, the PF requires its own interrupts. The number of requested MSI-X vectors for the PF is loaded from the *MSI_X_N* fields in the EEPROM up to maximum of 64 MSI-X vectors. It is reflected in the *Table Size* field in the PCIe MSI-X capability structure.

### 7.3.4.3.1 MSI-X Vectors Used by Physical Function (PF)

PF is responsible for the timer and other interrupt causes that include the VM to PF mailbox cause (explained in the virtualization sections). These events are reflected in EICR[30:16] and MSI-X vectors are the same as the non-IOV mode (illustrated in Figure 7-21). When there are less than the maximum possible active VF's, some of the Tx and Rx queues can be associated with the PF. These queues can be used for the sake of additional VM's serviced by the hypervisor (the same as VMDq mode) or some Kernel applications handled by the hypervisor. Tx and Rx mapping to the IVAR registers is shown in Figure 7-22 and mapping to the EICR, EICR(1),...EICR(2) registers as well as the MSI-X vectors is shown in Figure 7-23. See Section 7.3.4.3.3 for MSI-X vectors mapping of PF and VF's to the EITR registers.

**Note:** Software should not assign MSI-X vectors in the PF to Tx and Rx queues that are assigned to other VF's. In the case that VF's become active after the PF used the relevant Tx and Rx queues, it is the responsibility of the PF driver to clear all pending interrupts of the associated MSI-X vectors.

### 7.3.4.3.2 MSI-X Vectors Used by Virtual Functions (VFs)

Each of the VFs in IOV mode is allocated separate IVAR(s) called VFIVAR registers, and a separate IVAR_MISC called VFIVAR_MISC register. The VFIVAR_MISC maps the mailbox interrupt of the VF to its VFEICR and the MSI-X vector. The VFIVAR registers map the Tx and Rx interrupts of the VF to its VFEICR and the MSI-X vector. The mapping is similar to the mapping in the PF as shown in Figure 7-24 with the following comments:

- Each VF cannot have more than three MSI-X vectors. It has only three active bits in the VFEICR register while VFEICR.bit_num is associated with MSI-X vector (bit_num).
- The Tx and Rx interrupt can be mapped only to MSI-X 0 and MSI-X 1 (associated with VFEICR.0 and VFEICR.1).
- The mailbox interrupt can be mapped to any of the three MSI-X vectors. However, when all three of them are allocated by the operating system, software should map the mailbox to MSI-X 2 (associated with VFEICR.2). This rule should be kept since only VFEICR.0 and VFEICR.1 have ITR registers (VFEITR-0 and VFEITR-1).
- Association between the Tx and Rx queues and the VFIVAR registers is shown in the Figure 7-24, Figure 7-25 and Figure 7-26 for IOV-64 (64 VF's), IOV-32 and IOV-16. The colored boxes in the figures show the mapping between VF Rx and Tx queues to VFIVAR registers while the dashed boxes show the physical IVAR registers and the associated physical Rx and Tx queues.

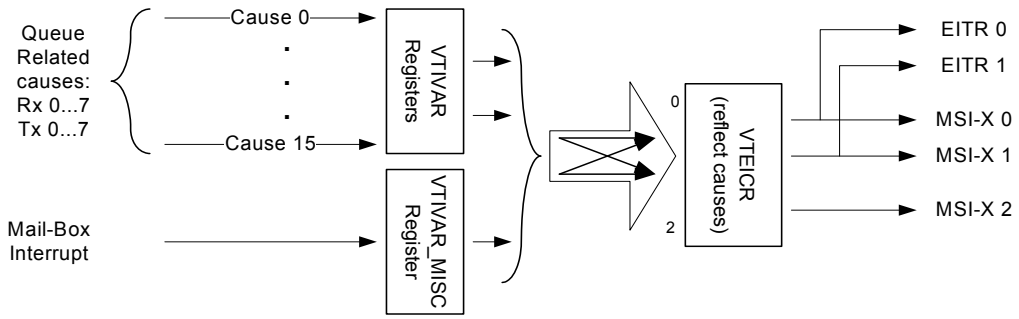



**Figure 7-24 VF Interrupt Cause Mapping (MSI-X, IOV)**

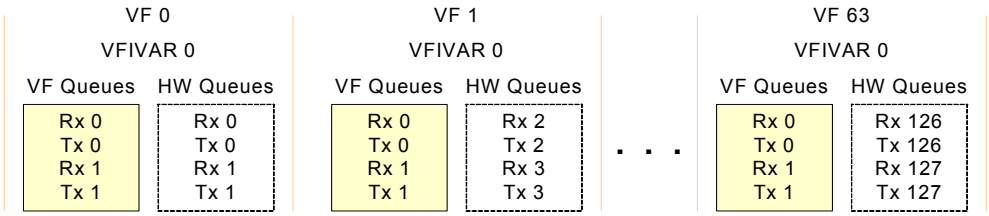



**Figure 7-25 VF Mapping of Rx and Tx Queue to VFIVAR in 64 VF's Mode**

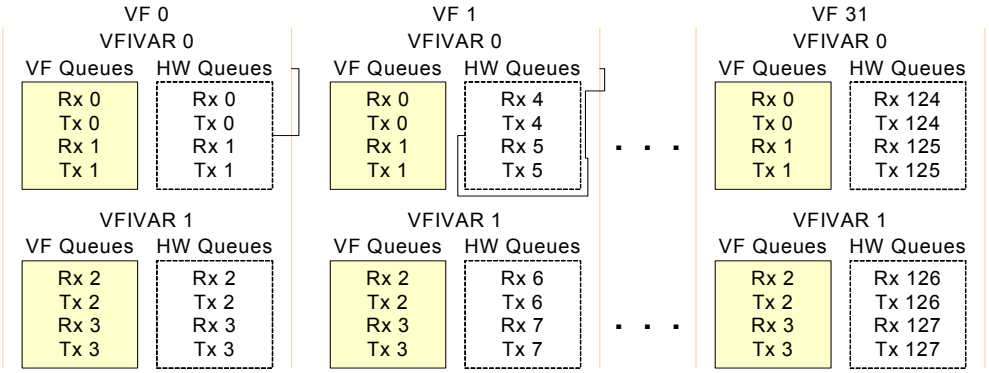



**Figure 7-26 VF Mapping of Rx and Tx Queue to VFIVAR in 32 VF's Mode**

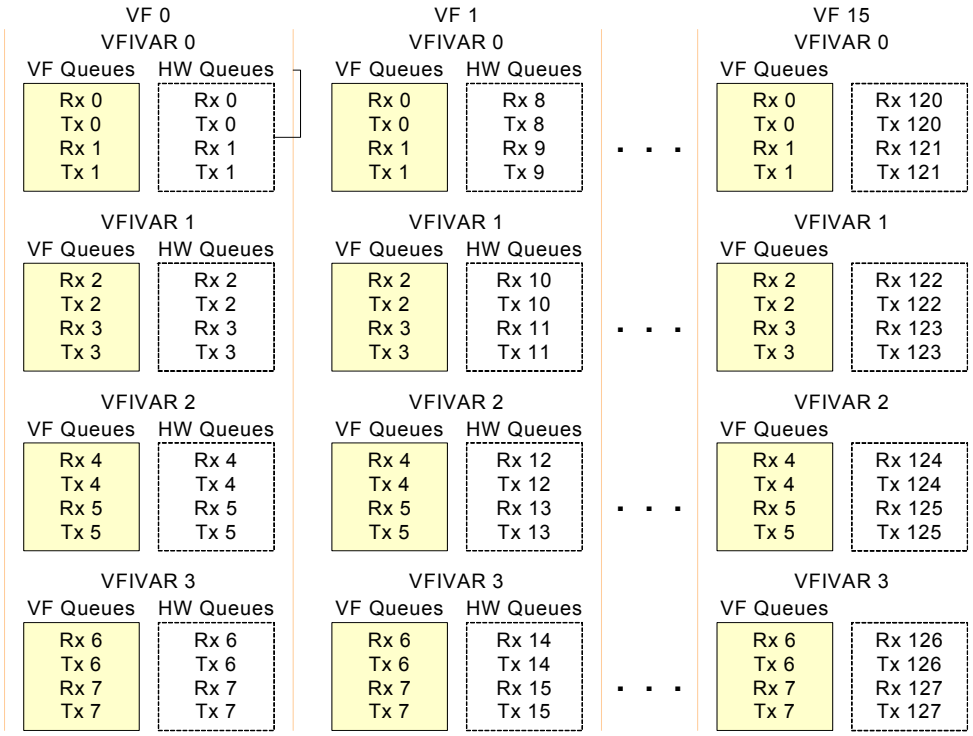
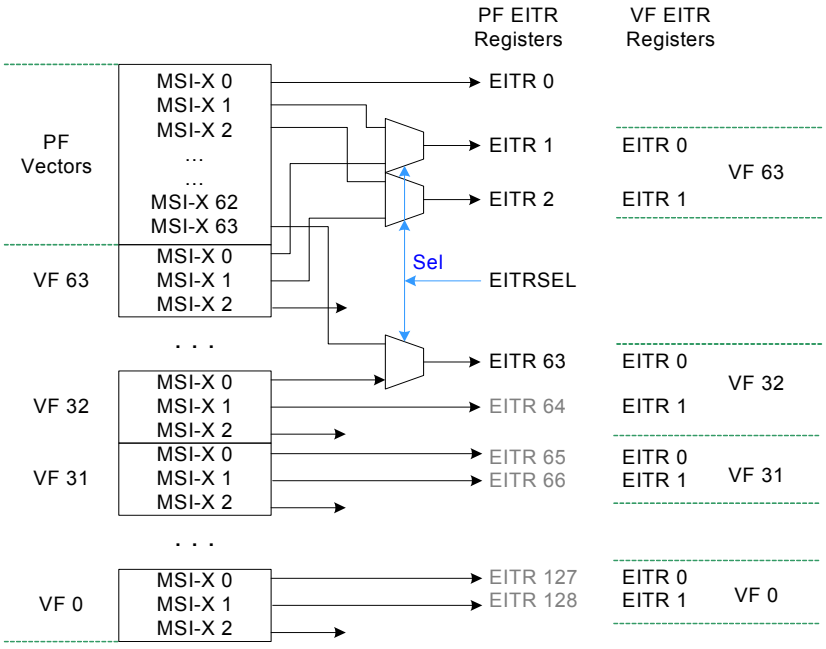



**Figure 7-27 VF Mapping of Rx and Tx Queue to VFIVAR in 16 VF's Mode**

### 7.3.4.3.3 MSI-X Vectors Mapping to EITR

EITR registers are aimed for Tx and Rx interrupt throttling. In IOV mode, the Tx and Rx queues might belong to either the PF or to the VF's. EITR(1...63) are multiplexed between the PF and the VF's as configured by the EITRSEL register. Figure 7-28 and Table 7-47 show the multiplexing logic and required software settings. For any active VF

(starting from VF32 and above), software should program the matching bit in the EITRSEL to 1b. For any EITR that belongs to a VF, software should not map any interrupt causes in the PF to an MSI-X vector that is associated with the same EITR register.




MSI-X 2 on each VF has no associated EITR register. It is useful for the mailbox interrupts that do not require interrupt moderation.

**Figure 7-28 PF / VF MSI-X Vectors Mapping to EITR**

**Table 7-47 PF / VF MSI-X Vectors Mapping Table to EITR Registers**

| VM Active | EITRSEL.N Setting | MSI-X Routing to EITR |
|---|---|---|
| Non-IOV or VF(32...63) inactive | EITRSEL must be set to 0x0000 | MSI-X(1...63) -> EITR(1...63) |
| VF(32) active | EITRSEL[0] must be set to 1b | VF(32) MSI-X(0) -> EITR(63) |
| VF(33) active | EITRSEL[1] must be set to 1b | VF(33) MSI-X(1) -> EITR(62) VF(33) MSI-X(0) -> EITR(61) |
| VF(34) active | EITRSEL[2] must be set to 1b | VF(34) MSI-X(1) -> EITR(60) VF(34) MSI-X(0) -> EITR(59) |
| | . . . | |
| VF(62) active | EITRSEL[30] must be set to 1b | VF(62) MSI-X(1) -> EITR(4) VF(62) MSI-X(0) -> EITR(3) |
| VF(63) active | EITRSEL[31] must be set to 1b | VF(63) MSI-X(1) -> EITR(2) VF(63) MSI-X(0) -> EITR(1) |

# 7.4 802.1q VLAN Support

The 82599 provides several specific mechanisms to support 802.1q VLANs:

- Optional adding (for transmits) and stripping (for receives) of IEEE 802.1q VLAN tags.
- Optional ability to filter packets belonging to certain 802.1q VLANs.

## 7.4.1 802.1q VLAN Packet Format

The following table compares an untagged 802.3 Ethernet packet with an 802.1q VLAN tagged packet:

| 802.3 Packet | #Octets | 802.1q VLAN Packet | #Octets |
|---|---|---|---|
| DA | 6 | DA | 6 |
| SA | 6 | SA | 6 |
| Type/Length | 2 | 802.1q Tag | 4 |
| Data | 46-1500 | Type/Length | 2 |
| CRC | 4 | Data | 46-1500 |
| | | CRC* | 4 |

**Note:** The CRC for the 802.1q tagged frame is re-computed, so that it covers the entire tagged frame including the 802.1q tag header. Also, maximum frame size for an 802.1q VLAN packet is 1522 octets as opposed to 1518 octets for a normal 802.3z Ethernet packet.

## 7.4.2 802.1q Tagged Frames

For 802.1q, the *Tag Header* field consists of four octets comprised of the Tag Protocol Identifier (TPID) and Tag Control Information (TCI); each taking two octets. The first 16 bits of the tag header makes up the TPID. It contains the protocol type that identifies the packet as a valid 802.1q tagged packet.

The two octets making up the TCI contain three fields as follows:

- User Priority (UP)
- Canonical Form Indicator (CFI). Should be set to 0b for transmits. For receives, the device has the capability to filter out packets that have this bit set. See the *CFIEN* and *CFI* bits in the VLNCTRL
- VLAN Identifier (VID)

| Octet 1 | | | | | | | | Octet 2 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UP | | | CFI | VID | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

## 7.4.3 Transmitting and Receiving 802.1q Packets

Since the 802.1q tag is only four bytes, adding and stripping of tags can be done completely in software. (In other words, for transmits, software inserts the tag into packet data before it builds the transmit descriptor list, and for receives, software strips the 4-byte tag from the packet data before delivering the packet to upper layer software). However, because adding and stripping of tags in software adds overhead for the host, the 82599 has additional capabilities to add and strip tags in hardware. See Section 7.4.3.1 and Section 7.4.3.2.

### 7.4.3.1 Adding 802.1q Tags on Transmits

Software might instruct the 82599 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 82599 inserts a VLAN tag into the packet that it transmits over the wire. The Tag Protocol Identifier — TPID (VLAN Ether Type) field of the 802.1q tag comes from the DMATXCTL.VT, and the Tag Control Information (TCI) of the 802.1q tag comes from the VLAN field of the legacy transmit descriptor or the *VLAN Tag* field of the advanced data transmit descriptor.

### 7.4.3.2 Stripping 802.1q Tags on Receives

Software might instruct the 82599 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 82599 strips the 4-byte VLAN tag from the packet, and stores the TCI in the *VLAN Tag* field of the receive descriptor.

The 82599 also sets the *VP* bit in the receive descriptor to indicate that the packet had a VLAN tag that was stripped. If the RXDCTL.VME bit is not set, the 802.1q packets can still be received if they pass the receive filter, but the VLAN tag is not stripped and the VP bit is not set.

## 7.4.4 802.1q VLAN Packet Filtering

VLAN filtering is enabled by setting the VLNCTRL.VFE bit to 1b. If enabled, hardware compares the *Type* field of the incoming packet to a 16-bit field in the VLAN Ether Type (VET) register. If the *VLAN Type* field in the incoming packet matches the VET register, the packet is then compared against the VLAN Filter Table Array for acceptance.

The VLAN filter register VTFA, is a vector array composed of 4096 bits. The VLAN ID (VID) is a 12-bit field in the VLAN tag that is used as an index pointer to this vector. If the VID in a received packet points to an active bit (set to 1b), the packet matches the VLAN filter. The 4096-bit vector is comprised of 128 x 32 bit registers. The upper 7 bits of the VID selects one of the 128 registers while the lower 5 bits map the bit within the selected register.

Two other bits in the VLNCTRL register, *CFIEN* and *CFI*, are also used in conjunction with 802.1q VLAN filtering operations. *CFIEN* enables the comparison of the value of the *CFI* bit in the 802.1q packet to the Receive Control register *CFI* bit as acceptance criteria for the packet.

**Note:** The *VFE* bit does not effect whether the VLAN tag is stripped. It only effects whether the VLAN packet passes the receive filter.

## 7.4.5 Double VLAN and Single VLAN Support

The 82599 supports a mode where all received and sent packets have at least one VLAN tag in addition to the regular tagging that might optionally be added. In this document, when a packet carries two VLAN headers, the first header is referred to as an outer VLAN and the second header as an inner VLAN header (as listed in the table that follows). This mode is used for systems where the near end switch adds the outer VLAN header containing switching information. This mode is enabled by the following configuration:

- This mode is activated by setting the DMATXCTL.GDV and the *Extended VLAN* bit in the CTRL_EXT register.
- The Ethertype of the VLAN tag used for the additional VLAN is defined in the *VET EXT* field in the EXVET register.

### Cross functionality with Manageability

The 82599 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 82599 is set to double VLAN mode as well as the VLAN Ethertype(s). When operating in a double VLAN mode, control packets sent by the manageability controller with no VLAN headers should not activate any hardware offload other than LinkSec encapsulation.

**Table 7-48 Transmit Handling of Packets with VLAN Header(s)**

| MAC Address | Outer VLAN | Inner VLAN | L2 Payload | Ethernet CRC |
|---|---|---|---|---|

Transmit functionality on the outer VLAN header

- A packet with a single VLAN header is assumed to have only the outer VLAN.
- The outer VLAN header must be added by software as part of the Tx data buffers.
- Hardware does not relate to the outer VLAN header other than the capability of skipping it for parsing inner fields.
- Hardware expects that any transmitted packet (see the disclaimer that follows) has at least the outer VLAN added by software. For any offload that hardware might provide in the transmit data path, hardware assumes that the outer VLAN is present. For those packets that an outer VLAN is not present, any offload that relates to inner fields to the Ethertype might not be provided.

Transmit functionality on the inner VLAN header

- The inner VLAN header can be added by software in one of the following methods:
  - The header is included in the transmit data buffers.
  - The 16-bit portion of the header that includes the priority tag, CFI and VLAN ID are included in the transmit descriptor. The VLAN Ethertype is taken from the *VT* field in the DMATXCTL register.
  - In IOV mode, the priority tag, CFI and VLAN ID can be taken from the PFVMVIR (see details in Section 7.10.3.9.2)
- Hardware identifies and skips the VLAN header for parsing inner fields.
- DCB — The user priority of the packet is taken from the inner VLAN. The traffic class is dictated by the Tx queue.
- Pool Filtering — Destination pool(s) and anti-spoofing functionality is based on the Ethernet MAC address and inner VLAN (if present) as described in Section 7.10.3.4 and Section 7.10.3.9.2.

### 7.4.5.1 Receive Handling of Packets with VLAN Header(s)

Receive functionality on the outer VLAN header

- If the packet carries a single VLAN header, it is assumed as the outer header and is treated as such.
- 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. In the case of mismatch, the packet is handled as unknown packet type at which time hardware does not provide any offloads other than LinkSec processing. Also, hardware skips the header for parsing inner fields and provides any supported offload functions.
- The outer VLAN header is posted as is to the receive data buffers.

Receive functionality on the inner VLAN header

- Hardware checks the Ethertype of the inner VLAN header against the programmed value in the VLNCTRL.VET. VLAN header presence is indicated in the Status.VP bit in the Rx descriptor.

- If the RXDCTL.VME is set, the inner VLAN is stripped by hardware while the priority tag, CFI and VLAN ID are indicated in the *VLAN Tag* field in the Rx descriptor.
- Hardware identifies and skips the VLAN header for parsing inner fields and provides any supported offload functions.
- L2 packet filtering is based on the VLAN ID in the inner VLAN header.
- Pool Filtering — Destination pool(s) are defined by the Ethernet MAC address and inner VLAN (if presence) as described in Section 7.10.3.3.
- DCB — The user priority of the packet is taken from the inner VLAN. In the absence of inner VLAN, the packet is assumed as user priority 0 (least priority). See Section 7.4.5.2 for the absence of any VLAN headers.

### 7.4.5.2 Packets with no VLAN headers in Double VLAN Mode

There are some cases when packets might not carry any VLAN headers, even when extended VLAN is enabled. A few examples for packets that might not carry any VLAN header are: flow control and priority flow control, LACP, LLDP, GMRP, and optional 802.1x packets. When it is expected to transmit untagged packets by software in double VLAN mode the software must not enable VLAN anti-spoofing and VLAN validation nor transmit to receive switching.

#### Transmitted functionality

DCB — The traffic class in the Tx data path is directed by the Tx queue of the transmitted packet.

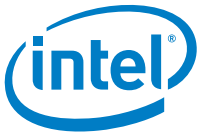
Transmit offload functionality — Software should not enable any offload functions other than LinkSec.

#### Receive functionality

DCB — Assume user priority 0 (lowest priority).

Receive offload functionality — pool and queue are selected by the Ethernet MAC address or ETQF/ETQS registers. LinkSec offload is functional. Filtering to host and manageability remains functional.

The *Extended VLAN* bit in the CTRL_EXT register and DMATXCTL.GDV are not set. Hardware expects that Rx and Tx packets might not carry a VLAN header or a single VLAN header. Hardware does not relate to the programming of the *VET EXT* field in the EXVET register. Tx and Rx handling of packets with double VLAN headers is unexpected.



### 7.4.5.3 Packet Priority in Single and Double VLAN Modes

This section summarizes packet handling in both single and double VLAN modes. The user priority of a packet is meaningful in DCB mode when multiple traffic classes are enabled as well as LLIs. The user priority is extracted from the packets as listed in the following table.

**Table 7-49 Packet Handling in Single and Double VLAN Modes**

| Packet type | Single VLAN | Double VLAN |
|---|---|---|
| Packet with no VLAN | User priority = 0 | User priority = 0 |
| Packet with 1 VLAN | The user priority field in the VLAN header in the packet | User priority = 0 |
| Packet with 2 VLANs | Erroneous case: The user priority field in the **outer** VLAN header in the packet | The user priority field in the **inner** VLAN header in the packet |

# 7.5 Direct Cache Access (DCA)

DCA is a method to improve network I/O performance by placing some posted inbound writes directly within CPU cache. DCA potentially eliminates cache misses due to inbound writes.

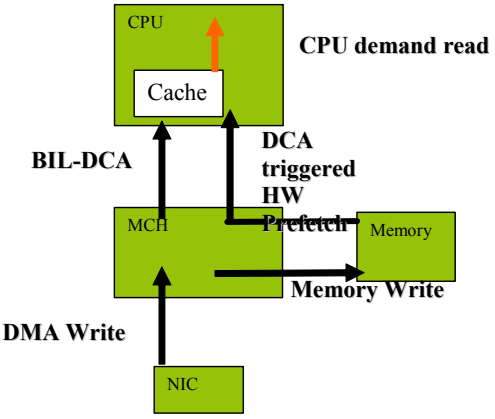



**Figure 7-29 Diagram of DCA Implementation on FSB System**

As Figure 7-29 illustrates, DCA provides a mechanism where the posted write data from an I/O device, such as an Ethernet NIC, can be placed into CPU cache with a hardware pre-fetch. This mechanism is initialized upon a power good reset. A device driver for the I/O device configures the I/O device for DCA and sets up the appropriate CPU ID and bus ID for the device to send data. The device then encapsulates that information in PCIe TLP headers, in the tag field, to trigger a hardware pre-fetch to the CPU cache.

DCA implementation is controlled by separate registers (DCA_RXCTRL and DCA_TXCTRL) for each transmit and receive queue. In addition, a DCA disable bit can be found in the DCA_CTRL register, and a DCA_ID register can be found for each port, in order to make the function, device, and bus numbers visible to the driver.

The DCA_RXCTRL and DCA_TXCTRL registers can be written by software on the fly and can be changed at any time. When software changes the register contents, hardware applies changes only after all the previous packets in progress for DCA have completed.

However, in order to implement DCA, the 82599 has to be aware of the IOAT version used. The software driver initializes the 82599 to be aware of the bus configuration. The *DCA Mode* field in the DCA_CTRL register defines the system configuration:

1. Legacy DCA: The DCA target ID is derived from CPU ID.
2. DCA 1.0: The DCA target ID is derived from APIC ID.

Both modes are described as follows.

# 7.5.1 PCIe TLP Format for DCA

Figure 7-30 shows the format of the PCIe TLP for DCA.

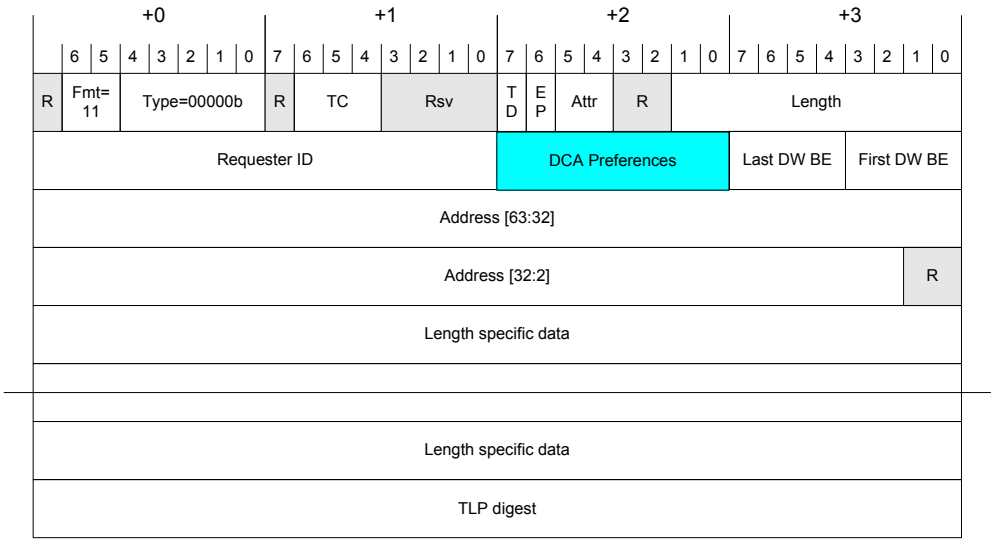



**Figure 7-30 PCIe Message Format for DCA**

The DCA preferences field has the following formats.

For legacy DCA systems:

| Bits | Name | Description |
|---|---|---|
| 0 | DCA indication | 0b = DCA disabled.<br>1b = DCA enabled. |
| 4:1 | DCA target ID | The DCA target ID specifies the target cache for the data. |

For DCA 1.0 systems:

| Bits | Name | Description |
|---|---|---|
| 7:0 | DCA target ID | 0000.0000b: DCA is disabled.<br>Other: Target core ID derived from APIC ID.25915 |

**Note:** All functions within a the 82599 have to adhere to the tag encoding rules for DCA writes. Even if a given function is not capable of DCA, but other functions are capable of DCA, memory writes from the non-DCA function must set the tag field to 00000000b.

# 7.6 LEDs

The 82599 implements four output drivers intended for driving external LED circuits per port. Each of the four LED outputs can be individually configured to select the particular event, state, or activity, which is indicated on that output. In addition, each LED can be individually configured for output polarity as well as for blinking versus non-blinking (steady-state) indication.

The configuration for LED outputs is specified via the LEDCTL Register. Furthermore, the hardware-default configuration for all LED outputs can be specified via EEPROM fields thereby supporting LED displays configurable to a particular OEM preference.

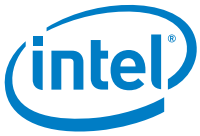
Each of the four LED's can be configured to use one of a variety of sources for output indication. For more information on the MODE bits see LEDCTL register (see Section 8.2.3.1.6).

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 source functions slightly different from the others when BLINK is enabled. The LED is:

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



# 7.7 Data Center Bridging (DCB)

See Section 4.6.11 for the DCB configuration sequence and Section 11.5 for the expected performance of DCB functionality.

## 7.7.1 Overview

DCB is a set of features that improve the capability of Ethernet to handle multiple traffic types (such as LAN, storage, IPC) by answering the various needs of those types. DCB enables multiple traffic types that have different requirements of packet delivery, bandwidth allocation and delay. Each traffic type can have one or several user priorities, or Traffic Classes (TCs). For example, IPC might have a high priority class for synchronization messages between servers and lower priority traffic class for bulk traffic exchange between servers. Most of the DCB functions impact the transmit traffic generated from the end node to the network (traffic generation). The receive data path needs be compliant with the requirements of DCB and provide the required functions as a traffic termination point.

DCB system requirements include:

1. Bandwidth grouping — For effective multiplexing that simulates a separate link for the separate types of traffic, DCB requires that traffic types be recognized as groups in the bandwidth and priority handling by nodes in the network. Traffic types are associated to Bandwidth Groups (BWGs). The system needs to be able to allocate bandwidth to the BWGs in a way that emulates that group being on its own separate link.
2. Bandwidth fairness — DCB multiplexing functions (transmit) and de-multiplexing functions (receive) need to guarantee minimum allocation of bandwidth to traffic types and traffic classes. Fairness between groups comes first, then fairness between TCs. If system resources (such as PCIe bandwidth) limit total throughput, then the available bandwidth should be distributed among consumers proportionally to their allocations.
3. Latency of operation — DCB multiplexing and de-multiplexing functions need to allow minimum latency for some TCs. Arbitration mechanisms, packet buffers, descriptor queues and flow control algorithm need to be defined and designed to allow this. The best example is the control/sync traffic in IPC. The expectation for end-to-end IPC control is measured in the low 10's of µs for the 82599 and is expected to drop to a singe digit µs later. Some elements in multimedia traffic also bear similar requirements. Although some of the end-to-end delays can be quite long, the individual contribution of the arbitration in each node must be kept to a minimal. End-to-end budgets do not comprehend large delays within transmission nodes.
4. No-drop behavior and network congestion management — The end node must be able to guarantee no-drop behavior for some TCs or some packets within TCs. As a termination point in receive, it is the end node’s responsibility to properly control traffic coming from the network to achieve this end. For traffic generation in transmit, the end station must be able to positively respond to flow control from the network as the must have tool to prevent packet drop. It also needs to participate in network congestion management.

5. Compatibility with existing systems. — The DCB implementation needs to be usable by IT using known configurations and parameters, unless new ones are made expressly available. For example, DCB implementation cannot assume new knowledge regarding bandwidth allocation of traffic types that do not have known bandwidth requirements.

The layer 2 features of DCB implemented in the 82599 are:

1. Multi-class priority arbitration and scheduling — The 82599 implements an arbitration mechanism on its transmit data path. The arbitration mechanism allocates bandwidth between TC in BWGs and between Virtual Machines (VMs) or Virtual Functions (VFs) in a virtualization environment. The BWGs can be used to control bandwidth and priority allocated to traffic types. Typically BWGs should be used to represent traffic types. TC arbitration allows control of bandwidth and priority control within BWGs as well as within the entire link bandwidth. The arbitration is designed to respect the bandwidth allocations to BWGs. The priority allocation allows minimization of delay for specific TCs. In the 82599, TCs and user priorities are processed on a packet-by-packet basis based on the 802.1p identifier in the 802.1Q-tag.
2. Class-based flow control (PFC — Priority Flow Control) — Class-based flow control functionality is similar to the IEEE802.3X link flow control. It is applied separately to the different TCs.
   - Transmit response to class-based flow control from the ingress switch it is connected to.
   - Receive class-based flow control commands to the switch in response to packet buffers filling status.
3. DMA queuing per traffic type — Implementation of the DCB transmit, minimization of software processing and delays require implementation of separate DMA queues for the different traffic types. The 82599 implements 128 descriptor queue in transmit and 128 descriptor queues in receive.
4. Multiple Buffers — The 82599 implements separate transmit and receive packet buffers per TC.
5. Rate-limiter per Tx queue — limiting the transmit data rate for each Tx queue.

### Latency requirements:

Quantitative latency requirements are defined for a single 64-byte packet at the highest priority traffic class. Latency is defined separately for transmit and receive:

- Transmit latency — measured from a tail update until the packet is transmitted on the wire. It is assumed that a single packet is submitted for this TC and its latency is then measured in the presence of traffic belonging to other TCs.
- Receive latency — measured from packet reception from the wire and until the descriptor is updated on PCIe.

Figure 7-31 shows the latency requirements as previously defined.

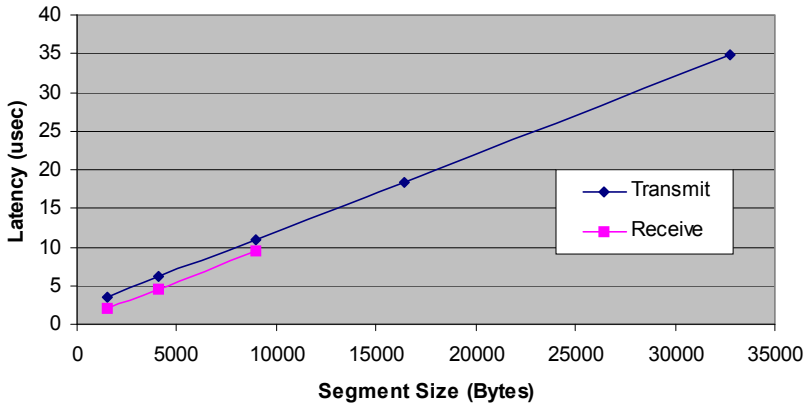



**Figure 7-31 Latency Requirements**

**Note:** In DCB mode, it is assumed all traffic is tagged (contains a VLAN header) — except for Layer2 frames with special Ethernet MAC addresses that goes untagged. GMRP frames (special Ethernet MAC addresses starting with 0x0180c20000) must, however, go tagged. Untagged packets must be delivered to the host and are assumed to belong to User Priority 7.

Note that any BCN signaling is terminated at the network's edge. At initialization, every component exchanges its capabilities with its peer via a Capability Exchange (DCX) protocol carried over dedicated Link Layer Discovery Protocol (LLDP) frames. Support for these protocols is transparent to the hardware implementation, and as a result, is not described in this document.

## 7.7.2 Transmit-side Capabilities

**Note:** When configured for DCB mode, the the 82599 driver should only use advanced transmit descriptors. Refer to Section 7.2.3.2.3. Using legacy transmit descriptors is not allowed.

### 7.7.2.1 Transmit Rate Scheduler (RS)

#### 7.7.2.1.1 Basic Rate Control Operation

Rate control is defined in terms of maximum payload rate, and not in terms of maximum packet rate. This means that each time a rate controlled packet is sent, the next time a new packet can be sent out of the same rate controlled queue is relative to the packet size of the last packet sent. The minimum spacing in time between two starts of packets sent from the same rate controlled queue is recalculated in hardware on every packet again, by using the following formula:

```
MIFS = PL x RF
```

Where:

- Packet Length (PL), is the Layer2 length (such as without preamble and IPG) in bytes of the previous packet sent out of that rate controller. It is an integer ranging from 64 to 9 K (at least 14-bits).
- RF = 10 Gb/s / target rate (rate factor) is the ratio between the nominal link rate and the target maximum rate to achieve for that rate-controlled queue. It is dynamically updated either by software via the RTTBCNRC register, or by hardware via the rate-drift mechanism, as described in Section 7.7.2.1.2. It is a decimal number ranging from 1 to 1,000 (10 Mb/s minimum target rate). For example, at least 10-bits before the hexadecimal point and 14-bits after as required for the maximum packet length by which it is multiplied. For links at 1 Gb/s, the rate factor must be configured relatively to the link speed, replacing 10 Gb/s by 1 Gb/s in the above formula.
- Minimum Inter Frame Space (MIFS) is the minimum delay in bytes units, between the starting of two Ethernet frames issued from the same rate-controlled queue. It is an integer ranging from 76 to 9,216,012 (at least 24-bits). In spite of the 8-byte resolution provided at the internal data path, the byte-level resolution is required here to maintain an acceptable rate resolution (at 1% level) for the small packets case and high rates.

**Note:** It might be that a pipeline implementation causes the MIFS calculated on a transmitted packet to be enforced only on the subsequent transmitted packet.

Time Stamps — A rate-scheduling table contains the accumulated interval MIFS, for each rate-controlled descriptor queue separately, and is stored as an absolute Time Stamp (TS) relative to an internal free running timer. The TS value points to the time in the future at which a next data read request can be sent for that queue. Whenever updating a TimeStamp:

```
TimeStamp(new) = TimeStamp(old) + MIFS
```

When a descriptor queue starts to be rate controlled, the first interval MIFS value is equal to 0 (TS equal to the current timer value) — without taking into account the last packet sent prior to rate control. When the TS value stored becomes equal to or smaller than the current free running timer value, it means that the switch is on and that the queue starts accumulating compensation times from the past (referred as a negative TS). When the TS value stored is strictly greater than the current free running timer value, it means that the switch is off (referred as a positive TS.

```
(CurrentTime) < TimeStamp       <-->    switch is “off”

(CurrentTime) >= TimeStamp      <-->    switch is “on”
```

MMW — The ability to accumulate negative compensation times that saturates to a Max Memory Window (MMW) time backward. MMW size is configured per TC via the *MMW_SIZE* field of the RTTBCNRM register, and is expressed in 1 KB units of payload, ranging from 0 up to 2 KB units (at least 11-bits). The MMW_SIZE configured in KB units of payload has to be converted in time interval MMW_TIME expressed in KB, before a new time stamp is checked for saturation. It is computed for each queue according to its associated Rate Factor (RF) using the following formula:

```
MMW_TIME = MMW_SIZE x RF
```

**Note:** MMW_TIME is rounded by default to a 1 KB precision level and must be at least 31 bits long. Hence, the time stamp byte-level values stored must be at least 32-bits long for properly handling the wrap-around case. 29-bits are required for the internal free running timer clocked once every 8 bytes.

Whenever updating a time stamp verify:

```
TimeStamp(old) + MIFS >= (CurrentTime) – MMW_TIME
```

and then the time stamp is updated according to the non-saturated formula:

```
TimeStamp(new) = TimeStamp(old) + MIFS
```

Otherwise, enforced saturation by assigning:

```
TimeStamp(new) = (CurrentTime) – MMW_TIME + MIFS
```

**Note:** Non-null MMW introduces some flexibility in the way controlled rates are enforced. It is required to avoid overall throughput losses and unfairness caused by rate-controlled packets over-delayed, consequently to packets inserted in between. Between two rate-limited packets spaced by at least the MIFS interval, non-rate-limited packets, or rate-limited packets from other rate-controlled queues, might be inserted. If a rate controlled packet has been delayed by more time than it was required for rate control (because of arbitration between VMs or TCs), the next MIFS accumulates from the last time the queue was switched on by the rate scheduling table — and not from the current time. Refer to Figure 7-32 for visualizing the effect of MMW.

MMW_SIZE set to 0 must be supported as well.

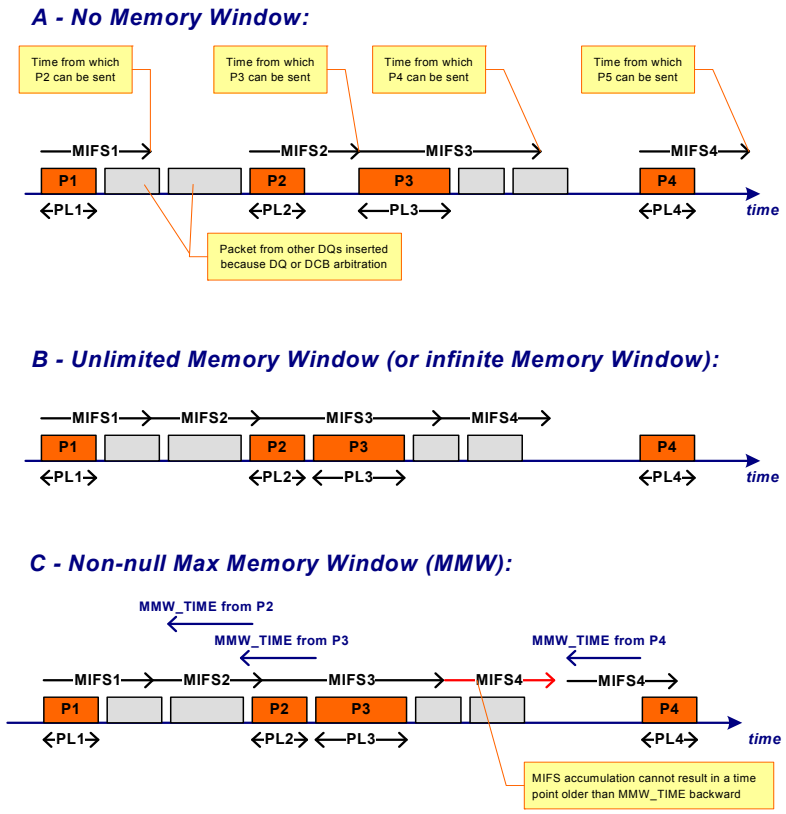



**Figure 7-32 Minimum Inter-Frame Spacing for Rate-Controlled Frames (in Orange)**

### 7.7.2.1.2 Rate Drift

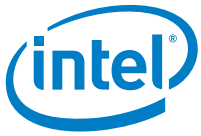
Periodically, at fixed intervals in time, the rate factors of all rate-controlled queues must be increased internally by a small amount. The periodic interval in time at which rate drift mechanism is triggered is configured via the *DRIFT_INT* field in RTTBCNRD register. the rate-drift mechanism done in hardware is enabled by setting the *DRIFT_ENA* bit in RTTBCNRD register; otherwise, it is assumed that it is handled by software.

The rate-drift mechanism is essential for fairness and rate recovery of rate-controlled flows reduced to very low rates.

**Note:** For providing accurate rate-drift intervals, the rate-drift mechanism must be started immediately once the interval in time has elapsed — without waiting for the next time stamp table scan cycle to start.

Rates are increased in a multiplicative manner, by multiplying the rates with a fixed value slightly *greater* than unity. It is thus similar to multiplying the rate factors by a fixed value slightly *smaller* than unity, which is referred to as the drift factor. It is configured via the *DRIFT_FAC* field in RTTBCNRD register. The rate-drift mechanism saturates if the full line rate has been recovered (when the rate factor has been decreased down to unity):

```
Rate-Factor(new) = max(1, Rate-Factor(old) x Drift-Factor)
```



For example, if a periodic rate increase of 3% is desired, then a drift factor of 1/ 1.03=0.97087... must be configured.

**Note:** One disadvantage of the multiplicative rate increase method results in smaller increases for low-rated flows and larger increases for high-rated flows. An additive method has been envisaged instead, increasing the rate factors by small additive steps on each interval, but it has been dropped off because it had poor chances of being standardized by IEEE 802.1au.

A queue that has recovered the full line rate via the rate-drift mechanism (rate factor decreased down to one) is not considered as a rate-controlled queue, its corresponding *RS_ENA* bit in the RTTBCNRC register must be internally self-cleared, and it should stop tagging its frames with the RLT option. Refer to Section 7.7.2.1 for further details on CM-tagging.

### 7.7.2.2 User Priority to Traffic Class Mapping

DCB-enabled software is responsible for classifying any Tx packet into one of the eight 802.1p user priorities, and to assure it is tagged accordingly by either software or hardware. The driver dispatches classified Tx traffic into the Tx queues attached to the proper TC, according to a UP-to-TC Tx mapping policy decided by the IT manager.

**Note:** When configured for DCB mode or when using the Tx rate-limiting functionality, the 82599 software driver should only use advanced transmit descriptors. Refer to Section 7.2.3.2.3. DO NOT use legacy transmit descriptors.

***Caution:*** When translating XON/XOFF priority flow control commands defined per UP into commands to the Tx packet buffers, the 82599 is required to use the same UP-to-TC Tx mapping table that software is using. The RTTUP2TC register must be configured by software accordingly. Refer to Section 3.7.7.1.3 for details on priority flow control.

### 7.7.2.3 VM-Weighted Round-Robin Arbiters

The 82599 implements VM-weighted arbiter(s) for virtualized environments and according to the following case:

- If DCB is enabled, there is one such arbiter per TC, arbitrating between the descriptor queues attached to the TC (one queue per VF). Bandwidth allocation to VMs is enforced at the descriptor plane, per each TC separately. The VM arbiter instantiated for each TC is aimed to elect the next queue for which a data read request is sent in case the TC is elected for transmission by the next level arbiter. For example, the TC weighted strict priority descriptor plane arbiter.
- If DCB is disabled, there is one single VM weighted arbiter, arbitrating between pools of descriptor queues, where a pool is formed by the queues attached to the same VF. Bandwidth allocation to VMs is enforced at the descriptor plane, between the pools, where queues within a pool are served on a frame-by-frame round-robin manner.

Refer to the different arbitration schemes where virtualization is enabled, as shown in Figure 7-17.

**Note:** In this section, VM is considered a generic term used to refer to the arbitrated entity, whether it is a Tx descriptor queue within the TC or whether it is a pool of Tx descriptor queues. In the later case, pool parameters are allocated only to the lowest indexed queue within the pool, taken as the representation of the entire pool.

#### 7.7.2.3.1 Definition and Description of Parameters

Credits: Credits regulate the bandwidth allocated to VMs. As part of the Weighted Round Robin (WRR) algorithm, each VM has pre-allocated credits. They are decremented upon each transmission and replenished cyclically. The ratio between the credits of the VMs represents the relative bandwidth percentage allocated to each VM (within the TC for the DCB enabled case). The 82599 effectively maintains one table that represents these ratios. Note that credits can get negative values down to the maximum sized frame allowed on the TC/pool.

**Note:** The absolute value of the credits has no direct bearing on the allocation of bandwidth. The ratio between the credits does. However, since credits can accumulate only up to twice the credit refills, the refills should be allocated as low as possible but must be set greater than the maximum sized frame allowed on the TC or on the pool.

WRR: The algorithm implemented in the 82599 for VM arbitration.

Table 7-50 (T1) defines the VMs and their bandwidth allocation. The following elements are defined in this table:

**Table 7-50 Bandwidth Allocation to VMs**

| T1: VM Bandwidth Allocation | | | |
|---|---|---|---|
| $VM_N$ | VM Refill | VM Max Credits | VM Min Credits |
| 0 | MSS:1,024 KB | 2xMSS:2,048 KB | -MSS |
| 1 | MSS:1,024 KB | 2xMSS:2,048 KB | -MSS |
| 2 | MSS:1,024 KB | 2xMSS:2,048 KB | -MSS |
| 3 | MSS:1,024 KB | 2xMSS:2,048 KB | -MSS |
| ... | | | |
| 15 | MSS:1,024 KB | 2xMSS:2,048 KB | -MSS |
| † Due to a pipelined implementation, the VM credits range is enlarged by one MSS, beyond negative limits. | | | |
| †† All values are implemented with 64-byte granularity (a value of one corresponds to 64 bytes of credit). | | | |

VM: Configuration — The unique Tx descriptor queue attached to a VF within a TC, or the pool of Tx descriptor queues attached to the same VF.

VM Credit Refill: Configuration — The 82599's WRR algorithm implement credit refill as the technique for percentage allocation to VMs. The credits refill are added to each VM credit count on each completion of the full round of the algorithm (after all the VMs had their chance to send what they had in store or at least one frame).

The 82599's driver needs to calculate the VM credit refill to match the percent allocated through management (such as in the MIB). Since the WRR arbitration is self timed, the ratio between the credits refill is the only defining parameter for the VMs. However, the refills must be greater or equal to the maximum sized frame allowed on the TC or on the pool in order to guaranty transmission of at least one frame on each recycling round. The 82599 allows a value of 1.5 KB to 1,024 KB for a dynamic range of x1000.

VM Maximum Accumulated Credits: Deducted from Refill Configuration — In order to prevent the use of stale credits, the number of credits each VM can accumulate upon refill is limited. The credits for each VM can only reach twice their refill. The maximum range for the credits is thus -9.5 KB to 2,048 KB, assuming negative credits can accumulate up to a maximum sized frame (9.5 KB [9728 bytes] if jumbo frames are allowed), and where positive credits can accumulate up to twice the maximum credit refill.

VM Credits: Run time parameter — VM credits is a running counter for each VM. It holds the number of available credits in 64-byte units. The algorithm runs sequentially between the VMs and enables transmission for those VMs that have enough pending credits (their credit number is greater than zero).

**Table 7-51 Registers Allocation for Tx VM Arbiters**

| Attribute | Tx VM Arbiter |
|---|---|
| VM Control registers | RTTDT1C |
| VMC Status registers | RTTDT1S |
| VM credit refill | CRQ |
| VM credits | CCC |

### 7.7.2.3.2 WRR Arbiter Algorithm

Round Robin — The round-robin aspect of the VM WRR arbiter resides in the fact that once a VM has been granted for a data read request, the next VMs are checked in a cyclic round-robin order, even if the granted VM still has credits for another data read request.

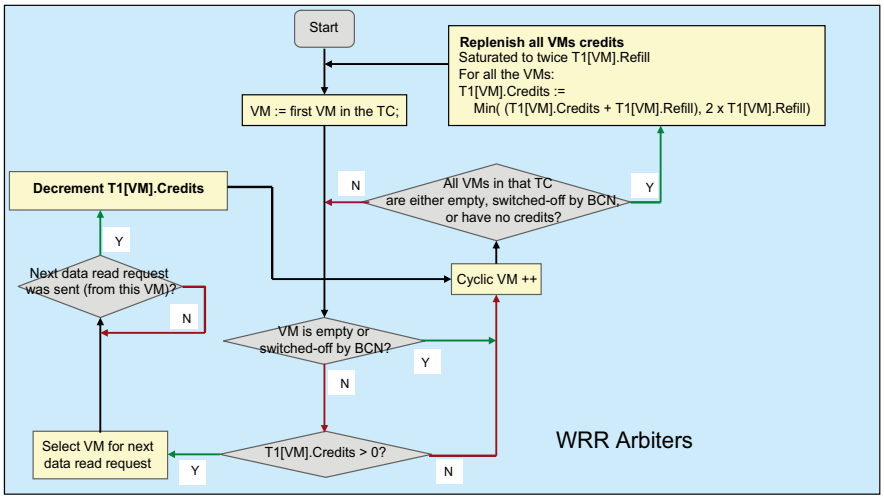



**Figure 7-33 Tx VM WRR Arbiters Operation**

#### 7.7.2.4 Tx TC Weighted Strict Priority Arbiters

In DCB, multiple traffic types are essentially multiplexed over the Ethernet 10 Gb/s network. There is a need to allow different behavior to different traffic flows as they pass through multiple Ethernet switches and links. For example, for LAN, SAN and IPC connections are consolidated on a single Ethernet link. Each traffic type (BWG) is guaranteed the bandwidth it has been allocated and is prevented from usurping bandwidth from other types. However, if a BWG does not use its bandwidth, that bandwidth is made available to the other BWGs. The same holds for TCs within a BWG. If allocated some bandwidth, TCs are guaranteed to have it, and if unused, that bandwidth can be used by the other TCs within the BWG. Information regarding bandwidth allocation for some TCs might not be available. In the case of LAN, the entire allocation of bandwidth within the LAN link is typically undefined in today's networks. The arbitration scheme includes Group Strict Priorities (GSP) to cover for that. TCs for which the *GSP* bit is set are limited by the total throughput allocated to their BWG rather than to TC allocation.

Link bandwidth is divided among the BWGs for guaranteed minimum behavior. For example: LAN:
4 Gb/s, SAN: 4 Gb/s, IPC: 2 Gb/s. The 82599 supports two types of bandwidth allocation within BWGs. TCs can be either allocated bandwidth or be used as in strict priority. If a TC does not use all of its allocated bandwidth, that bandwidth is recycled to other TCs in the BWG.

The 82599 implements two replications of the weighted TC arbiter:

- One in the descriptor plane, arbitrating between the different descriptor queues, deciding which queue is serviced next. It is aimed to enforce the TC bandwidth allocation and prioritization scheme for a case when PCIe bandwidth is smaller than the link bandwidth.
- A second in the packet plane, at the output of the packet buffers, deciding which packet to transmit next. It is aimed to enforce the TC bandwidth allocation and prioritization scheme for a case when PCIe bandwidth is greater than the link bandwidth.

The condition for entry into the bandwidth allocation algorithm sequence differs for the descriptor and data arbiters:

- The descriptor arbiter queries whether there is at least one queue attached to a TC that is not empty, not switched off by the rate scheduler, with positive VM weighted arbiter credits (when relevant), and for which the destined packet buffer has room for the worst case maximum sized frame. This last condition is controlled by RTTDCS.BPBFSM.
- The packet arbiter queries whether the packet buffer has a packet to send and whether it is not stalled by priority flow control.

##### 7.7.2.4.1 Definition and Description of Parameters

User Priority (UP): There are eight traffic priorities, determined by 802.1p tag bits on an Ethernet link. The Q-Tag field holds UP's. Per 802.1p, Priority #7 is the highest priority. User priorities are assigned by the application or the system to certain usage classes, such as manageability, IPC control channel, VoIP. An additional bit within the VLAN TCI field (bit 5 - DEI)defines whether the packet has a no drop requirement. This bit is not being used by DCB mechanisms.

User Bandwidth Group (UBWG): a user bandwidth group is a management parameter that is a binding of user priorities into bandwidth groups for provisioning purposes. The hardware implementation does not recognize the UBWG entity.

Traffic Class (TC): incoming packets are placed in traffic classes. Per the DCB functional specification, there might be a 1:1 mapping between UP and TC, or more than one priority can be grouped into a single class. Such grouping does not cross boundaries of traffic BWGs. the 82599 implements eight or four TCs and maps them to UP's according to a programmable register. This provides the best flexibility for the IT manager. However, when more than one UP is mapped to the same TC, they must have the same no-drop policy network wide.

Packet Buffer (PB): TCs are mapped to packet buffers in a straightforward 1:1 mapping. Packets are also placed in packet buffers based on their class assignments.

Traffic Bandwidth Group (BWG): For bandwidth allocation and grouping, one or more TC can be grouped into a Traffic Bandwidth Group (BWG). A BWG is a logical association at a node, and has no markings inside a packet header. End stations and switches are independent in their definition and allocation of grouping of different TCs. Consistency of behavior throughout the network is handled by the UBWG provisioning mechanism.

One or more TCs can be grouped in a BWG. BWGs are allocated a percentage of bandwidth of available Ethernet link. The allocated bandwidth for BWG can be further divided among the TCs that are inside the BWG.

Credits: Credits regulate the bandwidth allocated to BWGs and TCs. As part of the WSP algorithm, each BWG and TC has pre-allocated credits. Those are decremented upon each transmission and replenished cyclically. The ratio between the credits of the BWGs represents the relative bandwidth percentage allocated to each BWG. The ratio between the credits of the TCs represents the relative bandwidth percentage allocated to each TC within a BWG. The 82599 effectively maintains one table that represents both ratios at once. Note that credits can get negative values down to the maximum sized frame allowed on the TC.

Maximum Credit Value: The maximum credit value establishes a limit for the running sum of credits allotted to a class or group. This value prevents stacking up stale credits that can be added up over a relatively long period of time and then used by TCs all at once, altering fairness and latency.

**Note:** The absolute value of the credits has no direct bearing on the allocation of bandwidth. The ratio between the credits does. However, the absolute value might have substantial impact on the algorithm behavior. Larger absolute values can impact the latency of high priority queues and their ability to serve bursts with minimum latency, whereas too small credit values might impact the correct functionality in presence of jumbo frames. The speed of the algorithm implementation should also be taken into account. The value of the maximum credit limit are also in principle not part of the main WSP algorithm. However, they impact the fairness of bandwidth reallocation between queues in case some queues do not transmit the full amount they have been permitted to. Also, small values prevent correct functionality of jumbo frames. From high-level simulations it appears that credits should be allocated as low as possible based on the speed of the algorithm. Maximum credit values for TCs should be 1.5x to 2x the size of the maximum entity expected in that TC.

Weighted Strict Priority (WSP): The algorithm implemented in the 82599 for TC arbitration.

Group Strict Priority (GSP): Refer to the sections that follow for details.

Link Strict Priority (LSP): Refer to the sections that follow for details.

Table 7-52 (T2) defines the TCs, their association to BWGs and their bandwidth allocation. The following elements are defined in this table:

**Table 7-52 Bandwidth Allocation to TCs and BWGs**

| T2: Traffic Class Bandwidth Allocation Within a BWG | | | | | | |
|---|---|---|---|---|---|---|
| $TC_N$ | BWG | TC Refill | TC Max Credits | LSP | GSP | TC Min Credits (according to GSP 0/1) |
| 0 | 1 | 64 bytes to 32 KB | 64 bytes to 256 KB | 0/1 | 0/1 | -MSS / -2,048 KB |
| 1 | 2 | 64 bytes to 32 KB | 64 bytes to 256 KB | 0/1 | 0/1 | -MSS / -2,048 KB |
| 2 | 0 | 64 bytes to 32 KB | 64 bytes to 256 KB | 0/1 | 0/1 | -MSS / -2,048 KB |
| 3 | 1 | 64 bytes to 32 KB | 64 bytes to 256 KB | 0/1 | 0/1 | -MSS / -2,048 KB |
| 4 | 2 | 64 bytes to 32 KB | 64 bytes to 256 KB | 0/1 | 0/1 | -MSS / -2,048 KB |
| 5 | 3 | 64 bytes to 32 KB | 64 bytes to 256 KB | 0/1 | 0/1 | -MSS / -2,048 KB |
| 6 | 1 | 64 bytes to 32 KB | 64 bytes to 256 KB | 0/1 | 1/1 | -MSS / -2,048 KB |
| 7 | 0 | 64 bytes to 32 KB | 64 bytes to 256 KB | 0/1 | 0/1 | -MSS / -2,048 KB |

† Due to a pipelined implementation, the TC credits range is enlarged by one MSS in both directions, beyond its positive and negative limits.

†† All values are implemented with 64-byte granularity (a value of one corresponds to 64 bytes of credit).

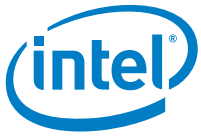
TC: Configuration — The traffic type associated to the packet buffer where incoming packets are kept before transmission (or discard — not implemented in transmit in the 82599). TC7 is the highest priority TC.

BWG: Configuration — Traffic BWG that a TC belongs to.

TC Credit Refill: Configuration — The 82599's WSP algorithm implements credit refill as the technique for traffic class percentage allocation. The credits refill are added to each TC credit count on each completion of the full round of the algorithm (after all TCs had their chance to send what they had in store and if they had credits for it).

The 82599's driver needs to calculate the TC Credit refill to match the percentage allocated through management (in the MIB). The TC credit refill table includes, in one table both the TC and the BWG. Since the WSP arbitration is self timed, the ratio between the credits refill is the only defining parameter for the TC and BWG. The absolute values of the refill have significance as to the rate of distribution between the queues and can have impact on latency and on the momentary stability of the bandwidth fairness. Results from simulations indicate that the quantum for the refill should be small to prevent large swings. It should not be too small as to create overhead in the mechanism due to the execution time; however. The 82599 allows a value of 64 bytes to 32 KB, A dynamic range of x500. The usage model is likely to call for a smallest refill value of 256 bytes to 512 bytes. This leaves a dynamic range of 80x-200x. For example, if a queue is assigned 1% and this is translated into a 256-byte increment, another assigned with 99% would have a refill value of 25344 bytes.

TC MaxCredit: Configuration — In order to prevent use of stale credits, the number of credits each TC can accumulate upon refill is limited. The TC credit can only reach MaxCredit, beyond that its value gets recycled to other queues. Refer to the recycling mode for more details. The maximum range for the MaxCredit is 256 KB. This high range value was inherited from the 82598 that had to deal with entire LSOs, but is not really necessary for the 82599.



**Note:** Full testing of many values for maximum value is unnecessary. Hierarchical testing should be applied. For the random test, four to six values should be sufficient. It is important that the testing includes values that are relevant to the interesting zone of maximum value. For example, in the 0.8x to 2x the relevant largest entity in the class. Although class with an expected shorter packet could use a smaller MaxCredit, it is recommended that testing fully covers the cases where all the classes have similar values of MaxCredit, as it is a possible variant use of the algorithm.

Link Strict Priority (LSP): Configuration — If set, this bit specifies that this TC can transmit without any restriction of credits. This effectively means that this TC can take up the entire link bandwidth, unless preempted by higher priority traffic. If this bit is set, then TC.CreditRefill must be set to 0b to ensure fair bandwidth allocation. Preferably, the algorithm implementation should disregard non-zero values in all its calculations.

Group Strict Priority (GSP): Configuration — This bit defines whether strict priority is enabled or disabled for this TC within its BWG. If TC.GSP is set to 1b, the TC is scheduled for transmission using strict priority. It does not check for availability of TC.Credits. It does check whether the BWG of this TC has credits (such as the amount of traffic generated from this TC is still limited by the BWG allocated for the BWG (T3. BWGP). If this bit is set, then TC.CreditRefill values can be set to 0b, if a non-zero value is configured, TC credits are reduced first from the GSP TC and if reached to zero from other TCs in the group, if the refill credits are configured to zero the TC credits are reduced from the other TCs in the BWG.

**Note:** Since the TC.GSP parameter relates to individual TCs, some BWGs might have both TC's with bandwidth allocation and TC's with GSP. This is a hybrid usage mode that is complex to validate and is possibly secondary in importance.

Usage Note — It is possible that a TC using LSP dominates the link bandwidth if there are no packets waiting and eligible in higher priority TC's. To guarantee correct bandwidth allocation, all TC's with the unlimited bit set should be in the same traffic BWG (high priority group). Note that this is different than a typical DCB deployment considered where BWG is created with functional grouping, like LAN, SAN, and IPC etc. TC's with the *LSP* bit set should be the first to be considered by the scheduler (the first TC's). For example, from queue 7 to queue 5 with the other five TC's for groups with bandwidth allocation. These are not strict requirements, if these rules are not followed, undesirable behavior could occur in some cases. A group containing only TC's with the unlimited bit set effectively has zero BWG credits since TC's with the LSP unlimited bit set should have TC credits set to zero. Use of LSP / TC.GSP should be restricted to UP/TC's that service trusted applications.

TC Credits: Run-time parameter — TC credits is a running algebraic counter for each TC. It holds the number of available credits in 64-byte units. The algorithm runs sequentially between the TC's and enables transmission for those TCs that have positive pending credits. Note that credits can get negative values down to the maximum sized frame allowed on the TC.

Table 7-53 (T3) defines the hierarchy of BWG similarly to T2 defining the hierarchy within the BWG's. T3 implementation should include eight rows.

**Table 7-53 Line Bandwidth Allocations to Bandwidth Groups (BWG) (with example)**

| T3: Line bandwidth allocation to Traffic Bandwidth Groups (BWG) | | | | |
|---|---|---|---|---|
| **BWG** | **BWG Refill Credits** | **BWG MaxCredit** | **BWG Credit** | **Description** |
| 0 | = Σ[TC] Credit Refill | = Σ[TC]. MaxCredit | -/+ 2,048 KB | IPC |
| 1 | = Σ[TC] Credit Refill | = Σ[TC]. MaxCredit | -/+ 2,048 KB | SAN |
| 2 | = Σ[TC] Credit Refill | = Σ[TC]. MaxCredit | -/+ 2,048 KB | LAN |
| 3 | = Σ[TC] Credit Refill | = Σ[TC]. MaxCredit | -/+ 2,048 KB | Manageability |
| - | | | | |
| - | | | | |
| - | | | | |
| 7 | | | | |

† Due to a pipelined implementation, the BWG credits range is enlarged by one MSS in both directions, beyond its positive and negative limits.

BWG: Configuration — This is the number of the traffic BWG that is three bits wide. This field corresponds to the TC.BWG field in Table 7-52.

BWG Refill Credits: A virtual number — The credits provisioned for this BWG. The credits ratio between the BWG's should reflect the ratio of bandwidth between the BWG's. In the actual implementation, this number is the sum of the credit Refills of the TC's associated with this BWG.

BWG MaxCredit: A virtual number — The maximum credits for a BWG. Credits in the BWG.Credit counter are limited to this value. Credits that should have been refilled above this value are lost. In effect, due to the self-timed cyclic nature of the WSP algorithm, those credits are distributed between all BWG's. In the actual implementation, this number is the sum of the MaxCredit of the TC's associated with this BWG.

BWG.Credit: Run-time parameter — A running algebraic counter that is decremented for each transmission. At the end of each cycle, this counter is synchronized with the sum of the TC.Credit counter associated with this BWG. The synchronization algorithm depends on the recycling mode. refer to the sections that follow for details about arbitration configurations. Note that credits can get negative values down to the maximum sized frame allowed on the BWG.

### 7.7.2.4.2 Arbiters Conventions

The WSP scheme previously described is written with the data plane arbiter in mind. However, the same scheme is used by the transmit descriptor plane arbiter and a subset of it is used by the receive data arbiter. To distinguish between the two arbiters, attributes of the each arbiter are prefixed as depicted in Table 7-54.

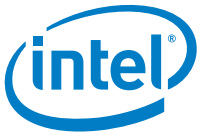


**Table 7-54 Attributes of Tx Arbiters**

| Attribute | Tx Packet Arbiter | Tx Descriptor Arbiter |
|---|---|---|
| TC | P-TC | D-TC |
| BWG | P-BWG | D-BWG |
| TC Credit Refill | P-TC Credit Refill | D-TC Credit Refill |
| TC MaxCredit | P-TC MaxCredit | D-TC MaxCredit |
| LSP | P-LSP | D-LSP |
| GSP | P-GSP | D-GSP |
| TC Credits | P-TC Credits | D-TC Credits |
| BWG Refill Credits | P-BWG Refill Credits | D-BWG Refill Credits |
| BWG MaxCredits | P-BWG MaxCredits | D-BWG MaxCredits |
| BWG Credits | P-BWG Credits | D-BWG Credits |

Table 7-55 lists the register fields that contain the relevant attributes from Table 7-54.

**Table 7-55 Registers Allocation for Tx TC Arbiters**

| Attribute | Tx Packet Arbiter | Tx Descriptor Arbiter |
|---|---|---|
| TC Control registers | RTTPT2C | Reserved |
| TC Status registers | RTTPT2S | RTTDT2S |
| BWG | BWG | BWG |
| TC credit refill | CRQ | CRQ |
| TC MaxCredit | MCL | MCL |
| LSP | LSP | LSP |
| GSP | GSP | GSP |
| TC credits | CCC | CCC |

### 7.7.2.4.3 Tx TC WSP Arbitration Configurations

RR / WSP: Global Configuration bits — When this bit is set, the arbitration is in WSP mode. When reset, the arbitration is in flat frame-based round robin mode. In RR mode, one frame is transmitted from each packet buffer in its turn. BWG and TC parameters do not apply.

Recycle Mode: Global Configuration bits.

Architecture Overview of Recycle — As a result of GSP transmits and TCs that reach their maximum credit limit, the credit count of a BWG might not match the total credit count of its TCs (refer to the sections that follow for more details). It is not merely an arithmetic issue. The WSP algorithm, dual hierarchy behaves as a maximum allocation algorithm within the BWG's and minimum allocation algorithm between the BWG's. Since the recycle is self timed, when a BWG does not transmit all of its allocated bandwidth within a cycle, at the end of the cycle its bandwidth is in effect reallocated to all BWG's. This results in a minimum allocation behavior. Inside the BWG's; however, this notion of self timing does not exist. Some explicit mechanism is required to recycle bandwidth within a BWG rather than to all the BWG's — The requirement to have a minimum allocation behavior.

- A BWG credit count might not match the total credit count of its TCs in the following cases:
  - A TC is defined as GSP — when a GSP is selected, the BWG credits are decremented but no TC is deducted. Therefore, the BWG credit count would be lower that the credit count of its TCs.
  - Max credits during refill — If a TC reaches its max credits value during refill, then some credits are lost for that TC. However, the BWG for that TC is provided with the full refill count. Therefore, the BWG credit count would be higher that the credit count of its TCs.
- One bit per TC arbiter governs the recycle mode of the WSP algorithm:
  - 0: No Recycling — At the end of each full arbitration cycle all TC's are refilled with their TC.Refill up to their TC.MaxCredits values. All BWG.Credit are loaded by the sum of the BWG TC.Credit.
  - 1: Recycle — TC credits for TC's that have reached their maximum are recycled to other TC's of the BWG. The operation is calculated based on the BWG.Credit and the TC.Credits after their refill. The difference between them is the BWG.Recycle value.

    Positive BWG.Recycle - The recycle algorithm adds credits from the BWG.Recycle to the TC.Credits starting from the highest priority TC in the BWG down, considering the TC.MaxCredit, until BWG.Recycle is zero.

    Negative BWG.Recycle - The recycle algorithm subtracts credits from the BWG.Recycle to the TC.Credits starting from the lowest priority TC in the BWG up, until BWG.Recycle is zero.

A separate set of configuration parameters exists for each of the three TC arbiters as listed in Table 7-56.

**Table 7-56 Configuration Parameters for the Tx and Rx TC Arbiters**

| Parameter | Tx Packet Arbiter | Tx Descriptor Arbiter | Rx Packet Arbiter |
|---|---|---|---|
| | RTTPCS | RTTDCS | RTRPCS |
| RR / WSP mode | TPPAC | TDPAC | RAC |
| Recycle mode | TPRM | TDRM | RRM |

### 7.7.2.4.4 Tx TC WSP Arbiter Algorithm

The Transmit Packet Plane Arbitration Control (TPPAC) bit in the RTTPCS registers determines the scheduler type (RR or WSP).

Strict Priority — The strict-priority aspect of the TC WSP arbiter resides in the fact that once a TC has been granted for a data read request or for transmission, the highest priority TCs are checked (again) in a strict-priority order, starting from TC7, even if the granted TC still has credits for another data read request or transmission.

**Note:** The descriptor plane arbiter can't issue a data read request unless there is an unused request for data. Therefore the arbiter stalls in its current state each time there are no data read requests available. The next arbitration decision is only done once there is at least one free data read request.

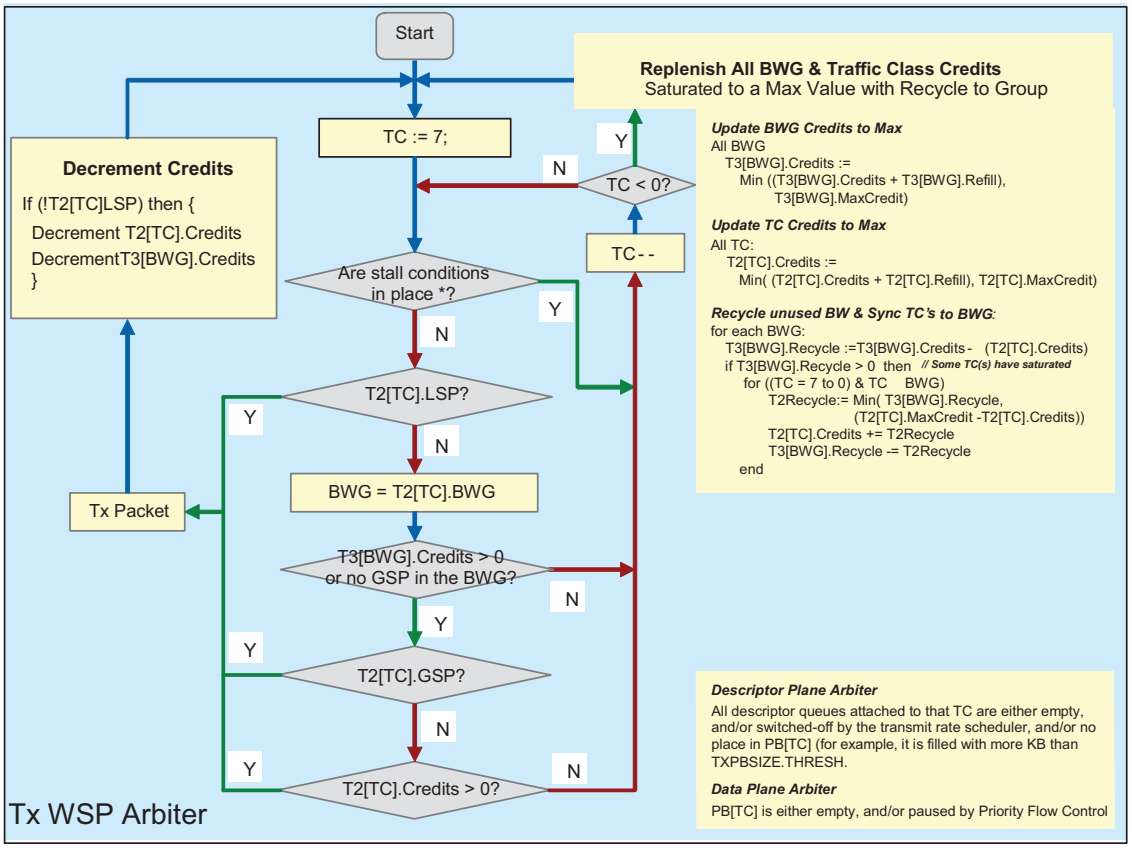



**Figure 7-34 Tx TC WSP Arbiters Operation**

## 7.7.3 Receive-Side Capabilities

### 7.7.3.1 User Priority to Traffic Class Mapping

To enable different TC support for incoming packets and proper behavior per TC, the 82599's receive packet buffer is segmented into several packet buffers. The 82599 supports the following configurations of packet buffer segmentation:

- DCB disabled — Single buffer of 512 KB
- DCB enabled with 4 TCs — 4 buffers, 128 KB each
- DCB enabled with 8 TCs — 8 buffers, 64 KB each
- DCB enabled with 8 TCs — 8 buffers, buffers 3:0 are 80 KB and buffers 7:4 are 48 KB

Incoming packets are transferred from the packet buffers into data buffers in system memory. Data buffers are arranged around descriptor rings described in Section 7.1.9. Each descriptor queue is assigned dynamically to a given TC (and therefore to a packet buffer) as described in Section 7.1.2.

Configuration registers:

- The size of each buffer is defined by the RXPBSIZE[0-7] registers. Note that it is possible to configure the buffers at 1 KB granularity
- A received packet is assigned by the 82599 to a TC, and is thus routed to the corresponding Rx packet buffer according to its *User Priority* field in the 802.1Q tag and according to a UP to TC mapping table loaded into the RTRUP2TC register.

***Caution:*** Different UP to TC mappings can be loaded in each direction Tx and Rx, as per RTTUP2TC and RTRUP2TC registers, respectively. But in such a case, when a packet is looped back by the internal VM to VM switch, it is routed to the Rx packet buffer that corresponds to the TC that was used in Tx.

### 7.7.3.2 Rx PB Weighted Strict Priority Arbiter

The 82599's Rx arbiter determines the order in which packets are written from the different packet buffers into system memory. Note that each packet buffer by itself is drained in the order packets arrived, as long as it deals with packets destined to the same Rx queue.

The arbitration algorithm between the receive packet buffers is WSP, similar to the TC scheme on the transmit side. Motivation for this scheme is as follows:

1. The major consideration is to prevent any delay in delivery of high-priority traffic.
2. Allocation of credits controls the bandwidth allocated to the different packet buffers.
3. A secondary mean of bandwidth allocation is the priority flow control. By altering the flow control high watermark, the 82599 can effectively (if coarsely) regulate bandwidth allocation to types of traffic.

Table 7-57 (T4) defines the TCs and their bandwidth allocation. The following elements are defined in this table:

**Table 7-57 Bandwidth Allocation to Traffic Classes and Bandwidth Groups**

| T4: Packet Buffer Bandwidth Allocation Within a BWG | | | | | | |
|---|---|---|---|---|---|---|
| **PB** | **BWG** | **PB Refill** | **PB Max Credits** | **LSP** | **GSP** | **PB Min Credits (according to GSP 0/1)** |
| 0 | 1 | 64 bytes to 32 KB | 64 bytes to 256 KB | 0/1 | 0/1 | -MSS / -2,048 KB |
| 1 | 2 | 64 bytes to 32 KB | 64 bytes to 256 KB | 0/1 | 0/1 | -MSS / -2,048 KB |
| 2 | 0 | 64 bytes to 32 KB | 64 bytes to 256 KB | 0/1 | 0/1 | -MSS / -2,048 KB |
| 3 | 1 | 64 bytes to 32 KB | 64 bytes to 256 KB | 0/1 | 0/1 | -MSS / -2,048 KB |
| 4 | 2 | 64 bytes to 32 KB | 64 bytes to 256 KB | 0/1 | 0/1 | -MSS / -2,048 KB |
| 5 | 3 | 64 bytes to 32 KB | 64 bytes to 256 KB | 0/1 | 0/1 | -MSS / -2,048 KB |
| 6 | 1 | 64 bytes to 32 KB | 64 bytes to 256 KB | 0/1 | 0/1 | -MSS / -2,048 KB |
| 7 | 0 | 64 bytes to 32 KB | 64 bytes to 256 KB | 0/1 | 0/1 | -MSS / -2,048 KB |

† Due to a pipelined implementation, the PB credits range is enlarged by one MSS in both directions, beyond its positive and negative limits.

†† All values are implemented with 64-byte granularity (a value of one corresponds to 64 bytes of credit).

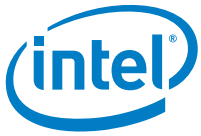


Table 7-58 (T5) defines the hierarchy of BWG similarly to T4 defining the hierarchy within the BWG's. T4 implementation should include eight rows.

**Table 7-58 Packet Buffer Allocations to BWGs (with Example)**

| T5: Line Bandwidth Allocation to Traffic BWGs | | | | |
|---|---|---|---|---|
| **BWG** | **BWG Credit Refill** | **BWG MaxCredit** | **BWG Credit** | **Description** |
| 0 | = Σ[TC] Credit Refill | = Σ[TC]. MaxCredit | -/+ 2,048KB | IPC |
| 1 | = Σ[TC] Credit Refill | = Σ[TC]. MaxCredit | -/+ 2,048KB | SAN |
| 2 | = Σ[TC] Credit Refill | = Σ[TC]. MaxCredit | -/+ 2,048KB | LAN |
| 3 | = Σ[TC] Credit Refill | = Σ[TC]. MaxCredit | -/+ 2,048KB | MGMT. |
| - | | | | |
| - | | | | |
| - | | | | |
| 7 | | | | |

The attributes of the Rx packet arbiter are described in Section 7.7.2.4.2.

### 7.7.3.2.1 Rx TC Arbitration Configurations

Table 7-59 lists the register fields that control the Rx arbiter.

**Table 7-59 Registers Allocation for DCB Rx Arbiters**

| Attribute | Rx Packet Arbiter |
|---|---|
| PB Control registers | RTRPT4C |
| PB Status registers | RTRPT4S |
| BWG | BWG |
| PB credit refill | CRQ |
| PB MaxCredit | MCL |
| LSP | LSP |
| GSP | GSP |
| PB credits | CCC |

Configuration parameters for the Rx packet arbiter are defined and listed in Section 7.7.2.4.1.

### 7.7.3.2.2 Rx TC WSP Arbiter Algorithm

The Rx packet arbiter operates in RR or WSP mode, configured through the *RAC* bit in the RTRPCS register. The WSP arbiter operation is described as follows.

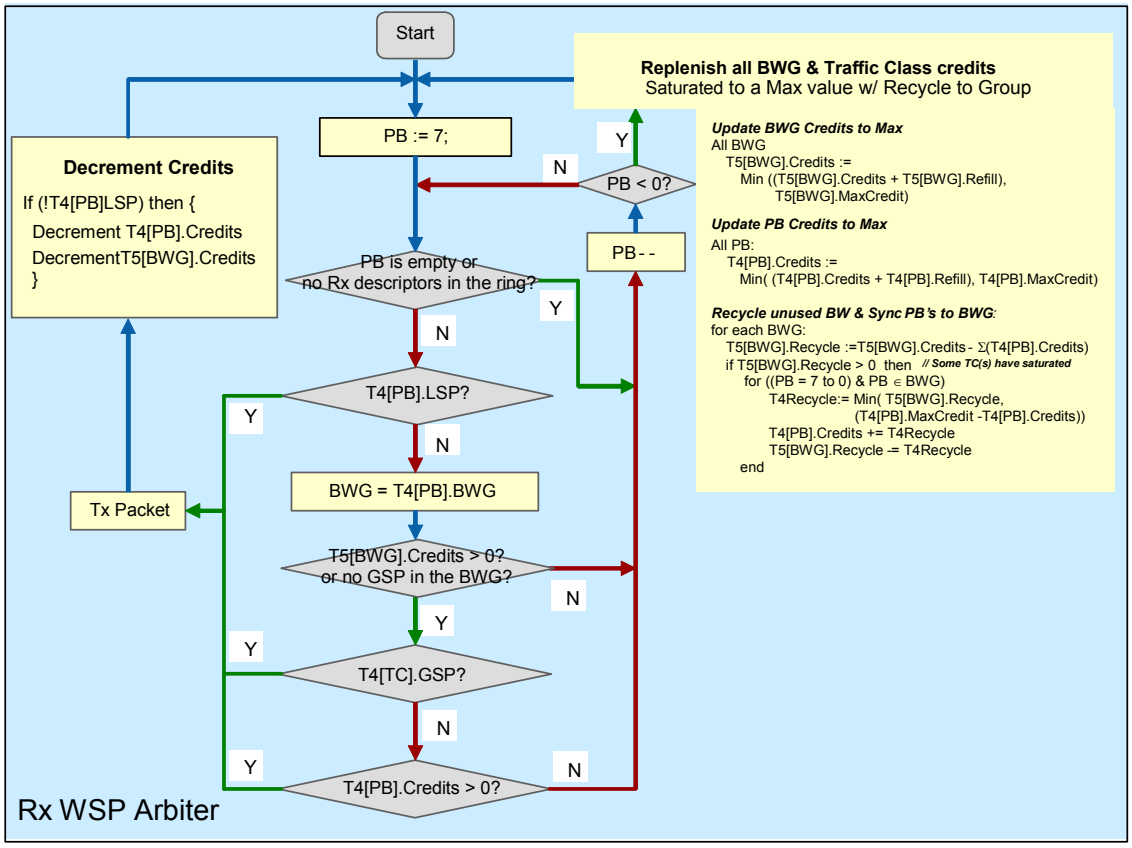



**Figure 7-35 Rx Packet WSP Arbiter Operation**

# 7.8 LinkSec

LinkSec (or MACsec, 802.1AE) is a MAC-level encryption/authentication scheme defined in IEEE 802.1AE that uses symmetric cryptography. The 802.1AE defines AES-GCM 128-bit key as a mandatory cipher suite that can be processed by the 82599. The LinkSec implementation, enabled as detailed in Section 4.6.12, supports the following:

- GCM AES 128-bit offload engine in the Tx and Rx data path that support 10 Gb/s wire speed.
- Both host and manageability controller traffic can be processed by the GCM AES engines.
- Support a single, secure Connectivity Association (CA):
  - Single Secure Connection (SC) on transmit data path.
  - Single SC on receive data path.
  - Each SC supports two Security Associations (SAs) for seamless re-keying.
- At any given time, either the manageability controller or the host can act as key agreement entity (KaY – in 802.1AE spec terminology). For example, control and access the offloading engine (SecY in 802.1AE specification terminology).
  - Arbitration semaphores indicate whether the manageability controller or the host acts as the KaY.
  - Tamper resistance — When the manageability controller acts as KaY it can disable accesses from the host to SecY's address space. When the host acts as the KaY no protection is provided.
- Provide statistic counters as listed in the Section 8.3.5.6.
- Support replay protection with replay window equal to zero. Packets that fail replay validation are posted with a replay error in the Rx descriptor. The packets are posted to the host regardless of strict versus check mode described later on in this section.
- Receive memory structure:
  - New LinkSec offload receive status indication in the receive descriptors. LinkSec offload must not be used with the legacy receive format but rather use the extended receive descriptor format.
  - LinkSec header/tag can be posted to the KaY for debug.
- Support VLAN header location according to IEEE 802.1AE (first header inner to the LinkSec tag).
- When LinkSec offload is enabled, Ethernet CRC must be enabled as well by setting both TXCRCEN and RXCRCSTRP bits in the HLREG0 register.

## 7.8.1 Packet Format

LinkSec defines frame encapsulation format as follows.

**Table 7-60 Legacy Frame Format**

| | | | | |
|---|---|---|---|---|
| MAC DA, SA | VLAN (optional) | Legacy Type / Len | LLC data (may include IP/TCP and higher level payload) | CRC |
| | ← — - — - — - — - — - — - — - — - — - — - User Data — - — - — - — - — - — - — - — - — - — - → | | | |

**Table 7-61 LinkSec Encapsulation**

| | | | | |
|---|---|---|---|---|
| MAC DA, SA | LinkSec header (SecTag) | User data (optional encrypted) | LinkSec ICV (tag) | CRC |

## 7.8.2 LinkSec Header (SecTag) Format

**Table 7-62 Sectag Format**

| | | | | |
|---|---|---|---|---|
| LinkSec Ethertype | TCI and AN | SL | PN | SCI (optional) |
| 2 bytes | 1 byte | 1 byte | 4 bytes | 8 bytes |

### 7.8.2.1 LinkSec Ethertype

The MACsec Ethertype comprises octet 1 and octet 2 of the SecTAG. It is included to allow:

a. Coexistence of MACsec capable systems in the same environment as other systems.

b. Incremental deployment of MACsec capable systems.

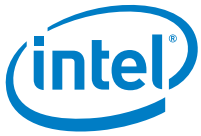
c. Peer SecY's to communicate using the same media as other communicating entities.

d. Concurrent operation of key agreement protocols that are independent of the MACsec protocol and the current cipher suite.

e. Operation of other protocols and entities that make use of the service provided by the SecY's uncontrolled port to communicate independently of the Key agreement state.

**Table 7-63 LinkSec Ethertype**

| Tag Type | Name | Value |
|---|---|---|
| 802.1AE security TAG | LinkSec Ethertype | 88-E5 |



### 7.8.2.2 TCI and AN

**Table 7-64 TCI and AN Description**

| Bit(s) | Description |
|---|---|
| 7 | Version Number (V).<br>supports only version 0. Packets with other version value are discarded by the 82599. |
| 6 | End Station (ES).<br>When set, indicates that the sender is an end station. As a result, SCI is redundant and causes the SC bit to be cleared. Currently, should be always 0b. |
| 5 | Secure Channel (SC).<br>Equals 1b when the SCI field is active. If the ES bit is set the SC must be cleared. Since only ES equals zero is supported, the SCI field must be active by setting the LSECTXCTRL.AISCI. |
| 4 | Single Copy Broadcast (SCB).<br>Cleared to 0b unless SC supports EPON. Should always be 0b. |
| 3 | Encryption (E).<br>Set to 1b when user data is encrypted. (see the note that follows). |
| 2 | Changed Text (C).<br>Set to 1b if the data portion is modified by the integrity algorithm. For example, if non-default integrity algorithm is used or if packet is encrypted. (see the note that follows). |
| 1:0 | Association Number (AN).<br>2-bit value defined by control channel to uniquely identify SA (Keys, etc.). |

**Note:** The combination of the *E* bit equals 1b and the *C* bit equals 0b is reserved for KaY packets. The LinkSec logic ignores these packets on the receive path and transfers them to KaY as is (no LinkSec processing and no LinkSec header strip). the 82599's implementation never issues a packet in which the *E* bit is cleared and the *C* bit is set, although can tolerate such packets on receive.

#### 7.8.2.2.1 Short Length

**Table 7-65 SL Field Description**

| Bit(s) | Description |
|---|---|
| 7:6 | Reserved, set to 0b. |
| 5:0 | Short Length (SL).<br>Number of octets in the secure data field from the end of SecTag to the beginning of ICV if it is less then 48 octets, else SL value is 0b. |

#### 7.8.2.2.2 Packet Number (PN)

The LinkSec engine increments it for each packet on the transmit side. The PN is used to generate the initial value (IV) for the crypto engines. When KaY is establishing a new SA, it should set the initial value of PN to 1b. See more details on PN exhausting in Section 7.8.5.1.

### 7.8.2.3 Secure Channel Identifier (SCI)

The SCI is composed of the Ethernet MAC address and port number as listed in the following table. If the *SC* bit in TCI is not set, the SCI is not encoded in the SecTag.

**Table 7-66 SCI Field Description**

| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 |
|---|---|---|---|---|---|---|---|
| Source Ethernet MAC Address | | | | | | Port Number | |

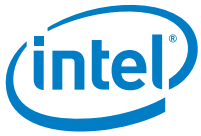
### 7.8.2.4 Initial Value (IV) Calculation

The IV is the initial value used by the Tx and Rx authentication engines. The IV is generated from the PN and SCI as described in the 802.1AE specification.

## 7.8.3 LinkSec Management – KaY (Key Agreement Entity)

Kay management is done by the host or the manageability controller. The ownership of LinkSec management is as follows:

1. Initialization at power up or after wake on LAN.
   - In most cases the manageability controller wakes before the host, so:
     - If the manageability controller can be a KaY, it establishes an SC (authentication and key exchange).
     - If the manageability controller cannot be a KaY the only way for it to communicate is through a dedicated Ethernet MAC address or VLAN. This means that the switch must support settings that enable specific Ethernet MAC Address or VLAN to bypass LinkSec.
   - When the host is awake:
     - If the manageability controller acted as KaY, the host should authenticate itself and transfer its ability to authenticate to the manageability controller in order for the manageability controller to transfer ownership over the LinkSec hardware. At this stage, the system operates in proxy mode where the host manages the secured channel while the manageability controller piggybacks on it.



— If the manageability controller wasn't KaY, the host takes ownership over the LinkSec hardware and establishes an SC (authentication and key exchange). The manageability controller mode of operation does not change and it continues to communicate through a dedicated Ethernet MAC address or VLAN.

2. Host in Sx state — manageability controller active:

— If the manageability controller is not Kay capable, then the SC should be reset by link reset or by sending a log-off packet (1af) and then the manageability controller can return to VLAN solution (or remain in such).

— If the manageability controller is KaY capable, the host should notify the manageability controller that it retires KaY ownership and the manageability controller should retake it.

## 7.8.4 Receive Flow

The 82599 might concurrently receive packets that contain LinkSec encapsulation as well as packets that do not include LinkSec encapsulation. This section describes the incoming packet classification.

- Examine the user data for a SecTAG:
  - — If no SecTag, post the packet with a cleared LinkSec bit in the *Packet Type* field of the receive descriptor.
- Validate frames with a SecTAG:
  - — The MPDU comprises at least 18 octets
  - — Octets 1 and 2 compose the MACsec Ethertype (88E5)
  - — The *V* bit in the TCI is cleared
  - — If the *ES* or the *SCB* bit in the TCI is set, then the *SC* bit is cleared
  - — Bits 7 and 8 of octet 4 of the SecTAG are cleared SL <= 0x3F
  - — If the *C* and *SC* bits in the TCI are cleared, the MPDU comprises 24 octets plus the number of octets indicated by the *SL* field if that is non-zero and at least 72 octets otherwise
  - — If the *C* bit is cleared and the *SC* bit set, then the MPDU comprises 32 octets plus the number of octets indicated by the *SL* field if that is non-zero and at least 80 octets otherwise
  - — If the *C* bit is set and the *SC* bit cleared, then the MPDU comprises 8 octets plus the minimum length of the ICV as determined by the cipher suite in use at the receiving SecY, plus the number of octets indicated by the *SL* field if that is non-zero and at least 48 additional octets otherwise
  - — If the *C* and *SC* bits are both set, the frame comprises at least 16 octets plus the minimum length of the ICV as determined by the cipher suite in use at the receiving SecY, plus the number of octets indicated by the *SL* field if that is non-zero and at least 48 additional octets otherwise
- Extract and decode the SecTAG as specified in Section 7.8.2.
- Extract the user data and ICV as specified section Section 7.8.1.

- Assign the frame to an SA:
  - If a valid SCI, use it to identify the SC
  - Select SA according to AN value
  - If no valid SC or no valid SA found, drop the packet
  - If SCI is omitted, use default SC
  - Select SA according to AN value
  - If no valid SC (or more then SC active) or no valid SA found drop packet
- Perform a preliminary replay check against the last validated PN
- Provide the validation function with:
  - The SA Key (SAK)
  - The SCI for the SC used by the SecY to transmit
  - The PN
  - The SecTAG
  - The sequence of octets that compose the secure data
  - The ICV
- Receive the following parameters from the cipher suite validation operation
  - A valid indication, if the integrity check was valid and the user data could be recovered
  - The sequence of octets that compose the user data
- Update the replay check
- Issue an indication to the controlled port with the DA, SA, and priority of the frame as received from the receive de-multiplexer, and the user data provided by the validation operation

**Note:** All the references to clauses are to the IEEE P802.1AE/D5.1 document from January 19, 2006.

### 7.8.4.1 Receive Modes

There are four modes of operation defined for LinkSec Rx as defined by the LSECRXCTRL.LSRXEN field:

1. Bypass (LSRXEN = 00b) — in this mode, LinkSec is not offloaded. There is no authentication or decrypting of the incoming traffic. The LinkSec header and trailer are not removed and these packets are forwarded to the host or the manageability controller according to the regular L2 MAC filtering. The packet is considered as untagged (no VLAN filtering). No further offloads are done on LinkSec packets.
2. Check (LSRXEN = 01b) — in this mode, incoming packets with matching key are decrypted and authenticated according to the LinkSec tag. In this mode both good and erroneous packets are forwarded to host (with the relevant error indication). The only cases where packets are dropped are: erroneous encrypted packets (with the 'C' bit in the SecTag header is set) or erroneous packets with replay error if replay protection is enabled in the LSECRXCTRL registers. The Check mode is expected to be

used mainly for debug purposes. In this mode, it may be useful to set also the "Post LinkSec header" bit in the LSECRXCTRL register which controls both SecTag and ICV to be posted to host memory. Note that the header is not removed from KaY packets.

3. Strict (LSRXEN = 10b) — in this mode, incoming packets with matching key are decrypted and authenticated according to the LinkSec tag. The LinkSec header and trailer might be removed from these packets and the packets are forwarded to the host only if the decrypting or authentication was successful. Additional offloads are possible on LinkSec packets. The header is not removed from KaY packets.
4. Drop (LSRXEN = 11b) — in this mode, LinkSec is not offloaded and LinkSec packets are dropped. There is no authentication or decrypting of the incoming traffic.

### 7.8.4.2 Receive SA Exhausting – Re-Keying

The seamless re-keying mechanism is explained in the following example.

KaY establishes SC0 SC and sets SA0 as the active SA by writing the key in register LinkSec RX Key, writing the AN in LSECRXSA[0], and setting the *SA Valid* bit in the same register. This clears the *Frame Received* bit. On the first packet that arrived to SA0, the frame received automatically sets the *Frame Received* bit. Only at this time the KaY can and should initiate SA1 in the same manner as for SA0. When a frame of SA1 arrives, SA0 retires and can be used for the next SA.

**Note:** The same mechanism should be used for all RX SCs.

### 7.8.4.3 Receive SA Context and Identification

Upon arrival of a secured frame the context of the SecTag is verified. This context of the SecTag is described in Section 7.8.2. In order to process the secured frame it should be associated with one of the SA keys. The identification is done by comparing the SCI data with LinkSec RX SC registers and the appropriate SC is selected. To ensure that the SC bit in the TCI of the frame is not set and more than one SC is valid belongs to the frame considered as erroneous and transferred to error handling if only one SC is valid, this SC is selected in this case SC. The incoming frame AN field is compared to the AN field of the Link RX SA register of the selected SC in order to select an SA. The selected SA PN (register LinkSec RX SA PN) field is compared to the incoming PN which should be equal or greater than the LinkSec RX SA PN value, otherwise this frame is dropped. On a match, the selected SA key is used for the secured frame processing.

### 7.8.4.4 Receive Statistic Counters

A detailed list and description of the LinkSec RX statistics counters can found in Section 8.3.5.6.

## 7.8.5 Transmit Data Path

The 82599 might concurrently transmit packets that contain LinkSec encapsulation as well as packets that do not include LinkSec encapsulation. This section describes the transmit packet classification, transmit descriptors and statistic counters.

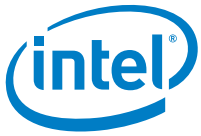
**Note:** Since flow control (PAUSE) packets are part of the MAC service they should not go through the LinkSec logic.

1. Assign the frame to an SA by adding the AN according to SA select bit in the LSECTXSA register.
2. Assign the next PN variable for that SA to be used as the value of the PN in the SecTAG based on the value in the appropriate (according to SA) LSECTXPN register.
3. Encode the octets of the SecTAG according to the setting in LSECTXCTRL register.
4. Provide the protection function of the current cipher suite with:
   a. The SA Key (SAK).
   b. The SCI for the SC used by the SecY to transmit.
   c. The PN.
   d. The SecTAG.
   e. The sequence of octets that compose the user data.
5. Receive the following parameters from the cipher suite protection operation:
   a. The sequence of octets that compose the secure data.
   b. The ICV.
6. Issue a request to the transmit multiplexer with the destination and source Ethernet MAC addresses, and priority of the frame as received from the controlled port, and an MPDU comprising the octets of the SecTAG, secure data, and the ICV concatenated in that order.

### 7.8.5.1 Transmit SA Exhausting – Re-Keying

the 82599 supports a single SC on the transmit data path with a seamless re-keying mechanism. The SC might act with one of two optional SAs. The SA is selected statically by the Active SA field in the LSECTXSA register. Once the KaY entity (could be either software or hardware as defined by the LinkSec Ownership field in the LSWFW register) changes the setting of the SA Select field in the LSEXTXSA register the Active SA field is getting the same value on a packet boundary. The next packet that is processed by the transmit LinkSec engine uses the updated SA.

The KaY should switch between the two SAs before the PN is exhausted. In order to protect against such event, hardware generates a LinkSec packet number interrupt to KaY when the PN reaches the exhaustion threshold as defined in the LSECTXCTRL register. The exhaustion threshold should be set to a level that enables the KaY to switch between SA's faster then the PN might be exhausted. If the KaY is slower than it should be, then the PN might be increment above planned. Hardware guarantees that the PN never repeats itself, even if the KaY is slow. Once the PN reaches a value of 0xFF…F0, hardware clears the Enable Tx LinkSec field in the LSECTXCTRL register to 00b. Clearing



the Enable Tx LinkSec field, hardware disables LinkSec offload before the PN could wrap around and then might repeat itself.

**Note:** Potential race conditions are possible as follows. the 82599 might fetch a transmit packet (indicated as TxPacketN) from the host memory (host or manageability controller packet). KaY can change the setting of the Tx SA Index. The TxPacketN can use the new TX SA Index if the TX SA index was updated before the TxPacketN propagated to the transmit LinkSec engine. This race is not critical since the receiving node should be able to process the previous SA as well as the new SA in the re-keying transition period.

#### 7.8.5.2 Transmit SA Context

Upon transmission of a secured frame, the SA associated data is inserted into the *SecTag* field of the frame. The SecTag data is composed from the LinkSec Tx registers. The SCI value is taken from LinkSec TX SCI Low and High registers unless instructed to omit SCI. The AN value is taken from the active LinkSec TX SA and the PN from the appropriate LinkSec TX SA PN.

#### 7.8.5.3 Transmit Statistic Counters

A detailed list and description of the LinkSec TX statistics counters can found in Section 8.2.3.13.

### 7.8.6 LinkSec and Manageability

See Section 10.4.

### 7.8.7 Key and Tamper Protection

LinkSec provides the network administrator protection to the network infrastructure from hostile or unauthorized devices. Since the local host operating system can itself be compromised, hardware protects vital LinkSec context from software access. There are two levels of protection:

- Disable host read access to the LinkSec Keys (keys are write-only)
- Disable host access to LinkSec logic while the firmware manages the LinkSec SC.

### 7.8.7.1 Key Protection

The LinkSec keys are protected against read accesses at all times. Both software and firmware are not able to read back the keys that hardware uses for transmit and receive activity. Instead, hardware enables the software and firmware reading a signature enabling to verify proper programming of the device. The signature is a byte XOR operation of the Tx and Rx keys readable in the LSECTXSUM and LSECRXSUM fields in the LSECCAP register.

### 7.8.7.2 Tamper Protection

In a scenario where the host failed authentication and as a result cannot act as the KaY, the manageability controller disables the host access to network and manages the LinkSec channel while the host operating system is already up and running. In such cases, hardware provides the required hooks to protect LinkSec connectivity against hostile software. The manageability controller firmware can disable write accesses generated by the host CPU (on the PCI interface) by setting the *Lock LinkSec Logic* (bit 12) bit in the LSWFW register. Setting this bit can generate an interrupt to the host in case it is enabled by the host in the IMS register.

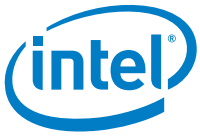
## 7.8.8 LinkSec Statistics

### 7.8.8.1 Rx Statistics

After receiving a packet, one and only one of the statistics in Table 7-67 applies. The precedence order of the statistics is also defined in Table 7-67.

**Table 7-67 Rx Statistics**

| Register Name | 802.1ae Name | Priority | Notes |
|---|---|---|---|
| LSECRXBAD | InPktsBadTag | 2 | Packet is dropped in strict mode or in check mode when the *C* bit is one. |
| LSECRXUNSCI | InPktsUnknownSCI | 3 | Used only in check mode. Packet is forwarded to the host if the *C* bit is zero. |
| LSECRXNOSCI | InPktsNoSCI | 3 | Packet is dropped in strict mode or in check mode when the *C* bit is one. |
| LSECRXUNSA | InPktsUnusedSA | 4 | Packet is dropped in strict mode or in check mode when the *C* bit is one.<br>*Note:* This statistic reflects the sum of InPktsUnusedSA for all SAs. |
| LSECRXNUSA | InPktsNotUsingSA | 4 | Used only in check mode. Packet is forwarded to the host if the *C* bit is zero.<br>*Note:* This statistic reflects the sum of InPktsUnusedSA for all SAs. |



## Table 7-67 Rx Statistics (Continued)

| Register Name | 802.1ae Name | Priority | Notes |
|---|---|---|---|
| LSECRXLATE | InPktsLate | 5 | |
| n/a | InPktsOverrun | n/a | The 82599 supports wire-speed decryption and thus this statistic is not needed. |
| LSECRXNV[SA#] | InPktsNotValid | 6 | Packet is dropped in strict mode or in check mode when the *C* bit is one. |
| LSECRXINV[SA#] | InPktsInvalid | 6 | Used only in check mode. Packet is forwarded to the host if the *C* bit is zero. |
| LSECRXDELAY | InPktsDelayed | 7 | |
| GPRC | InPktsUnchecked | n/a | This statistic is relevant only in bypass mode. In this case, this statistic is reflected in the regular GPRC statistic. |
| LSECRXOK[SA#] | InPktsOK | 8 | |

# 7.9 Time SYNC (IEEE1588 and 802.1AS)

## 7.9.1 Overview

Measurement and control applications are increasingly using distributed system technologies such as network communication, local computing, and distributed objects. Many of these applications are enhanced by having an accurate system-wide sense of time achieved by having local clocks in each sensor, actuator, or other system device. Without a standardized protocol for synchronizing these clocks, it is unlikely that the benefits are realized in the multi-vendor system component market. Existing protocols for clock synchronization are not optimum for these applications. For example, Network Time Protocol (NTP) targets large distributed computing systems with Millisecond (ms) synchronization requirements. The 1588 standard specifically addresses the needs of measurement and control systems:

- Spatially localized
- Microsecond (µs) to sub-µs accuracy
- Administration free
- Accessible for both high-end devices and low-cost, low-end devices

**Note:** The time sync mechanism activation is possible in full-duplex mode only. There are no limitations on the wire speed although the wire speed might affect the accuracy.

## 7.9.2 Flow and Hardware/Software Responsibilities

The operation of a Precision Time Protocol (PTP) enabled network is divided into two stages: initialization and time synchronization.

At the initialization stage, every master-enabled node starts by sending sync packets that include the clock parameters of its clock. Upon receipt of a sync packet, a node compares the received clock parameters to its own and if the received parameters are better, then this node moves to a slave state and stops sending sync packets. While in slave state, the node continuously compares the incoming packet to its currently chosen master and if the new clock parameters are better, than the master selection is transferred to this master clock. Eventually the best master clock is chosen. Every node has a defined time-out interval that if no sync packet was received from its chosen master clock it moves back to a master state and starts sending sync packets until a new best master clock (PTP) is chosen.

The time synchronization stage is different to master and slave nodes. If a node is in a master state it should periodically send a sync packet that is time stamped by hardware on the TX path (as close as possible to the PHY). After the sync packet, a Follow_Up packet is sent that includes the value of the time stamp kept from the sync packet. In addition, the master should time stamp Delay_Req packets on its Rx path and return to the slave that sent the time stamp value using a Delay_Response packet. A node in a slave state should time stamp every incoming sync packet and if it came from its

selected master, software uses this value for time offset calculation. In addition, it should periodically send Delay_Req packets in order to calculate the path delay from its master. Every sent Delay_Req packet sent by the slave is time stamped and kept. With the value received from the master with Delay_Response packet, the slave can now calculate the path delay from the master to the slave. The synchronization protocol flow and the offset calculation are shown in Figure 7-36.

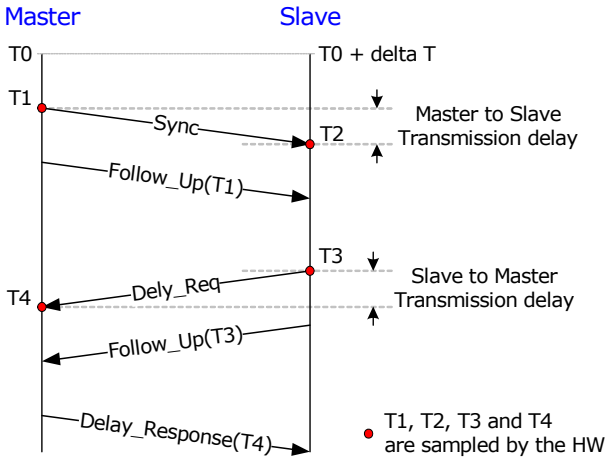



**Figure 7-36 Sync Flow and Offset Calculation**

Hardware responsibilities are:

1. Identify the packets that require time stamping.
2. Time stamp the packets on both Rx and Tx paths.
3. Store the time stamp value for software.
4. Keep the system time in hardware and give a time adjustment service to software.
5. Maintain auxiliary features related to the system time.

Software responsibilities are:

1. Manageability controller protocol execution, which means defining the node state (master or slave) and selection of the master clock if in slave state.
2. Generate PTP packets, consume PTP packets.
3. Calculate the time offset and adjust the system time using a hardware mechanism for that.
4. Enable configuration and usage of the auxiliary features.

| Action | Responsibility | Node Role |
|---|---|---|
| Generate a sync packet with time stamp notification in the descriptor. | Software | Master |
| Time stamp the packet and store the value in registers (T1). | Hardware | Master |
| Time stamp incoming sync packet, store the value in register and store the sourceID and sequenceID in registers (T2). | Hardware | Slave |
| Read the time stamp from register put in a Follow_Up packet and send. | Software | Master |
| Once received, the Follow_Up store T2 from registers and T1 from Follow_up packet. | Software | Slave |
| Generate a Delay_Req packet with time stamp notification in the descriptor. | Software | Slave |
| Time stamp the packet and store the value in registers (T3). | Hardware | Slave |
| Time stamp incoming Delay_Req packet, store the value in register and store the sourceID and sequenceID in registers (T4). | Hardware | Master |
| Read the time stamp from register and send back to slave using a Delay_Response packet. | Software | Master |
| Once received, the Delay_Response packet calculate offset using T1, T2, T3 and T4 values. | Software | Slave |

#### 7.9.2.1 TimeSync Indications in Rx and Tx Packet Descriptors

Some indications need to be transferred between software and hardware regarding PTP packets. On the Tx path, software should set the 1588 bit in the Tx packet descriptor (bit 9). On the Rx path, hardware has two indications to transfer to software, one is to indicate that this packet is a PTP packet (whether time stamp is taken or not). This is also for other types of PTP packets needed for management of the protocol and this bit is set only for the L2 type of packets (the PTP packet is identified according to its Ethertype). PTP packets have the *L2Type* bit in the packet type field set (bit 9) and the Ethertype matches the filter number set by software to filter PTP packets. The UDP type of PTP packets don't need such indication since the port number (319 for event and 320 all the rest PTP packets) directs the packets toward the time sync application. The second indication is TS (bit 14) to indicate to software that time stamp was taken for this packet. Software needs to access the time stamp registers to get the time stamp values.

### 7.9.3 Hardware Time Sync Elements

All time sync hardware elements are reset to their initial values (as defined in Section 8.0) upon MAC reset. The clock driving the time sync elements is the DMA clock which frequency depends on the link speed. Upon change in link speed some of the time sync parameters should be changed accordingly. For details please see Table 7-68.

### 7.9.3.1 System Time Structure and Mode of Operation

The time sync logic contains an up counter to maintain the system time value. This is a 64-bit counter that is built from the SYSTIML and SYSTIMH registers. When operating as a master, the SYSTIMH and SYSTIML registers should be set once by software according to the general system. When operating as a slave, software should update the system time on every sync event as described in Section 7.9.3.3. Setting the system time is done by a direct write to the SYSTIMH register and a fine tune setting of the SYSTIML register using the adjustment mechanism described in Section 7.9.3.3.

Read access to the SYSTIMH and SYSTIML registers should execute in the following manner:

1. Software reads register SYSTIML, at this stage hardware should latch the value of SYSTIMH.
2. Software reads register SYSTIMH, the latched (from last read from SYSTIML) value should be returned by hardware.

Upon an increment event, the system time value should increment its value by the value stored in TIMINCA.*incvalue*. An increment event happens every TIMINCA.*incperiod* cycle if its one then an increment event should occur on every clock cycle. The *incvalue* defines the granularity in which the time is represented by the SYSTMH/L registers. For example, if the cycle time is 16 ns and the *incperiod* is one then and the *incvalue* is 16 then the time is represented in nanoseconds if the *incvalue* is 160 then the time is represented in 0.1 ns units and so on. The *incperiod* helps to avoid inaccuracy in cases where T value cannot be represented as a simple integer and should be multiplied to get to an integer representation. The *incperiod* value should be as small as possible to achieve best accuracy possible.

**Table 7-68 Recommended Values for incvalue and incperiod and the outcome SYSTIME**

| Link Speed | Clock Frequency | Recommended Incvalue | Recommended Incperiod | SYSTIML / SYSTIMH Time Units | SYSTIML / SYSTIMH Granularity |
|---|---|---|---|---|---|
| 10 Gb/s | 156.25 MHz | 16000000 (0xF42400) | 2 | $0.8 \times 10^{-15}$ | 12.8 ns |
| 1 Gb/s | 15.625 MHz | 16000000 | 2 | $8 \times 10^{-15}$ | 128 ns |
| 100 Mb/s | 1.5625 MHz | 16000000 | 2 | $80 \times 10^{-15}$ | 1.28 µs |

**Note:** Best accuracy is achieved at lowest permitted Incperiod equals two and as high as possible Incvalue.

### 7.9.3.2 Time Stamping Mechanism

The time stamping logic is located as close as possible to the PHY. Figure 7-37 shows the exact point in time where the time value is captured by the hardware relative to the packet content. This is to reduce delay uncertainties originated from implementation differences. While the time stamp is sampled at a very late phase in the data path, the 82599 does not insert it to the transferred packet. Instead, the 82599 supports the two-step operation as follows for Tx and Rx.

### Tx time stamping

The time stamp logic is activated if enabled by the TSYNCTXCTL.EN bit and the time stamp bit in the packet descriptor is set. In this case, hardware captures the packet's transmission time in the TXSTMPL and TXSTMPH registers. Software is responsible to read the transmission time and append it in the Follow_Up packet as shown in Figure 7-36.

### Rx time stamping

On the Rx, this logic parses the traversing frame. If it is matching the message type defined in RXMTRL register, the following packet's parameters are latched: The reception time stamp is stored in the RXSTMPL and RXSTMPH registers. The SourceuuID and SequenceID are stored in the RXSATRL and RXSATRH registers. In addition, two status bits are reported in the Rx descriptor: PTP packet indication (this bit is set only for L2 packets since on the UDP packets the port number direct the packet to the application) and the *TS* bit to identify that a time stamp was taken for this packet (stored in the RXSTMPL and RXSTMPH registers).

**Note:** The time stamp values are locked in the RXSTMPL and RXSTMPH registers until software accesses them. As long as software does not read these registers, hardware does not capture the time stamp of further Rx packets. In order to avoid potential deadlocks, it is recommended that software read the Rx time stamp registers at some time after sync or Delay_Req packets are expected. It would overcome erroneous cases on which the hardware latches a packet reception time while the packet's content was not posed properly to the software.

Reception consecutive packets that are able to latch its reception time stamp are not supported by the 82599. The RXSATRL and RXSATRH registers may not contain sufficient information to identify uniquely a specific client. Therefore, Master software must not initiate consecutive sync requests before the previous response is received.

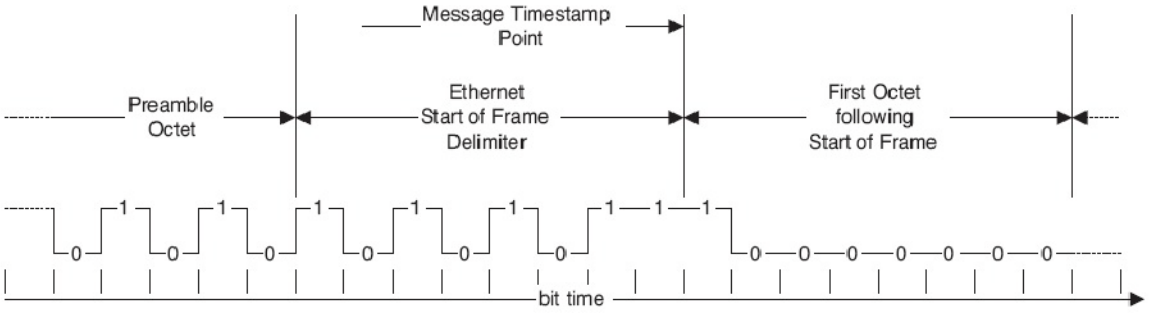
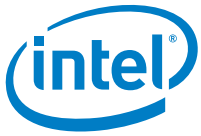



**Figure 7-37 Time Stamp Point**



#### 7.9.3.3 Time Adjustment Mode of Operation

A node in a time sync network can be in one of two states: master or slave. When a time sync entity is in a master state, it should synchronize other entities to its system clock. In this case, no time adjustments are needed. When the entity is in slave state, it should adjust its system clock by using the data arrived with the Follow_Up and Delay_Response packets and to the time stamp values of Sync and Delay_Req packets. When having all the values software on the slave entity can calculate its offset in the following manner.

After an offset calculation, the system time register should be updated. This is done by writing the calculated offset to TIMADJL and TIMADJH registers. The order should be as follows:

1. Write the lower portion of the offset to TIMADJL.
2. Write the high portion of the offset to TIMADJH to the lower 31 bits and the sign to the most significant bit.

After the write cycle to TIMADJH the value of TIMADJH and TIMADJL should be added to the system time.

### 7.9.4 Time Sync Related Auxiliary Elements

The time sync logic implements three types of auxiliary element using the precise system timer and SDPs. The time sync block implements two of each features while the possible options of connecting them to SDPs are:

| | | | | | | |
|---|---|---|---|---|---|---|
| SDP2 | Time Stamp 0 | Time Stamp 0 | Target Time 1 | Time Stamp 0 | Time Stamp 0 | Time Stamp 0 |
| SDP3 | Time Stamp 1 | Target Time 0 | Target Time 0 | Time Stamp 1 | Time Stamp 1 | Target Time 0 |
| SDP6 | CLK0 | CLK0 | CLK0 | Target Time 1 | Target Time 1 | Target Time 1 |
| SDP7 | CLK1 | CLK1 | CLK1 | CLK1 | Target Time 0 | CLK1 |

Selecting the SDP functionality is done by programming the TimeSync Auxiliary control register and the Extended SDP Control register.

#### 7.9.4.1 Target Time

The target time register is used to get a time triggered event to hardware using an SDP pin. Each target time register is structured the same as the system time register. If the value of the system time is equal to the value written to one of the target time registers, a change in level occurs on one the selected SDP outputs. The accuracy of the comparison is defined by the value of *Mask* field in the TSAUXC register. The target time register also can be used for adjustment of the configurable clock out. Each target time register has an enable bit located in the Auxiliary Control register. After receiving a target time event, the enable bit is cleared and needs to be set again by software to get another target time event.

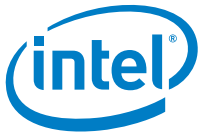
### 7.9.4.2 Time Stamp Events

After a change in level of an input from one of the SDP pins, a time stamp of the system time is captured into one of the two auxiliary time stamp registers.

## 7.9.5 PTP Packet Structure

The time sync implementation supports both the 1588 V1 and V2 PTP frame formats. The V1 structure can come only as UDP payload over IPv4 while the V2 can come over L2 with its Ethertype or as a UDP payload over IPv4 or IPv6. The 802.1AS uses only the layer 2 V2 format. Note that PTP frame structure over UDP is not supported in the 82599 for IP tunneling packets.

| Offset in Bytes | V1 Fields | V2 Fields | |
|---|---|---|---|
| Bits | 7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0 | |
| 0 | versionPTP | transportSpecific[1] | messageId |
| 1 | | Reserved | versionPTP |
| 2 | versionNetwork | messageLength | |
| 3 | | | |
| 4 | Subdomain | SubdomainNumber | |
| 5 | | Reserved | |
| 6 | | flags | |
| 7 | | | |
| 8 | | Correction Field | |
| 9 | | | |
| 10 | | | |
| 11 | | | |
| 12 | | | |
| 13 | | | |
| 14 | | | |
| 15 | | | |
| 16 | | Reserved | |
| 17 | | | |
| 18 | | | |
| 19 | | | |



| 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<br>(only part of the field is captured in the RXSATRL and RXSATRH registers) |
| 21 | Source communication technology | |
| 22 | Sourceuuid | |
| 23 | | |
| 24 | | |
| 25 | | |
| 26 | | |
| 27 | | |
| 28 | sourceportid | |
| 29 | | |
| 30 | sequenceId | sequenceId |
| 31 | | |
| 32 | control | control |
| 33 | reserved | logMessagePeriod |
| 34 | falgs | n/a |
| 35 | | |

1. Should all be zero.

**Note:** Only the fields with the bold italic format colored red are of interest to hardware.

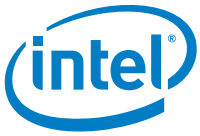
| | | | |
|---|---|---|---|
| Ethernet (L2) | VLAN (Optional) | PTP Ethertype | PTP message |
| | | | |
| Ethernet (L2) | IP (L3) | UDP | PTP message |

When a PTP packet is recognized (by Ethertype or UDP port address) on the Rx side the version should be checked if it is V1 then the control field at offset 32 should be compared to message field in register described in Section 8.2.3.26.6, otherwise the byte at offset 0 should be used for comparison to the rest of the needed field are at the same location and size for both V1 and V2.

| Enumeration | Value |
|---|---|
| PTP_SYNC_MESSAGE | 0 |
| PTP_DELAY_REQ_MESSAGE | 1 |
| PTP_FOLLOWUP_MESSAGE | 2 |
| PTP_DELAY_RESP_MESSAGE | 3 |
| PTP_MANAGEMENT_MESSAGE | 4 |
| reserved | 5–255 |

| MessageId | Message Type | Value (hex) |
|---|---|---|
| PTP_SYNC_MESSAGE | Event | 0 |
| PTP_DELAY_REQ_MESSAGE | Event | 1 |
| PTP_PATH_DELAY_REQ_MESSAGE | Event | 2 |
| PTP_PATH_DELAY_RESP_MESSAGE | Event | 3 |
| Unused | | 4-7 |
| PTP_FOLLOWUP_MESSAGE | General | 8 |
| PTP_DELAY_RESP_MESSAGE | General | 9 |
| PTP_PATH_DELAY_FOLLOWUP_MESSAGE | General | A |
| PTP_ANNOUNCE_MESSAGE | General | B |
| PTP_SIGNALLING_MESSAGE | General | C |
| PTP_MANAGEMENT_MESSAGE | General | D |
| Unused | | E-F |

If V2 mode is configured in Section 8.2.3.26.15,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 register described at Section 8.2.3.26.6.



# 7.10 Virtualization

## 7.10.1 Overview

I/O virtualization is a mechanism that can be used to share I/O resources among several consumers. For example, in a virtual system, multiple operating systems are loaded and each operates as though the entire system's resources were at its disposal. However, for the limited number of I/O devices, this presents a problem because each operating system might be in a separate memory domain and all the data movement and device management has to be done by a Virtual Machine Monitor (VMM). VMM access adds latency and delay to I/O accesses and degrades I/O performance. Virtualized devices are designed to reduce the burden of the VMM by making certain functions of an I/O device shared among multiple guest operating systems or a Virtual Machine (VM), thereby allowing each VM direct access to the I/O device.

The 82599 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 Next Generation VMDq mode in this section.

The virtualization offloads capabilities provided by the 82599 apart from the replication of functions defined in the PCI SIG IOV specification are part of Next Generation VMDq.

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 82599 provides for these modes.

This section assumes a single-root implementation of IOV and no support for multi-root.

### 7.10.1.1 Direct Assignment Model

The direct assignment support in the 82599 is built according to the following model of the software environment.

It is assumed that one of the software drivers sharing the port hardware behaves as a master driver (Physical Function or PF driver). This driver is responsible for the initialization and the handling of the common resources of the port. All the other drivers (Virtual Function drivers or VF drivers) might read part of the status of the common parts but cannot change them. The PF driver might run either in the VMM or in some service operating system. It might be part of an IOVM or part of a dedicated service operating system.

In addition, part of the non time-critical tasks are also handled by the PF driver. For example, access to CSR through the I/O space or access to the configuration space are available only through the master interface. Time-critical CSR space like control of the Tx and Rx queue or interrupt handling is replicated per VF, and directly accessible by the VF driver.

**Note:** In some systems with a thick hypervisor, the service operating system might be an integral part of the VMM. For these systems, each reference to the service operating system in the sections that follow refer to the VMM.

#### 7.10.1.1.1 Rationale

The direct assignment model enables each of the VMs to receive and transmit packets with minimum of overhead. Non time-critical operations such as initialization and error handling can be done via the PF driver. In addition, it is important that the VMs can operate independently with minimal disturbance. It is also preferable that the VM interface to hardware should be as close as possible to the native interface in non-virtualized systems in order to minimize the software development effort.

The main time critical operations that require direct handling by the VM are:

- Maintenance of the data buffers and descriptor rings in host memory. In order to support this, the DMA accesses of the queues associated to a VM should be identified as such on the PCIe using a different requester ID.
- Handling of the hardware ring (tail bump and head updates)
- Interrupts handling

The capabilities needed to provide independence between VMs are:

- Per VM reset and enable capabilities
- Tx rate control
- Allocating separate CSR space per VM. This CSR space is organized as close as possible to the regular CSR space to enable sharing of the base driver code.

**Note:** The rate control and VF enable capabilities are controlled by the PF.

### 7.10.1.2 System Overview

The following drawings show the various elements involved in the I/O process in a virtualized system. Figure 7-38 shows the flow in software Next Generation **VMDq** mode and Figure 7-39 shows the flow in IOV mode.

This section assumes that in IOV mode, the driver on the guest operating system is aware that it operates in a virtual system (para-virtualized) and there is a channel between each of the VM drivers and the PF driver allowing message passing such as configuration request or interrupt messages. This channel can use the mailbox system implemented in the 82599 or any other means provided by the VMM vendor.

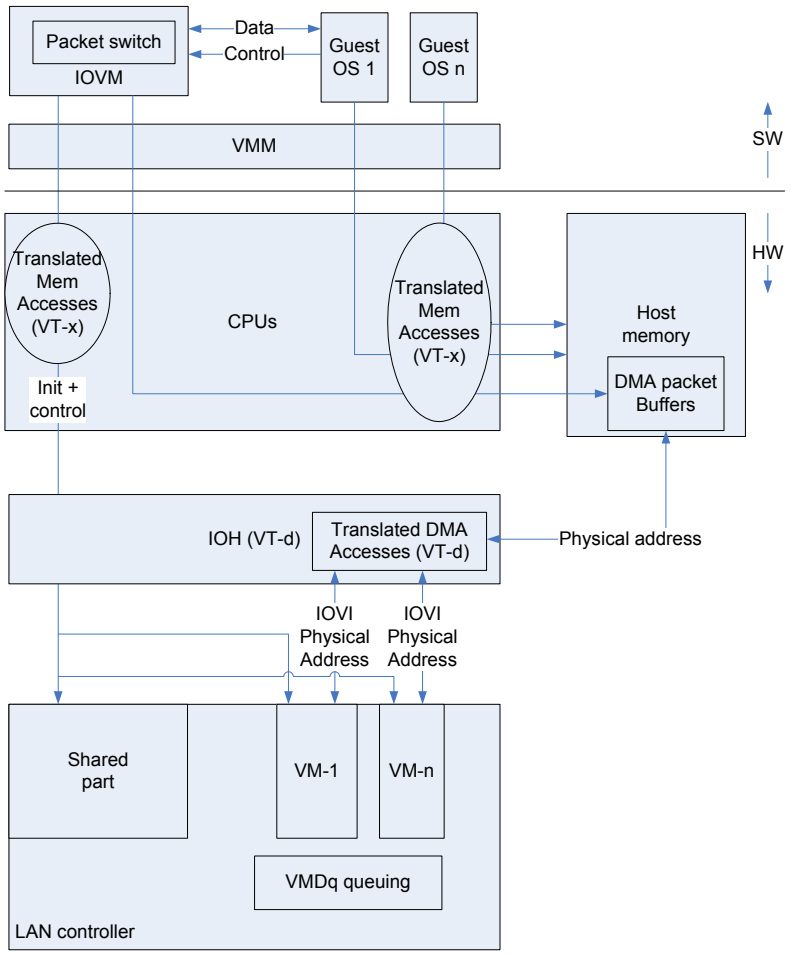



**Figure 7-38 System Configuration for Next Generation VMDq Mode**

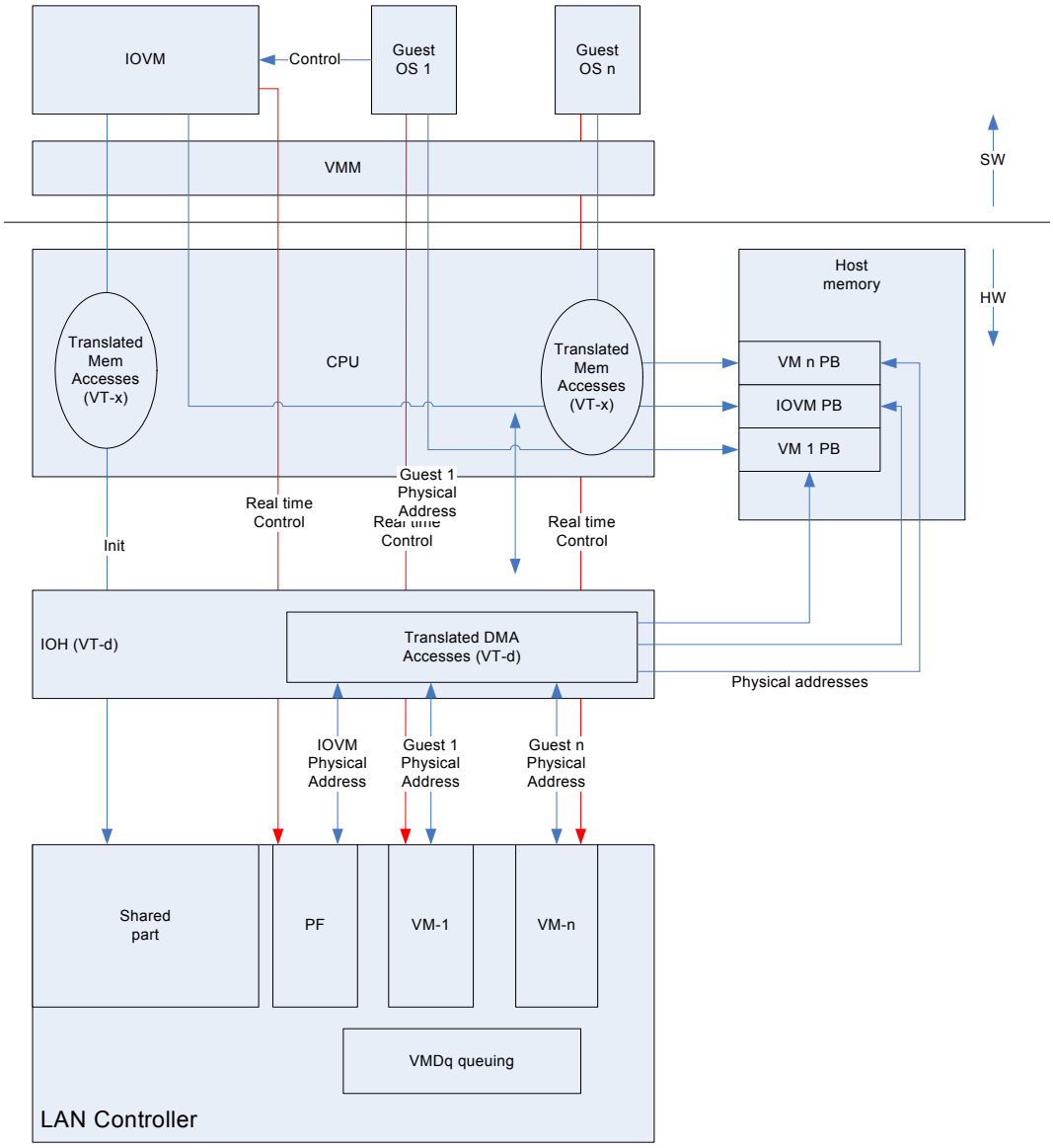
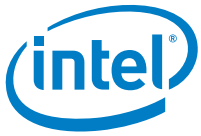



**Figure 7-39 System Configuration for IOV Mode**



## 7.10.2 PCI-SIG SR-IOV Support

### 7.10.2.1 SR-IOV Concepts

SR-IOV defines the following entities in relation to I/O virtualization:

- Virtual Image (VI): Part of the I/O resources are assigned to a A VM.
- I/O Virtual Intermediary (IOVI) or I/O Virtual Machine (IOVM): A special VM that owns the physical device and is responsible for the configuration of the physical device.
- Physical function (PF): A function representing a physical instance — One port for the 82599. The PF driver is responsible for the configuration and management of the shared resources in the function.
- Virtual Function (VF): A part of a PF assigned to a VI.

### 7.10.2.2 Configuration Space Replication

The SR-IOV specification defines a reduced configuration space for the virtual functions. Most of the PCIe configuration of the VFs comes from the PF.

This section describes the expected handling of the different parts of the configuration space for virtual functions. It deals only with the parts relevant to the 82599.

Details of the configuration space for virtual functions can be found in Section 9.5.

#### 7.10.2.2.1 Legacy PCI Configuration Space

The legacy configuration space is allocated to the PF only and emulated for the VFs. A separate set of BARs and one bus master enable bit is allocated in the SR-IOV capability structure in the PF and is used to define the address space used by the entire set of VFs.

All the legacy error reporting bits are emulated for the VF. See Section 7.10.2.4 for details.

#### 7.10.2.2.2 Memory BARs Assignment

The SR-IOV specification defines a fixed stride for all the VF BARs, so that each VF can be allocated part of the memory BARs at a fixed stride from the a basic set of BARs. In this method, only two decoders per replicated BAR per PF are required and the BARs reflected to the VF are emulated by the VMM.

The only BARs that are useful for the VFs are BAR0 and BAR3, so only those are replicated. The following table lists the existing BARs and the stride used for the VFs:

**Table 7-69 BARs in the 82599 (64-bit BARs)**

| BAR | Type | Usage | Requested Size per VF (=Stride) |
|---|---|---|---|
| 0, 1 | Mem | CSR space | Maximum (16 KB, page size). For page size see Section 9.4.4.8 for more details. |
| 2 | n/a | Not used | n/a |
| 3, 4 | Mem | MSI-X | Maximum (16 KB, page size). |
| 5 | n/a | Not used | n/a |

BAR0 of the VFs are a sparse version of the original PF BAR and include only the register relevant to the VF. For more details see Section 7.10.2.7.

Figure 7-40 shows the different BARs in an IOV-enabled system:

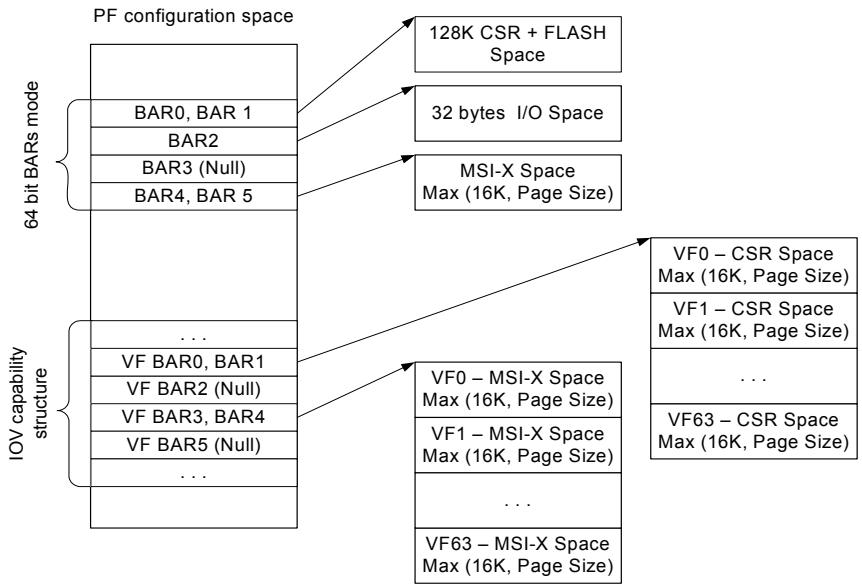
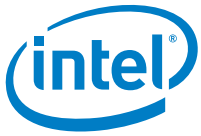



**Figure 7-40 BARs in an IOV-enabled System**

## 7.10.2.2.3 PCIe Capability Structure

The PCIe capability structure is shared between the PF and the VFs. The only relevant bits that are replicated are:

1. Transaction pending
2. Function Level Reset (FLR). See Section 7.10.2.3 for details.



#### 7.10.2.2.4 MSI and MSI-X Capabilities

Both MSI and MSI-X are implemented in the 82599. MSI-X vectors can be assigned per VF. MSI is not supported for the VFs.

See Section 9.3.8.1 for more details of the MSI-X and PBA tables implementation.

#### 7.10.2.2.5 VPD Capability

VPD is implemented only once and is accessible only from the PF.

#### 7.10.2.2.6 Power Management Capability

The 82599 does not support power management per VF. The power management registers exist for each VF, but only the D0 power state is supported.

#### 7.10.2.2.7 Serial ID

The serial ID capability is not supported in VFs.

#### 7.10.2.2.8 Error Reporting Capabilities (Advanced and Legacy)

All the bits in this capability structure are implemented only for the PF. Note that the VMs see an emulated version of this capability structure. See Section 7.10.2.4 for details.

### 7.10.2.3 FLR Capability

The *FLR* bit is required per VF. Setting of this bit resets only a part of the logic dedicated to the specific VF and does not influence the shared part of the port. This reset should disable the queues, disable interrupts and the stop receive and transmit process per VF.

Setting the PF *FLR* bit resets the entire function.

### 7.10.2.4 Error Reporting

Error reporting includes legacy error reporting and Advanced Error Reporting (AER) or role-based capability.

The legacy error management includes the following functions:

1. Error capabilities enablement. These are set by the PF for all the VFs. Narrower error reporting for a given VM can be achieved by filtering of the errors by the VMM. This includes:
   a. SERR# Enable
   b. Parity Error Response
   c. Correctable Reporting Enable

d. Non-Fatal Reporting Enable

e. Fatal Reporting Enable

f. UR Reporting Enable

2. Error status in the configuration space. These should be set separately for each VF. This includes:

   a. Master Data Parity Error

   b. Signaled Target Abort

   c. Received Target Abort

   d. Master Abort

   e. SERR# Asserted

   f. Detected Parity Error

   g. Correctable Error Detected

   h. Non-Fatal Error Detected

   i. Unsupported Request Detected

AER capability includes the following functions:

1. Error capabilities enablement. The *Error Mask*, and *Severity* bits are set by the PF for all the VFs. Narrower error reporting for a given VM can be achieved by filtering of the errors by the VMM. These includes:

   a. Uncorrectable Error Mask Register

   b. Uncorrectable Error Severity Register

   c. Correctable Error Mask Register

   d. ECRC Generation Enable

   e. ECRC Check Enable

2. Non-Function Specific Errors Status in the configuration space.

   a. Non-Function Specific Errors are logged in the PF

   b. Error logged in one register only

   c. VI avoids touching all VFs to clear device level errors

   d. The following errors are not function specific

      - All Physical Layer errors
      - All Link Layer errors
      - ECRC Fail
      - UR, when caused by no function claiming a TLP
      - Receiver Overflow
      - Flow Control Protocol Error
      - Malformed TLP
      - Unexpected Completion

3. Function Specific Errors Status in the configuration space.
   a. Allows Per VF error detection and logging
   b. Help with fault isolation
   c. The following errors are function specific
      - Poisoned TLP received
      - Completion Timeout
      - Completer Abort
      - UR, when caused by a function that claims a TLP
      - ACS Violation
4. Error logging. Each VF has it's own header log.
5. Error messages. In order to ease the detection of the source of the error, the error messages should be emitted using the requester ID of the VF in which the error occurred.

#### 7.10.2.5 Alternative Routing ID (ARI) and IOV Capability Structures

In order to allow more than eight functions per end point without requesting an internal switch, as usually needed in virtualization scenarios, the PCI-SIG defines the ARI capability structure. This is a new capability that enables an interpretation of the *Device* and *Function* fields as a single identification of a function within the bus. In addition, a new structure used to support the IOV capabilities reporting and control is defined. Both structures are described in sections Section 9.4.3 and Section 9.4.4. Refer to the following section for details on the Requester ID (RID) allocation to VFs.

#### 7.10.2.6 RID Allocation

RID allocation of the VF is done using the *Offset* field in the IOV structure. This field should be replicated per VF and is used to do the enumeration of the VFs.

Each PF includes an offset to the first associated VF. This pointer is a relative offset to the Bus/Device/Function (BDF) of the first VF. The *Offset* field is added to PF's requester ID to determine the requester ID of the next VF. An additional field in the IOV capability structure describes the distance between two consecutive VF's requester IDs.

##### 7.10.2.6.1 BDF Layout

###### 7.10.2.6.1.1 ARI Mode

ARI allows interpretation of the device ID part of the RID as part of the function ID inside a device. Thus, a single device can span up to 256 functions. In order to ease the decoding, the least significant bit of the function number points to the physical port number. The *Next* bits indicate the VF number. The following table lists the VF RIDs.

The layout of RID's used by the 82599 is reported to the operating system via the PCIe IOV capability structure. See Section 9.4.4.6.

**Table 7-70 RID per VF — ARI Mode**

| Port | VF# | B,D,F | Binary | Notes |
|---|---|---|---|---|
| 0 | PF | B,0,0 | B,00000,000 | PF |
| 1 | PF | B,0,1 | B,00000,001 | PF |
| 0 | 0 | B,16,0 | B,10000,000 | Offset to first VF from PF is 128. |
| 1 | 0 | B,16,1 | B,10000,001 | |
| 0 | 1 | B,16,2 | B,10000,010 | |
| 1 | 1 | B,16,3 | B,10000,011 | |
| 0 | 2 | B,16,4 | B,10000,100 | |
| 1 | 2 | B,16,5 | B,10000,101 | |
| ... | | | | |
| 0 | 63 | B,31,6 | B,11111,110 | |
| 1 | 63 | B,31,7 | B,11111,111 | Last |

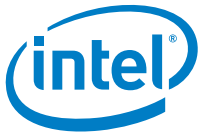
### 7.10.2.6.1.2 Non-ARI Mode

When ARI is disabled, non-zero devices in the first bus cannot be used, thus a second bus is needed to provide enough RIDs. In this mode, the RID layout is as follows:

**Table 7-71 RID per VF — Non-ARI Mode**

| Port | VF# | B,D,F | Binary | Notes |
|---|---|---|---|---|
| 0 | PF | B,0,0 | B,00000,000 | PF |
| 1 | PF | B,0,1 | B,00000,001 | PF |
| 0 | 0 | B+1,16,0 | B+1,10000,000 | Offset to first VF from PF is 384. |
| 1 | 0 | **B+1**,16,1 | B+1,10000,001 | |
| 0 | 1 | **B+1**,16,2 | B+1,10000,010 | |
| 1 | 1 | **B+1**,16,3 | B+1,10000,011 | |
| 0 | 2 | **B+1**,16,4 | B+1,10000,100 | |
| 1 | 2 | **B+1**,16,5 | B+1,10000,101 | |
| ... | | | | |
| 0 | 63 | **B+1**,31,6 | B+1,11111,110 | |
| 1 | 63 | **B+1**,31,7 | B+1,11111,111 | Last |

**Note:** When the device ID of a physical function changes (because of LAN disable or LAN function select settings), the VF device IDs changes accordingly.



### 7.10.2.7 Hardware Resources Assignment

The main resources to allocate per VM are queues and interrupts. The assignment is static. If a VM requires more resources, it might be allocated to more than one VF. In this case, each VF gets a specific Ethernet MAC address/VLAN tag in order to enable forwarding of incoming traffic. The two VFs are then teamed in software.

#### 7.10.2.7.1 PF Resources

A possible use of the PF is for a configuration setting without transmit and receive capabilities. In this case, it is not allocated to any queues but is allocated to one MSI-X vector.

The PF has access to all the resources of all VMs, but it is not expected to make use of resources allocated to active VFs.

#### 7.10.2.7.2 Assignment of Queues to VF

See Section 7.2.1.2.1 for allocating Tx queues.

See Section 7.1.2.2 for allocating Rx queues.

The following table lists the Tx and Rx queues to VF allocation.

**Table 7-72 Queue to VF Allocation**

| VF | Queues in 16 VMs Mode | Queues in 32 VMs Mode | Queues in 64 VMs Mode |
|---|---|---|---|
| 0 | 0-7 | 0-3 | 0-1 |
| 1 | 8-15 | 4-7 | 2-3 |
| ... | ... | ... | ... |
| 15 | 120-127 | ... | ... |
| ... | | ... | ... |
| 31 | | 124-127 | ... |
| ... | | | ... |
| 63 | | | 126-127 |

#### 7.10.2.7.3 Assignment of MSI-X Vector to VF

See Section 7.3.4.3 for allocating MSI-X vectors in IOV mode.

### 7.10.2.8 CSR Organization

CSRs can be divided into three types:

- Global Configuration registers that should be accessible only to the PF. For example, link control and LED control. These types of registers also include all of the debug features such as the mapping of the packet buffers and is responsible for most of the CSR area requested by the 82599. This includes per VF configuration parameters that can be set by the PF without performance impact.
- Per-VF parameters — For example, per VF reset, interrupt enable, etc. Multiple instances of these parameters are used only in an IOV system and only one instance is needed for non IOV systems.
- Per-queue parameters that should be replicated per queue — For example, head, tail, Rx buffer size, DCA tag, etc. These parameters are used by both a VF in an IOV system and by the PF in a non-IOV mode.

In order to support IOV without distributing the current drivers operation in legacy mode, the following method is used:

- The PF instance of BAR0 continues to contain the legacy and control registers. It is accessible only to the PF. The BAR enables access to all the resources including the VF queues and other VF parameters. However, it is expected that the PF driver does not access these queues in IOV mode.
- The VF instances of BAR0 provide control on the VF specific registers. These BARs have the same mapping as the original BAR0 with the following exceptions:
  a. Fields related to the shared resources are reserved.
  b. The queues assigned to a VF are mapped at the same location as the first same number of queues of the PF.
- Assuming some backward compatibility is needed for IOV drivers, The PF/VF parameters block should contain a partial register set as described in Section 8.3.

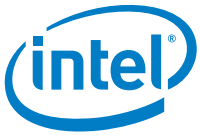
### 7.10.2.9 SR IOV Control

In order to control the IOV operation, the physical driver is provided with a set of registers. These include:

- The mailbox mechanism described in the next section.
- The switch and filtering control registers described in Section 7.10.3.10.
- PFVFLRE register indicating that a VFLR reset occurred in one of the VFs (bitmap).

#### 7.10.2.9.1 VF-to-PF Mailbox

The VF drivers and the PF driver require some means of communication between them. This channel can be used for the PF driver to send status updates to the VFs (such as link change, memory parity error, etc.) or for the VF to send requests to the PF (add to VLAN).



Such a channel can be implemented in software, but requires enablement by the VMM vendors. In order to avoid the need for such an enablement, the 82599 provides such a channel that enables direct communication between the two drivers.

The channel consists of a mailbox. Each driver can then receive an indication (either poll or interrupt) when the other side wrote a message.

Assuming a maximum message size of 64 bytes (one cache line), a memory of 64 bytes x 64 VMs =
4 KB. 512 bytes is provided per port. The RAM is organized as follows:

### Table 7-73 Mailbox Memory

| RAM Address | Function | PF BAR 0 Mapping[1] | VF BAR 0 Mapping[2] |
|---|---|---|---|
| 0 — 63 | VF0 <-> PF | 0 — 63 | VF0 + MBO |
| 64 — 127 | VF1 <-> PF | 64 — 127 | VF1 + MBO |
| .... | | | |
| (4 KB-64) — (4 KB-1) | VF63<-> PF | (4 KB-64) — (4 KB-1) | VF63 + MBO |

1. Relative to mailbox offset.
2. MBO = mailbox offset in VF CSR space.

In addition for each VF, the VFMailbox and PFMailbox registers are defined in order to coordinate the transmission of the messages. These registers contain a semaphore mechanism to enable coordination of the mailbox usage.

The PF driver can decide which VFs are allowed to interrupt the PF to indicate a mailbox message using the PFMBIMR mask register.

The following flows describe the usage of the mailbox:

### Table 7-74 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 VMFBMEM. | | |
| 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. |

### Table 7-74 PF-to-VF Messaging Flow (Continued)

| Step | PF Driver | Hardware | VF #n driver |
|---|---|---|---|
| 8 | | | Set the VFMailbox.ACK bit. |
| 9 | | Indicate an interrupt to PF. | |
| 10 | Clear PFMailbox[n].PFU | | |

1. The PF might implement a timeout mechanism to detect non-responsive VFs.

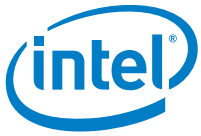
### Table 7-75 VF-to-PF Messaging Flow

| Step | PF Driver | Hardware | VF #n Driver |
|---|---|---|---|
| 1 | | | Set VFMailbox.VFU. |
| 2 | | Set VFU bit if VFMailbox[n].PFU is cleared. | |
| 3 | | | Read VFMailbox [n] and check that VFU bit was set. Otherwise wait and go to step 1. |
| 4 | | | Write message to relevant location in VFMBMEM. |
| 5 | | | Set the VFMailbox.REQ bit. |
| 6 | | Indicate an interrupt to PF. | |
| 7 | Read PFMBICR to detect which VF caused the interrupt. | | |
| 8 | Read the adequate message from VFMBMEM. | | |
| 9 | Set the PFMailbox.ACK bit. | | |
| 10 | | Indicate an interrupt to VF #n. | |
| 11 | | | Clear VFMailbox.VFU. |

The content of the message is hardware independent and is determined by software.

The messages currently assumed by this specification are:

- Registration to VLAN/multicast packet/broadcast packets — A VF can request to be part of a given VLAN or to get some multicast/broadcast traffic.
- Reception of large packet — Each VF should notify the PF driver what is the largest packet size allowed in receive.
- Get global statistics — A VF can request information from the PF driver on the global statistics.
- Filter allocation request — A VF can request allocation of a filter for queuing/ immediate interrupt support.



- Global interrupt indication.
- Indication of errors.

## 7.10.2.10 DMA

### 7.10.2.10.1 RID

Each VF is allocated a RID. Each DMA request should use the RID of the VM that requested it. See Section 7.10.2.6 for details.

### 7.10.2.10.2 Sharing of the DMA Resources

The outstanding requests and completion credits are shared between all the VFs. The tags attached to read requests are assigned the same way as in a non-virtualized setting, although in VF systems tags can be re-used for different RIDs. See Section 3.1.3.1.

### 7.10.2.10.3 DCA

The DCA enable is common to all the devices (all PFs and VFs). Given a DCA enabled device, each VM might decide for each queue, on which type of traffic (data, headers, Tx descriptors, Rx descriptors) the DCA should be asserted and what is the CPU ID assigned to this queue.

**Note:** There are no plans to virtualize DCA in the IOH. Thus, the physical CPU ID should be used in the programming of the CPUID field.

## 7.10.2.11 Timers and Watchdog

### 7.10.2.11.1 TCP Timer

The TCP timer is available only to the Physical Function (PF). It might indicate an interrupt to the VFs via the mailbox mechanism.

### 7.10.2.11.2 IEEE 1588

IEEE 1588 is a per-link function and thus is controlled by the PF driver. The VMs have access to the real time clock register.

### 7.10.2.11.3 Watchdog

The watchdog was originally developed for pass-through NICs where virtualization is not a viable. Thus, this functionality is used only by the PF.

#### 7.10.2.11.4 Free Running Timer

The free running timer is a PF driver resource the VMs can access. This register is read only to all VFs and is reset only by the PCI reset.

### 7.10.2.12 Power Management and Wake Up

Power management is a PF resource and is not supported per VF.

### 7.10.2.13 Link Control

The link is a shared resource and as such is controllable only by the PF. This includes interface settings, speed and duplex settings, flow control settings, etc. The flow control packets are sent with the station Ethernet MAC address stored in the EEPROM. The watermarks of the flow control process and the time-out value are also controllable by the PF only. In a DCB environment, the parameters of the per TC flow control are also part of the PF responsibilities.

Linksec is a per-link function and is controlled by the PF driver.

Double VLAN is a network setting and as such should be common to all VFs.

#### 7.10.2.13.1 Special Filtering Options

Pass bad packets is a debug feature. As such, pass bad packets is available only to the PF. Bad packets are passed according to the same filtering rules of the regular packets.

**Note:** Pass bad packets might cause guest operating systems to get unexpected packets. As a result, it should be used only for debug purposes of the entire system.

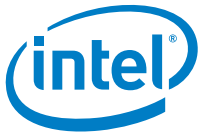
Receiving long packets is enabled separately per Rx queue in the RXDCTL registers. As this impacts the flow control thresholds, the PF should be made aware of the decision of all the VMs. Because of this, the setup of TSO packets is centralized by the PF and each VF might request this setting.

## 7.10.3 Packet Switching

### 7.10.3.1 Assumptions

The following assumptions are made:

- The required bandwidth for the VM-to-VM loopback traffic is low. That is, the PCIe bandwidth is not congested by the combination of the VM-to-VM and the regular incoming traffic. This case is handled but not optimized for. Unless specified otherwise, Tx and Rx packets should not be dropped or lost due to congestion caused by loopback traffic.



- If the buffer allocated for the VM-to-VM loopback traffic is full, it is acceptable to back pressure the transmit traffic of the same TC. This means that the outgoing traffic might be blocked if the loopback traffic is congested.
- The decision on local traffic is done only according to the Ethernet DA address and the VLAN tag. There is no filtering according to other parameters (IP, L4, etc.). The switch has no learning capabilities. In case of double VLAN mode, the inner VLAN is used for the switching functionality.
- The forwarding decisions are based on the receive filtering programming.
- No packet switching between TCs.
- Coexistence with IPSEC offload: Any loopback VM-to-VM traffic should not use the IPSEC offload (the *IPSEC* bit must be cleared in the advanced Tx data descriptor). IPsec processing of Tx packets destined to a local VM must be handled by software.
- Coexistence with TimeSync: time stamp is not sampled for any VM-to-VM loopback traffic.
- Coexistence with Double VLAN: When double VLAN is enabled by DMATXCTL.GDV and it is expected to transmit untagged packets by software, transmit-to-receive packet switching should not be enabled.

### 7.10.3.2 Pool Selection

Pool selection is described in the following sections. A packet might be forwarded to a single pool or replicated to multiple pools. Multicast and broadcast packets are cases of replication, as is mirroring.

The following capabilities determine the destination pools of each packet:

- 128 Ethernet MAC address filters (RAH/RAL registers) for both unicast and multicast filtering. These are shared with L2 filtering. For example, the same Ethernet MAC addresses are used to determine if a packet is received by the switch and to determine the forwarding destination.
- 64 shared VLAN filters (PFVLVF and PFVLVFB registers) — each VM can be made a member of each VLAN.
- Hash filtering of unicast and multicast addresses (if the direct filters previously mentioned are not sufficient)
- Forwarding of broadcast packets to multiple pools
- Forwarding by Ethertype
- Mirroring by pool, VLAN, or link

### 7.10.3.3 Rx Packets Switching

Rx packet switching is the second of three stages that determine the destination of a received packet. The three stages are defined in Section 7.1.2.

As far as switching is concerned, it doesn’t matter whether the 82599’s virtual environment operates in IOV mode or in Next Generation **VMDq** mode.

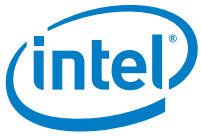
When operating in replication mode, broadcast and multicast packets can be forwarded to more than one pool, and is replicated to more than one Rx queue. Replication is enabled by the Rpl_En bit in the PFVTCTL register.

#### 7.10.3.3.1 Replication Mode Enabled

When replication mode is enabled, each broadcast/multicast packet can go to more than one pool. Finding the pool list of any packet is provided in the following steps:

1. Exact unicast or multicast match — If there is a match in one of the exact filters (RAL/RAH), for unicast or multicast packets, use the MAC Pool Select Array (MPSAR[n]) bits as a candidate for the pool list.
2. Broadcast — If the packet is a broadcast packet, add pools for which their PFVML2FLT.BAM bit (Broadcast Accept Mode) is set.
3. Unicast hash — If the packet is a unicast packet, and the prior steps yielded no pools, check it against the Unicast Hash Table (PFUTA). If there is a match, add pools for which their PFVML2FLT.ROPE bit (Accept Unicast Hash) is set.
4. 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.
5. 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.
6. VLAN groups — This step is relevant only when VLAN filtering is enabled by the VLNCTRL.VFE bit. 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]. Untagged packets: enable only pools with their PFVML2FLT.AUPE bit set. If there is no match, the pool list should be empty.

   **Note:** In a VLAN network, untagged packets are not expected. Such packets received by the switch should be dropped, unless their destination is a virtual port set to receive these packets. The setting is done through the PFVML2FLT.AUPE bit. It is assumed that VMs for which this bit is set are members of a default VLAN and thus only MAC queuing is done on these packets.
7. 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).
8. Ethertype filters — If one of the Ethertype filters (ETQF) is matched by the packet and queuing action is requested and the Pool Enable bit in the ETQF is set, the pool list is set to the pool pointed to by the filter.
9. 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.
10. 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 VMVLAN register.

c. Uplink port mirroring — PFMRCTL.UPME is set, the pool list is not empty.

d. PFVFRE — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list. The PFVFRE register blocks reception by a VF while the PF configures its registers. Note that this stage appears twice in order to handle mirroring cases.

#### 7.10.3.3.2 Replication Mode Disabled

When replication mode is disabled, software should take care of multicast and broadcast packets and check which of the VMs should get them. In this mode, the pool list of any packet always contains one pool only according to the following steps:

1. Exact unicast or multicast match — If the packet DA matches one of the exact filters (RAL/RAH), use the MAC Pool Select Array (MPSAR[n]) bits as a candidate for the pool list.
2. Unicast hash — If the packet is a unicast packet, and the prior step 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.
3. VLAN groups — This step is relevant only when VLAN filtering is enabled by the VLNCTRL.VFE bit. 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]. Untagged packets: enable only pools with their PFVML2FLT.AUPE bit set. If there is no match, the pool list should be empty.
4. 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).
5. Multicast or broadcast — If the packet is a multicast or broadcast packet and was not forwarded in step 1 and 2, set the default pool bit in the pool list (from PFVTCTL.DEF_PL).
6. 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.
7. PFVFRE — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list. The PFVFRE register blocks reception by a VF while the PF configures its registers.

The following software limitations apply when replication is disabled:

- Software must not set more than one bit in the bitmaps of the exact filters. Note that multiple bits can be set in an RAH register as long as it's guaranteed that the packet is sent to only one queue by other means (such as VLAN).
- Software must not set per-VM promiscuous bits (multicast or broadcast).
- Software must not set the *ROPE* bit in more than one PFVML2FLT register.
- Software should not activate mirroring.

### 7.10.3.4 Tx Packets Switching

Tx switching is used only in a virtualized environment to serve VM-to-VM traffic. Packets that are destined to one or more local VMs are directed back (loopback) to the Rx packet buffers. Enabling Tx switching is done by setting the PFDTXGSWC.LBE bit. Tx to Rx switching always avoids packet drop as if flow control is enabled. Therefore, the software must set the FCRTH[n].RTH fields regardless if flow control is activated on the 82599.

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:

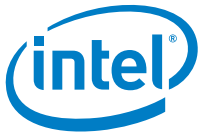
- Loopback is disabled when the network link is disconnected. It is expected (but not required) that system software (including VMs) does not post packets for transmission when the link is disconnected. Note that packets posted by system software for transmission when the link is down are buffered.
- Loopback is disabled when the RXEN (*Receive Enable*) bit is cleared.
- Loopback packets are identified by the *LB* bit in the receive descriptor.

**Note:** When Tx switching is enabled, the host must avoid sending packets longer than 9.5 KB as this hangs the Tx path.

#### 7.10.3.4.1 Replication Mode Enabled

When replication mode is enabled, the pool list for any packet is determined according to the following steps:

1. Exact unicast or multicast match — If there is a match in one of the exact filters (RAL/RAH), for unicast or multicast packets, take the MPSAR[n] bits as a candidate for the pool list.
2. Broadcast — If the packet is a broadcast packet, add pools for which their PFVML2FLT.BAM bit (Broadcast Accept Mode) is set.
3. Unicast hash — If the packet is a unicast packet, and the prior steps yielded no pools, check it against the Unicast Hash Table (PFUTA). If there is a match, add pools for which their PFVML2FLT.ROPE bit (Accept Unicast Hash) is set.
4. 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.
5. 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.



6. Filter source pool — The pool from which the packet was sent is removed from the pool list unless the PFVMTXSW.LLE bit is set.
7. VLAN groups — This step is relevant only when VLAN filtering is enabled by the VLNCTRL.VFE bit. 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]. Untagged packets: enable only pools with their PFVML2FLT.AUPE bit set. If there is no match, the pool list should be empty.
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.
9. PFVFRE — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list (pre mirroring step). Refer to the notes after step 11.
10. Mirroring — Each of the following three mirroring rules adds its destination pool (PFMRCTL.MP) to the pool list if the following applies:
    a. Pool mirroring — PFMRCTL.VPME is set and one of the bits in the pool list matches one of the bits in the PFMRVM register.
    b. VLAN port mirroring — PFMRCTL.VLME is set and the index of the VLAN of the packet in the PFVLVF table matches one of the bits in the VMVLAN register.
    c. Downlink port mirroring — PFMRCTL.DPME is set and the packet is sent to the network.
11. PFVFRE — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list (post mirroring step). Refer to the following notes.

**Note:** The PFVFRE filtering is applied only after the decision to forward the packet to network and/or local pool (based on MAC address and VLAN). If a packet that matches an exact MAC address is set to be forwarded to a local pool, it is not sent to the network regardless of the PFVFRE setting. Therefore, when a pool is disabled, software should also clear its exact MAC address filters before clearing the PFVFRE.

### 7.10.3.4.2 Replication Mode Disabled

When replication mode is disabled, software should take care of multicast and broadcast packets and check which of the VMs should get them. In this mode the pool list for any packet always contains one pool only according to the following steps:

1. Exact unicast or multicast match — If the packet DA matches one of the exact filters (RAL/RAH), take the MPSAR[n] bits as a candidate for the pool list.
2. 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.
3. VLAN groups — This step is relevant only when VLAN filtering is enabled by the VLNCTRL.VFE bit. 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]. Untagged packets: enable only pools with their PFVML2FLT.AUPE bit set. If there is no match, the pool list should be empty.

4. 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).
5. Filter source pool — The pool from which the packet was sent is removed from the pool list unless the PFVMTXSW.LLE bit is set.
6. 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.
7. PFVFRE — If any bit in the PFVFRE register is cleared, clear the respective bit in the pool list.

**Note:** The PFVFRE filtering is applied only after the decision to forward the packet to network and/or local pool (based on MAC address and VLAN). If a packet that matches an exact MAC address is set to be forwarded to a local pool, it is not sent to the network regardless of the PFVFRE setting. Therefore, when a pool is disabled, software should also clear its exact MAC address filters before clearing the PFVFRE.

The following software limitations apply when replication is disabled:

1. It is software's responsibility not to set more than one bit in the bitmaps of the exact filters. Note that multiple bits can be set in an RAH register as long as it is guaranteed that the packet is sent to only one queue by other means (such as VLAN)
2. Software must not set per-VM promiscuous bits (multicast or broadcast).
3. Software must not set the *ROPE* bit in more than one PFVML2FLT register.
4. Software should not activate mirroring.

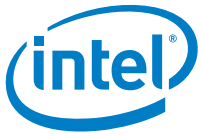
### 7.10.3.5 Mirroring Support

The 82599 supports four separate mirroring rules, each associated with a destination pool (mirroring can be done into up to four pools). Each rule is programmed with one of the four mirroring types:

1. Pool mirroring — reflect all the packets received to a pool from the network.
2. Uplink port mirroring — reflect all the traffic received from the network.
3. Downlink port mirroring — reflect all the traffic transmitted to the network.
4. VLAN mirroring — reflect all the traffic received from the network in a set of given VLANs (either from the network or from local VMs).

**Note:** Reflecting all the traffic received by any of the pools (either from the network or from local VMs) is supported by enabling mirroring of all pools.

Mirroring and replication on FCoE traffic is not supported on receive if the ETQF filters define FCoE packets and on transmit if the packets are indicated as FCoE (by setting the FCoE bit in the TUCMD field in the Transmit Context Descriptor).



Mirroring modes are controlled by a set of rule control registers:

- PFMRCTL — controls the rules to be applied and the destination port.
- PFMRCTL — controls the VLAN ports as listed in the PFVLVF table taking part in the VLAN mirror rule.
- PFMRVM — controls the pools taking part in the pool mirror rule.

## 7.10.3.6 Offloads

The general rule is that offloads are executed as configured for the pool and queue associated with the receive packet. Some special cases:

- If a packet is directed to a single pool, then offloads are determined by the pool and queue for that packet.
- If a packet is replicated to more than one pool, then each copy of the packet is offloaded according to the configuration of its pool and queue.
- If replication is disabled, offloads are determined by the unique destination of the packet.

The following subsections describe exceptions to the previously described special cases.

### 7.10.3.6.1 Local Traffic Offload

The following capabilities are not supported on the loopback path:

- The Ethertype filters do not apply.
- Padding to a legal packet size is not supported.
- The following offload capabilities are only supported if XSUM offload is provided on the Tx path for the packet: RSS, 5-tuple filters, VLAN strip. The reason is that when XSUM is not offloaded, software does not provide the necessary offload offsets with the Tx packet.
- Header split/replication is not supported for NFS.
- Receive Side Coalescing (RSC) is not supported.
- FCoE offloads are not supported.
- IPSec offload is not supported.

### 7.10.3.6.2 Rx Traffic Offload

- Security offloads (LinkSec, IPsec) are managed globally and not per pool.
- CRC offload is a global policy. CRC strip is enabled or disabled for all received packets.

### 7.10.3.7 Congestion Control

- Tx packets going through the local switch are stored in the Rx packet buffer, similar to packets received from the network. Tx to Rx switching always avoids packet drop as if flow control is enabled. Therefore, the software must set the FCRTH[n].RTH fields regardless if flow control is activated on the 82599.

The 82599 guarantees that one TC flow is not affected by congestion in another TC.

Receive and local traffic are provided with the same priority and performance expectations. Packets from the two sources are merged in the Rx packet buffers, which can in general support both streams at full bandwidth. Any congestion further in the pipeline (such as lack of PCIe bandwidth) evenly affects Rx and local traffic.

### 7.10.3.8 Tx Queue Arbitration and Rate Control

In order to guarantee each pool with adequate bandwidth, a per-pool bandwidth control mechanism is added to the 82599. Each Tx pool gets a percentage of the transmit bandwidth and is guaranteed it can transmit within its allocation. This arbitration is combined with the TC arbitration. See additional details on DCB Tx capabilities in Section 7.7.2.2.

### 7.10.3.9 Security Features

The 82599 allows some security checks on the inbound and outbound traffic of the switch.

#### 7.10.3.9.1 Inbound Security

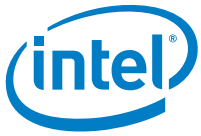
Each incoming packet (either from the LAN or from a local VM) is filtered according to the VLAN tag so that packets from one VLAN cannot be received by pools that are not members of that VLAN.

#### 7.10.3.9.2 Outbound Security

MAC anti-spoofing

Each pool is associated with one or more Ethernet MAC addresses on the receive path. The association is determined through the MPSAR registers. The MAC anti-spoofing capability insures that a VM always uses a source Ethernet MAC address on the transmit path that is part of the set of Ethernet MAC addresses defined on the Rx path. A packet with a non-matching SA is dropped, preventing spoofing of the Ethernet MAC address. This feature is enabled in the PFVFSPOOF.MACAS field, and can be enabled per Tx pool.

**Note:** Anti-spoofing is not available for VMs that hide behind other VMs whose Ethernet MAC addresses are not part of the RAH/RAL Ethernet MAC Address registers. In this case, anti-spoofing should be done by software switching, handling these VMs.



### VLAN anti-spoofing

Each pool is associated with one or more VLAN tags on the receive path. The association is determined through the PFVLVF and PFVLVFB registers. The VLAN anti-spoofing capability insures that a VM always uses a VLAN tag on the transmit path that is part of the set of VLAN tags defined on the Rx path. A packet with a non-matching VLAN tag is dropped, preventing spoofing of the VLAN tag. This feature is enabled in the PFVFSPOOF.VLANAS field, and can be enabled per Tx pool.

**Note:** If VLAN anti-spoofing is enabled, then MAC anti-spoofing must be enabled as well.

When double VLAN is enabled by DMATXCTL.GDV and it is expected to transmit untagged packets by software, VLAN anti-spoofing should not be enabled.

### VLAN tag validation

In PCI-SIG IOV scenarios the driver might be malicious, and thus may fake a VLAN tag. The 82599 provides the ability to override the VLAN tag inserted by a VM. The possible behaviors are controlled by the PFVMVIR[n] registers as follows:

- Use descriptor value — to be used in case of a trusted VM that can decide which VLAN to send. This option should also be used in case one VM is member of multiple VLANs.
- Always insert default VLAN — this mode should be used for non-trusted or non-VLAN aware VMs. In this case, any VLAN insertion command from the VM is ignored. If a packet is received with a VLAN, the packet should be dropped.
- Never insert VLAN — This mode should be used in a non-VLAN network. In this case, any VLAN insertion command from the VM is ignored. If a packet is received with a VLAN, the packet should be dropped.

**Note:** The VLAN insertion settings should be done before any of the queues of the VM are enabled.

When double VLAN is enabled by DMATXCTL.GDV and it is expected to transmit untagged packets by software, VLAN validation should not be enabled.

## 7.10.3.10 Switch Control

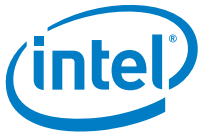
The PF driver has some control of the switch logic. The following registers are available to the PF for this purpose:

PFVTCTL: - VT Control register — contains the following fields:

- Replication Enable (Rpl_En) — enables replication of multicast and broadcast packets — both in incoming and local traffic. If this bit is cleared, Tx multicast and broadcast packets are sent only to the network and Rx multicast and broadcast packets are sent to the default pool.
- Default Pool (DEF_PL) — defines the target pool for packets that passed L2 filtering but didn’t pass any of the pool filters. This field is invalid when the Dis_Def_Pool bit is set.
- Disable Default Pool (Dis_Def_Pool) — disables acceptance of packets that failed all pool filters.

- PFVFRE — Enables/disables reception of packets from the link to a specific VF. Used during initialization of the VF. See Section 4.2.2.2 for more details.
- PFDTXGSWC (LBE) — VMDQ loopback enables switching of Tx traffic to the Rx path for VM-to-VM communication.
- PFVFSPOOF — MAC Anti-spoof Enable (MACAS) — enables filtering of Tx packet for anti-spoof.
- Local Loopback Enable (LLE) — defines whether or not to allow loopback of a packet from a certain pool into itself.
- Queue Drop Enable (PFQDE) register — A register defining global policy for drop enable functionality when no descriptors are available. It lets the PF override the per-queue SRRCTL[n] Drop_En setting. PFQDE should be used in SR-IOV mode as described in Section 4.6.11.3.1.
- PFVML2FLT — Receive Overflow Multicast Packets (ROMPE) — accept multicast hash — Defines whether or not a pool accepts packets that match the multicast MTA table.
- Receive MAC Filters Overflow (ROPE) — accept unicast hash — Defines whether or not a pool accepts packets that match the unicast PFUTA table.
- Broadcast Accept (BAM) — Defines whether or not a pool accepts broadcast packets.
- Multicast Promiscuous (MPE) — Defines whether or not a pool accepts all multicast packets.
- Accept Untagged Packets Enable (AUPE) — Defines whether or not a pool accepts untagged VLAN packets.
- Mirror Control — See Section 7.10.3.5.
- PFVFTE — Enables/disables transmission of packets to the link to a specific VF. Used during initialization of the VF. See Section 4.2.2.2 for more details.
- PFVLVF/PFVLVFB — VLAN queuing table — A set of 64 VLAN entries with an associated bitmap, one bit per pool. Bits are set for each pool that participates in this VLAN.
- Unicast Table Array (PFUTA) — a 4 Kb array that covers all combinations of 12 bits from the MAC destination address. A received unicast packet that misses the MAC filters is compared against the PFUTA. If the relevant bit in the PFUTA is set, the packet is routed to all pools for which the *ROPE* bit is set.
- Multicast Table Array (MTA) — a 4 Kb array that covers all combinations of 12 bits from the MAC destination address. A received multicast packet that misses the MAC filters is compared against the MTA. If the relevant bit in the MTA is set, the packet is routed to all pools for which the *ROMPE* bit is set.

In addition, the rate-control mechanism is programmed as described in Section 7.7.2.2.



## 7.10.4 Virtualization of Hardware

This section describes additional features used in both IOV and Next Generation **VMDq** modes.

### 7.10.4.1 Per-pool Statistics

Part of the statistics are by definition shared and cannot be allocated to a specific VM. For example, CRC error count cannot be allocated to a specific VM, as the destination of such a packet is not known if the CRC is wrong.

All the non-specific statistics are handled by the PF driver in the same way it is done in non-virtualized systems. A VM might request a statistic from the PF driver but might not access it directly.

The conceptual model used to gather statistics in a virtualization context is that each queue pool is considered as a virtual link and the Ethernet link is considered as the uplink of the switch. Thus, any packet sent by a pool is counted in the Tx statistics, even if it was forwarded to another pool internally or was dropped by the MAC for some reason. In the same way, a replicated packet is counted in each of the pools receiving it.

The following statistics are provided per pool:

- Good packet received count
- Good packet transmitted count
- Good octets received count
- Good octets transmitted count
- Multicast packets received count

**Note:** All the per VF statistics are read only and wrap around after reaching their maximum value.

# 7.11 Receive Side Coalescing (RSC)

The 82599 can merge multiple received frames from the same TCP/IP connection (referred to as flow in this section) into a single structure. The 82599 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 82599 digests received packets and categorizes them by their TCP/IP connections (flows). For each flow, hardware coalesces the packets as shown in Figure 7-41 and Figure 7-42 (the colored parameters are explained in the RSC context table and receive descriptor sections). The 82599 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 82599 opens and closes the RSC contexts autonomously with no need for any software intervention. Software needs only to enable RSC in the selected receive queues.

Figure 7-41 shows a top level flow diagram that is used for RSC functionality. The following sections provide a detailed explanation of this flow as well as the memory structures and device settings that support the RSC functionality.

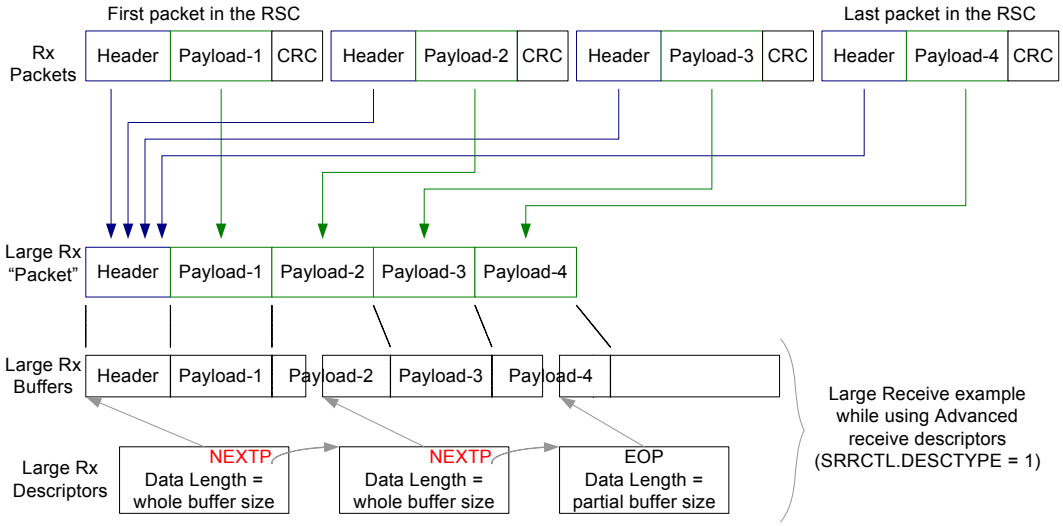



**Figure 7-41 RSC Functionality (No Header Split)**

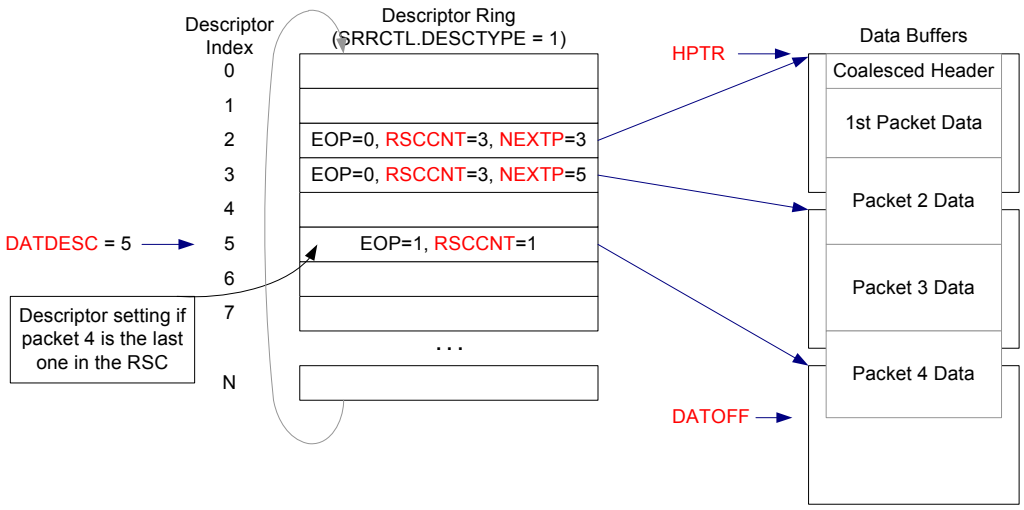



**Figure 7-42 RSC Functionality (No Header Split)**

**Note:** Software might abort reception to any queue at any time. For example: VFLR or queue disable. Following these settings, hardware aborts further DMA(s) and descriptor completions. Specifically, active RSC(s) in the specific queue(s) are not completed. In such cases there could be completed packets and RSC(s) hidden from software by prior incomplete RSC(s).

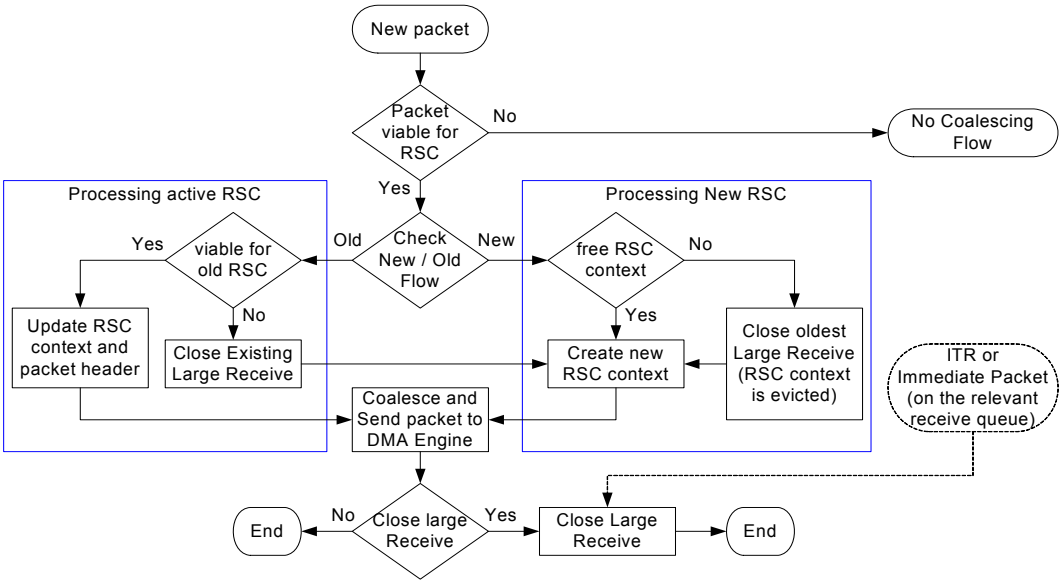
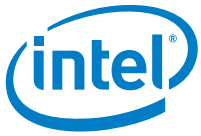



**Figure 7-43 RSC Event Flow**

### 7.11.1 Packet Viability for RSC Functionality

Incoming packets can be good candidates for RSC offload when the following conditions are met. If any of the these conditions are not met, the received packet is processed in the legacy (non-coalescing) scheme.

- RSC is not disabled globally by the RFCTL.RSC_DIS bit. Note that in SR-IOV mode the RSC must be disabled globally by setting the RFCTL.RSC_DIS bit.
- 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.
- The SRRCTL[n].BSIZEHEADER (header buffer size) must be larger than the packet header (even if header split is not enabled). A minimum size of 128 bytes for the header buffer addresses this requirement.
- The SRRCTL[n].BSIZEPACKET (packet buffer size) must be 2 KB at minimum.
- The received packet has no MAC errors and no TCP/IP checksum errors. MAC errors are: CRC error or undersize frame received or oversize frame received or error control byte received in mid-packet or illegal code byte received in mid-packet.
- If the *Length* field in the IP header does not cover the entire packet (as the case for padding bytes) then the received packet is not a candidate for RSC.
- If the packet carries LinkSec encapsulation, the LinkSec offload is activated on the packet with no errors.
- The packet type is TCP/IPv4 (non-SNAP) with optional VLAN header.
- IP header does not carry any option headers.
- NFS packets can be coalesced only if NFS filtering is disabled by setting both RFCTL.NFSW_DIS and RFCTL.NFSR_DIS bits to 1b. Furthermore, the PSR_type1 bit (header split on NFS) must be turned off in all PSRTYPE[n] registers.
- If NFS coalescing is not required, software should set both RFCTL.NFSW_DIS and RFCTL.NFSR_DIS bits to 0b.
- The packet does not carry IPsec encapsulation (regardless if IPsec offload is enabled).
- The TCP segment is not fragmented.
- The following TCP flags are inactive: FIN, SYN, RST, PSH, URG, ECE, CWR, NS and the other three reserved TCP flags (see TCP Flags mapping in Table 7-76).
- The *ECT* and *CE* bits in the *TOS* field in the IP header are not equal to 11b (see the flags in Table 7-77).
- The packet does not carry any TCP option headers.
- Virtualization rule 1: RSC is not supported for switched packet transmitted from a local VM.
- Virtualization rule 2: When a Rx packet is replicated or mirrored, it might be coalesced only on the Rx queue that belongs to the source VM.
- Note that there are no limitations on the maximum packet length including jumbo packets.



- If there is already an active RSC for the matched flow, then a few additional conditions should be met as listed in Section 7.11.4.

The supported packet format is as follows:

| Size | Packet fields |
|---|---|
| 6 Byte | Destination Ethernet MAC address |
| 6 Byte | Source Ethernet MAC address |
| [8 / 16 Byte] | Optional LinkSec header (supported by RSC only if LinkSec offload is enabled and the hardware extracts this header) |
| [4 Byte] | Optional VLAN |
| [4 Byte] | Optional 2nd VLAN (double VLAN) |
| 2 Byte | Ethernet type field equals 0x0800 (MS byte first on the wire) |
| 20 Byte | IPv4 header with no options |
| 20 Byte | Basic TCP header (no options — refer to the rows that follow) |
| [10 Byte] | Optional TCP time stamp header:<br>1 Byte Kind 0x08<br>1 Byte Length 0x0A<br>4 Byte TS value variable<br>4 Byte TS echo reply variable |
| [1 Byte] | Optional TCP no operation header<br>1 Byte Kind 0x01 |
| [1 Byte] | Optional TCP end of option header list<br>1 Byte Kind 0x00 |
| Variable length | TCP payload (RSC candidate must have payload size greater than zero) |
| [8 / 18 Byte] | Optional LinkSec Integrity Checksum Value — ICV (supported by RSC only if LinkSec offload is enabled and the hardware extracts this field) |

### Table 7-76 Packet Format Supported by RSC

| 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | NS | CWR | ECE | URG | ACK | PSH | RST | SYN | FIN |

### Table 7-77 IP TOS Field — Bit Map

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TOS (DS) | | | | | | ECT | CE |

**Table 7-78 TCP Time-Stamp Option Header (RFC 1323)**

| 1 byte: First on the wire | 1 byte | 4 byte | 4 bytes: Last on the wire |
|---|---|---|---|
| Kind = 0x8 | Length = 10 | TS Value (TSval) | TS Echo Reply (TSecr) |

## 7.11.2 Flow Identification and RSC Context Matching

TCP/IP packet's flow is identified by its four tuples: Source / Destination IP addresses and Source / Destination TCP port numbers. These tuples are compared against the *Flow Identification* fields stored in the active RSC contexts (listed in Table 7-79). 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 Section 7.11.3.
- If the packet's flow matches an active RSC context then the packet might be appended to the existing RSC as described in Section 7.11.4.

**Table 7-79 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 zero when RSC completes. |
| 1 bytes | 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 1 for IPv4). |
| 2 bytes | TCPDPORT | TCP destination port. |
| 2 bytes | TCPSPORT | TCP source port. |

## Table 7-79 RSC Context (Continued)

| Size | Name | Description |
|---|---|---|
| 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). |
| 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.<br>RSCACKTYPE case: The ACK sequence number in the last good packet.<br>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 7.11.5.3). |
| 2 bytes | DATDESC | Data descriptor is the active descriptor index. Initialized by the first packet in the RSC to the first descriptor. It is updated to the active descriptor at a packet DMA completion. |
| 2 bytes | DATOFF | Offset within the data buffer. The data of the first packet in a large receive is the same as the legacy (non-coalescing) definition. Following a DMA completion, it points to the beginning of the data portion of the next packet. |
| 13 bytes | | Total. |

1. These parameters are extracted from the first packet that opens (activate) the context.
2. All parameters are set by the first packet that opens the context while some are dynamic.

## 7.11.3 Processing New RSC

Defining the RSC context parameters activates a new large receive. If a received packet does not match any active RSC context, the packet starts (opens) a new one. If there is no free context, the oldest active large receive is closed and its evicted context is used for the new large receive.

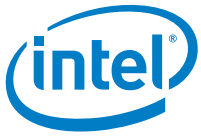
### 7.11.3.1 RSC Context Setting

The 82599 extracts the flow identification and RSC header parameters from the packet that opens the context (the first packet in a large receive that activates an RSC context). The context parameters can be divided into categories: flow identification; RSC header and DMA parameters.

## 7.11.4 Processing Active RSC

Received packets that belong to an active RSC can be added to the large receive if all the following conditions are met:

- The L2 header size equals the size of previous packets in the RSC as recorded in the internal IPOFF parameter in the RSC context table.
- The packet header length as reported in the HDR_LEN field is assumed to be the same as the first packet in the RSC (not checked by hardware).
- The ACK bit in the TCP header is 1b or equals to the RSCACK bit in the RSC context (an active RSCACK context and inactive received ACK bit is defined as no match).
- The packet type remains the same as indicated by the RSCACKTYPE bit in the RSC context. Packet type can be either ACK packet (with no TCP payload) or other.
- For non-RSCACKTYPE (packet with TCP payload): The sequence number in the TCP header matches the expected value in the RSC context (RSCSEQ).
- For RSCACKTYPE: The ACK sequence number in the TCP header is greater than the RSCSEQ number in the RSC context. Note that the 82599 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 82599 closes the old large receive it is guaranteed that there is at least one free context.



If the received packet meets all the above conditions, the 82599 appends this packet to the active large receive and updates the context as follows. The packet is then DMA'ed to the RSC buffers (as described in Section 7.11.5).

- Update the TCP ACK: The RSCACK in the large receive context gets the value of the ACK bit in the TCP header in the received packet.
- Update the expected sequence number for non-RSCACKTYPE: The RSCSEQ in the large receive context is incremented by the value of the TCP payload size of the received packet.
- Update the expected sequence number for RSCACKTYPE: The RSCSEQ in the large receive context is updated to the value of the ACK sequence number field in the received packet.
- Update the total length: The RSCIPLEN in the large receive context is incremented by the value of the TCP payload size of the received packet. The value of the *Total Length* field in the IP header in the received packet gets the updated RSCIPLEN. Note that in RSCACKTYPE packets the received payload size is zero.
- Note that LinkSec encapsulation (if it exists) is stripped first by hardware. In this case, hardware also strips the Ethernet padding (if it exists).
- IP header checksum is modified to reflect the changes in the *Total Length* field as follows (note that there is no special process for RSCACKTYPE packets):

1's {(RSCIPLEN — Packet total length) + 1's (Packet IP header checksum)} while...

  - — Packet total length is the total length value in the received packet.
  - — Packet IP header checksum stands for the IP header checksum field in the received packet.
  - — 1's operation defines a ones complement.
  - — Plus (+) operation is a cyclic plus while the carry out is fed as a carry in.

- TCP header checksum is left as is in the first packet in the RSC and is set to zero on any succeeding packets.
- Update the DMA parameters.
  - — The RSCCNT is initialized to 0x1 on each new descriptor. It is then incremented by one on each packet that starts on the same descriptor as long as it does not exceed a value of 0xF. When the RSCCNT is set to 0xF (14 packets) the RSC completes.
  - — Decrement by one the Remaining Descriptors (RSCDESC) for each new descriptor.
  - — Update the receive descriptor index (DATDESC) for each new descriptor.
  - — Update the offset within the data buffer (DATOFF) at the end of the DMA to its valid value for the next packet.
- All other fields are kept as defined by the first packet in the large receive.

## 7.11.5 Packet DMA and Descriptor Write Back

The Figure 7-44 shows a top view of the RSC buffers using advanced receive descriptors and header split descriptors.

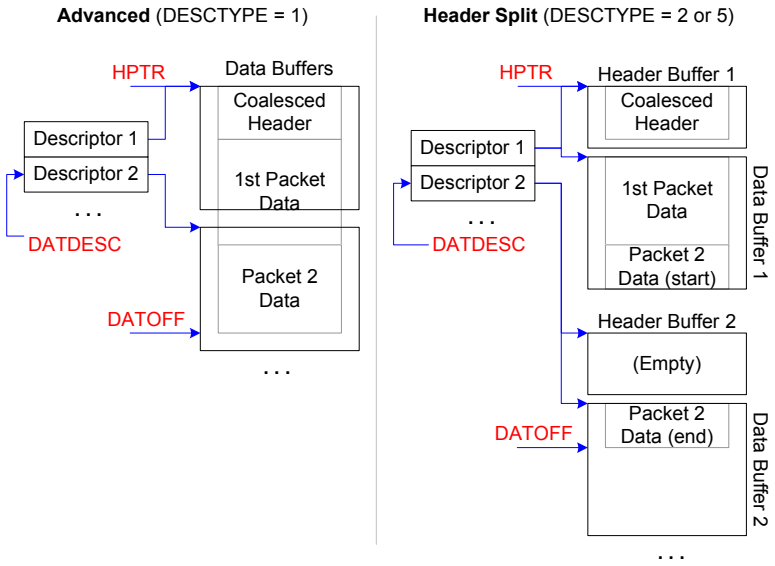
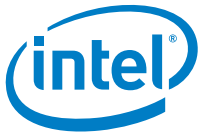



**Figure 7-44 RSC — Header and Data Buffers**

### 7.11.5.1 RSC Descriptor Indication (Write Back)

Following reception of each packet, the 82599 posts the packet data to the data buffers and updates the coalesced header in its buffer. Any completed descriptor is indicated (write back) by setting the fields listed in the following table. A descriptor is defined as the last one when an RSC completes. Section 7.11.5.1 summarizes all the causes for RSC completion. Any other descriptor in the middle of the RSC is indicated (write back) when the hardware requires the next descriptor so it can report the NEXTP field explained as follows.

| Fields on the Last Descriptors of Large Receive | Fields on All Descriptors Except for the Last One |
|---|---|
| **EOP**: End of packet, and all other fields that are reported together with the EOP. | **NEXTP**: Points to the next descriptor of the same large receive. |
| **DD**: indicates that this descriptor is completed by the hardware and can be processed by the software. | |
| **RSCCNT**: indicates the number of coalesced packets in this descriptor. | |



### 7.11.5.2 Received Data DMA

On the first packet of a large receive, the entire packet is posted to its buffers in host memory. On any other packet, the packet's header and data are posted to host memory as detailed in Section 7.11.5.3 and Section 7.11.5.4.

### 7.11.5.3 RSC Header

The RSC header is stored at the beginning of the first buffer when using advanced receive descriptors, or at the header buffer of the first descriptor when using header split descriptors. (It is defined by the internal HPTR parameter in the RSC context - see Figure 7-44).

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=0) - The received packet header is posted as is to host memory. Note that if the received packet includes padding bytes, these bytes are discarded.

### 7.11.5.4 Large Receive Data

The data of a coalesced packet is posted to its buffer by the DMA engine as follows.

Ethernet CRC.

- When RSC is enabled on any queue, the global CRC strip must be set (HLREG0.RXCRCSTRP =1b).

Packet data spans on a single buffer.

- The data of the received packet spans on a single buffer if buffer has the required space.
- The DMA engine posts the packet data to its buffer pointed to by DATDESC descriptor at an offset indicated by the DATOFF.

Packet data spans on multiple buffers.

- The data of the received packet spans across multiple buffers when it is larger than a single buffer or larger than the residual size of the current buffer.
- When a new buffer is required (new descriptor) the DMA engine writes back to the completed descriptor linking it to the new one (Section 7.11.5.1 details the indicated descriptor fields).
- Decrement the RSCDESC parameter by one and update the DATDESC for each new opened descriptor.

DMA completion.

- Following DMA completion, set the DATOFF to the byte offset of the next packet.

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).

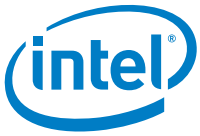
Not enough descriptors due to RSCDESC exhaust.

- If the received packet requires more descriptors than indicated by the internal RSCDESC parameter, then the 82599 completes the current large receive while the new packet starts a new large receive.

### 7.11.6 RSC Completion and Aging

This section summarizes all causes of large receive completion (the first three cases repeat previous sections).

- A packet of a new flow is received while there are no free RSC contexts. the 82599 completes (closes) the oldest large receive (opened first). The new packet starts a new large receive using the evicted context.
- The received packet cannot be added to the active large receive due to one of the following cases (indicated also in Section 7.11.4). In these cases the existing RSC completes and the received packet opens a new large receive.
  - The sequence number does not meet expected value.
  - The receive packet includes a time stamp TCP option header while there was no time stamp TCP option header in the first packet in the RSC.
  - There is not enough space in the RSC buffer(s) for the packet data. Meaning, the received packet requires a new buffer while the RSC already exhausted all permitted buffers defined by the RSCCTL[n].MAXDESC.
  - The received packet requires a new buffer while its descriptor wraps around the descriptor ring.
- When a packets is received while there are no more descriptors in the receive queue and the SRRCTL.Drop_En bit is set, the large receive completes and the new packet is discarded.
- EITR expiration while interrupt is enabled — RSC completion is synchronized with interrupt assertion to the host. It enables software to process the received frames since the last interrupt. See more details and EITR setting in Section 7.3.2.1.1.
- EITR expiration while interrupt is disabled — The ITR counter continues to count even when its interrupt is disabled. Every time the timer expires it triggers RSC completion on the associated Rx queues.
- LLI packet reception — All active RSC's on the same Rx queue complete and then the interrupt is asserted. Hardware then triggers RSC completion on all other queues associated with this interrupt.



- Low number of available descriptors — Whenever crossing the number of free Rx descriptors, the receive descriptor minimum threshold size defined in the SRRCTL[n] registers an LLI event is generated that affects RSC completion as well.
- Interrupt assertion by setting the EICS register has the same impact on packet reception as described in Section 7.3.1.2.1.
- Auto RSC Disable — When the interrupt logic triggers RSC completion it might also auto-disable further coalescing by clearing the RSCINT.RSCEN bit. Auto RSC disablement is controlled by the RSCINT.AUTORSC bit. In this mode, hardware also re-enables RSC (by setting the RSCINT.RSCEN bit back to 1b) when the interrupt is re-enabled by the EIMS (either by software or hardware as described in Section 7.3.1.3 and Section 7.3.1.5).

**Note:** In some cases packets that do not meet coalescing conditions might have active RSC of the same flow. As an example: received packets with ECE or CWR TCP flags. Such packets bypass completely the RSC logic (posted as single packets), and do not cause a completion of the active RSC. The active RSC would eventually be closed by either reception of a legitimate packet that is processed by the RSC logic but would not have the expected TCP sequence number. Or, an interrupt event closes all RSC's in its Rx queue. When software processes the packets, it gets them in order even though the RSC completes after the previous packet(s) that bypassed the RSC logic.

Any interrupt closes all RSC's on the associated receive queues. Therefore, when ITR is not enabled any receive packet causes an immediate interrupt and receive coalescing should not be enabled on the associated Rx queues.

# 7.12 IPsec Support

## 7.12.1 Overview

This section defines the hardware requirements for the IPsec offload ability included in the 82599. IPsec offload is the ability to handle (in hardware) a certain amount of the total number of IPsec flows, while the remaining are still handled by the operating system. It is the operating system's responsibility to submit to hardware the most loaded flows, in order to take maximum benefits of the IPsec offload in terms of CPU utilization savings. Establishing IPsec Security Associations between peers is outside the scope of this document, since it is handled by the operating system. In general, the requirements on the driver or on the operating system for enabling IPsec offload are not detailed here.

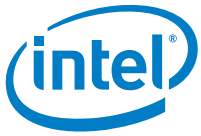
When an IPsec flow is handled in software, since the packet might be encrypted and the integrity check field already valid, and as IPv4 options might be present in the packet together with IPsec headers, the 82599 processes it like it does for any other unsupported Layer4 protocol, and without performing on it any layer4 offload.

Refer to section Section 4.6.12 for security offload enablement.

## 7.12.2 Hardware Features List

### 7.12.2.1 Main Features

- Offload IPsec for up to 1024 Security Associations (SA) for each of Tx and Rx.
  - On-chip storage for both Tx and Rx SA tables
  - Tx SA index is conveyed to hardware via Tx context descriptor
  - Deterministic Rx SA lookup according to a search key made of SPI, destination IP address, and IP version type (IPv6 or IPv4)
- IPsec protocols:
  - IP Authentication Header (AH) protocol for authentication
  - IP Encapsulating Security Payload (ESP) for authentication only
  - IP ESP for both authentication and encryption, only if using the same key for both
- Crypto engines:
  - For AH or ESP authentication only: AES-128-GMAC (128-bit key)
  - For ESP encryption and authentication: AES-128-GCM (128-bit key)
- IPsec encapsulation mode: transport mode, with tunnel mode only in receive

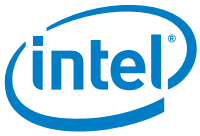


  - In Tx, packets are provided by software already encapsulated with a valid IPsec header, and
    - for AH with blank ICV inside
    - for ESP single send, with a valid ESP trailer and ESP ICV (blank ICV)
    - for ESP large send, without ESP trailer and without ESP ICV
  - In Rx, packets are provided to software encapsulated with their IPsec header and for ESP with the ESP trailer and ESP ICV,
    - where up to 255 bytes of incoming ESP padding is supported, for peers that prefer hiding the packet length
- IP versions:
  - IPv4 packets that do not include any IP option
  - IPv6 packets that do not include any extension header (other than AH/ESP extension header)
- Rx status reported to software via Rx descriptor:
  - Packet type: AH/ESP (in the SECSTAT field)
  - IPsec offload done (SA match), in the *IPSSA* field
- One Rx error reported to software via Rx descriptor in the following precedence order: no error, invalid IPsec protocol, packet length error, authentication failed (*SECERR* field)

### 7.12.2.2 Cross Features

- When IPSec offload is enabled, Ethernet CRC must be enabled as well by setting both TXCRCEN and RXCRCSTRP bits in the HLREG0 register
- Segmentation: full coexistence (TCP/UDP packets only)
  - increment IPsec Sequence Number (SN) and Initialization Vector (IV) on each additional segment
- Checksum offload: full coexistence (Tx and Rx)
  - IP header checksum
  - TCP/UDP checksum
- IP fragmentation: no IPsec offload done on IP fragments
- RSS: full coexistence, hash on the same fields used without IPsec (either 4-tuples or 2-tuples)
- LinkSec offload:
  - A device interface is operated in either LinkSec offload or IPsec offload mode, but not both of them altogether
  - If both IPsec and LinkSec encapsulations are required on the same packets, the 82599's interface is operated in LinkSec offload mode, while IPsec is performed by the operating system

- Virtualization:
  - Full coexistence in VMDq mode
  - in IOV mode, all IPsec registers are owned by the VMM/PF. For example, IPsec can be used for VMotion traffic.
  - No coexistence with VM-to-VM switch, IPsec packets handled in hardware are not looped back by the 82599 to another VM. Tx IPsec packets destined to a local VM must be handled in software and looped back via the software switch. However, an anti-spoofing check is performed on any IPsec packet.
- DCB: full coexistence
  - Priority flow control, with special care to respect timing considerations
  - Bandwidth allocation scheme enforced on IPsec packets since 802.1p field is always sent in clear text
  - CM-tagging takes place at Layer2 and then does not interfere with IPsec
- FCoE: no interaction as FCoE packets are not IP packets
- RSC: no coexistence
- Jumbo frames: When the SECTXCTRL.STORE_FORWARD bit is set (as required for IPSec offload), the maximum supported jumbo packet size is 9.5 KB (9728 bytes). This limitation is valid for all packets regardless if they are offloaded by hardware or carry IPSec encapsulation altogether.
- 802.1x: no interaction
- Teaming: no interaction
- TimeSync:
  - TimeSync IEEE 1588v1 UDP packets must not be encapsulated as IPsec packets
  - No interaction with TimeSync 1588v2 Layer2 packets
- Layer2 encapsulation modes:
  - IPsec offload is not supported for flows with SNAP header
  - IPsec offload coexists with double VLAN encapsulations
- Tunneled IPsec packets in receive: IPsec offload supported, but no other Layer4 offload performed
- NFS and any other Layer5 Rx filter: NFS or Layer5 packets encapsulated over ESP (whether IPsec is offloaded in hardware or not) and over a Layer4 protocol other than TCP are not parsed, nor recognized
- SCTP Rx offload: partial coexistence with SCTP CRC32 offload for IPSec-AH packets only.
- SCTP Tx offload: full coexistence with SCTP CRC32 offload for both IPSec-AH and IPSec-ESP packets.
- Manageability traffic: IPsec offload ability is controlled exclusively by the host, and thus manageability traffic could use IPsec offload only if it is coordinated/configured with/by the host. For IPsec flows handled by software:
  - If manageability and host entities share some IP address(es), then manageability should coordinate any use of IPsec protocol with the host. Note it should be true for previous devices that do not offer IPsec offload.



  - If manageability and host entities have totally separate IP addresses, then manageability can use IPsec protocol (as long as it is handled by the manageability controller software)
- Header split:
  - Supported for SAs handled in hardware, IP boundary split includes the IPsec header
  - For SAs handled in software, no header split done

## 7.12.3 Software/Hardware Demarcation

The followings items are not supported by hardware but might be supported by operating system/driver:

- Multicast SAs
- IPsec protocols:
  - Both AH and ESP protocols on the same SA or packet
  - ESP for encryption only
  - ESP for both authentication and encryption using different keys and/or different crypto engines
- Crypto engines:
  - AES-256, SHA-1, AES-128-CBC, or any other crypto algorithm
- Tx IPsec packets encapsulated in tunnel mode
- Extended Sequence Number (ESN)
- IP versions:
  - IPv4 packets that include IP option
  - IPv6 packets that include extension headers other than the AH/ESP extension headers
- Anti-replay check and discard of incoming replayed packets
- Discard of incoming dummy ESP packets (packets with protocol value 59)
- IPsec packets that are IP fragments
- ESP padding content check
- IPsec statistics
- IPsec for flows with SNAP header

**Note:** For SCTP and other Layer4 header types, or for tunneled packets, hardware does not care what is there when doing Rx IPsec processing. Everything after the IP/IPsec headers can be opaque to hardware (consider as IP payload). IPsec processing can be done on any packet that has a matching SA and appropriate IP options/extension headers. There is no expectation that hardware determines what is in the packet beyond the IP/IPsec headers before decrypting/authenticating the packet. The most important point is that hardware should not corrupt or drop incoming IPsec packets —

in any situation. When hardware decides and starts performing IPsec offload on a packet, it should pursue the offload until the packet's end — at the price of eventually not doing other Layer3/4 offloads on it. It is always acceptable for hardware not to start doing the IPsec offload on a matched SA, if it knows it is an unsupported encapsulation. For example, one of the three cases: IPv4 option, IPv6 extensions, or SNAP.

## 7.12.4 IPsec Formats Exchanged Between Hardware and Software

This section describes the IPsec packet encapsulation formats used between software and hardware by an IPsec packet concerned with the offload in either Tx or Rx direction.

In Rx direction, the IPsec packets are delivered by hardware to software encapsulated as they were received from the line, whether IPsec offload was done or not, and when it was done, whether authentication/decrypting has succeeded or failed.

### 7.12.4.1 Single Send

In Tx direction, single-send IPsec packets are delivered by software to hardware already encapsulated and formatted with their valid IPsec header and trailer contents, as they should be run over the wire — except for the *ICV* field that is filled with zeros, and the ESP payload destined to be encrypted that is provided in clear text before IPsec encryption.

### 7.12.4.2 Single Send with TCP/UDP Checksum Offload

For single-send ESP packets with TCP/UDP checksum offload, the checksum computing includes the TCP/UDP header and payload before hardware encryption occurred and without the ESP trailer and ESP ICV provided by software. Software provides the length of the ESP trailer plus ESP ICV in a dedicated field of the Tx context descriptor (IPS_ESP_LEN field) to signal hardware when to stop TCP/UDP checksum computing.

Software calculates a full checksum for the IP pseudo-header as in the usual case. The protocol value used in the IP pseudo-header must be the TCP/UDP protocol value and not the AH/ESP protocol value that appears in the IP header. This full checksum of the pseudo-header is placed in the packet data buffer at the appropriate offset for the checksum calculation.

The byte offset from the start of the DMA'ed data to the first byte to be included in the TCP/UDP checksum (the start of the TCP header) is computed as in the usual case: MACLEN+IPLEN. It assumes that IPLEN provided by software in the Tx context descriptor is the sum of the IP header length with the IPsec header length.

**Note:** For the IPv4 header checksum offload, hardware cannot rely on the *IPLEN* field provided by software in the Tx context descriptor, but should rely on the fact that no IPv4 options are present in the packet. Consequently, for IPsec offload packets, hardware always computes IP header checksum over a fixed amount of 20 bytes.

### 7.12.4.3 TSO TCP/UDP

In Tx direction, TSO IPsec packets are delivered by software to the 82599 already encapsulated and formatted with only their valid IPsec header contents — except for the *ICV* field included in AH packets headers that is filled with zeros, and to the ESP payload destined to be encrypted that is provided in clear text before any encryption. No ESP trailer or ESP ICV are appended to TSO by software. It means that hardware has to append the ESP trailer and ESP ICV on each segment by itself, and to update IP total length / IP payload length accordingly.

The next header of the ESP trailer to be appended by hardware is taken from TUCMD.L4T field of the Tx context descriptor.

By definition TSO requires on each segment that the IP total length / IP payload length be updated, and the IP header checksum and TCP/UDP checksum be re-computed. But for the TSO of IPsec packets, the *SN* and the *IV* fields must be increased by one in hardware on each new segment (after the first one) as well.

Software calculates a partial checksum for the IP pseudo-header as in the usual case. The protocol value used in the IP pseudo-header must be the TCP/UDP protocol value and not the AH/ESP protocol value that appears in the IP header. This partial checksum of the pseudo header is placed in the packet data buffer at the appropriate offset for the checksum calculation.

The byte offset from the start of the DMA'ed data to the first byte to be included in the TCP checksum (the start of the TCP/UDP header) is computed as in the usual case: MACLEN+IPLEN. It assumes that IPLEN provided by software in the Tx context descriptor is the sum of the IP header length with the IPsec header length.

For TSO ESP packets, the TCP/UDP checksum computing includes the TCP/UDP header and payload before hardware encryption occurred and without the ESP trailer and ESP ICV appended by hardware. The 82599 thus stops TCP/UDP checksum computing after the amount of bytes given by L4LEN + MSS. It is assumed that the MSS value placed by software in the Tx context descriptor specifies the maximum TCP/UDP payload segment sent per frame, not including any IPsec header or trailer — and not including the TCP/UDP header.

**Note:** For IPv4 header checksum computing, refer to the note in section Section 7.12.4.2.

Shaded fields in the tables that follow correspond to fields that need to be updated per each segment.

### Table 7-80 IPv4 TSO ESP Packet Provided by Software

| | 0 3 | 4 7 | 8 15 | 16 19 23 | 24 31 |
|---|---|---|---|---|---|
| 1 | Ver | Hlen | TOS | IP Total Length | |
| 2 | Identification | | | Flags | Fragment Offset |
| 3 | TTL | | Protocol = ESP | Header Checksum | |
| 4 | Source IPv4 Address | | | | |
| 5 | Destination IPv4 Address | | | | |
| 1 | Security Parameter Index (SPI) | | | | |
| 2 | Sequence Number (SN) | | | | |
| 3 | Initialization Vector (IV) | | | | |
| 4 | | | | | |
| 1 | TCP/UDP Header | | | | |
| | TCP/UDP Payload | | | | |

### Table 7-81 IPv6 TSO ESP Packet Provided by Software

| | 0 3 | 4 7 | 8 15 | 16 23 | 24 31 |
|---|---|---|---|---|---|
| 1 | Ver | Priority | Flow Label | | |
| 2 | IP Payload Length | | | Next Header = ESP | Hop Limit |
| 3 | Source IPv6 Address | | | | |
| 4 | | | | | |
| 5 | | | | | |
| 6 | | | | | |
| 7 | Destination IPv6 Address | | | | |
| 8 | | | | | |
| 9 | | | | | |
| 10 | | | | | |
| 1 | Security Parameter Index (SPI) | | | | |
| 2 | Sequence Number (SN) | | | | |
| 3 | Initialization Vector (IV) | | | | |
| 4 | | | | | |
| 1 | TCP/UDP Header | | | | |
| | TCP/UDP Payload | | | | |

## 7.12.5 TX SA Table

IPsec offload is supported only via advanced transmit descriptors. See Section 7.2.3.2.4 for details.

### 7.12.5.1 Tx SA Table Structure

The Tx SA table contains additional information required by the AES-128 crypto engine to authenticate and encrypt data. This information is not run over the wire together with the IPsec packets, but is exchanged between the IPsec peers' operating system during the SA establishment process. When the IKE software does a key computation it computes four extra bytes using a pseudo-random function (it generates 20 bytes instead of 16 bytes that it needs to use as a key) and the last four bytes are used as a salt value.

The SA table in Tx is a 1024 x 20-byte table loaded by software. Each line in the table contains the following fields:

**Table 7-82 TX SA Table**

| AES-128 KEY | AES-128 SALT |
|---|---|
| 16 bytes | 4 bytes |

Refer to Section 7.12.7 for a description of the way these fields are used by the AES-128 crypto engine.

Each time an unrecoverable memory error occurs when accessing the Tx SA tables, an interrupt is generated and the transmit path is stopped until the host resets the 82599. Packets that have already started to be transmitted on the wire are sent with a wrong CRC.

Upon reset, the 82599 clears the contents of the Tx SA table. Note that access to Tx SA table is not guaranteed for 10 µs after the reset command.

### 7.12.5.2 Access to Tx SA Table

#### 7.12.5.2.1 Write Access

1. Software writes the IPSTXKEY 0...3 and/or IPSTXSALT register(s).
2. Software writes the IPSTXIDX register with the SA_IDX field carrying the index of the SA entry to be written, and with the *Write* bit set (*Read* bit cleared).
3. Hardware issues a Write command into the SA table, copying the IPSTXKEY (16 bytes) and the IPSTXSALT (4 bytes) registers into the table entry pointed by the SA_IDX field configured in IPSTXIDX register. It then clears the *Write* bit in IPSTXIDX register.

4. Software starts/resumes sending IPsec offload packets with the *IPsec SA IDX* field in the Tx context descriptor pointing to valid/invalid SA entries. A valid SA entry contains updated key and salt fields currently in use by the IPsec peers.

#### 7.12.5.2.2 Read Access

1. Software writes the IPSTXIDX register with the index of the SA entry to be read, and with the *Read* bit set (*Write* bit cleared).
2. Hardware issues a Read command from the SA table, copying into the registers the IPSTXKEY (16 bytes) and the IPSTXSALT (4 bytes) values from the table entry pointed by the SA_IDX field configured in the IPSTXIDX register. It then clears the *Read* bit in IPSTXIDX register.
3. Software reads the IPSTXKEY 0...3 and/or IPSTXSALT register(s).

## 7.12.6 TX Hardware Flow

### 7.12.6.1 Single Send without TCP/UDP Checksum Offload

1. Extract IPsec offload request from the IPSEC bit of the POPTS field in the advanced Tx transmit data descriptor.
2. If IPsec offload is required for the packet (IPSEC bit was set), then extract the *SA_IDX*, *Encryption*, and *IPSEC_TYPE* fields from the Tx context descriptor associated to that flow.
3. Fetch the AES-128 KEY and Salt from the Tx SA entry indexed by SA_IDX and according to the *Encryption* and *IPSEC_TYPE* bits to determine which IPsec offload to perform.
4. For AH, zero the mutable fields.
5. Compute ICV and encryption data (if required for ESP) over the appropriate fields, according to the operating rules described in Section 7.12.7, and making use of the AES-128 KEY and Salt fields fetched in step 3.
6. Insert ICV at its appropriate location and replace the plaintext with the ciphertext (if required for ESP).

### 7.12.6.2 Single Send with TCP/UDP Checksum Offload

1. Extract the IPsec offload command from the *IPSEC* bit of the *POPTS* field in the advanced Tx transmit data descriptor.
2. If IPsec offload is required for the packet (*IPSEC* bit was set), then extract the *SA_IDX*, *Encryption*, *IPSEC_TYPE*, and *IPS_ESP_LEN* fields from the Tx context descriptor associated to that flow.

3. Fetch the AES-128 KEY and Salt from the Tx SA entry indexed by SA_IDX, and according to the *Encryption* and *IPSEC_TYPE* bits to determine which IPsec offload to perform.
4. Compute the byte offset from the start of the DMA'ed data to the first byte to be included in the checksum (the start of the TCP header) as specified in Section 7.12.4.2.
5. Compute TCP/UDP checksum until either the last byte of the DMA data or for ESP packets, up to IPS_ESP_LEN bytes before it. As in the usual case, *implicitly* pad out the data by one zeroed byte if its length is an odd number.
6. Sum the full checksum of the IP pseudo header placed by software at its appropriate location with the TCP/UDP checksum computed in step 5. Overwrite the checksum location with the 1's complement of the sum.
7. For AH, zero the mutable fields.
8. Compute ICV and encrypt data (if required for ESP) over the appropriate fields, according to the operating rules described in Section 7.12.7, and making use of the AES-128 KEY and Salt fields fetched in step 3.
9. Insert ICV at its appropriate location and replace the plaintext with the ciphertext (if required for ESP).

### 7.12.6.3 TSO TCP/UDP

1. Extract the IPsec offload command from the *IPSEC* bit of the *POPTS* field in the advanced Tx transmit data descriptor.
2. If IPsec offload is required for the packet (*IPSEC* bit was set), then extract the *SA_IDX*, *Encryption*, and *IPSEC_TYPE* fields from the Tx context descriptor associated to that flow.
3. Fetch the AES-128 KEY and Salt from the Tx SA entry indexed by SA_IDX, and according to the *Encryption* and *IPSEC_TYPE* bits to determine which IPsec offload to perform.
4. Fetch the packet header from system memory, up to IPLEN+L4LEN bytes from the start of the DMA'ed data.
5. Overwrite the TCP SN with the stored accumulated TCP SN (if it is not the first segment).
6. Fetch (next) MSS bytes (or the remaining bytes up to PAYLEN for the last segment) from system memory and from the segment formed by packet header and data bytes, while storing the accumulated TCP SN.
7. Compute the byte offset from the start of the DMA'ed data to the first byte to be included in the checksum (the start of the TCP header) as specified in Section 7.12.4.3.
8. Compute TCP/UDP checksum until the last byte of the DMA data. As in the usual case, *implicitly* pad out the data by one zeroed byte if its length is an odd number.

9. For both IPv4 and IPv6, hardware needs to factor in the TCP/UDP length (typically L4LEN+MSS) to the software-supplied pseudo header partial checksum. It then sums to obtain a full checksum of the IP pseudo header with the TCP/UDP checksum computed in step 7. Overwrite the TCP/UDP checksum location with the 1's complement of the sum.
10. Increment by one the *AH/ESP SN* and *IV* fields on every segment (excepted to the first segment), and store the updated *SN* and *IV* fields with other temporary statuses stored for that TSO (one
TSO set of statuses per Tx queue).
11. For ESP, append the ESP trailer: 0-3 padding bytes, padding length, and next header = TCP/UDP protocol value, in a way to get the 4-byte alignment as described in Section 7.12.4.3.
12. Compute the IP total length / IP payload length and compute IPv4 header checksum as described in the note of Section 7.12.4.1. Place the results in their appropriate location.
13. For AH, zero the mutable fields.
14. Compute ICV and encryption data (if required for ESP) over the appropriate fields, according to the operating rules described in Section 7.12.7, and making use of the AES-128 KEY and Salt field fetched in step 3.
15. Insert ICV at its appropriate location and replace the plaintext with the ciphertext (if required for ESP).
16. Go back to step 4 to process the next segment (if necessary).

## 7.12.7 AES-128 Operation in Tx

The AES-128-GCM crypto engine used for IPsec is the same AES-128-GCM crypto engine used for LinkSec. It is referred throughout the document as an AES-128 black box, with 4-bit string inputs and 2-bit string outputs, as shown in Figure 7-45. Refer to the GCM specification for the internal details of the engine. The difference between IPsec and LinkSec, and between the different IPsec modes reside in the set of inputs presented to the box.

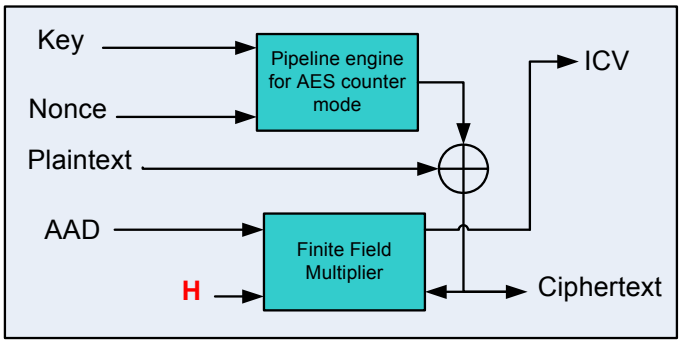



**Figure 7-45 AES-128 Crypto Engine Box**

- Key — 128-bits AES-128 KEY field (secret key) stored for that IPsec flow in the Tx SA table:

  ```
  Key = AES-128 KEY
  ```

- Nonce — 96-bits initialization vector used by the AES-128 engine, which is distinct for each invocation of the encryption operation for a fixed key. It is formed by the AES-128 Salt field stored for that IPsec flow in the Tx SA table, appended with the Initialization Vector (IV) field included in the IPsec packet:

  ```
  Nonce = [AES-128 SALT, IV]
  ```

  The nonce, also confusingly referred as IV in the GCM specification, is broken into two pieces — a fixed random part salt and increasing counter part IV, so the salt value goes with the packet as the fixed part. The purpose behind using the salt value is to prevent offline dictionary-type attacks in hashing case, to prevent predictable patterns in the hash.

- AAD — Additional Authentication Data input, which is authenticated data that must be left un-encrypted.
- Plaintext — Data to be both authenticated and encrypted.
- Ciphertext — Encrypted data, whose length is exactly that of the plaintext.
- ICV — 128-bit Integrity Check Value (referred also as authentication tag).
- H — is internally derived from the key.

**Note:** The square brackets in the formulas is used as a notation for concatenated fields.

### 7.12.7.1 AES-128-GCM for ESP — Both Authenticate and Encryption

```
AAD = [SPI, SN]
Plaintext = [TCP/UDP header, TCP/UDP payload, ESP trailer]
```

**Note:** Unlike other IPsec modes, in this mode, the *IV* field is used only in the nonce, and it is not included in either the plaintext or the AAD inputs.

ESP trailer does not include the *ICV* field.

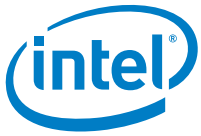
### 7.12.7.2 AES-128-GMAC for ESP — Authenticate Only

```
AAD = [SPI, SN, IV, TCP/UDP header, TCP/UDP payload, ESP trailer]
Plaintext = [] = empty string, no plaintext input in this mode
```

**Note:** ESP trailer does not include the *ICV* field.

### 7.12.7.3 AES-128-GMAC for AH — Authenticate Only

```
AAD = [IP header, AH header, TCP/UDP header, TCP/UDP payload]
Plaintext = []= empty string, no plaintext input in this mode
```



**Note:** Both IP header and AH header contain mutable fields that must be zeroed prior to be entered into the engine. Among other fields, the AH header includes *SPI*, *SN*, and *IV* fields.

## 7.12.8 RX Descriptors

IPsec offload is supported only via advanced receive descriptors. See Section 7.1.6 for details.

## 7.12.9 Rx SA Tables

### 7.12.9.1 Rx SA Tables Structure

The Rx SA tables contain additional information required by the AES-128 crypto engine to authenticate and decrypt the data. This information is not run over the wire together with the IPsec packets, but is exchanged between the IPsec peers' operating system during the SA establishment process. When the IKE software does a key computation it computes four extra bytes using a pseudo-random function (it generates 20 bytes instead of 16 bytes that it needs to use as a key) and the last four bytes are used as a salt value.

SPI is allocated by the receiving operating system in a unique manner. However, in a virtualized context, guest operating systems can allocate SPI values that collide with the SPI values allocated by the VMM/PF. Consequently, the SPI search must be completed by comparing the destination IP address with the IP addresses of the VMM/PF, which are stored in a separate table. Guest operating systems could thus use the proposed IPsec offload as long as their SAs are configured via the VMM/PF. It is assumed that refreshing the SAs would be done once every several minutes, and would thus not overload the VMM/PF.

There are three Rx SA tables in the 82599:

- IP address table — 128 entries
- SPI table — 1K entries
- KEY table — 1K entries

They are loaded by software via indirectly addressed CSRs, as described in Section 7.12.9.2.

**Table 7-83 IP Address Table**

| IP Address |
|---|
| 16 bytes |

### Table 7-84 SPI Table

| SPI | IP Index<br>(points to IP address table) |
|---|---|
| 4 bytes | 1 bytes |

### Table 7-85 KEY Table

| IPsec Mode | AES-128 KEY | AES-128 SALT |
|---|---|---|
| 1 byte | 16 bytes | 4 bytes |

The *IPsec Mode* field contains the following bits:

- VALID
- IPv6
- PROTO
- DECRYPT

It is assumed that the SPI and IP address tables are implemented internally in CAM cells, while the KEY table uses RAM cells. When an incoming IPsec packet (which does not includes option in IPv4 or another extension header in IPv6) is detected, hardware first looks up for the destination IP address to match one of the IP addresses stored in the IP address table. If there is a match, the index of that IP Addr. match is used together with the *SPI* field extracted from the packet for a second lookup into the SPI table. If there is again a match, then the index of that SPI+IP Index match is used to retrieve the SA parameters from the KEY table. The packet is finally considered to get an SA match only after inspecting the corresponding entry in the KEY table, as long as all the following conditions are met:

- *Valid* bit is set
- *IPv6* bit match with the IP version (IPv6/IPv4) of the incoming IPsec packet
- *Proto* bit match with the AH/ESP type of the incoming IPsec packet

Each time an unrecoverable memory ECC error occurs when accessing one of the Rx SA tables, an interrupt is generated and the receive path is stopped until the host resets the 82599.

Upon reset, the 82599 clears the contents of the Rx KEY table and software is required to invalidate the entire IP address and SPI CAM tables by clearing their contents. Access to Rx SA tables is not guaranteed for 10 μs after the Reset command.

## 7.12.9.2 Access to Rx SA Tables

### 7.12.9.2.1 Write Access

1. Software writes the IP address table via the IPSRXIPADDR 0...3 registers
2. Software writes the IPSRXIDX register with the following:
   a. *Table* bits combination corresponding to the Rx SA table to be written (such as 01b for IP address table)
   b. TB_IDX field pointing to the index to be written within the table
   c. *Write* bit set (*Read* bit cleared)
3. Hardware issues a Write command into the Rx SA table pointed by the Table bits combination, copying the concerned register(s) into the entry pointed by the *TB_IDX* field configured in the IPSRXIDX register. It then clears the *Write* bit in IPSRXIDX register.
4. Software performs steps 1 to 3 twice, first for writing the SPI table via IPSRXSPI, IPSRXIPIDX registers, and second for writing the KEY table via IPSRXKEY 0...3, IPSRXSALT, and IPSRXMOD registers.

Each time an entry in the IP address or SPI table is not valid/in-use anymore, software is required to invalidate its content by clearing it. For the IP table, an entry must be invalidated by software each time there is no more SPI entry that points to it; while for the SPI table, software must invalidate any entry as soon as it is not valid/not used anymore.

### 7.12.9.2.2 Read Access

1. Software writes the IPSRXIDX register with the *Table* and *TB_IDX* fields corresponding to the Rx SA table and entry to be read, and with the *Read* bit set (*Write* bit cleared).
2. Hardware issues a Read command from the Rx SA table and entry pointed by *Table* bits combination and *TB_IDX* field, copying each field into its corresponding register. It then clears the *Read* bit in IPSRXIDX register.
3. Software reads the corresponding register(s).

***Caution:*** There is an internal limitation in that only one single Rx SA table can be read accessed by software at a time. Hence, it is recommended that the entire read process, from steps 1 to 3, be repeated successively for each Rx SA table separately.

## 7.12.10 RX Hardware Flow without TCP/UDP Checksum Offload

1. Detect an IPsec header not encapsulated over a SNAP header is present without any IPv4 option or other IPv6 extension header encapsulated before it, and determine its type AH/ESP.
2. If such an IPsec header is present (as announced by the IP protocol field for IPv4 or by the next header for IPv6), then extract the SPI, destination IP address, and IP version (IPv4 or IPv6), and use these fields for the lookups into the Rx SA tables as described in Section 7.12.9.1. Also report the IPsec protocol found in the *Security* bits of the *Extended Status* field in the advanced Rx descriptor.
3. If there is a SA match for that packet, fetch the IPsec Rx mode from the SA entry, and according to the *Proto* and *Decrypt* bits determine which IPsec offload to perform. Also, set the *IPSSA* bit of the *Extended Status* field in the advanced Rx descriptor. If there was no SA match, then clear the *IPSSA* bit, report no error in *Security* error bits of the *Extended Errors* field in the advanced Rx descriptor, and stop processing the packet for IPsec.
4. If the *Proto* field recorded in the Rx SA table does not match the *IP Protocol* field (next header for IPv6) seen in the packet, then report it via the *Security* error bits of the *Extended Errors* field in the advanced Rx descriptor, and stop processing the packet for IPsec.
5. Fetch the AES-128 KEY and Salt from the matched Rx SA entry.
6. For AH, zero the mutable fields.
7. Make sure the AH/ESP header is not truncated, and for ESP make sure whether or not the packet is 4-bytes aligned. If not, report it via the *Security* error bits of the *Extended Errors* field in the advanced Rx descriptor, but processing of the packet for IPsec might be completed (if it has already started). A truncated IPsec packet is a valid Ethernet frame (at least 64 bytes) shorter than:
   a. ESP – at least 40 bytes following the IP header (16 [ESP header] + 4 [min. padding, pad_len, NH] + 16 [ICV] + 4 [CRC])
   b. AH over IPv4 – at least 40 bytes following the IP header (20 [AH header] + 16 [ICV] + 4 [CRC])
   c. AH over IPv6 – at least 44 bytes following the IP header (20 [AH header] + 4 [ICV padding] + 16 [ICV] + 4 [CRC])
8. Compute ICV and decrypt data (if required for ESP) over the appropriate fields, according to the operating rules described in Section 7.12.12, and making use of the AES-128 KEY and SALT fields fetched in step 5.
9. Compare the computed ICV with the *ICV* field included in the packet at its appropriate location, and report the comparison status match/fail via the *Security* error bits of the Extended Errors field in the advanced Rx descriptor.

## 7.12.11 RX Hardware Flow with TCP/UDP Checksum Offload

Perform the RX hardware flow described in Section 7.12.10 and add the following steps:

10. Start computing the checksum from the TCP/UDP header's beginning — found according to the Rx parser logic updated for IPsec formats. Do not perform Layer4 offloads if unsupported IPsec encapsulation is detected. For example, tunneled IPsec, IPv4 options or IPv6 extensions after the IPsec header.
11. For ESP, stop checksum computing before the beginning of the ESP trailer — found from the end of packet according to the padding length field content. As in the usual case, *implicitly* pad out the data by one zeroed byte if its length is an odd number.
12. Store the next header extracted from the AH header/ESP trailer into the *Packet Type* field of the advanced Rx descriptor, but use the TCP/UDP protocol value in the IP pseudo header used for the TCP/UDP checksum. Also compute the TCP/UDP packet length to be inserted in the IP pseudo header (excluding any IPsec header or trailer).
13. Compare the computed checksum value with the TCP/UDP checksum included in the packet. Report the comparison status in the Extended Errors field of the advanced Rx descriptor.

## 7.12.12 AES-128 Operation in Rx

The AES-128 operation in Rx is similar to the operation in Tx, while for decryption, the encrypted payload is fed into the plaintext input, and the resulted ciphertext stands for the decrypted payload. Refer to Section 7.12.7 for the proper inputs to use in every IPsec mode.

### 7.12.12.1 Handling IPsec Packets in Rx

The following table lists how IPsec packets are handled according to some of their characteristics.

**Table 7-86 Summary of IPsec Packets Handling in Rx**

| IP Fragment | IPv4 Option or IPv6 Extensions or SNAP | IP Version | SA Match | IPsec Offload in Hardware | Layer4/3 Offload in Hardware | Header Split | AH/ESP Reported in Rx Desc. |
|---|---|---|---|---|---|---|---|
| Yes | Yes | v4 | Don't care | No | IP checksum only | Up to IPsec header included | Yes |
| Yes | Yes | v6 | Don't care | No | No | Up to IP fragment extension included | No |
| Yes | No | v4 | Don't care | No | IP checksum only | Up to IPsec header included | Yes |
| No | Yes | v4 | Don't care | No | IP checksum only | No | Yes |

## Table 7-86 Summary of IPsec Packets Handling in Rx (Continued)

| IP Fragment | IPv4 Option or IPv6 Extensions or SNAP | IP Version | SA Match | IPsec Offload in Hardware | Layer4/3 Offload in Hardware | Header Split | AH/ESP Reported in Rx Desc. |
|---|---|---|---|---|---|---|---|
| No | Yes | v6 | Don't care | No | No | Up to first unknown or IPsec extension header, excluded | No[1] |
| No | No | v4 | Yes | Yes | Yes[2] | Yes[3] | Yes |
| No | No | v4 | No | No | IP checksum only | No | Yes |
| No | No | v6 | Yes | Yes | Yes[4] | Yes[3] | Yes |
| No | No | v6 | No | No | No | No | Yes |

1. Exception to SNAP IPsec packets that are reported as AH/ESP in Rx descriptor.
2. No Layer4 offload done on packets with IPSec error.
3. According to definition made in PSRTYPE[n] registers.
4. No Layer4 offload done on packets with IPSec error.

# 7.13 Fibre Channel over Ethernet (FCoE)

## 7.13.1 Introduction

Fibre Channel (FC) is the predominant protocol used in Storage Area Networks (SAN). Fibre Channel over Ethernet (FCoE) is used to connect an Ethernet storage initiator and legacy FC storage targets.

The FC protocol is based on high reliability of the communication link between the initiator and the storage target. It assumes an extremely low error rate of $10^{-12}$ and no packet drop. DCB extends Ethernet through class-based flow control in such a way that FC-like no-drop is guaranteed as required by FC. Doing so, FC protocol can be transposed to an Ethernet link by Layer 2 encapsulation that is defined by the FCoE protocol. Figure 7-46 shows a connection between an FCoE initiator and legacy FC targets.

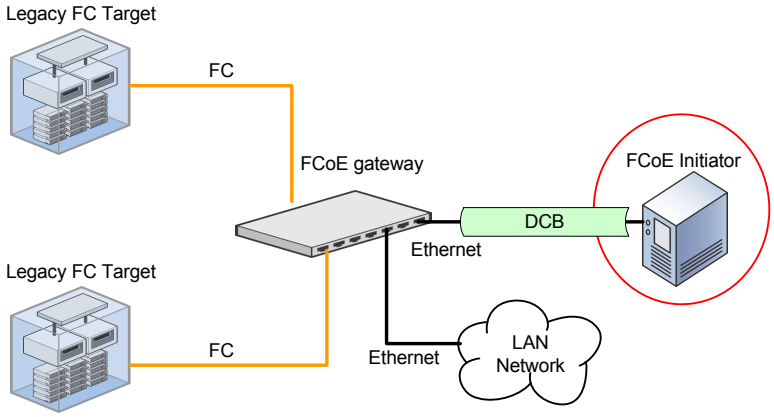



**Figure 7-46 Connecting an FCoE Initiator to FC Targets**

Existing FC HBAs used to connect between an FC initiator and FC targets provide full offload of the FC protocol to the initiator to maximize storage performance. In order to compete with this market, the 82599 offloads the main data path of I/O Read and Write commands to the storage target.

### 7.13.1.1 FC Terminology

Useful background on FC framing and its Ethernet encapsulation can be found in Section A.5. More comprehensive material can be found in the FIBRE CHANNEL FRAMING AND SIGNALING-2 (FC-FS-2) specification. Following are some of the most common terms used extensively in the sections that describe the FCoE functionality.

FC Exchange - Complete FC read or FC write flow. It starts with a read or write request by the initiator (the host system) until it receives a completion indication from the target (the remote disk).

FC Sequence - An FC exchange is composed of multiple FC sequences. An FC sequence can be single or multiple frames that are sent by the initiator or the target. Also, each FC sequence has a unique sequence ID.

FC Frame - FC frames are the smallest units sent between the initiator and the target. The FC-FS-2 specification defines the maximum frame size as 2112 bytes. Each FC frame includes an FC header and optional FC payload. It also may include extended headers and FC optional headers. Extended headers other than Virtual Fabric Tagging (VFT) are not expected in an FCoE network and FC optional headers are not used in most cases as well.

Data Frame - FC frames that carry read or write data.

FCP_RSP Frame - FC control frames are sent from the target to the initiator, which defines the completion of an FC read or write exchange.

## 7.13.2 FCoE Transmit Operation

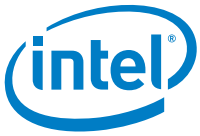
Transmit FCoE offload is enabled by setting the TUCMD.FCoE bit in the transmit context descriptor. The 82599 supports the following offload capabilities: FC CRC calculation and insertion, FC padding insertion and FC segmentation. These capabilities are described in the following sections.

### 7.13.2.1 FCoE Transmit Cross Functionality

After setting the TUCMD.FCoE bit, hardware digests the packet's content before it is sent to the wire. In this case, software must enable hardware offload for additional tasks as follows:

**Table 7-87 FCoE Transmit Cross Functionality**

| Cross Function | Requirements |
|---|---|
| Ethernet CRC insertion | Software must enable Ethernet CRC insertion by setting the *IFCS* bit in the transmit data descriptor. The Ethernet CRC covers the entire packet. Enabling FCoE offloading, hardware modifies the packet content and must also adjust the Ethernet CRC. |
| LinkSec offload | LinkSec encapsulation covers the entire Ethernet packet payload (it includes both FCoE content and Ethernet padding). When packets carry LinkSec encapsulation on the wire, LinkSec offload by hardware should be activated. |
| VLAN header | It is assumed that any FCoE has a VLAN header. In the case of double VLAN mode, the packet must have the two VLAN headers. |
| SNAP packet | The 82599 does not provide FCoE offload for FCoE frame over SNAP. |
| Traffic rate control | FC traffic relies on a high quality link that guarantees no packet loss. It is expected that any lost traffic protocols supported by the network are enabled by the 82599 as well. |
| FC and PFC | |
| Virtualization | It is expected that the VMM abstract the FCoE functionality to the VM(s). FCoE setting and FCoE traffic is expected only by the VMM accessing the LAN via the PF. |
| TCP/IP and UDP/IP offload | FCoE traffic is L2 traffic (not over IP). Any setting of TCP/IP and UDP/IP offload capabilities are not applicable and do not impact FCoE offload functions. |
| Transmit descriptors | Software must use the advanced transmit descriptor to activate either FC CRC offload or TSO functionality. |



### 7.13.2.2 FC Padding Insertion

FC frames always consist of a whole number of four bytes. If user data is not composed of a whole number of four bytes, then the FC frames contain padding bytes with a zero value. The length of the padding bytes can be any number between zero to three so together with the user data, the length of the FC frames has a whole number of four bytes. The length of the padding bytes is indicated by software in the *Fill Bytes* field in the FC header. This field is used by the receiving end node (target) to extract these bytes. Hardware does not use this field to identify the required length of the padding bytes. Instead, it checks the transmit buffer size indicated by the *PAYLEN* field in the transmit data descriptor. The length of the padding bytes added by hardware equals:

2's complement {two LS bits of (PAYLEN minus MACLEN)}. While PAYLEN is defined in the Tx data descriptor and MACLEN is defined in the Tx context descriptor.

The 82599 auto-pads the frame with the required zero bytes when FCoE offload is enabled (TUCMD.FCoE bit is set). In TSO, padding bytes are added only on the last frame since the MSS must be a whole number of four bytes.

### 7.13.2.3 SOF Placement

During a single send, the *SOF* field is taken as is from the FCoE header in the data buffer.

### 7.13.2.4 EOF Insertion

The 82599 automatically inserts the *End of Frame* field when the TUCMD.FCoE bit in the transmit context descriptor is set. The EOF codes that are inserted into the transmitted packets are stored in the TEOFF register. The TEOFF register contains four EOF codes named EOF0...EOF3 that are supported by the transmit FCoE offload. By default, these values are programmed into the following values: EOF0 = EOFn; EOF1 = EOFt; EOF2 = EOFni; EOF3 = EOFa. The *EOF* flag in the *FCoEF* field in the transmit context descriptor define an index value as listed in the Table 7-88.

**Table 7-88 EOF Codes in Single Send**

| **EOF Bits in the Context Descriptor (ORIE bit in the Context Descriptor must be set to 1b)** | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| **Inserted EOF Code** | EOF0 (EOFn) | EOF1 (EOFt) | EOF2 (EOFni) | EOF3 (EOFa) |

### 7.13.2.5 FC CRC Insertion

FC CRC calculation is one of the most CPU intensive tasks in large transactions. The 82599 offloads the FC CRC calculation when the *FCoE* bit is set in the *TUCMD* field within the transmit context descriptor. The 82599 calculates and adds the FC CRC before packet transmission but after the required FC padding bytes are already added.

The CRC polynomial used by the FC protocol is the same one as used in FDDI and Ethernet as shown in the following equation. While CRC bytes are transmitted in big endian byte ordering (MS byte first on the wire):

$$X_{32}+X_{26}+X_{23}+X_{22}+X_{16}+X_{12}+X_{11}+X_{10}+X_8+X_7+X_5+X_4+X_2+X+1.$$

The size of FCoE payload on which FC CRC is calculated is indicated in the context and data descriptors as follows. Figure 7-47 specifies the FCoE frame and the relevant parameters to CRC calculation.

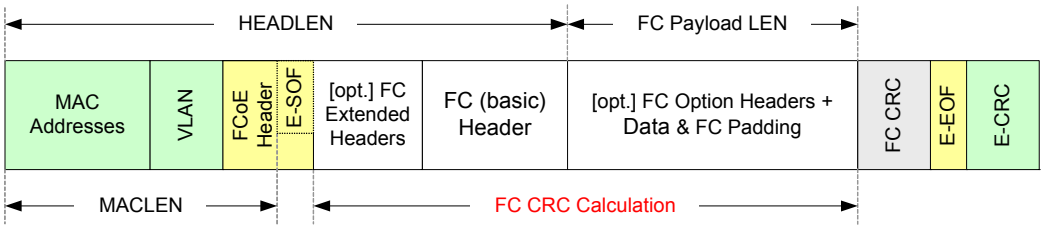



**Figure 7-47 FCoE Frame and Relevant Transmit Descriptor Parameters**

FC CRC Calculation Beginning

FC CRC calculation starts after the FCoE header. It equals to byte offset of MACLEN + 4, while the *MACLEN* field in the transmit context descriptor is the byte offset of the last Dword in the FCoE header that contains the SOF flag.

FC CRC Calculation End

FC CRC calculation ends at the end of the FC Payload LEN shown in Figure 7-47 (eight bytes before the Ethernet CRC).

## 7.13.2.6 Host Data Buffers Content for a Single Packet Send

The Table 7-89 lists the data prepared by software when transmit FCoE offload is enabled (the *FCoE* bit in the *TUCMD* field is set in the transmit context descriptor).

**Table 7-89 Transmit FCoE Packet Data Provided by Software (for TUCMD.FCoE = 1)**

| Ethernet MAC Addresses | VLAN Header | FCoE Header | FC Frame (provided by software) | | |
|---|---|---|---|---|---|
| | | | FC Header | FC Option Header(s) | Opt. Data |

Listed below are fields in the transmitted FCoE frame that are not included in the data buffers (in host memory) as shown in Figure 7-89.

VLAN Header — The VLAN header could be part of the data buffer or in the transmit descriptor depending on *VLE* bit in the *CMD* field in the transmit descriptor.

EOF — The EOF is defined by the *EOF* fields and *ORIE* bit in the context descriptor (more details in Section 7.2.3.2.3)

FC-CRC — The 82599 calculates and inserts the FC CRC bytes.

FC-Padding — The 82599 calculates the padding length and inserts these bytes as required (all zeros).

Ethernet CRC — Insertion should be enabled by the *IFCS* bit in transmit data descriptor.

LinkSec Header and Digest — When the link is secured by LinkSec, then LinkSec offload must be enabled and the LinkSec encapsulation is added by hardware.

## 7.13.2.7 FCoE Transmit Segmentation Offload (TSO)

FCoE segmentation enables the FCoE software to initiate a transmission of multiple FCoE packets up to a complete FC sequence with a single header in host memory (single instruction). It is activated by using the advanced Tx context descriptor (DTYP equals 0010b) and setting both the TUCMD.FCoE in the context descriptor and setting the DCMD.TSE bit in the transmit data descriptor. The 82599 splits the transmitted content to multiple packets as defined by the *MSS* field in the Tx context descriptor.

TSO Parameters

- The frame header includes the Ethernet MAC addresses, VLAN Tag, FCoE header and the FC header. The header size is defined in the context descriptor by the HEADLEN and MACLEN as illustrated.
- The *SOF* and *EOF* fields are defined by the SOF, ORIS, EOF and ORIE fields in the context descriptor as described in Section 7.13.2.3 and Section 7.13.2.4.
- MSS – the maximum segment size in the context descriptor that define the FC data (payload) size on each packet other than the last frame which can be smaller.

### 7.13.2.7.1 Host Data Buffers Content for TSO Offload

The Figure 7-48 shows the data in host memory when FCoE TSO is activated. The TSO header is repeated on all frames of the TSO. The header includes static and dynamic fields that are modified by hardware from packet-to-packet. The payload size is reflected in all frames.

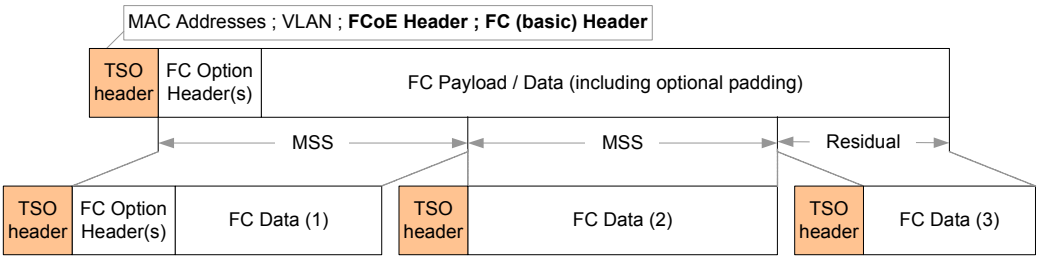



**Figure 7-48 FCoE TSO Provided by the FCoE Driver**

FCoE Header

The FCoE packet header must not span more than two buffers. For best bus use it is recommended that the header be located in a single buffer (the first one).

- Ethernet MAC addresses are the source and destination Ethernet MAC addresses

- VLAN tag can be provided by the driver as part of the packet header or as part of the data descriptor.
- FCoE header (shown in Figure 7-48) includes the FCoE Ethernet type, FCoE Version and SOF flag. Software should leave the *SOF* fields as zero while hardware inserts it according to the *SOF* and *ORIS* bits in the Tx context descriptor.
- FC (basic) header.

FCoE TSO — Payload

- FC option headers.
- FC data to be segmented
- The payload may or may not include the optional FC padding bytes. Hardware adds any required padding bytes not included in the data buffers according to the PAYLEN field in the data descriptor.

Modified fields between consecutive frames within TSO are described in the following sections.

#### 7.13.2.7.2 Dynamic Start of Frame in TSO

During TSO the *SOF* field in the data buffer is replaced by hardware according to the values of the *SOF* and *ORIS* bits in the transmit context descriptor. In this case the value of the *SOF* field in the data buffer is ignored (for future expansion software should set it to zero). The SOF codes that are inserted to the transmitted packets are stored in the TSOFF register. The TSOFF register contains four SOF codes named as SOF0...SOF3 that are supported by the transmit FCoE offload. By default these values are programmed to the following values: SOF0 = SOFi2; SOF1 = SOFi3; SOF2 = SOFn2; SOF3 = SOFn3. The SOF flag and *Orientation Start* (ORIS) bit in the *FCoEF* field in the transmit context descriptor define an index value. This index is used to extract the SOF code that is inserted to the packet as listed in the Table 7-90. The *ORIS* bit defines if the TSO starts an FC sequence or if the first frame on the FC sequence is already sent.

**Table 7-90 SOF Codes in TSO**

| ***SOF* Bit in the Context Descriptor** | ***ORIS* Bit in the Context Descriptor** | **SOF Code in the First Frame** | **SOF Code in Other Frames** | **SOF Code while TSO = Single Frame** |
|---|---|---|---|---|
| 1 (Class 3) | 1 (sequence start) | SOF1 (SOFi3) | SOF3 (SOFn3) | SOF1 (SOFi3) |
| 1 (Class 3) | 0 (not a sequence start) | SOF3 (SOFn3) | SOF3 (SOFn3) | SOF3 (SOFn3) |
| 0 (Class 2) | 1 (sequence start) | SOF0 (SOFi2) | SOF2 (SOFn2) | SOF0 (SOFi2) |
| 0 (Class 2) | 0 (not a sequence) | SOF2 (SOFn2) | SOF2 (SOFn2) | SOF2 (SOFn2) |

#### 7.13.2.7.3 Dynamic FC Header fields in TSO

F_CTL — Table 7-91 lists those fields in the F_CTL that are modified between consecutive frames of a TSO. If a TSO is transmitted by a single packet all F_CTL fields are taken from the data buffer (as if it is the last frame in the TSO).

### Table 7-91 F_CTL Codes in TSO

| F_CTL Bits | last frame in TSO when the *ORIE* bit in the Tx context descriptor is set. | Any other frame |
|---|---|---|
| Fill Bytes (1:0) | Taken from the F_CTL(1:0) in the data buffer. It defines the length of the FC padding required to make the FC data a complete multiply of four bytes. | 00b |
| Continue Sequence Condition (7:6) | Taken from the F_CTL(7:6) in the data buffer. The continue sequence condition is meaningful only if F_CTL(19) is set and F_CTL(16) is cleared. | 00b |
| Sequence Initiative (16) | Taken from the F_CTL(16) in the data buffer. The sequence initiative is meaningful only if F_CTL(19) is also set. | 0b |
| End Sequence (19) | Taken from the F_CTL(19) in the data buffer. The end sequence should be set to 1b by software only if the frame is the last one of a sequence. | 0b |

DF_CTL

Table 7-92 lists those fields in the DF_CTL that can be modified between consecutive frames of a TSO. Note that the ESP Header presence bit is not listed in this table. When ESP Hea*der* is present, software must not use a TSO that spans across multiple packets. If a TSO is transmitted by a single packet all DF_CTL fields are taken from the data buffer (as if it is the first frame in the TSO).

### Table 7-92 DF_CTL Codes in TSO

| DF_CTL Fields | 1st frame in TSO when *ORIS* bit in the Tx context descriptor is set. | Any other frame |
|---|---|---|
| Device Header Indication (1:0) | Taken from the data buffer. | 00b |
| Association Header Indication (4) | Taken from the data buffer. | 0b |
| Network Header Indication (5) | Taken from the data buffer. | 0b |

SEQ_CNT

SEQ_CNT in the first frame is taken from the SEQ_CNT field in the FC header in the data buffers. On any other frame, the value of SEQ_CNT is incremented by one from its value in the previous frame. The SEQ_CNT wrap-to-zero after reaching a value of 65,535.

PARAM

The PARAM field in the first frame is taken from the *PARAM* field in the FC header in the data buffers. If the FCoEF.PARINC bit is set in the transmit context descriptor, the value of the PARAM becomes dynamic. In that case, the *PARAM* is incremented by hardware by the MSS value on each frame. Software should set the FCoEF.PARINC bit when the *PARAM* field indicates the data offset (*Relative Offset Present* bit in the F_CTL field is set).

#### 7.13.2.7.4 Dynamic End Of Frame Fields

FC_CRC — Calculated and inserted on each frame as described in section Section 7.13.2.5.

FC_Padding — Calculated the number of required padding bytes and inserted them on the last frame as described in section Section 7.13.2.2.

EOF — As explained for a single send, the EOF flag is appended to the transmitted packets while values are taken from the TEOFF register. The FCoE flag index to the TEOFF register is defined by the *EOF* flag and *Orientation End (ORIE*) bit in the *FCoEF* field in the transmit context descriptor as listed in Table 7-93.

**Table 7-93 EOF Codes in TSO**

| EOF Bits in the Context Descriptor | ORIE Bit in the Context Descriptor | Last Frame of the TSO or TSO in Single Frame | Other Frames of the TSO |
|---|---|---|---|
| 00 (EOFn) | 0 (not a sequence end) | EOF0 (EOFn) | EOF0 (EOFn) |
| 00 (EOFn) | 1 (sequence end) | EOF1 (EOFt) | EOF0 (EOFn) |
| Else = Reserved | N/A | N/A | N/A |

## 7.13.3 FCoE Receive Operation

The 82599 can offload the following tasks from the CPU while processing FCoE receive traffic: FC CRC check, receive coalescing and Direct Data placement (DDP). These offload options are described in the sections that follow.

DDP functionality is not provided for control packets or data packets that do not meet DDP criteria (described later in the sections that follow). In those cases, hardware posts the packets to the legacy Rx queues as is (header and trailer are not stripped including SOF, EOF, FC padding and FC CRC bytes). When DDP functionality is enabled, only the FC payload is posted to the user buffers. If the packet's header should be indicated to the legacy Rx queues, all bytes starting at the destination Ethernet MAC address until the FC header and optionally FC header(s) inclusive are posted to the legacy buffer.

### 7.13.3.1 FCoE Receive Cross Functionality

FCoE receive offload capabilities coexist with other functions in the 82599 are listed as follows:

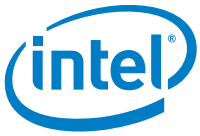


**Table 7-94 FCoE Receive Cross Functionality**

| Cross Function | Requirements |
|---|---|
| Ethernet CRC check | There is no enforcement on save bad frames policy. In the case of save bad frames, packets with bad Ethernet CRC are posted to the legacy receive queue even if DDP is enabled. FC payload of bad packets are never posted directly to the user buffers. |
| Ethernet padding extraction | There is no enforcement on the Ethernet padding extraction. When DDP is enabled, hardware posts the FC payload to the user buffers. When DDP is not enabled the entire packets are posted to the legacy receive queues with or without the Ethernet padding according to the device setting. |
| LinkSec offload | LinkSec encapsulation covers the entire Ethernet packet payload. If the traffic includes LinkSec, hardware must process first the LinkSec encapsulation uncovering the FCoE plain text to the FCoE offload logic. If the LinkSec processing is not enabled and the packets include LinkSec encapsulation, then the packets are posted to the matched legacy receive queue. If the LinkSec processing is enabled but fails for any reason, the packet can still be posted to the matched legacy queue according to save bad frames policy. |
| VLAN header | It is assumed that any FCoE has a VLAN header. In the case of double VLAN mode, the packet must have the two VLAN headers. |
| SNAP packet | The 82599 does not provide FCoE offload for FCoE frame over SNAP. |
| FC and PFC | FC traffic relies on a high-quality link that guarantees no packet loss. It is expected that any lost traffic protocols supported by the network is enabled by the 82599 as well. |
| Virtualization | It is expected that VM(s) generate FC write requests to the VMM. FCoE setting and FCoE traffic is expected only by the VMM accessing the physical function. |
| TCP/IP and UDP/IP offload | FCoE traffic is L2 traffic (not over IP). Any setting of TCP/IP and UDP/IP offload capabilities are not applicable and do not impact FCoE offload functions. |
| Jumbo frames | Maximum expected clear text FC frame size is 2140 bytes (FC header + FC payload + FC CRC). Adding optional FC crypto, plus FCoE encapsulation, plus optional LinkSec encapsulation packet might exceed the 2200 bytes. In order to enable FCoE traffic, jumbo packet reception should be enabled. |
| Receive descriptors in the legacy Rx queues | When FC CRC offload or DDP functionality are enabled, software must use the advanced descriptors in the associated legacy Rx queues (SRRCTL.DESCTYPE = 001b). The legacy Rx buffers must be larger than the maximum expected packet size so any Rx packets span on a single buffer. |

### 7.13.3.2 FC Receive CRC Offload

FC CRC calculation is one of the most CPU intensive tasks in TSO transactions. The 82599 offloads the receive FC CRC integrity check while trashing the CRC bytes and FC padding bytes.

The 82599 recognizes FCoE frames in the receive data path by their FCoE Ethernet type and the FCoE version in the FCoE header. The Ethernet type that hardware associates with FCoE is defined in the ETQF register by setting the *FCoE* bit with a specific Ethernet type value. The supported FCoE versions by the Rx offload logic are defined by FCRXCTRL.FCOEVER. FCoE packets that do not match the previously described Ethernet type and FCoE versions are ignored by the Rx FCoE logic.

The 82599 reconstructs the FC CRC while processing the incoming bytes and compares it against the received FC CRC. The frame is considered a good FC packet if the previous comparison matches and it is considered as a bad FC packet otherwise.

The FC CRC integrity check is meaningful only if all the following conditions are met:

- The received frame contains a correct Ethernet CRC
- If the received frame includes LinkSec encapsulation then LinkSec offload must be enabled and LinkSec integrity is found OK.

The length of the FC padding bytes that hardware trashes are defined in the *Fill Bytes* field in the FC frame control (F_CTL). The *Fill Bytes* field can have any value between zero to three that makes the FC frame a whole number of Dwords. It is expected that the *Fill Bytes* field would be zero except for last data frames within a sequence.

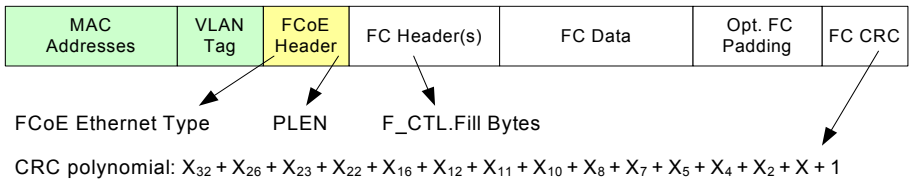



**Figure 7-49 Relevant FCoE and FC Fields for CRC Receive Offload**

## 7.13.3.3 Large FC Receive

Large FC receive includes two types of offloads. The 82599 can save a data copy by posting the received FC payload directly to the kernel storage cache or the user application space (in the remainder of the document there is no difference between the two cases and it is named as user buffers). When the packet's payload are posted directly to the user buffers their headers can still be posted to the legacy receive queues. The 82599 saves CPU cycles by reducing the data copy and also minimize CPU processing by posting only the packet's headers that are required for software.

Figure 7-50 shows the mapping of received FCoE frames to the legacy Rx queue and the user buffers. Figure 7-51 shows a top level overview of the large FC receive flow. The remaining sections detail the large FC receive functionality as follows:

- Enabling large FC receive — Section 7.13.3.3.1
- FC read exchange flow — Section 7.13.3.3.2
- FC write exchange flow — Section 7.13.3.3.3
- FCoE receive filtering (Frame types and rules) — Section 7.13.3.3.5, Section 7.13.3.3.10 and Section 7.13.3.3.12
- User descriptors — Section 7.13.3.3.7
- Header posting to the legacy receive queues and FC exceptions — Section 7.13.3.3.13 and Section 7.13.3.3.14
- Interrupts — Section 7.13.3.3.15

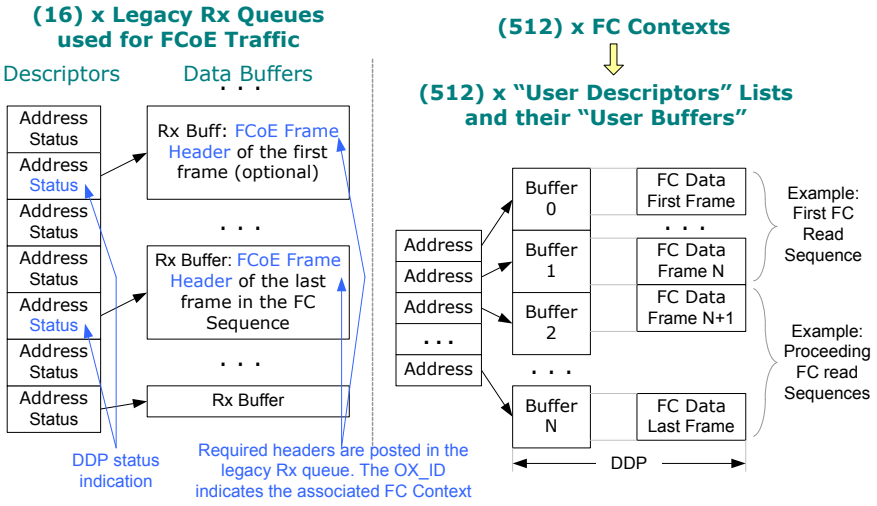



**Figure 7-50 Large FC Reception to User Buffers and Legacy Rx Queue**

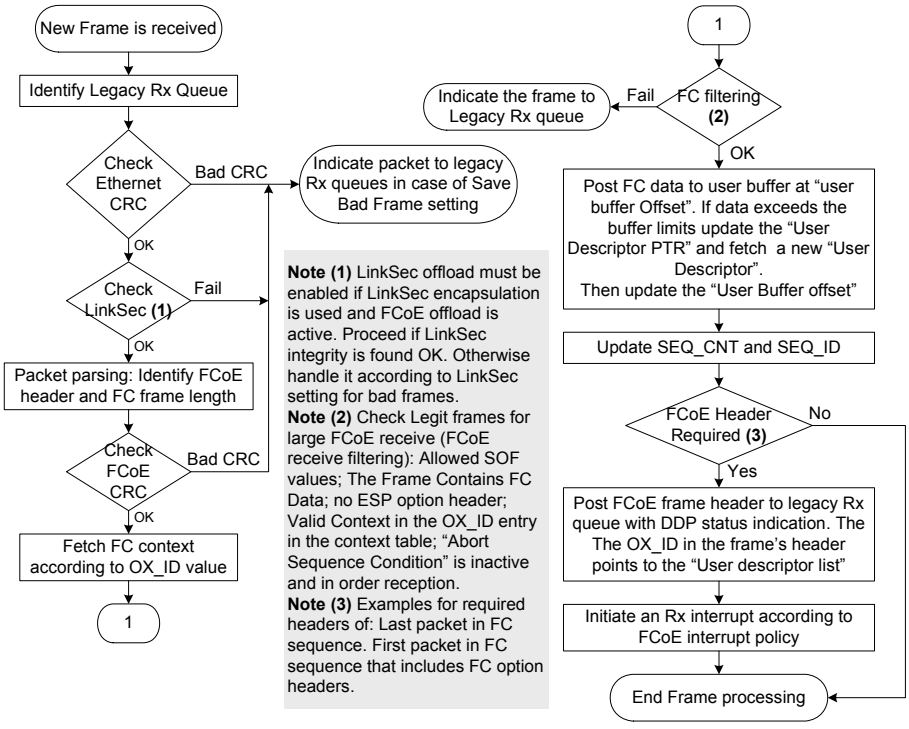



**Figure 7-51 FCoE Large Receive Flow Diagram**

#### 7.13.3.3.1 Enabling Large FC Receive

Large FC receive offload is enabled per each outstanding read or write exchange by programming the FCoE context table with the flow parameters. Setting the FC context for read or write exchange is done at run time. It is expected that a read context is programmed before the read request is initiated to the remote target and write context is programmed before the target sends the ready indication to the initiator. Unless the FC context is invalidated, software must not modify it in the middle of a transaction (see Section 7.13.3.3.5 for details on context invalidation). For more details on FCoE initialization flow see Section 4.6.9.

#### 7.13.3.3.2 FC Read Exchange Flow

Figure 7-52 shows an example of an FC (class 3) read request. This flow is detailed in this section.

1. The software checks if the read request can use large FC receive offload depending on FC context resources and some criteria as listed in Section 7.13.3.3.5. Section 7.13.3.3.12 describes a proposed software flow to manage the FC contexts.
2. If the previous conditions are not met, software can initiate the FC read request according the flow described in Figure 7-52 while the received frames are posted to the legacy receive queues.

- If the previous conditions are met, software locks the relevant user buffers (the target buffers for the FC read request) and program the FC context table. It then initiates the FC read request according the flow shown in Figure 7-52. The payload of the received frames is posted directly to the user buffers. Some of the packet's headers (only the required ones) are posted to the legacy receive queues. The FC header in the packet's header contains the OX_ID field. This field indicates to software its context and its user buffer list. During nominal operation, all packets' headers except packets with FC optional headers are trashed by the hardware minimizing software overhead.

3. The target sends the FCP_RSP frame type indicating the completion of the read exchange. As a response, the hardware invalidates the FC read context (if it was used) and indicates the number of bytes posted directly to the user buffers in the receive descriptor (see Section 7.13.3.3.13). Software indicates the read completion to the application.

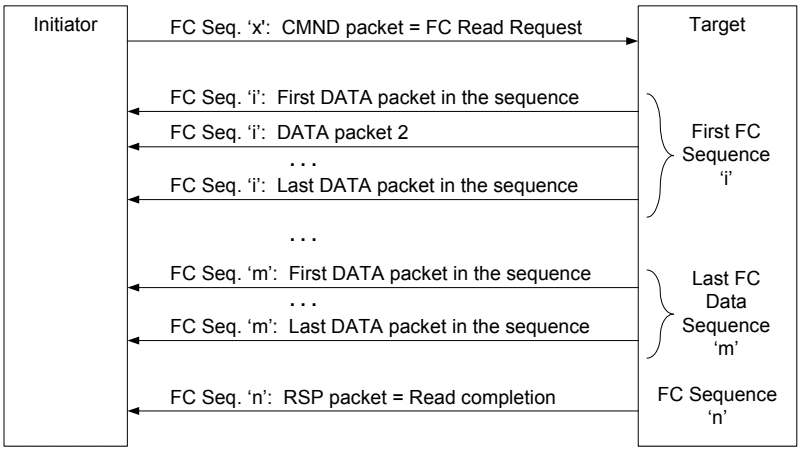



**Figure 7-52 Example for FC Class 3 Read Exchange Flow**

## 7.13.3.3.3 FC Write Exchange Flow

Figure 7-53 shows an example of an FC (class 3) write request (on which the Seq_CNT starts from zero on each new sequence). This flow is detailed in the sections that follow.

1. The host (originator) sends an FC write request to the target (responder).
2. Software in the target checks if the write request can use large FC receive offload depending on FC context resources and some criteria as listed in Section 7.13.3.3.5.
3. If the previous conditions are met, software can use DDP for this FC write exchange.
4. The target software locks the relevant user buffers (the target buffers for the FC write request) and program the FC context table. It then initiates the FC ready indication to the host.
5. As a response, the host sends the data frames to be written to the target. The frames are received in the target. If DDP is used, the FC payload is posted directly to the user buffers while "most" (see additional details below) packet's headers are trashed minimizing software overhead.
6. The host marks the last data frame it was requested to send by setting the *Sequence Initiative* bit in the F_CTL field.
7. The target identifies the last data frame and invalidates the DDP context. As indicated above, during nominal operation, "most" packet's headers are trashed. Only headers that have meaningful content are posted to host memory as: Headers of packets with FC optional headers and the header of the last packet in a sequence with active sequence initiative bit are posted to the legacy receive queues. The hardware indicates the number of bytes posted directly to the user buffers in the receive descriptor (see Section 7.13.3.3.13). Note that the FC header contains the RX_ID field that can be used by software to identifies its associated DDP context and user buffer list.
8. The target may repeat step 4, which is followed by step 5 until the entire requested data is transferred.

9. The target sends the FCP_RSP frame indicating to the initiator the completion of the write exchange.

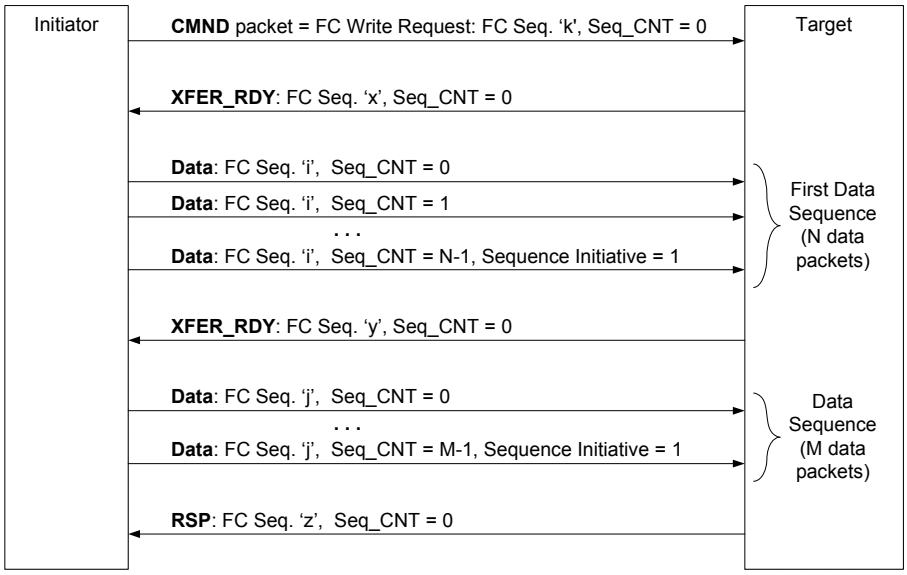



**Figure 7-53 Example for FC Class 3 Write Exchange Flow**

### 7.13.3.3.4 EOF and SOF Flags identification

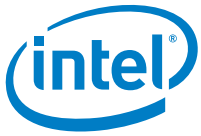
As part of the DDP functionality, hardware identifies the SOF and EOF flags in the received packets. The flags identification is based on a setting of the RSOFF and REOFF registers. These registers are identical to the TSOFF and TEOFF registers and should be programmed by software to the same values.

### 7.13.3.3.5 FCoE Receive Filtering

Received FCoE frames are associated to one of the legacy receive queues according to the scheme described in Section 7.1.2. When the legacy receive queue is enabled, large FC receive functionality is enabled as well if a matched FC receive context is defined. The data is posted to the user buffers that are pointed to by the FC receive context. Some of the headers of these frames that are required for the data processing are posted to the legacy receive queue (see Section 7.13.3.3.13).

FCoE frames that carry FC class 3 or class 2 data can be posted to large receive buffers if they meet the following conditions:

- If the received packet carries LinkSec encapsulation it must be offloaded (and de-capsulated) by hardware.
- The FC context table contains valid context that matches the exchange ID in the received frame. Hardware checks the RX_ID for write data packets sent by the initiator. These packets are identified by the *Exchange Context* bit in the F_CTL header equals zero (originator of exchange). Hardware checks the OX_ID for read response data packets sent by the target. These packets are identified by the *Exchange Context* bit in the F_CTL header equals one (responder of exchange).



- Frames are identified as FCoE frame type according to the Ethernet type in the FCoE header. The Ethernet type that hardware associates with FCoE is defined in the ETQF registers by setting the *FCoE* bit with a specific Ethernet type value.
- The FC frame carry class 2 or class 3 content as defined by the SOF flag. The SOF in the FCoE header equals SOFi2 or SOFn2 or SOFi3 or SOFn3.
- The FCoE version in the received frame is equal or lower than FCRXCTRL.FCOEVER.
- The frame contains data content (with data payload) as defined in the *Routing Control* field (R_CTL) in the FC header:
  - R_CTL.Information (least significant four bits) equals 0x1 (solicited data)
  - R_CTL.Routing (most significant four bits) equals 0x0 (device data)
  - Frames that do not contain device data are not posted to the user buffers. Still these frames are compared against the expected SEQ_ID and SEQ_CNT in the FC context and update these parameters as described in Section 7.13.3.3.5.
- The FC frame does not include ESP header (bit 6 in the DF_CTL field within the FC header is cleared). Frames that include ESP option headers are posted to the legacy receive queue. For good use of hardware resources, software should not program the large FC receive context table with flows that carry an ESP header.
- The FC frame does not include any FC extended headers. For good use of hardware resources, software should not program the large FC receive context table with flows that carry extended headers.
- The first packet received to a new context is identified as the first FC frame in the exchange. This packet is expected to have the SOFi2 or SOFi3 codes. The SEQ_ID on the first packet may have any value.
- The frame is received in order as defined in Section 7.13.3.3.7 and does not carry any exception errors as defined in Section 7.13.3.3.14.
- The first frame on each FC sequence is identified by the SOFi2 or SOFi3 codes in the *SOF* field in the FCoE header. It is expected that the SEQ_ID is changed for any new sequence.
- The last frame on each FC sequence is identified by an active *End Sequence* flag in the F_CTL field in the FC header. It is expected to receive the EOFt code in the *EOF* field; however, hardware does not check this rule.

Other frames (that do not meet the previous conditions) are posted to the legacy receive queues according to the generic Rx filtering rules.

### 7.13.3.3.6 DDP Context

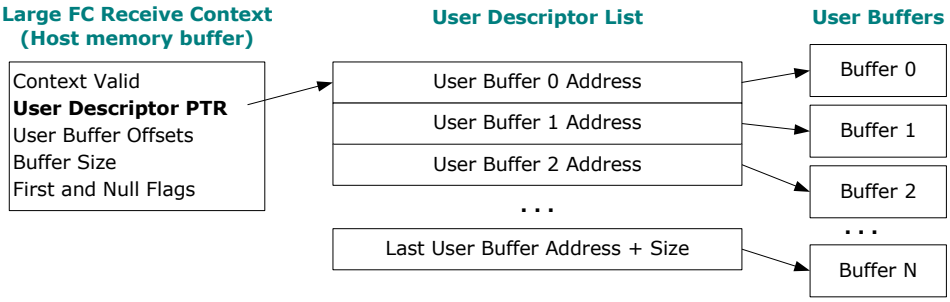
Hardware can provide DDP offload for up to 512 concurrent outstanding FC read or write exchanges. Each exchange has an associated FC context in hardware. Contexts are identified by the exchange ID (OX_ID for FC read and RX_ID for FC write). The exchange ID is a 16-bit field so that a system could theoretically generate up to 64 K concurrent outstanding FC read requests and 64 K concurrent outstanding FC write requests. Hardware contains 512 contexts for the 512 concurrent outstanding exchanges. Using exchange ID values between 0 to 511, software can benefit from the DDP offload. Any exchange ID value in the range of 0 to 511 can be used for either read or write exchange but not for both.

The FC context is a set of parameters used to identify a frame and its user buffers in host memory. The context parameters are split into two categories (according to the internal hardware implementation): DMA context (FCPTRL, FCPTRH, FCBUFF and FCDMARW registers) and filter context (FCFLT, FCPARAM and FCFLTRW register) as listed in Table 7-95 and shown in Figure 7-54.

Software should program both the DMA context and filter context making the context usable. During reception, hardware updates some of the parameters if the packet matches all criteria detailed in Section 7.13.3.3.5. Initialization values and the updated ones are listed in this section.

### Table 7-95 Large FC Context Table

| Exchange ID | DMA Context (FCPTRL, FCPTRH, FCBUFF, FCDMARW) | | Filter Context (FCFLT and FCFLTRW) | |
|---|---|---|---|---|
| | DMA Flags | User Descriptor | Filter Flags | In Order Reception |
| 0 | Valid, First, Count | Size, Offset, Pointer | Valid, First | Seq_ID, Seq_CNT,PARAM |
| 1 | Valid, First, Count | Size, Offset, Pointer | Valid, First | Seq_ID, Seq_CNT,PARAM |
| 2 | Valid, First, Count | Size, Offset, Pointer | Valid, First | Seq_ID, Seq_CNT,PARAM |
| . . . | . . . | | . . . | |
| 511 | Valid, First, Count | Size, Offset, Pointer | Valid, First | Seq_ID,Seq_CNT,PARAM |




### Figure 7-54 Large FC Receive Context Related to the User Buffers

DMA Context Valid (1 bit) and Filter Context Valid (1 bit) — These bits indicates the validity of this context.

**Note:** During programming time, software should enable first the DMA context. When software disables a context it should invalidate first the filter context. See more details on context invalidation in Section 7.13.3.3.10.

Filter First (1 bit) and DMA First (1 bit) — The first received frame that matches an active context in the filter unit is marked by the filter. This marking is used by the DMA unit as an indication that reception to this context has been started. The DMA context does not accept packets from the filter unit unless it received successfully the packet that was marked as the first one (see the section on exception handling in Section 7.13.3.3.14). The Filter First flag should be cleared by software when programming the context. Hardware sets this bit when the filter unit recognizes the first packet that matches a valid context. The DMA First bit should be cleared by software when programming the context.

Hardware sets this bit when the DMA unit received packet that matches a valid context and marked as first by the filter unit.

Buffer Count (8 bit) — This field defines the number of remaining user buffers in the list. At programming time, software sets the buffer count to the number of the allocated user buffers. During reception, hardware decrements the buffer count as each of them completes. The number of active buffers equals the buffer count value while 0x00 equals 256.

Buffer Size (2 bit) — This field defines the user buffer size used in this context. It can be 4 KB, 8 KB, 16 KB or 64 KB. All buffers except the first one and the last one are full size. The address of all buffers is aligned to the buffer size in the context. The first buffer may start at a non-zero offset. The size of the last buffer may be smaller than the buffer size as defined by the last buffer size parameter.

User Buffer Offset (16 bit) — This field defines the byte offset within the current buffer to which the next packet should be posted. At context programming, the software sets the user buffer offset to the beginning of the first buffer. During reception, hardware updates this field at the end of each packet processing for the next received packet.

Last User Buffer Size (16 bit) — This field defines the size of the last user buffer in byte units.

User Descriptor PTR (8 byte) — The user buffers are indicated by a list of pointers named as user descriptors (see Section 7.13.3.3.9 for a description of the user descriptors). The user descriptor PTR in the FC context is a pointer to the user descriptor list. At programming time, software sets the user descriptor PTR to the beginning of the user descriptor list. During reception, hardware increments the user descriptor PTR by eight (the size of the user descriptor) when it completes a buffer and requires the next one.

SEQ_ID (8bit) — The sequence ID identifies the sequence number sent by the target. An FC read or write exchange can be composed of multiple sequences depending on the target implementation. The SEQ_ID has a different value for each sequence and does not necessarily increment sequentially. Hardware uses the SEQ_ID for checking in-order reception as described in Section 7.13.3.3.7. Hardware updates the SEQ_ID in the context table according to the value of the SEQ_ID in the incoming frame. The initialization value during programming could be of any value. For future compatibility software should set it to zero.

SEQ_CNT (16 bit) — SEQ_CNT is an index of the expected FC frames within a sequence or within the entire exchange depending on the target implementation. Hardware uses the SEQ_CNT for checking in order reception as described in Section 7.13.3.3.7. On read context, software should initialize SEQ_CNT to zero. On write context, software should initialize SEQ_CNT to SEQ_CNT + 1 of the last packet of the same exchange received from the initiator. For each in-order reception, hardware sets SEQ_CNT in the context to the value of the received SEQ_CNT + 1.

PARAM (32 bit) — The PARAM field in the FC header may indicate the data offset within the FC IO exchange. It is indicated as an offset by the *Relative Offset Present* bit in the F_CTL field in the FC header. In this case, the *PARAM* field indicates the expected value of the next received packet. At programming time, software should initialize it to zero. During reception, hardware increments the *PARAM* by the size of the FC payload if it is used as an offset. The FC payload size equals the packet size minus the length of its header and trailer. While the header for this purpose includes all bytes starting at the Ethernet destination address up to and including the basic FC header, the trailer includes the FC CRC, FC padding, EOF including the three reserved bytes, and the Ethernet CRC.

#### 7.13.3.3.7 In Order Reception Checking

Hardware checks in-order reception by *SEQ_ID*, *SEQ_CNT* and *PARAM* fields. These parameters should meet the expected values (as follows) in order to pass in-order reception's criteria.

PARAM — When the PARAM field is used as an offset (as indicated by the Relative Offset Present bit in the F_CTL field in the FC header), the PARAM field in the received packet should be the same as the PARAM field in the FC context. Software should initialize this parameter to the expected received value (equals to zero in read exchanges).

SEQ_ID, SEQ_CNT — SEQ_ID identifies the FC sequence and SEQ_CNT is the FC frame index within the entire exchange or within the sequence (according to specific vendor preference). SEQ_CNT in the received packet could be either the same as the SEQ_CNT in the FC context or it could start from zero for new SEQ_ID, which is different than the SEQ_ID in the context. Software should initialize SEQ_CNT to the expected received value (equals zero in read exchanges). SEQ_ID on the first packet is always assumed to be a new value even if by chance it equals to the initial value in the context.

#### 7.13.3.3.8 Accessing the Large FC Receive Context

The 82599 supports a large number of FC contexts while each context contains about 16 bytes. In order to save consumed memory space, the FC context is accessed by indirect mapping. This section describes how the DMA and filter contexts are accessed. The DMA context is consist of the FCPTRL, FCPTRH and FCBUFF registers while read and write accesses are controlled by the FCDMARW register. The filter context is consist of the FCFLT register while read and write accesses are controlled by the FCFLTRW register.

DMA Context Programming — Software should program the FCPTRL, FCPTRH and FCBUFF registers by the required setting. It then programs the FCDMARW register with the following content:

- FCoESEL should be set by the required context index (OX_ID or RX_ID values)
- The *WE* bit is set to 1b for write access while the *RE* bit is set to 0b.
- LASTSIZE should be set to the relevant value for the context

DMA Context Read — Software should program the FCDMARW register as follows and then read the context on the FCPTRL, FCPTRH, FCBUFF and FCDMARW registers

- Software should initiate two consecutive write cycles to the FCDMARW register with the following setting: FCoESEL should be set to the required FCoE read index while both *WE* and *RE* should be set to 0b.
- FCoESEL should be set by the required context index (OX_ID or RX_ID values).
- *RE* bit should be set to 1b for read access while *WE*, and *LASTSIZE* fields are set to 0b.
- LASTSIZE should be set to 0b. It is ignored by hardware when the *RE* bit is set to 1b. When reading *FCDMARW* the *LASTSIZE* field reflects the context content.

Filter Context Programming — Software should program the FCFLT register by the required setting. It then programs the FCFLTRW register with the following content:

- FCoESEL should be set by the required context index (OX_ID or RX_ID values).
- WE bit is set to 1b for a write access while RE bit is set to 0b.

Filter Context Read — Software should program the FCFLTRW register as follows and then read the context on the FCFLT register:

- FCoESEL should be set by the required context index (OX_ID or RX_ID values).
- RE bit should be set to 1b for a read access while WE bit is set to 0b.

#### 7.13.3.3.9 User Descriptor Structure and User Descriptor List

The buffers in host memory could be either user application memory or storage cache named as user buffers. In both cases the buffers must be locked (against software) and converted to physical memory up front.

The user descriptor list is a contiguous list of pointers to the user buffers. The buffers are aligned to their size as defined in the FC context. The first buffer can start at a non-zero offset as the software defines it in the FC context. All other buffers start at a zero offset. The last buffer can be smaller than the full size as defined in the FC context.

**Table 7-96 FC User Descriptor**

| 63 | 0 |
|---|---|
| User buffer address defined in byte units. N LS bits must be set to zero while N equals 12 for a 4 KB buffer size, 13 for 8 KB buffer size, 14 for 16 KB buffer size and 16 for 64 KB buffer size. | |

#### 7.13.3.3.10 Invalidating FC Receive Context

During nominal activity, hardware invalidates autonomously the FC contexts. The target indicates a completion of an FC read by sending the FCP_RSP frame. Hardware identifies the FCP_RSP frame and invalidates the FC context that matches the OX_ID in the incoming frame. The FCP_RSP frame is posted to the legacy Rx queues with appropriate status indication. Hardware identifies the FCP_RSP frame by the following criteria:

- The frame is identified as FCoE frame by its Ethernet type
- R_CTL.Information (least significant four bits) equals 0x7 (command status)
- R_CTL.Routing (most significant four bits) equals 0x0 (device data)

Context that is invalidated autonomously by hardware is indicated by setting the *FCSTAT* field in the receive descriptor to 10b. When software gets this indication it can unlock the user buffers instantly and re-use the context for a new FC exchange.

In some erroneous cases software might invalidate a context before a read exchange completes (such as a time out event). In such cases, software should clear the *Filter Context Valid* bit and then the *DMA Context Valid* bit. Hardware invalidates the context at a packet's boundaries. Therefore, after software clears the *DMA Context Valid* bit, software should either poll it until it is granted (cleared) by hardware or optionally software could wait ~100 µs (guaranteed time for any associated DMA cycles to complete). In addition, software should also ensure that the receive packet buffer does not contain any residual packets of the same flow. See Section 4.6.7.1 for the required software flow. Only then the software can unlock the user buffers and re-use the context for a new FC exchange.

#### 7.13.3.3.11 Invalidating FC Write Context

During nominal activity, hardware invalidates autonomously the FC contexts. The initiator indicates a completion of a granted portion of an FC write by sending a data frame with active sequence initiative flag. After receiving this type of frame, hardware invalidates the matched FC context. The header of this frame is posted to the legacy Rx queues with appropriate status indication. Hardware identifies this frame by the following criteria:

- The frame is identified as FCoE frame by its Ethernet type
- R_CTL -> Information (least significant four bits) equals 0x1 (solicited data)
- R_CTL -> Routing (most significant four bits) equals 0x0 (device data)
- F_CTL -> Sequence initiative equals 1b indicating transfer initiative to the target
- F_CTL -> End sequence" equals 1b indicating last frame in a sequence

Context that is invalidated autonomously by hardware is indicated by setting the *FCSTAT* field in the receive descriptor to 10b. When software gets this indication, it can unlock the user buffers instantly and re-use the context for a new FC exchange. If the FC write is not complete, software can re-use the same context for the completion of the exchange. It can also define a new user buffer list and indicate it to hardware by programming the DMA context. It then can enable the filter context by setting the *Re-Validation* bit the *WE* bit and the *FCoESEL* field in the FCFLTRW register.

Software can also invalidate a context in case of a time out event or other reasons. Software invalidation flow is described in Section 7.13.3.3.10.

#### 7.13.3.3.12 OX_ID and RX_ID Pool Management

As previously indicated, hardware enables Large FC receive offload for up to 512 concurrent outstanding read or write requests. In some cases more than 512 concurrent outstanding requests are generated by the FCoE stack. Therefore, software would need to manage two separate queues for the requests: one queue for those FC requested supported by the large FC receive offload and another one for those requests that do not gain the large FC receive offload. Software should claim an entry in the context table, and its associated OX_ID or RX_ID for the duration of the read or write requests, respectively. Once a request completes and its context is invalidated, software can re-use its context entry for a new request.

Table 7-97 defines an example for an OX_ID list that can be used for new FC read requests managed by software at initialization time and during run time. Similarly, this table could be helpful for write requests and their RX_ID or shared pool for both read and write requests.

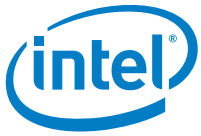


**Table 7-97 Software OX_ID Table**

| Init State of the OX_ID Table | Run-Time Events | Updated State of the OX_ID Table | Run-Time Events | Updated State of the OX_ID Table | Run-Time events | Updated State of the OX_ID Table |
|---|---|---|---|---|---|---|
| 0 | Software is using 50 OX_ID values supported by large FC receive. | 50 | The following FC read requests are completed (and released by software) in the following order: 44, 21, 9, 0. | 50 | Software is using additional 50 OX_ID values supported by large FC receive.<br>The following FC read requests are completed (and released by software) in the following order: 75, 10, 38.<br>Ordering between software requests and releases does not matter in this example. | 100 |
| 1 | | 51 | | 51 | | 101 |
| . . . | | . . . | | . . . | | . . . |
| . . . | | . . . | | . . . | | 510 |
| . . . | | . . . | | . . . | | 511 |
| . . . | | 510 | | 510 | | 44 |
| . . . | | 511 | | 511 | | 21 |
| . . . | | | | 44 | | 9 |
| . . . | | | | 21 | | 0 |
| . . . | | | | 9 | | 75 |
| 510 | | | | 0 | | 10 |
| 511 | | | | | | 38 |

*SW Note:* Software is aware of which read requests can be offloaded by the large FC receive and use OX_IDs in the hardware range (0 to 511) only for those ones.

### 7.13.3.3.13 Packets and Headers Indication in the Legacy Receive Queue

The following packets or packets' headers are posted to the legacy receive queues:

- All FCoE frames that are not offloaded by the DDP logic
- Any packet with exception errors as described in Section 7.13.3.3.14
- Headers of packets posted to the user buffers by the DDP logic that contain meaningful data (as detailed in Section 7.13.3.3.2 and Section 7.13.3.3.3)

There are a few new fields in the receive descriptor dedicated to FCoE described in Section 7.1.6.2:

- Packet Type — FCoE packets are identified by their Ethernet type that is programmed in the ETQF registers.
- FCoE_PARAM — Reflects the value of the PARAM field in the DDP context.
- FCSTAT — FCoE DDP context indication.
- FCERR — FCoE Error indication. DDP offload is provided only when no errors are found.
- FCEOFs and FCEOFe — Status indication on the EOF and SOF flags in the Rx packet.

### 7.13.3.3.14 Exception Handling

Table 7-98 lists the exception errors related to FC receive functionality. Packets with any of the following exception errors are posted to the legacy receive queues with no DDP unless specified differently. In these cases, the exception error is indicated in the *Extended Error* field in the receive descriptor. The exceptions are listed in priority order in the table with highest priority first. Other then the EOF exception, any high priority exception hides all other ones with a lower priority.

**Table 7-98 Exception Error Table**

| Event Description | Actions and Indications |
|---|---|
| Unsupported FCoE version (Rx Version > FCRXCTRL.FCOEVER) | The packet is identified as an FCoE packet type. DDP context parameters are left intact. Speculative CRC check is done. The packet is posted to legacy Rx queue regardless of CRC correctness (independent of FCRXCTRL.SavBad setting). If the packet matches the FCoE redirection table, the packet is posted to Rx queue index defined by the FCRETA[0].<br>RDESC.STATUS.FCSTAT = 00b.<br>RDESC.ERRORS.FCERR = 100b. |
| Incorrect FC CRC (see note 1). | Increment bad FC CRC count. FC context parameters are left intact. The packet can be posted to the legacy receive queues only if the FCRXCTRL.SavBad is set. If the packet matches the FCoE redirection table, the packet is posted to Rx queue index defined by the FCRETA[0].<br>RDESC.STATUS.FCSTAT = 00b.<br>RDESC.ERRORS.FCERR = 001b. |
| Rx packet with ESP option header. | If it matches the DDP context then auto invalidate the filter context while keeping the parameters intact. Note that this exception is not expected since software should not enable a context to an exchange that uses ESP encapsulation.<br>RDESC.STATUS.FCSTAT = 00b / 01b / 10b.<br>RDESC.ERRORS.FCERR = 000b. |
| Received EOFa or EOFni or any unrecognized EOF or SOF flags. | If it matches the DDP context then auto invalidate the filter context while keeping the parameters intact.<br>RDESC.STATUS.FCSTAT = 00b / 01b / 10b.<br>RDESC.ERRORS.FCERR = 010b (even if no DDP match).<br>RDESC.ERRORS.FCEOFe = 1b.<br>RDESC.STATUS.FCEOFs = 1b. |
| Received non-zero abort sequence condition in FC read exchange. | If it matches the DDP context then auto invalidate the filter context while keeping the parameters intact.<br>RDESC.STATUS.FCSTAT = 00b / 01b / 10b.<br>RDESC.ERRORS.FCERR = 010b (even if no DDP match). |
| Out of order reception of packet that matches a DDP context (see note 2). | Auto invalidate the filter context while keeping the parameters intact.<br>RDESC.STATUS.FCSTAT = 01b.<br>RDESC.ERRORS.FCERR = 100b. |
| Received unexpected EOF / SOF:<br>1) New sequence ID and SOF is not SOFi.<br>2) Last packet in a sequence and EOF is not EOFt. | No DDP while filter context is updated (if matched and other parameters are in order).<br>RDESC.STATUS.FCSTAT = 00b. / 01b.<br>RDESC.ERRORS.FCERR = 000b.<br>RDESC.ERRORS.FCEOFe = 1b.<br>RDESC.STATUS.FCEOFs = 0b. |

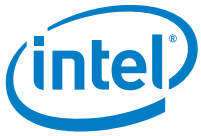


**Table 7-98 Exception Error Table (Continued)**

| Event Description | Actions and Indications |
|---|---|
| The DMA unit gets FCoE packets while it missed the packet that was marked as first by the filter unit (see note 3). | Filter context parameters are updated while DMA context parameters are left intact.<br>RDESC.STATUS.FCSTAT = 01b / 10b / 11b.<br>RDESC.ERRORS.FCERR = 011b or 101b. |
| Last user buffer is exhausted (not enough space for the FC payload). | The filter context is updated while DMA context is auto invalidated.<br>RDESC.STATUS.FCSTAT = 01b / 10b / 11b.<br>RDESC.ERRORS.FCERR = 101b. |
| Legacy receive queue is not enabled or no legacy receive descriptor. | The entire packet is dropped. Auto invalidates the DMA context while the filter context remains active and continues to be updated regularly. Once legacy descriptors become valid again, packets are posted to the legacy queues with the following indication.<br>RDESC.STATUS.FCSTAT = 01b / 10b / 11b.<br>RDESC.ERRORS.FCERR = 101b. |
| Packet missed (lost) by the Rx packet buffer. Normally a case when flow control is not enabled or flow control does not work properly. | The entire packet is dropped (by the Rx packet buffer). Auto invalidate the DMA context while the filter context remains active and continues to be updated regularly. Once the Rx packet buffer gets free, further Rx packets are posted to the legacy queues with the following indication.<br>RDESC.STATUS.FCSTAT = 00b / 01b / 10b / 11b.<br>RDESC.ERRORS.FCERR = 110b.<br>The software might ignore this error when FCSTAT equals 00b. |

***Note (1):*** Out of order might be one of the following cases. SEQ_CNT does not meet expected value. The *PARAM* field in the Rx packet does not match the DDP context. SEQ_ID keeps the same value as the previous packet in a new sequence identified by the presence of SOFi code in the *SOF* field.

***Note (2):*** Lost sync between Filter and DMA contexts could be a result of context invalidation by software together with misbehaved target that sends packet with no host request.

## 7.13.3.3.15 FC Exchange Completion Interrupt

One of the performance indicators of an initiator is measured by the number of I/O operations per second it can generate. The number of FC exchanges per second is affected mainly by the CPU overhead associated with the FC exchange processing and software latencies. The number of concurrent outstanding FC exchanges supported by large FC receive is limited by hardware resources. Reducing the latency associated with processing completions increases the number of FC exchanges per second that the system supports.

The 82599 enables LLI for FCP_RSP frames or last FC data frame in a sequence with active *Sequence Initiative* flag. Any such frames can generate an LLI interrupt if the *FCOELLI* bit in the FCRXCTRL register is set.

Similarly, reducing the latency associated with processing FC write exchange can increase responder performance. During an FC write exchange, the originator handles the initiative to the responder after it sends all the data that the responder is ready to receive. Therefore, the 82599 enables LLI after receiving the last packet in a sequence with the *Sequence Initiative* bit set in the F_CTL field. The LLI is enabled by the same *FCOELLI* bit in the FCRXCTRL register previously indicated.

# 7.14 Reliability

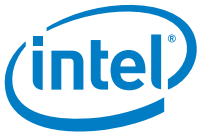
## 7.14.1 Memory Integrity Protection

All the 82599 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 82599 halts the traffic and sets the ECC error interrupt. Software is then required to initiate a complete initialization cycle to resume nominal operation.

## 7.14.2 PCIe Error Handling

For PCIe error events and error reporting see Section 3.1.7.



# 8.0 Programming Interface

## 8.1 Address Regions

The 82599's address space is mapped into four regions with the PCI-Based Address Registers (BARs) listed in Table 8-1 and explained more in Section 9.3.6.1 and Section 9.3.6.2.

**Table 8-1 the 82599 Address Regions**

| Addressable Content | Mapping Style | Region Size |
|---|---|---|
| Internal registers memories and Flash (memory BAR) | Direct memory mapped | 128 KB + Flash_Size |
| Flash (optional)[1] | Direct memory-mapped | 64 KB to 8 MB |
| Expansion ROM (optional)[2] | Direct memory-mapped | 64 KB to 8 MB |
| Internal registers and memories (optional)[2] | I/O window mapped | 32 bytes |
| MSI-X (optional) | Direct memory mapped | 16 KB |

1. The Flash space in the memory CSR and expansion ROM base address map is the same Flash memory. Accessing the memory BAR at offset 128 KB and expansion ROM at offset 0x0 are mapped to the Flash device at offset 0x0.
2. The internal registers and memories can be accessed though I/O space as explained in the sections that follow.

### 8.1.1 Memory-Mapped Access

#### 8.1.1.1 Memory-Mapped Access to Internal Registers and Memories

The internal registers and memories can be accessed as direct memory-mapped offsets from the memory CSR BAR. See the following sections for detailed descriptions of the Device registers.

In IOV mode, this area is partially duplicated per Virtual Function (VF). All replications contain only the subset of the register set that is available for VF programming.

### 8.1.1.2 Memory-Mapped Accesses to Flash

The external Flash can be accessed using direct memory-mapped offsets from the CSR BAR (BAR0 in 32-bit addressing or BAR0/BAR1 in 64-bit addressing). The Flash is only accessible if enabled through the EEPROM Initialization Control word. For accesses, the offset from the CSR BAR minus 128 KB corresponds to the physical address within the external Flash device.

### 8.1.1.3 Memory-Mapped Access to MSI-X Tables

The MSI-X tables can be accessed as direct memory-mapped offsets from BAR3. The MSIX registers are described in Section 8.2.3.6.

In IOV mode, this area is duplicated per VF.

### 8.1.1.4 Memory-Mapped Access to Expansion ROM

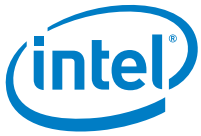
The external Flash can also be accessed as a memory-mapped expansion ROM. Accesses to offsets starting from the expansion ROM base address reference the Flash, provided that access is enabled through the EEPROM Initialization Control word, and if the expansion ROM base address register contains a valid (non-zero) base memory address.

## 8.1.2 I/O-Mapped Access

All internal registers and memories can be accessed using I/O operations. I/O accesses are supported only if an I/O base address is allocated and mapped (BAR2), the BAR contains a valid (non-zero value), and I/O address decoding is enabled in the PCIe configuration.

When an I/O BAR is mapped, the I/O address range allocated opens a 32-byte window in the system I/O address map. Within this window, two I/O addressable registers are implemented: IOADDR and IODATA. The IOADDR register is used to specify a reference to an internal register or memory, and then the IODATA register is used to access it at the address specified by IOADDR:

| Offset | Abbreviation | Name | RW | Size |
|---|---|---|---|---|
| 0x0 | IOADDR | Internal Register, Internal Memory, or Flash Location Address.<br>0x00000-0x1FFFF – Internal registers/memories.<br>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 capable. | RW | 4 bytes |
| 0x08-0x1F | Reserved | Reserved. | O | 4 bytes |



### 8.1.2.1 IOADDR (I/O Offset 0x0; RW)

The IOADDR register must always be written as a Dword access. Writes that are less than 32 bits are ignored. Reads of any size returns a Dword of data; however, the chipset or CPU might only return a subset of that Dword.

For software programmers, the IN and OUT instructions must be used to cause I/O cycles to be used on the PCIe bus. Because writes must be to a 32-bit quantity, the source register of the OUT instruction must be EAX (the only 32-bit register supported by the OUT command). For reads, the IN instruction can have any size target register, but it is recommended that the 32-bit EAX register be used.

Because only a particular range is addressable, the upper bits of this register are hard coded to zero. Bits 31 through 20 are not write-able and always read back as 0b.

At hardware reset (LAN_PWR_GOOD) or PCI reset, this register value resets to 0x00000000. Once written, the value is retained until the next write or reset.

### 8.1.2.2 IODATA (I/O Offset 0x04; RW)

The IODATA register must always be written as a Dword access when the IOADDR register contains a value for the internal register and memories (such as 0x00000-0x1FFFC). In this case, writes that are less than 32 bits are ignored.

Writes and reads to IODATA when the IOADDR register value is in an undefined range (0x20000-0x7FFFC) should not be performed. Results cannot be determined.

**Note:** There are no special software timing requirements on accesses to IOADDR or IODATA. All accesses are immediate except when data is not readily available or acceptable. In this case, the 82599 delays the results through normal bus methods (like split transaction or transaction retry).

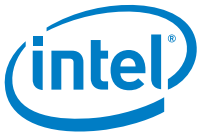
Because a register/memory read or write takes two I/O cycles to complete, software must provide a guarantee that the two I/O cycles occur as an atomic operation. Otherwise, results can be non-deterministic from the software viewpoint.

### 8.1.2.3 Undefined I/O Offsets

I/O offsets 0x08 through 0x1F are considered to be reserved offsets with the I/O window. Dword reads from these addresses return 0xFFFF; writes to these addresses are discarded.

### 8.1.3 Registers Terminology

| Shorthand | Description |
|---|---|
| RW | Read/Write.<br>A register with this attribute can be read and written. If written since reset, the value read reflects the value written. |
| RO | Read Only.<br>If a register is read only, writes to this register have no effect. |
| WO | Write Only.<br>Reading this register might not return a meaningful value. |
| RW1C | Read/Write Clear.<br>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. |
| W1C | Write to clear register.<br>Writing 1b to this register clears an event possibly reported in another register. |
| RC | Read Clear.<br>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.<br>Register that is set to 1b by software by writing a 1b to the register, and cleared to 0b by hardware. |
| Reserved | Reserved field can return any value on read access and must be set to its initial value on write access unless specified differently in the field description. |



# 8.2 Device Registers — PF

## 8.2.1 MSI-X BAR Register Summary PF

See Section 9.3.6.1 for the MSI-X BAR offset in 32-bit and 64-bit BAR options.

| Category | BAR 3 Offset | Alias Offset | Abbreviation | Name | RW |
|---|---|---|---|---|---|
| MSI-X | 0x0000 — (N-1)*0x10 | N/A | MSIXTADD | MSIX table entry lower address. | RW |
| MSI-X | 0x0004 — (N-1)*0x10 | N/A | MSIXTUADD | MSIX table entry upper address. | RW |
| MSI-X | 0x0008 — (N-1)*0x10 | N/A | MSIXTMSG | MSIX table entry message. | RW |
| MSI-X | 0x000C — (N-1)*0x10 | N/A | MSIXTVCTRL | MSIX table vector control. | RW |
| MSI-X | 0x2000 — 0x200C | N/A | MSIXPBA | MSI-X Pending bit array. | RO |

## 8.2.2 Registers Summary PF — BAR 0

All of the 82599's non-PCIe configuration registers are listed in the following table. These registers are ordered by grouping and are not necessarily listed in the order that they appear in the address space.

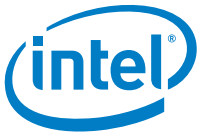
**Note:** All registers should be accessed as a 32-bit width on reads with an appropriate software mask, if needed. A software read/modify/write mechanism should be invoked for partial writes.

**Table 8-2 Register Summary**

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| **General Control Registers** | | | | | | |
| 0x00000 / 0x00004 | CTRL | Device Control Register | Target | RW | | 543 |
| 0x00008 | STATUS | Device Status Register | Target | RO | | 544 |
| 0x00018 | CTRL_EXT | Extended Device Control Register | Target | RW | | 544 |
| 0x00020 | ESDP | Extended SDP Control | Target | RW | | 545 |
| 0x00028 | I2CCTL | I2C Control | Target | RW | PERST | 549 |
| 0x00200 | LEDCTL | LED Control | Target | RW | | 549 |
| 0x05078 | EXVET | Extended VLAN Ether Type | Target | RW | | 551 |

## Table 8-2 Register Summary (Continued)

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| **EEPROM/Flash Registers** | | | | | | |
| 0x10010 | EEC | EEPROM/Flash Control Register | FLEEP | RW | | 552 |
| 0x10014 | EERD | EEPROM Read Register | FLEEP | RW | | 554 |
| 0x1001C | FLA | Flash Access Register | FLEEP | RW | | 555 |
| 0x10114 | EEMNGDATA | Manageability EEPROM Read/ Write Data | FLEEP | RW | | 556 |
| 0x10118 | FLMNGCTL | Manageability Flash Control Register | FLEEP | RW | | 557 |
| 0x1011C | FLMNGDATA | Manageability Flash Read Data | FLEEP | RW | | 557 |
| 0x01013C | FLOP | Flash Opcode Register | FLEEP | RW | | 558 |
| 0x10200 | GRC | General Receive Control | FLEEP | RW | | 558 |
| **Flow Control Registers** | | | | | | |
| 0x0431C / 0x03008 | PFCTOP | Priority Flow Control Type Opcode | MAC | RW | | 559 |
| 0x03200+4*n, n=0...3 | FCTTVn | Flow Control Transmit Timer Value n | DBU-Rx | RW | | 559 |
| 0x03220+4*n, n=0...7 | FCRTL[n] | Flow Control Receive Threshold Low | DBU-Rx | RW | | 560 |
| 0x03260+4*n, n=0...7 | FCRTH[n] | Flow Control Receive Threshold High | DBU-Rx | RW | | 560 |
| 0x032A0 | FCRTV | Flow Control Refresh Threshold Value | DBU-Rx | RW | | 561 |
| 0x0CE00 | TFCS | Transmit Flow Control Status | DBU-Tx | RO | | 561 |
| 0x03D00 | FCCFG | Flow Control Configuration | DBU-Rx | RW | | 561 |
| **PCIe Registers** | | | | | | |
| 0x11000 | GCR | PCIe Control Register | PCIe | RW | | 562 |
| 0x11010 | GSCL_1 | PCIe Statistic Control Register #1 | PCIe | RW | | 562 |
| 0x11014 | GSCL_2 | PCIe Statistic Control Registers #2 | PCIe | RW | | 563 |
| 0x011030+4*n, n=0...3 | GSCL_5_8 | PCIe Statistic Control Register #5...#8 | PCIe | RW | | 565 |

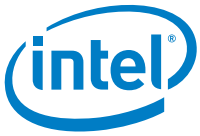


## Table 8-2 Register Summary (Continued)

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| 0x11020+4*n, n=0...3 | GSCN_0_3 | PCIe Statistic Counter Registers #0...#3 | PCIe | RO | | 565 |
| 0x10150 | FACTPS | Function Active and Power State to Manageability | FLEEP | RO | | 565 |
| 0x11040 | PCIEPHYADR | PCIe PHY Address Register | PCIe | RW | | 566 |
| 0x11044 | PCIEPHYDAT | PCIe PHY Data Register | PCIe | RW | | 567 |
| 0x10140 | SWSM | Software Semaphore Register | FLEEP | RW | | 567 |
| 0x10148 | FWSM | Firmware Semaphore Register | FLEEP | RW | | 567 |
| 0x10160 | SW_FW_SYNC | Software–Firmware Synchronization | FLEEP | RW | | 569 |
| 0x11050 | GCR_EXT | PCIe Control Extended Register | PCIe | RW | | 569 |
| 0x11064 | MREVID | Mirrored Revision ID | PCIe | RO | | 570 |
| 0x110B0 | PICAUSE | PCIe Interrupt Cause | PCIe | RO | | 570 |
| 0x110B8 | PIENA | PCIe Interrupts Enable | PCIe | RW | | 571 |
| **Interrupt Registers** | | | | | | |
| 0x00800 | EICR | Extended Interrupt Cause Register | Interrupt | RW1C | | 572 |
| 0x00808 | EICS | Extended Interrupt Cause Set Register | Interrupt | WO | | 573 |
| 0x00880 | EIMS | Extended Interrupt Mask Set/ Read Register | Interrupt | RWS | | 573 |
| 0x00888 | EIMC | Extended Interrupt Mask Clear Register | Interrupt | WO | | 574 |
| 0x00810 | EIAC | Extended Interrupt Auto Clear Register | Interrupt | RW | | 574 |
| 0x00890 | EIAM | Extended Interrupt Auto Mask Enable Register | Interrupt | RW | | 574 |
| 0x00A90+4*(n-1), n=1...2 | EICS[n] | Extended Interrupt Cause Set Registers | Interrupt | WO | | 575 |
| 0x00AA0+4*(n-1), n=1...2 | EIMS[n] | Extended Interrupt Mask Set/ Read Registers | Interrupt | RWS | | 575 |
| 0x00AB0+4*(n-1), n=1...2 | EIMC[n] | Extended Interrupt Mask Clear Registers | Interrupt | WO | | 575 |
| 0x00AD0+4*(n-1), n=1...2 | EIAM[n] | Extended Interrupt Auto Mask Enable registers | Interrupt | RW | | 575 |

## Table 8-2 Register Summary (Continued)

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| 0x00894 | EITRSEL | MSI to EITR Select | Interrupt | RW | | 575 |
| 0x00820+4*n, n=0...23 and 0x012300+4*(n-24), n=24...128 | EITR[n] | Extended Interrupt Throttle Registers | Interrupt | RW | | 576 |
| 0x0E800+4*n, n=0...127 | L34TIMIR[n] | L3 L4 Tuples Immediate Interrupt Rx | DBU-Rx | RW | | 576 |
| 0x0EC90 | LLITHRESH | LLI Size Threshold | DBU-Rx | RW | | 577 |
| 0x0EC60 / 0x05AC0 | IMIRVP | Immediate Interrupt Rx VLAN Priority Register | DBU-Rx | RW | | 577 |
| 0x00900+4*n, n=0...63 | IVAR[n] | Interrupt Vector Allocation | Interrupt | RW | | 578 |
| 0x00A00 | IVAR_MISC | Miscellaneous Interrupt Vector Allocation | Interrupt | RW | | 579 |
| 0x00898 | GPIE | General Purpose Interrupt Enable | Interrupt | RW | | 579 |
| **MSI-X Table Registers** | | | | | | |
| 0x110C0+4*n, n=0...7 / 0x11068 [n=0] | PBACL[n] | MSI-X PBA Clear | PCIe | RW | | 581 |
| **Receive Registers** | | | | | | |
| 0x05080 | FCTRL | Filter Control Register | Rx-Filter | RW | | 582 |
| 0x05088 | VLNCTRL | VLAN Control Register | Rx-Filter | RW | | 583 |
| 0x05090 | MCSTCTRL | Multicast Control Register | Rx-Filter | RW | | 583 |
| 0x0EA00+4*n, n=0...63 / 0x05480+4*n, n=0…15 | PSRTYPE[n] | Packet Split Receive Type Register | DBU-Rx | RW | | 584 |
| 0x05000 | RXCSUM | Receive Checksum Control | Rx-Filter | RW | | 585 |
| 0x05008 | RFCTL | Receive Filter Control Register | Rx-Filter | RW | | 586 |
| 0x05200+4*n, n=0...127 | MTA[n] | Multicast Table Array | Rx-Filter | RW | | 587 |
| 0x0A200+8*n, n=0...127 | RAL[n] | Receive Address Low | Rx-Filter | RW | | 587 |
| 0x0A204+8*n, n=0...127 | RAH[n] | Receive Address High | Rx-Filter | RW | | 587 |
| 0x0A600+4*n, n=0...255 | MPSAR[n] | MAC Pool Select Array | Rx-Filter | RW | | 588 |
| 0x0A000+4*n, n=0...127 | VFTA[n] | VLAN Filter Table Array | Rx-Filter | RW | | 588 |
| 0x0EC80 / 0x05818 | MRQC | Multiple Receive Queues Command Register | DBU-Rx | RW | | 589 |

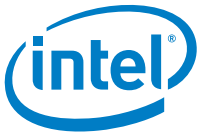


## Table 8-2 Register Summary (Continued)

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| 0x0EC70 | RQTC | RSS Queues Per Traffic Class Register | DBU-Rx | RW | | 590 |
| 0x0EB80+4*n, n=0...9 / 0x05C80+4*n, n=0...9 | RSSRK[n] | RSS Random Key Register | DBU-Rx | RW | | 591 |
| 0x0EB00+4*n, n=0...31 / 0x05C00+4*n, n=0...31 | RETA[n] | Redirection Table | DBU-Rx | RW | | 591 |
| 0x0E000+4*n, n=0...127 | SAQF[n] | Source Address Queue Filter | DBU-Rx | RW | | 592 |
| 0x0E200+4*n, n=0...127 | DAQF[n] | Destination Address Queue Filter | DBU-Rx | RW | | 592 |
| 0x0E400+4*n, n=0...127 | SDPQF[n] | Source Destination Port Queue Filter | DBU-Rx | RW | | 592 |
| 0x0E600+4*n, n=0...127 | FTQF[n] | Five Tuple Queue Filter | DBU-Rx | RW | | 593 |
| 0x0EC30 | SYNQF | SYN Packet Queue Filter | DBU-Rx | RW | | 594 |
| 0x05128+4*n, n=0...7 | ETQF[n] | EType Queue Filter | Rx-Filter | RW | | 594 |
| 0x0EC00+4*n, n=0...7 | ETQS[n] | EType Queue Select | DBU-Rx | RW | | 595 |
| **Receive DMA Registers** | | | | | | |
| 0x01000+0x40*n, n=0...63 and 0x0D000+0x40*(n-64), n=64...127 | RDBAL[n] | Receive Descriptor Base Address Low | DMA-Rx | RW | | 596 |
| 0x01004+0x40*n, n=0...63 and 0x0D004+0x40*(n-64), n=64...127 | RDBAH[n] | Receive Descriptor Base Address High | DMA-Rx | RW | | 596 |
| 0x01008+0x40*n, n=0...63 and 0x0D008+0x40*(n-64), n=64...127 | RDLEN[n] | Receive Descriptor Length | DMA-Rx | RW | | 596 |
| 0x01010+0x40*n, n=0...63 and 0x0D010+0x40*(n-64), n=64...127 | RDH[n] | Receive Descriptor Head | DMA-Rx | RO | | 597 |
| 0x01018+0x40*n, n=0...63 and 0x0D018+0x40*(n-64), n=64...127 | RDT[n] | Receive Descriptor Tail | DMA-Rx | RW | | 597 |
| 0x01028+0x40*n, n=0...63 and 0x0D028+0x40*(n-64), n=64...127 | RXDCTL[n] | Receive Descriptor Control | DMA-Rx | RW | | 597 |
| 0x01014+0x40*n, n=0...63 and 0x0D014+0x40*(n-64), n=64...127 / 0x02100+4*n, [n=0...15] | SRRCTL[n] | Split Receive Control Registers | DMA-Rx | RW | | 598 |
| 0x02F00 | RDRXCTL | Receive DMA Control Register | DMA-Rx | RW | | 599 |

## Table 8-2 Register Summary (Continued)

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| 0x03C00+4*n, n=0...7 | RXPBSIZE[n] | Receive Packet Buffer Size | DBU-Rx | RW | | 600 |
| 0x03000 | RXCTRL | Receive Control Register | DBU-Rx | RW | | 600 |
| 0x03190 | RXMEMWRAP | Rx Packet Buffer Flush Detect | DBU-Rx | RO | | 600 |
| 0x03028 | RSCDBU | RSC Data Buffer Control Register | DBU-Rx | RW | | 602 |
| 0x0102C+0x40*n, n=0...63 and 0x0D02C+0x40*(n-64), n=64...127 | RSCCTL[n] | RSC Control | DMA-Rx | RW | | 602 |
| **Transmit Registers** | | | | | | |
| 0x08100 | DTXMXSZRQ | DMA Tx TCP Max Allow Size Requests | DMA-Tx | RW | | 603 |
| 0x04A80 | DMATXCTL | DMA Tx Control | DMA-Tx | RW | | 603 |
| 0x04A88 | DTXTCPFLGL | DMA Tx TCP Flags Control Low | DMA-Tx | RW | | 604 |
| 0x04A8C | DTXTCPFLGH | DMA Tx TCP Flags Control High | DMA-Tx | RW | | 604 |
| 0x06000+0x40*n, n=0...127 | TDBAL[n] | Transmit Descriptor Base Address Low | DMA-Tx | RW | | 604 |
| 0x06004+0x40*n, n=0...127 | TDBAH[n] | Transmit Descriptor Base Address High | DMA-Tx | RW | | 605 |
| 0x06008+0x40*n, n=0...127 | TDLEN[n] | Transmit Descriptor Length | DMA-Tx | RW | | 605 |
| 0x06010+0x40*n, n=0...127 | TDH[n] | Transmit Descriptor Head | DMA-Tx | RO | | 605 |
| 0x06018+0x40*n, n=0...127 | TDT[n] | Transmit Descriptor Tail | DMA-Tx | RW | | 606 |
| 0x06028+0x40*n, n=0...127 | TXDCTL[n] | Transmit Descriptor Control | DMA-Tx | RW | | 606 |
| 0x06038+0x40*n, n=0...127 | TDWBAL[n] | Tx Descriptor Completion Write Back Address Low | DMA-Tx | RW | | 607 |
| 0x0603C+0x40*n, n=0...127 | TDWBAH[n] | Tx Descriptor Completion Write Back Address High | DMA-Tx | RW | | 608 |
| 0x0CC00+0x4*n, n=0...7 | TXPBSIZE[n] | Transmit Packet Buffer Size | DBU-Tx | RW | | 608 |
| 0x0CD10 | MNGTXMAP | Manageability Transmit TC Mapping | DBU-Tx | RW | | 608 |
| 0x08120 | MTQC | Multiple Transmit Queues Command Register | DMA-Tx | RW | | 609 |
| 0x04950 +0x4*n, n=0...7 | TXPBTHRESH | Tx Packet Buffer Threshold | DMA-Tx | RW | | 610 |

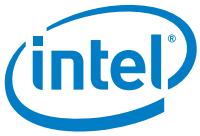


## Table 8-2 Register Summary (Continued)

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| **DCB Registers** | | | | | | |
| 0x02430 | RTRPCS | DCB Receive Packet plane Control and Status | DMA-Rx | RW | | 611 |
| 0x04900 | RTTDCS | DCB Transmit Descriptor Plane Control and Status | DMA-Tx | RW | | 611 |
| 0x0CD00 | RTTPCS | DCB Transmit Packet Plane Control and Status | DBU-Tx | RW | | 617 |
| 0x03020 | RTRUP2TC | DCB Receive User Priority to Traffic Class | DBU-Rx | RW | | 613 |
| 0x0C800 | RTTUP2TC | DCB Transmit User Priority to Traffic Class | DBU-Tx | RW | | 614 |
| 0x02140+4*n, n=0...7 | RTRPT4C[n] | DCB Receive Packet Plane T4 Config | DMA-Rx | RW | | 615 |
| 0x082E0+4*n, n=0...3 | TXLLQ[n] | Strict Low Latency Tx Queues | DMA-Tx | RW | | 616 |
| 0x02160+4*n, n=0...7 | RTRPT4S[n] | DCB Receive Packet plane T4 Status | DMA-Rx | RO | | 616 |
| 0x04910+4*n, n=0...7 | RTTDT2C[n] | DCB Transmit Descriptor plane T2 Config | DMA-Tx | RW | | 616 |
| 0x0CD20+4*n, n=0...7 | RTTPT2C[n] | DCB Transmit Packet Plane T2 Config | DBU-Tx | RW | | 617 |
| 0x0CD40+4*n, n=0...7 | RTTPT2S[n] | DCB Transmit Packet Plane T2 Status | DBU-Tx | RO | | 617 |
| 0x04980 | RTTBCNRM | DCB Transmit Rate–Scheduler MMW | DMA-Tx | RW | | 617 |
| 0x04904 | RTTDQSEL | DCB Transmit Descriptor Plane Queue Select | DMA-Tx | RW | | 618 |
| 0x04908 | RTTDT1C | DCB Transmit Descriptor Plane T1 Config | DMA-Tx | RW | | 618 |
| 0x0490C | RTTDT1S | DCB Transmit Descriptor Plane T1 Status | DMA-Tx | RO | | 618 |
| 0x04984 | RTTBCNRC | DCB Transmit Rate-Scheduler Config | DMA-Tx | RW | | 619 |
| 0x04988 | RTTBCNRS | DCB Transmit Rate-Scheduler Status | DMA-Tx | RW | | 619 |
| 0x0498C | RTTBCNRD | DCB Transmit Rate Scheduler Rate Drift | DMA-Tx | RW | | 620 |

## Table 8-2 Register Summary (Continued)

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| **DCA Registers** | | | | | | |
| 0x0100C+0x40*n, n=0...63 and 0x0D00C+0x40*(n-64), n=64...127 / 0x02200+4*n, [n=0...15] | DCA_RXCTRL[n] | Rx DCA Control Register | DMA-Rx | RW | | 621 |
| 0x0600C+0x40*n, n=0...127 | DCA_TXCTRL[n] | Tx DCA Control Register | DMA-Tx | RW | | 622 |
| 0x11070 | DCA_ID | DCA Requester ID Information Register | PCIe | RO | | 623 |
| 0x11074 | DCA_CTRL | DCA Control Register | PCIe | RW | | 622 |
| **Security Registers** | | | | | | |
| 0x08800 | SECTXCTRL | Security Tx Control | SEC-Tx | RW | | 624 |
| 0x08804 | SECTXSTAT | Security Tx Status | SEC-Tx | RO | | 625 |
| 0x08808 | SECTXBUFFAF | Security Tx Buffer Almost Full | SEC-Tx | RW | | 625 |
| 0x08810 | SECTXMINIFG | Security Tx Buffer Minimum IFG | SEC-Tx | RW | | 625 |
| 0x08D00 | SECRXCTRL | Security Rx Control | SEC-Rx | RW | | 626 |
| 0x08D04 | SECRXSTAT | Security Rx Status | SEC-Rx | RO | | 626 |
| **LinkSec Registers** | | | | | | |
| 0x08A00 | LSECTXCAP | LinkSec Tx Capabilities Register | SEC-Tx | RW | | 627 |
| 0x08F00 | LSECRXCAP | LinkSec Rx Capabilities Register | SEC-Rx | RW | | 627 |
| 0x08A04 | LSECTXCTRL | LinkSec Tx Control Register | SEC-Tx | RW | | 628 |
| 0x08F04 | LSECRXCTRL | LinkSec Rx Control Register | SEC-Rx | RW | | 628 |
| 0x08A08 | LSECTXSCL | LinkSec Tx SCI Low | SEC-Tx | RW | | 629 |
| 0x08A0C | LSECTXSCH | LinkSec Tx SCI High | SEC-Tx | RO | | 629 |
| 0x08A10 | LSECTXSA | LinkSec Tx SA | SEC-Tx | RW | | 630 |
| 0x08A14 | LSECTXPN0 | LinkSec Tx SA PN 0 | SEC-Tx | RW | | 630 |
| 0x08A18 | LSECTXPN1 | LinkSec Tx SA PN 1 | SEC-Tx | RW | | 630 |
| 0x08A1C+4*n, n=0...3 | LSECTXKEY0[n] | LinkSec Tx Key 0 | SEC-Tx | WO | | 631 |
| 0x08A2C+4*n, n=0...3 | LSECTXKEY1[n] | LinkSec Tx Key 1 | SEC-Tx | WO | | 631 |
| 0x08F08 | LSECRXSCL | LinkSec Rx SCI Low | SEC-Rx | RW | | 631 |

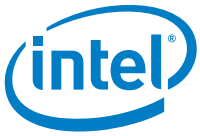


## Table 8-2 Register Summary (Continued)

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| 0x08F0C | LSECRXSCH | LinkSec Rx SCI High | SEC-Rx | RW | | 632 |
| 0x08F10+4*n, n=0...1 | LSECRXSA[n] | LinkSec Rx SA | SEC-Rx | RW | | 632 |
| 0x08F18+4*n, n=0...1 | LSECRXPN[n] | LinkSec Rx SA PN | SEC-Rx | RW | | 633 |
| 0x08F20+0x10*n+4*m, n=0...1, m=0...3 | LSECRXKEY[n,m] | LinkSec Rx Key | SEC-Rx | WO | | 633 |
| 0x08A3C | LSECTXUT | Tx Untagged Packet Counter | SEC-Tx | RO | | 634 |
| 0x08A40 | LSECTXPKTE | Encrypted Tx Packets | SEC-Tx | RO | | 634 |
| 0x08A44 | LSECTXPKTP | Protected Tx Packets | SEC-Tx | RO | | 634 |
| 0x08A48 | LSECTXOCTE | Encrypted Tx Octets | SEC-Tx | RO | | 634 |
| 0x08A4C | LSECTXOCTP | Protected Tx Octets | SEC-Tx | RO | | 635 |
| 0x08F40 | LSECRXUT | LinkSec Untagged Rx Packet | SEC-Rx | RO | | 636 |
| 0x08F44 | LSECRXOCTE | LinkSec Rx Octets Decrypted | SEC-Rx | RO | | 636 |
| 0x08F48 | LSECRXOCTP | LinkSec Rx Octets Validated | SEC-Rx | RO | | 636 |
| 0x08F4C | LSECRXBAD | LinkSec Rx Packet with Bad Tag | SEC-Rx | RO | | 636 |
| 0x08F50 | LSECRXNOSCI | LinkSec No SCI | SEC-Rx | RO | | 637 |
| 0x08F54 | LSECRXUNSCI | LinkSec Unknown SCI | SEC-Rx | RO | | 637 |
| 0x08F58 | LSECRXUC | LinkSec Rx Unchecked Packets | SEC-Rx | RO | | 638 |
| 0x08F5C | LSECRXDELAY | LinkSec Rx Delayed Packets | SEC-Rx | RO | | 638 |
| 0x08F60 | LSECRXLATE | LinkSec Rx Late Packets | SEC-Rx | RO | | 638 |
| 0x08F64+4*n, n=0...1 | LSECRXOK[n] | LinkSec Rx Packet OK | SEC-Rx | RO | | 639 |
| 0x08F6C+4*n, n=0...1 | LSECRXINV[n] | LinkSec Rx Invalid | SEC-Rx | RO | | 639 |
| 0x08F74+4*n, n=0...1 | LSECRXNV[n] | LinkSec Rx Not Valid | SEC-Rx | RC | | 639 |
| 0x08F7C | LSECRXUNSA | LinkSec Rx Unused SA | SEC-Rx | RC | | 639 |
| 0x08F80 | LSECRXNUSA | LinkSec Rx Not Using SA | SEC-Rx | RC | | 640 |
| **IPsec Registers** | | | | | | |
| 0x08900 | IPSTXIDX | IPsec Tx Index | SEC-Tx | RW | | 641 |
| 0x08908+4*n, n=0...3 | IPSTXKEY[n] | IPsec Tx Key Registers | SEC-Tx | RW | | 641 |

## Table 8-2 Register Summary (Continued)

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| 0x08904 | IPSTXSALT | IPsec Tx Salt Register | SEC-Tx | RW | | 642 |
| 0x08E00 | IPSRXIDX | IPsec Rx Index | SEC-Rx | RW | | 642 |
| 0x08E04+4*n, n=[0...3] | IPSRXIPADDR | IPsec Rx IP address Register | SEC-Rx | RW | | 643 |
| 0x08E14 | IPSRXSPI | IPsec Rx SPI Register | SEC-Rx | RW | | 643 |
| 0x08E18 | IPSRXIPIDX | IPsec Rx SPI Register | SEC-Rx | RW | | 643 |
| 0x08E1C+4*n, n=0...3 | IPSRXKEY[n] | IPsec Rx Key Register | SEC-Rx | RW | | 643 |
| 0x08E2C | IPSRXSALT | IPsec Rx Salt Register | SEC-Rx | RW | | 644 |
| 0x08E30 | IPSRXMOD | IPsec Rx Mode Register | SEC-Rx | RW | | 644 |
| **Timers Registers** | | | | | | |
| 0x0004C | TCPTIMER | TCP Timer | Target | RW | | 645 |
| **FCoE Registers** | | | | | | |
| 0x05100 | FCRXCTRL | FC Receive Control | Rx-Filter | RW | | 647 |
| 0x0ED00 | FCRECTL | FCoE Redirection Control | DBU-Rx | RW | | 648 |
| 0x0ED10+4*n, n=0...7 | FCRETA[n] | FCoE Redirection Table | DBU-Rx | RW | | 648 |
| 0x02410 | FCPTRL | FC User Descriptor PTR Low | DMA-Rx | RW | | 648 |
| 0x02414 | FCPTRH | FC User Descriptor PTR High | DMA-Rx | RW | | 649 |
| 0x02418 | FCBUFF | FC Buffer Control | DMA-Rx | RW | | 649 |
| 0x02420 | FCDMARW | FC Receive DMA RW | DMA-Rx | RW | | 650 |
| 0x05108 | FCFLT | FC FLT Context | Rx-Filter | RW | | 650 |
| 0x051D8 | FCPARAM | FC Offset Parameter | Rx-Filter | RW | | 650 |
| 0x05110 | FCFLTRW | FC Filter RW Control | Rx-Filter | WO | | 651 |
| **Flow Director Registers** | | | | | | |
| Global Settings Registers | | | | | | |
| 0x0EE00 | FDIRCTRL | Flow Director Filters Control Register | DBU-Rx | RW | | 652 |
| 0x0EE68 | FDIRHKEY | Flow Director Filters Lookup Table Hash Key | DBU-Rx | RW | | 653 |
| 0x0EE6C | FDIRSKEY | Flow Director Filters Signature Hash Key | DBU-Rx | RW | | 653 |

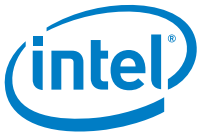


## Table 8-2 Register Summary (Continued)

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| 0x0EE3C | FDIRDIP4M | Flow Director Filters IPv4 Mask | DBU-Rx | RW | | 653 |
| 0x0EE40 | FDIRSIP4M | Flow Director Filters Source IPv4 Mask | DBU-Rx | RW | | 654 |
| 0x0EE44 | FDIRTCPM | Flow Director Filters TCP Mask | DBU-Rx | RW | | 654 |
| 0x0EE48 | FDIRUDPM | Flow Director Filters UDP Mask | DBU-Rx | RW | | 654 |
| 0x0EE74 | FDIRIP6M | Flow Director Filters IPv6 Mask | DBU-Rx | RW | | 655 |
| 0x0EE70 | FDIRM | Flow Director Filters Other Mask | DBU-Rx | RW | | 655 |
| **Global Status / Statistics Registers** | | | | | | |
| 0x0EE38 | FDIRFREE | Flow Director Filters Free | DBU-Rx | RW | | 656 |
| 0x0EE4C | FDIRLEN | Flow Director Filters Length | DBU-Rx | RC | | 656 |
| 0x0EE50 | FDIRUSTAT | Flow Director Filters Usage Statistics | DBU-Rx | RW / RC | | 656 |
| 0x0EE54 | FDIRFSTAT | Flow Director Filters Failed Usage Statistics | DBU-Rx | RW / RC | | 657 |
| 0x0EE58 | FDIRMATCH | Flow Director Filters Match Statistics | DBU-Rx | RC | | 657 |
| 0x0EE5C | FDIRMISS | Flow Director Filters Miss Match Statistics | DBU-Rx | RC | | 657 |
| **Flow Programming Registers** | | | | | | |
| 0x0EE0C+4*n, n=0…2 | FDIRSIPv6[n] | Flow Director Filters Source IPv6 | DBU-Rx | RW | | 657 |
| 0x0EE18 | FDIRIPSA | Flow Director Filters IP SA | DBU-Rx | RW | | 658 |
| 0x0EE1C | FDIRIPDA | Flow Director Filters IP DA | DBU-Rx | RW | | 658 |
| 0x0EE20 | FDIRPORT | Flow Director Filters Port | DBU-Rx | RW | | 658 |
| 0x0EE24 | FDIRVLAN | Flow Director Filters VLAN and FLEX bytes | DBU-Rx | RW | | 658 |
| 0x0EE28 | FDIRHASH | Flow Director Filters Hash Signature | DBU-Rx | RW | | 658 |
| 0x0EE2C | FDIRCMD | Flow Director Filters Command Register | DBU-Rx | RW | | 659 |
| **MAC Registers** | | | | | | |
| 0x04200 | PCS1GCFIG | PCS_1G Global Config Register 1 | MAC | RW | | 661 |
| 0x04208 | PCS1GLCTL | PCG_1G link Control Register | MAC | RW | | 661 |

## Table 8-2 Register Summary (Continued)

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| 0x0420C | PCS1GLSTA | PCS_1G Link Status Register | MAC | RO | | 662 |
| 0x04218 | PCS1GANA | PCS_1 Gb/s Auto-Negotiation Advanced Register | MAC | RW | | 663 |
| 0x0421C | PCS1GANLP | PCS_1GAN LP Ability Register | MAC | RO | | 663 |
| 0x04220 | PCS1GANNP | PCS_1G Auto-Negotiation Next Page Transmit Register | MAC | RW | | 664 |
| 0x04224 | PCS1GANLPNP | PCS_1G Auto-Negotiation LP's Next Page Register | MAC | RO | | 665 |
| 0x04240 | HLREG0 | MAC Core Control 0 Register | MAC | RW | | 666 |
| 0x04244 | HLREG1 | MAC Core Status 1 Register | MAC | RO | | 667 |
| 0x04248 | PAP | Pause and Pace Register | MAC | RW | | 668 |
| 0x0425C | MSCA | MDI Single Command and Address | MAC | RW | | 668 |
| 0x04260 | MSRWD | MDI Single Read and Write Data | MAC | RW | | 669 |
| 0x04268 | MAXFRS | Max Frame Size | MAC | RW | | 669 |
| 0x4288 | PCSS1 | XGXS Status 1 | MAC | RO | | 669 |
| 0x0428C | PCSS2 | XGXS Status 2 | MAC | RO | | 670 |
| 0x04290 | XPCSS | 10GBASE-X PCS Status | MAC | RO | | 670 |
| 0x04298 | SERDESC | SerDes Interface Control Register | MAC | RW | | 672 |
| 0x0429C | MACS | FIFO Status/CNTL report Register | MAC | RW | | 673 |
| 0x042A0 | AUTOC | Auto-Negotiation Control Register | MAC | RW | | 674 |
| 0x042A4 | LINKS | Link Status Register | MAC | RO | | 676 |
| 0x04324 | LINKS2 | Link Status Register 2 | MAC | RO | | 678 |
| 0x042A8 | AUTOC2 | Auto-Negotiation Control 2 Register | MAC | RW | | 679 |
| 0x042B0 | ANLP1 | Auto-Negotiation Link Partner Link Control Word 1 Register | MAC | RO | | 679 |
| 0x042B4 | ANLP2 | Auto-Negotiation Link Partner Link Control Word 2 Register | MAC | RO | | 680 |

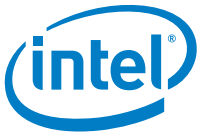


## Table 8-2 Register Summary (Continued)

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| 0x042D0 | MMNGC | MAC Manageability Control Register | MAC | RO | | 680 |
| 0x042D4 | ANLPNP1 | Auto-Negotiation Link Partner Next Page 1 register | MAC | RO | | 680 |
| 0x042D8 | ANLPNP2 | Auto-Negotiation Link Partner Next Page 2 register | MAC | RO | | 681 |
| 0x042E0 | KRPCSFC | KR PCS and FEC Control Register | MAC | RW | | 681 |
| 0x042E4 | KRPCSS | KR PCS Status Register | MAC | RO | | 682 |
| 0x042E8 | FECS1 | FEC Status 1 Register | MAC | RC | | 684 |
| 0x042EC | FECS2 | FEC Status 2 Register | MAC | RC | | 684 |
| 0x014F00 | CoreCTL | Core Analog Configuration Register | MAC | RW | | 684 |
| 0x014F10 | SMADARCTL | Core Common Configuration Register | MAC | RW | | 685 |
| 0x04294 | MFLCN | MAC Flow Control Register | MAC | RW | | 685 |
| 0x04314 | SGMIIC | SGMII Control Register | MAC | RW | | 686 |
| **Statistic Registers** | | | | | | |
| 0x04000 | CRCERRS | CRC Error Count | STAT | RC | | 687 |
| 0x04004 | ILLERRC | Illegal Byte Error Count | STAT | RC | | 687 |
| 0x04008 | ERRBC | Error Byte Count | STAT | RC | | 688 |
| 0x04034 | MLFC | MAC Local Fault Count | STAT | RC | | 688 |
| 0x04038 | MRFC | MAC Remote Fault Count | STAT | RC | | 688 |
| 0x04040 | RLEC | Receive Length Error Count | STAT | RC | | 688 |
| 0x08780 | SSVPC | Switch Security Violation Packet Count | DMA-Tx | RC | | 688 |
| 0x041A4 | LXONRXCNT | Link XON Received Count | STAT | RC | | 689 |
| 0x041A8 | LXOFFRXCNT | Link XOFF Received Count | STAT | RC | | 689 |
| 0x04140+4*n, n=0...7 | PXONRXCNT[n] | Priority XON Received Count | STAT | RC | | 690 |
| 0x04160+4*n, n=0...7 | PXOFFRXCNT[n] | Priority XOFF Received Count | STAT | RC | | 690 |
| 0x0405C | PRC64 | Packets Received [64 Bytes] Count | STAT | RW | | 690 |

## Table 8-2 Register Summary (Continued)

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| 0x04060 | PRC127 | Packets Received [65–127 Bytes] Count | STAT | RW | | 691 |
| 0x04064 | PRC255 | Packets Received [128–255 Bytes] Count | STAT | RW | | 691 |
| 0x04068 | PRC511 | Packets Received [256–511 Bytes] Count | STAT | RW | | 691 |
| 0x0406C | PRC1023 | Packets Received [512–1023 Bytes] Count | STAT | RW | | 691 |
| 0x04070 | PRC1522 | Packets Received [1024 to Max Bytes] Count | STAT | RW | | 692 |
| 0x04078 | BPRC | Broadcast Packets Received Count | STAT | RC | | 692 |
| 0x0407C | MPRC | Multicast Packets Received Count | STAT | RC | | 692 |
| 0x04074 | GPRC | Good Packets Received Count | STAT | RC | | 692 |
| 0x04088 | GORCL | Good Octets Received Count Low | STAT | RC | | 693 |
| 0x0408C | GORCH | Good Octets Received Count High | STAT | RC | | 693 |
| 0x041B0 | RXNFGPC | Good Rx Non-Filtered Packet Counter | STAT | RC | | 693 |
| 0x041B4 | RXNFGBCL | Good Rx Non-Filter Byte Counter Low | STAT | RC | | 693 |
| 0x041B8 | RXNFGBCH | Good Rx Non-Filter Byte Counter High | STAT | RC | | 693 |
| 0x02F50 | RXDGPC | DMA Good Rx Packet Counter | DMA-Rx | RC | | 694 |
| 0x02F54 | RXDGBCL | DMA Good Rx Byte Counter Low | DMA-Rx | RC | | 694 |
| 0x02F58 | RXDGBCH | DMA Good Rx Byte Counter High | DMA-Rx | RC | | 694 |
| 0x02F5C | RXDDPC | DMA Duplicated Good Rx Packet Counter | DMA-Rx | RC | | 694 |
| 0x02F60 | RXDDBCL | DMA Duplicated Good Rx Byte Counter Low | DMA-Rx | RC | | 694 |
| 0x02F64 | RXDDBCH | DMA Duplicated Good Rx Byte Counter High | DMA-Rx | RC | | 695 |
| 0x02F68 | RXLPBKPC | DMA Good Rx LPBK Packet Counter | DMA-Rx | RC | | 695 |

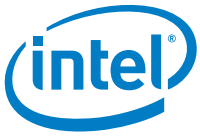


## Table 8-2 Register Summary (Continued)

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| 0x02F6C | RXLPBKBCL | DMA Good Rx LPBK Byte Counter Low | DMA-Rx | RC | | 695 |
| 0x02F70 | RXLPBKBCH | DMA Good Rx LPBK Byte Counter High | DMA-Rx | RC | | 695 |
| 0x02F74 | RXDLPBKPC | DMA Duplicated Good Rx LPBK Packet Counter | DMA-Rx | RC | | 696 |
| 0x02F78 | RXDLPBKBCL | DMA Duplicated Good Rx LPBK Byte Counter Low | DMA-Rx | RC | | 696 |
| 0x02F7C | RXDLPBKBCH | DMA Duplicated Good Rx LPBK Byte Counter High | DMA-Rx | RC | | 696 |
| 0x04080 | GPTC | Good Packets Transmitted Count | STAT | RO | | 696 |
| 0x04090 | GOTCL | Good Octets Transmitted Count Low | STAT | RC | | 696 |
| 0x04094 | GOTCH | Good Octets Transmitted Count High | STAT | RC | | 697 |
| 0x087A0 | TXDGPC | DMA Good Tx Packet Counter | DMA-Tx | RC | | 697 |
| 0x087A4 | TXDGBCL | DMA Good Tx Byte Counter Low | DMA-Tx | RC | | 697 |
| 0x087A8 | TXDGBCH | DMA Good Tx Byte Counter High | DMA-Tx | RC | | 697 |
| 0x040A4 | RUC | Receive Undersize Count | STAT | RC | | 697 |
| 0x040A8 | RFC | Receive Fragment Count | STAT | RC | | 698 |
| 0x040AC | ROC | Receive Oversize Count | STAT | RC | | 698 |
| 0x040B0 | RJC | Receive Jabber Count | STAT | RC | | 698 |
| 0x040B4 | MNGPRC | Management Packets Received Count | STAT | RO | | 698 |
| 0x040B8 | MNGPDC | Management Packets Dropped Count | STAT | RO | | 698 |
| 0x040C0 | TORL | Total Octets Received | STAT | RC | | 699 |
| 0x040C4 | TORH | Total Octets Received | STAT | RC | | 699 |
| 0x040D0 | TPR | Total Packets Received | STAT | RC | | 699 |
| 0x040D4 | TPT | Total Packets Transmitted | STAT | RC | | 699 |
| 0x040D8 | PTC64 | Packets Transmitted (64 Bytes) Count | STAT | RC | | 700 |

## Table 8-2 Register Summary (Continued)

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| 0x040DC | PTC127 | Packets Transmitted [65–127 Bytes] Count | STAT | RC | | 700 |
| 0x040E0 | PTC255 | Packets Transmitted [128–255 Bytes] Count | STAT | RC | | 700 |
| 0x040E4 | PTC511 | Packets Transmitted [256–511 Bytes] Count | STAT | RC | | 700 |
| 0x040E8 | PTC1023 | Packets Transmitted [512–1023 Bytes] Count | STAT | RC | | 701 |
| 0x040EC | PTC1522 | Packets Transmitted [Greater than 1024 Bytes] Count | STAT | RC | | 701 |
| 0x040F0 | MPTC | Multicast Packets Transmitted Count | STAT | RC | | 701 |
| 0x040F4 | BPTC | Broadcast Packets Transmitted Count | STAT | RC | | 701 |
| 0x04010 | MSPDC | MAC short Packet Discard Count | STAT | RC | | 702 |
| 0x04120 | XEC | XSUM Error Count | STAT | RC | | 702 |
| 0x02300+4*n, n=0...31 | RQSMR[n] | Receive Queue Statistic Mapping Registers | DMA-Rx | RW | | 702 |
| 0x02F40 | RXDSTATCTRL | Rx DMA Statistic Counter Control | DMA-Tx | RW | | 703 |
| 0x08600+4*n, n=0...31 / 0x07300+4*n, n=0…7 | TQSM[n] | Transmit Queue Statistic Mapping Registers | DMA-Tx | RW | | 703 |
| 0x01030+0x40*n, n=0...15 | QPRC[n] | Queue Packets Received Count | DMA-Rx | RC | | 704 |
| 0x01430+0x40*n, n=0...15 | QPRDC[n] | Queue Packets Received Drop Count | DMA-Rx | RC | | 704 |
| 0x1034+0x40*n, n=0...15 | QBRC_L[n] | Queue Bytes Received Count Low | DMA-Rx | RC | | 704 |
| 0x1038+0x40*n, n=0...15 | QBRC_H[n] | Queue Bytes Received Count High | DMA-Rx | RC | | 704 |
| 0x08680+0x4*n, n=0...15 / 0x06030+0x40*n, n=0...15 | QPTC | Queue Packets Transmitted Count | DMA-Tx | RC | | 704 |
| 0x08700+0x8*n, n=0...15 | QBTC_L[n] | Queue Bytes Transmitted Count Low | DMA-Tx | RC | | 705 |
| 0x08704+0x8*n, n=0...15 | QBTC_H[n] | Queue Bytes Transmitted Count High | DMA-Tx | RC | | 705 |
| 0x05118 | FCCRC | FC CRC Error Count | Rx-Filter | RC | | 705 |
| 0x0241C | FCOERPDC | FCoE Rx Packets Dropped Count | DMA-Rx | RC | | 706 |

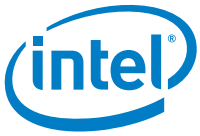


## Table 8-2 Register Summary (Continued)

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| 0x02424 | FCLAST | FC Last Error Count | DMA-Rx | RC | | 706 |
| 0x02428 | FCOEPRC | FCoE Packets Received Count | DMA-Rx | RC | | 706 |
| 0x0242C | FCOEDWRC | FCOE DWord Received Count | DMA-Rx | RC | | 706 |
| 0x08784 | FCOEPTC | FCoE Packets Transmitted Count | DMA-Tx | RC | | 706 |
| 0x08788 | FCOEDWTC | FCoE DWord Transmitted Count | DMA-Tx | RC | | 707 |
| **Wake-Up Control Registers** | | | | | | |
| 0x05800 | WUC | Wake Up Control Register | Rx-Filter | RW | | 708 |
| 0x05808 | WUFC | Wake Up Filter Control Register | Rx-Filter | RW | | 708 |
| 0x5838 | IPAV | IP Address Valid | Rx-Filter | RW | | 709 |
| 0x05840+8*n, n = 0...3 | IP4AT[n] | IPv4 Address Table | Rx-Filter | RW | | 710 |
| 0x05880+4*n, n = 0...3 | IP6AT[n] | IPv6 Address Table | Rx-Filter | RW | | 710 |
| 0x05900 | WUPL | Wake Up Packet Length | Rx-Filter | RO | | 710 |
| 0x05A00+4*n, n=0...31 | WUPM[n] | Wake Up Packet Memory (128 Bytes) | Rx-Filter | RO | | 710 |
| 0x09000 — 0x093FC, 0x09800 — 0x099FC | FHFT | Flexible Host Filter Table registers | Rx-Filter | RW | | 710 |
| **Management Filters Registers** | | | | | | |
| 0x5010 +4*n, n=0...7 | MAVTV[n] | Management VLAN TAG Value | Rx-Filter | RW | | 712 |
| 0x5030+4*n, n=0...7 | MFUTP[n] | Management Flex UDP/TCP Ports | Rx-Filter | RW | | 712 |
| 0x05190+4*n, n=0...3 | METF[n] | Management Ethernet Type Filters | Rx-Filter | RW | | 713 |
| 0x05820 | MANC | Management Control Register | Rx-Filter | RW | | 713 |
| 0x5824 | MFVAL | Manageability Filters Valid | Rx-Filter | RW | | 714 |
| 0x5860 | MANC2H | Management Control To Host Register | Rx-Filter | RW | | 714 |
| 0x5890+4*n, n=0...7 | MDEF[n] | Manageability Decision Filters | Rx-Filter | RW | | 715 |
| 0x05160+4*n, n=0...7 | MDEF_EXT[n] | Manageability Decision Filters | Rx-Filter | RW | | 716 |
| 0x58B0+0x10*m+4*n, m=0...3, n=0...3 | MIPAF | Manageability IP Address Filter | Rx-Filter | RW | | 716 |

## Table 8-2 Register Summary (Continued)

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| 0x5910+8*n, n=0...3 | MMAL[n] | Manageability Ethernet MAC Address Low | Rx-Filter | RW | | 717 |
| 0x5914+8*n, n=0...3 | MMAH[n] | Manageability Ethernet MAC Address High | Rx-Filter | RW | | 717 |
| 0x09400-0x097FC | FTFT | Flexible TCO Filter Table registers | Rx-Filter | RW | | 717 |
| 0x015F14 | LSWFW | LinkSec SW/FW Interface | MNG | RO | | 718 |
| **Time Sync (IEEE 1588) Registers** | | | | | | |
| 0x05188 | TSYNCRXCTL | Rx Time Sync Control Register | Rx-Filter | RW | | 720 |
| 0x051E8 | RXSTMPL | Rx Timestamp Low | Rx-Filter | RO | | 720 |
| 0x051A4 | RXSTMPH | Rx Timestamp High | Rx-Filter | RO | | 720 |
| 0x051A0 | RXSATRL | Rx Timestamp Attributes Low | Rx-Filter | RO | | 721 |
| 0x051A8 | RXSATRH | Rx Timestamp Attributes High | Rx-Filter | RO | | 721 |
| 0x05120 | RXMTRL | Rx Message Type Register Low | Rx-Filter | RW | | 721 |
| 0x08C00 | TSYNCTXCTL | Tx Time Sync Control Register | SEC-Tx | RW | | 721 |
| 0x08C04 | TXSTMPL | Tx Timestamp Value Low | SEC-Tx | RO | | 722 |
| 0x08C08 | TXSTMPH | Tx Timestamp Value High | SEC-Tx | RO | | 722 |
| 0x08C0C | SYSTIML | System Time Register Low | SEC-Tx | RW | | 722 |
| 0x08C10 | SYSTIMH | System Time Register High | SEC-Tx | RW | | 722 |
| 0x08C14 | TIMINCA | Increment Attributes Register | SEC-Tx | RW | | 722 |
| 0x08C18 | TIMADJL | Time Adjustment Offset Register low | SEC-Tx | RW | | 723 |
| 0x08C1C | TIMADJH | Time Adjustment Offset Register High | SEC-Tx | RW | | 723 |
| 0x08C20 | TSAUXC | TimeSync Auxiliary Control Register | SEC-Tx | RW | | 723 |
| 0x08C24 | TRGTTIML0 | Target Time Register 0 Low | SEC-Tx | RW | | 724 |
| 0x08C28 | TRGTTIMH0 | Target Time Register 0 High | SEC-Tx | RW | | 724 |
| 0x08C2C | TRGTTIML1 | Target Time Register 1 Low | SEC-Tx | RW | | 724 |
| 0x08C30 | TRGTTIMH1 | Target Time Register 1 High | SEC-Tx | RW | | 724 |



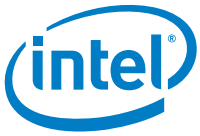
## Table 8-2 Register Summary (Continued)

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| 0x08C3C | AUXSTMPL0 | Auxiliary Time Stamp 0 Register low | SEC-Tx | RO | | 725 |
| 0x08C40 | AUXSTMPH0 | Auxiliary Time Stamp 0 Register high | SEC-Tx | RO | | 725 |
| 0x08C44 | AUXSTMPL1 | Auxiliary Time Stamp 1 Register low | SEC-Tx | RO | | 725 |
| 0x08C48 | AUXSTMPH1 | Auxiliary Time Stamp 1 Register high | SEC-Tx | RO | | 725 |
| **Virtualization PF Registers** | | | | | | |
| 0x051B0 | PFVTCTL | PF Virtual Control Register | Rx-Filter | RW | | 726 |
| 0x04B00+4*n, n=0...63 | PFMailbox[n] | PF Mailbox | Target | RW | | 726 |
| 0x00710+4*n, n=0...3 | PFMBICR[n] | PF Mailbox Interrupt Causes Register | Target | RW1C | | 727 |
| 0x00720+4*n, n=0...1 | PFMBIMR[n] | PF Mailbox Interrupt Mask Register | Target | RW | | 727 |
| 0x00600, 0x001C0 | PFVFLRE[n] | PF VFLR Events Indication | Target | RO | | 727 |
| 0x00700+4*n, n=0...1 | PFVFLREC[n] | PF VFLR Events Clear | Target | W1C | | 728 |
| 0x051E0+4*n, n=0...1 | PFVFRE[n] | PF VF Receive Enable | | RW | | 728 |
| 0x08110+4*n, n=0...1 | PFVFTE[n] | PF VF Transmit Enable | DMA-Tx | RW | | 728 |
| 0x02F04 | PFQDE | PF Queue Drop Enable Register | DMA-Rx | RW | | 728 |
| 0x05180+4 *n, n=0...1 | PFVMTXSW[n] | PF VM Tx Switch Loopback Enable | Rx-Filter | RW | | 729 |
| 0x08200+4*n, n=0...7 | PFVFSPOOF[n] | PF VF Anti Spoof Control | DMA-Tx | RW | | 729 |
| 0x08220 | PFDTXGSWC | PF DMA Tx General Switch Control | DMA-Tx | RW | | 729 |
| 0x08000+4*n, n=0...63 | PFVMVIR[n] | PF VM VLAN Insert Register | DMA-Tx | RW | | 730 |
| 0x0F000+4*n, n=0...63 | PFVML2FLT[n] | PF VM L2 Control Register | Rx-Filter | RW | | 730 |
| 0x0F100+4*n, n=0...63 | PFVLVF[n] | PF VM VLAN Pool Filter | Rx-Filter | RW | | 730 |
| 0x0F200+4*n, n=0...127 | PFVLVFB[n] | PF VM VLAN Pool Filter Bitmap | Rx-Filter | RW | | 731 |
| 0x0F400+4*n, n=0...127 | PFUTA[n] | PF Unicast Table Array | Rx-Filter | RW | | 731 |
| 0x0F600+4*n, n= 0...3 | PFMRCTL[n] | PF Mirror Rule Control | Rx-Filter | RW | | 732 |

**Table 8-2 Register Summary (Continued)**

| Offset / Alias Offset | Abbreviation | Name | Block | RW | Reset Source | Page |
|---|---|---|---|---|---|---|
| 0x0F610+4*n, n= 0...7 | PFMRVLAN[n] | PF Mirror Rule VLAN | Rx-Filter | RW | | 732 |
| 0x0F630+4*n, n= 0...7 | PFMRVM[n] | PF Mirror Rule Pool | Rx-Filter | RW | | 732 |

**Note:** (*) The MAC Manageability Control Register is read only to the host and read/write to manageability.



# 8.2.3 Detailed Register Descriptions — PF

## 8.2.3.1 General Control Registers

### 8.2.3.1.1 Device Control Register — CTRL (0x00000 / 0x00004; RW)

CTRL is also mapped to address 0x00004 to maintain compatibility with previous devices.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 1:0 | 0b | Reserved.<br>Write as 0b for future compatibility. |
| PCIe Master Disable | 2 | 0b | When set, the 82599 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. |
| LRST | 3 | 0b | Link Reset.<br>This bit performs a reset of the MAC, PCS, and auto negotiation functions and the entire Intel® 82599 10 GbE Controller (software reset) resulting in a state nearly approximating the state following a power-up reset or internal PCIe reset, except for the system PCI configuration. Normally 0b, writing 1b initiates the reset. This bit is self-clearing. Also referred to as MAC reset. |
| Reserved | 25:4 | 0b | Reserved. |
| RST | 26 | 0b | Device Reset.<br>This bit performs a complete reset of the 82599, resulting in a state nearly approximating the state following a power-up reset or internal PCIe reset, except for the system PCI configuration. Normally 0b, writing 1b initiates the reset. This bit is self-clearing. Also referred to as a software reset or global reset. |
| Reserved | 31:27 | 0x0 | Reserved. |

LRST and RST can be used to globally reset the entire Intel® 82599 10 GbE Controller. This register is provided primarily as a last-ditch software mechanism to recover from an indeterminate or suspected hung hardware state. Most registers (receive, transmit, interrupt, statistics, etc.) and state machines are set to their power-on reset values, approximating the state following a power-on or PCI reset. However, PCIe Configuration registers are not reset, thereby leaving the 82599 mapped into system memory space and accessible by a software device driver.

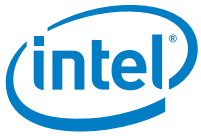
To ensure that a global device reset has fully completed and that the 82599 responds to subsequent accesses, programmers must wait approximately 1 ms after setting before attempting to check if the bit has cleared or to access (read or write) any other device register.

### 8.2.3.1.2 Device Status Register — STATUS (0x00008; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 1:0 | 0b | Reserved. |
| LAN ID | 3:2 | 0b | LAN ID.<br>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 | 0b | Reserved. |
| LinkUp | 7 | 0b | Linkup Status Indication.<br>This bit is Read/Write, and is useful for IOV mode. The PF software driver sets it according to Links register and PHY state. It is reflected in the VFSTATUS register indicating linkup to the VF drivers. |
| Reserved | 9:8 | 0b | Reserved. |
| Num VFs | 17:10 | 0x0 | The *Num VFs* field reflects the value of the Num VFs in the IOV capability structure<br>*Note:* Bit 17 is always 0b. |
| IOV Active | 18 | 0b | The *IO Active* bit, reflects the value of the *VF Enable (VFE)* bit in the IOV Control/ Status register. |
| PCIe Master Enable Status | 19 | 1b | This is a status bit of the appropriate CTRL.PCIe *Master Disable* bit.<br>0b = Associated LAN function does not issue any master request and all previously issued requests are complete.<br>1b = Associated LAN function can issue master requests. |
| Reserved | 31:20 | 0b | Reserved.<br>Reads as 0b. |

### 8.2.3.1.3 Extended Device Control Register — CTRL_EXT (0x00018; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 13:0 | 0x0 | Reserved. |
| PFRSTD (SC) | 14 | 0b | PF Reset Done.<br>When set, the RSTI bit in all the VFMailbox registers are cleared and the RSTD bit in all the VFMailbox regs is set. |
| Reserved | 15 | 0b | Reserved. |
| NS_DIS | 16 | 0b | No Snoop Disable.<br>When set to 1b, the 82599 does not set the no snoop attribute in any PCIe packet, independent of PCIe configuration and the setting of individual no snoop enable bits. When set to 0b, behavior of no snoop is determined by PCIe configuration and the setting of individual no snoop enable bits.<br>*Note:* If legacy descriptors are used, this bit should be set to 1b.<br>This bit must be set during Rx flow initialization for proper device operation. |



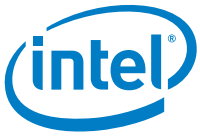
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RO_DIS | 17 | 0b | Relaxed Ordering Disable.<br>When set to 1b, the device does not request any relaxed ordering transactions. When this bit is cleared and the *Enable Relaxed Ordering* bit in the Device Control register is set, the device requests relaxed ordering transactions per queues as configured in the DCA_RXCTRL[n] and DCA_TXCTRL[n] registers. |
| Reserved | 25:18 | 0b | Reserved. |
| Extended VLAN | 26 | 0b | Extended VLAN.<br>When set, all incoming Rx packets are expected to have at least one VLAN with the Ether type as defined in EXVET register. The packets can have an inner-VLAN that should be used for all filtering purposes. All Tx packets are expected to have at least one VLAN added to them by the host. In the case of an additional VLAN request (VLE), the inner-VLAN is added by the hardware after the outer-VLAN is added by the host. This bit should only be reset by a PCIe reset and should only be changed while Tx and Rx processes are stopped.<br>The exception to this rule are MAC control packets such as flow control, 802.1x, LACP, etc. that never carry a VLAN tag of any type. |
| Reserved | 27 | 0b | Reserved. |
| DRV_LOAD | 28 | 0b | Driver loaded and the corresponding network interface is enabled.<br>This bit should be set by the software device driver after it was loaded and cleared when it unloads or at PCIe soft reset. The Manageability Controller (MC) loads this bit as an indication that the driver successfully loaded to it. |
| Reserved | 31:29 | 0b | Reserved. |

### 8.2.3.1.4 Extended SDP Control — ESDP (0x00020; RW)

This register is initialized only at LAN Power Good preserving the SDP states across software and PCIe resets. Some specific I/O pins are initialized in other resets in native mode as expected for the specific behavior and described explicitly as follows.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SDP0_DATA | 0 | 0b[1] | SDP0 Data Value.<br>Used to read (write) a value of the software-controlled I/O pin SDP0. If SDP0 is configured as an output (SDP0_IODIR = 1b), this bit controls the value driven on the pin. If SDP0 is configured as an input, all reads return the current value of the pin. |
| SDP1_DATA | 1 | 0b[1] | SDP1 Data Value.<br>Used to read (write) a value of the software-controlled I/O pin SDP1. If SDP1 is configured as an output (SDP1_IODIR = 1b), this bit controls the value driven on the pin. If SDP1 is configured as an input, all reads return the current value of the pin. |
| SDP2_DATA | 2 | 0b[1] | SDP2 Data Value.<br>Used to read (write) a value of software-controlled I/O pin SDP2. If SDP2 is configured as an output (SDP2_IODIR = 1b), this bit controls the value driven on the pin. If SDP2 is configured as an input, all reads return the current value of the pin. |

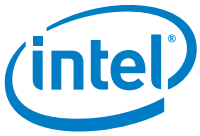
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SDP3_DATA | 3 | 0b[1] | SDP3 Data Value.<br>Used to read (write) a value of the software-controlled I/O pin SDP3. If SDP3 is configured as an output (SDP3_IODIR = 1b), this bit controls the value driven on the pin. If SDP3 is configured as an input, all reads return the current value of the pin. |
| SDP4_DATA[2] | 4 | 0b | SDP4 Data Value.<br>Used to read (write) a value of the software-controlled I/O pin SDP4. If SDP4 is configured as an output (SDP4_IODIR = 1b), this bit controls the value driven on the pin. If SDP4 is configured as an input, all reads return the current value of the pin. |
| SDP5_DATA | 5 | 0b | SDP5 Data Value.<br>Used to read (write) a value of the software-controlled I/O pin SDP5. If SDP5 is configured as an output (SDP5_IODIR = 1b), this bit controls the value driven on the pin. If SDP5 is configured as an input, all reads return the current value of the pin. |
| SDP6_DATA | 6 | 0b | SDP6 Data Value.<br>Used to read (write) a value of the software-controlled I/O pin SDP6. If SDP6 is configured as an output (SDP6_IODIR = 1b), this bit controls the value driven on the pin. If SDP6 is configured as an input, all reads return the current value of the pin. |
| SDP7_DATA | 7 | 0b | SDP7 Data Value.<br>Used to read (write) a value of the software-controlled I/O pin SDP7. If SDP7 is configured as an output (SDP7_IODIR = 1b), this bit controls the value driven on the pin. If SDP7 is configured as an input, all reads return the current value of the pin. |
| SDP0_IODIR | 8 | 0b[1] | SDP0 Pin Directionality.<br>Controls whether or not software-controlled pin SDP0 is configured as an input or output.<br>0b = Input<br>1b = Output |
| SDP1_IODIR | 9 | 0b[1] | SDP1 Pin Directionality.<br>Controls whether or not software-controlled pin SDP1 is configured as an input or output.<br>0b = Input<br>1b = Output |
| SDP2_IODIR | 10 | 0b[1] | SDP2 Pin Directionality. Controls whether or not software-controlled pin SDP2 is configured as an input or output.<br>0b = Input<br>1b = Output |
| SDP3_IODIR | 11 | 0b[1] | SDP3 Pin Directionality. Controls whether or not software-controlled pin SDP3 is configured as an input or output.<br>0b = Input<br>1b = Output |
| SDP4_IODIR[2] | 12 | 0b | SDP4 Pin Directionality. Controls whether or not software-controlled pin SDP4 is configured as an input or output.<br>0b = Input<br>1b = Output |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SDP5_IODIR | 13 | 0b | SDP5 Pin Directionality. Controls whether or not software-controlled pin SDP5 is configured as an input or output.<br>0b = Input<br>1b = Output |
| SDP6_IODIR | 14 | 0b | SDP6 Pin Directionality. Controls whether or not software-controlled pin SDP6 is configured as an input or output.<br>0b = Input<br>1b = Output |
| SDP7_IODIR | 15 | 0b | SDP7 Pin Directionality. Controls whether or not software-controlled pin SDP7 is configured as an input or output.<br>0b = Input<br>1b = Output |
| SDP0_NATIVE | 16 | 0b | SDP0 Operating Mode.<br>0b = Generic software controlled I/O by SDP0_DATA and SDP0_IODIR.<br>1b = Reserved. |
| SDP1_NATIVE | 17 | 0b[1] | SDP1 Operating Mode.<br>0b = Generic software controlled I/O by SDP1_DATA and SDP1_IODIR.<br>1b = Native mode operation (connected to hardware function). In this mode, the SDP1_IODIR must be set to 1b. |
| SDP2_NATIVE | 18 | 0b | SDP2 operating mode.<br>0b = Generic software controlled IO by SDP2_DATA and SDP2_IODIR.<br>1b = Native mode operation (Connected to hardware function). In this mode pin functions as defined by the SDP2_TSync_TT1 bit |
| SDP3_NATIVE | 19 | 0b | SDP3 Operating Mode.<br>0b = Generic software controlled I/O by SDP3_DATA and SDP3_IODIR.<br>1b = Native mode operation (connected to hardware function). In this mode pin functions as defined by the SDP3_TSync_TT0 bit. |
| SDP4_NATIVE[2] | 20 | 0b | SDP4 Operating Mode.<br>0b = Generic software controlled I/O by SDP4_DATA and SDP4_IODIR.<br>1b = Native mode operation (connected to hardware function). Drives optical module reset according to functionality defined by the SDP4_Function bit. |
| SDP5_NATIVE | 21 | 0b | SDP5 Operating Mode.<br>0b = Generic software controlled I/O by SDP5_DATA and SDP5_IODIR.<br>1b = Native mode operation (connected to hardware function). Drives optical module transmit disable according to functionality defined by the SDP5_Function bit. |
| SDP6_NATIVE | 22 | 0b | SDP6 Operating Mode.<br>0b = Generic software controlled I/O by SDP6_DATA and SDP6_IODIR.<br>1b = Native mode operation (connected to hardware function). In this mode, pin functions as defined by the SDP6_TSync_TT1 bit. |
| SDP7_NATIVE | 23 | 0b | SDP7 Operating Mode.<br>0b = Generic software controlled I/O by SDP7_DATA and SDP7_IODIR.<br>1b = Native mode operation (connected to hardware function). In this mode, pin functions as defined by the SDP7_TSync_TT0 bit. |
| Reserved | 25:24 | 0 | Reserved. |

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SDP2_TSync_TT1 | 26 | 0b | SDP2 Native Mode Functionality (SDP2_NATIVE = 1).<br>0b = TS0 functionality. Samples IEEE 1588 time stamp into Auxiliary Time Stamp 0 register on level change of SDP2 signal (For TS0 functionality, SDP2_IODIR should be configured as input).<br>1b = TT1 functionality. Asserts SDP2 to 1 when IEEE 1588 time stamp equals Target Time register 1 (For TT1 functionality, SDP2_IODIR should be configured as output). |
| SDP3_TSync_TT0 | 27 | 0b | SDP3 Native Mode Functionality (SDP3_NATIVE = 1).<br>0b = TS1 functionality. Samples IEEE 1588 time stamp into Auxiliary Time Stamp 1 register on level change of SDP3 signal (For TS1 functionality, SDP3_IODIR should be configured as input).<br>1b = TT0 functionality. Asserts SDP3 to 1b when IEEE 1588 time stamp equals Target Time register 0 (For TT0 functionality, SDP3_IODIR should be configured as output). |
| SDP4_Function[2] | 28 | 0b | SDP4 Native Mode Functionality (SDP4_NATIVE = 1).<br>0b = Pin functionality is driven by software (SDP4_data bit) except when the MAC is reset or when entering D3 power state when management functionality is disabled. In the previous case SDP4 pin moves to tri-state (by resetting SDP4_IODIR bit) and optical module is reset by placing an appropriate external pull-up or pull-down resistor on the SDP4 pin.<br>1b = SDP4 pin is driven high when the MAC is reset or powered down (D3 state).<br>SDP4_IODIR should be configured as output for this functionality. |
| SDP5_Function | 29 | 0b | SDP5 Native Mode Functionality (SDP5_NATIVE = 1).<br>0b = Pin functionality is driven by software (SDP5_data bit) except when the MAC is reset or when entering D3 power state when management functionality is disabled. In the previous case, SDP5 pin moves to tri-state (by resetting SDP5_IODIR bit) and optical module transmission is disabled by placing an appropriate external pull-up or pull-down resistor on the SDP5 pin.<br>1b = SDP5 pin is driven high when the MAC is reset or powered down (D3 state).<br>SDP5_IODIR should be configured as output for this functionality. |
| SDP6_TSync_TT1 | 30 | 0b | SDP6 Native Mode Functionality (SDP6_NATIVE = 1).<br>0b = CLK0 functionality. Drives a reference clock with the frequency defined in the Frequency Out 0 Control register (For CLK0 functionality, SDP6_IODIR should be configured as output).<br>1b = TT1 functionality. Asserts SDP6 to 1b when IEEE 1588 time stamp equals Target Time register 1 (For TT1 functionality, SDP6_IODIR should be configured as output). |
| SDP7_TSync_TT0 | 31 | 0b | SDP7 Native Mode Functionality (SDP7_NATIVE = 1).<br>0b = CLK1 functionality. Drives a reference clock with the frequency defined in the Frequency Out 1 Control register (for CLK1 functionality, SDP7_IODIR should be configured as output).<br>1b = TT0 functionality. Asserts SDP7 to 1b when IEEE 1588 time stamp equals Target Time register 1 (For TT0 functionality, SDP7_IODIR should be configured as output). |

1. Initial value can be configured using the EEPROM.
2. SDP4 in port 0 is a dedicated input pin for Security enablement. See Section 4.6.12.



### 8.2.3.1.5 I2C Control — I2CCTL (0x00028; RW)

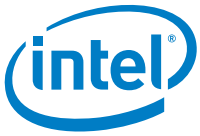
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| I2C_CLK_IN | 0 | 0b | I2C_CLK In Value.<br>Provides the value of I2C_CLK (input from external PAD). This bit is RO. |
| I2C_CLK_OUT | 1 | 1b | I2C_CLK Out Value.<br>Used to drive the value of I2C_CLK (output to PAD). |
| I2C_DATA_IN | 2 | 0b | I2C_DATA In Value.<br>Provides the value of I2C_DATA (input from external PAD). This bit is RO. |
| I2C_DATA_OUT | 3 | 1b | I2C_DATA Out Value.<br>Used to drive the value of I2C_DATA (output to PAD). |
| Reserved | 31:4 | 0x0 | Reserved. |

### 8.2.3.1.6 LED Control — LEDCTL (0x00200; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LED0_MODE | 3:0 | 0x0[1] | LED0 Mode.<br>This field specifies the control source for the LED0 output. An initial value of 0000b selects the LINK_UP indication. |
| Reserved | 4 | 0b[1] | Reserved. |
| GLOBAL_BLINK_MODE | 5 | 0b[1] | GLOBAL Blink Mode.<br>This field specifies the blink mode of all LEDs.<br>0b = Blink at 200 ms on and 200 ms off.<br>1b = Blink at 83 ms on and 83 ms off. |
| LED0_IVRT | 6 | 0b[1] | LED0 Invert.<br>This field specifies the polarity/inversion of the LED source prior to output or blink control. By default the output drives the cathode of the LED so when the LED output is 0b the LED is on.<br>0b = LED output is active low.<br>1b = LED output is active high. |
| LED0_BLINK | 7 | 0b[1] | LED0 Blink.<br>This field specifies whether or not to apply blink logic to the (inverted) LED control source prior to the LED output.<br>0b = Do not blink LED output.<br>1b = Blink LED output. |
| LED1_MODE | 11:8 | 0x1[1] | LED1 Mode.<br>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 | 0b[1] | Reserved. |

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LED1_IVRT | 14 | 0b[1] | LED1 Invert.<br>This field specifies the polarity/inversion of the LED source prior to output or blink control. By default the output drives the cathode of the LED so when the LED output is 0b the LED is on.<br>0b = LED output is active low.<br>1b = LED output is active high. |
| LED1_BLINK | 15 | 1b[1] | LED1 Blink.<br>This field specifies whether or not to apply blink logic to the (inverted) LED control source prior to the LED output.<br>0b = Do not blink LED output.<br>1b = Blink LED output. |
| LED2_MODE | 19:16 | 0x4[1] | LED2 Mode.<br>This field specifies the control source for the LED0 output. An initial value of 0100 selects LINK/ACTIVITY indication. |
| Reserved | 21:20 | 0[1] | Reserved |
| LED2_IVRT | 22 | 0[1] | LED2 Invert.<br>This field specifies the polarity/inversion of the LED source prior to output or blink control. By default the output drives the cathode of the LED so when the LED output is 0b the LED is on.<br>0b = LED output is active low.<br>1b = LED output is active high. |
| LED2_BLINK | 23 | 0[1] | LED2 Blink.<br>This field specifies whether or not to apply blink logic to the (inverted) LED control source prior to the LED output.<br>0b = Do not blink LED output.<br>1b = Blink LED output. |
| LED3_MODE | 27:24 | 0x5[1] | LED3 Mode.<br>This field specifies the control source for the LED0 output. An initial value of 0101b selects the 1 Gb/s link indication. |
| Reserved | 29:28 | 0b[1] | Reserved. |
| LED3_IVRT | 30 | 0b[1] | LED3 Invert.<br>This field specifies the polarity/inversion of the LED source prior to output or blink control. By default the output drives the cathode of the LED so when the LED output is 0b the LED is on.<br>0b = LED output is active low.<br>1b = LED output is active high. |
| LED3_BLINK | 31 | 0b[1] | LED3 Blink.<br>This field specifies whether or not to apply blink logic to the (inverted) LED control source prior to the LED output.<br>0b = Do not blink LED output.<br>1b = Blink LED output. |

1. These bits are read from the EEPROM.



The following mapping is used to specify the LED control source (MODE) for each LED output:

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

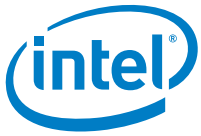
### 8.2.3.1.7 Extended VLAN Ether Type — EXVET (0x05078; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 15:0 | 0x0 | Reserved. |
| VET EXT | 31:16 | 0x8100 | Outer-VLAN Ether Type (VLAN Tag Protocol Identifier - TPID).<br>*Note:* This field appears in little endian (MS byte first on the wire). |

## 8.2.3.2 EEPROM/Flash Registers

### 8.2.3.2.1 EEPROM/Flash Control Register — EEC (0x10010; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| EE_SK | 0 | 0b | Clock input to the EEPROM.<br>When EE_GNT is set to 1b, the EE_SK output signal is mapped to this bit and provides the serial clock input to the EEPROM. Software clocks the EEPROM via toggling this bit with successive writes. |
| EE_CS | 1 | 0b | Chip select input to the EEPROM.<br>When EE_GNT is set to 1b, the EE_CS output signal is mapped to the chip select of the EEPROM device. Software enables the EEPROM by writing a 0b to this bit. |
| EE_DI | 2 | 0b | Data input to the EEPROM.<br>When EE_GNT is set to 1b, the EE_DI output signal is mapped directly to this bit. Software provides data input to the EEPROM via writes to this bit. |
| EE_DO (RO field) | 3 | X | Data output bit from the EEPROM.<br>The EE_DO input signal is mapped directly to this bit in the register and contains the EEPROM data output. This bit is read-only from a software perspective; writes to this bit have no effect. |
| FWE | 5:4 | 01b | Flash Write Enable Control.<br>These two bits control whether or not writes to the Flash are allowed.<br>00b = Flash erase (along with bit 31 in the FLA register).<br>01b = Flash writes disabled.<br>10b = Flash writes enabled.<br>11b = Not allowed. |
| EE_REQ | 6 | 0b | Request EEPROM Access.<br>Software must write a 1b to this bit to get direct EEPROM access. It has access when EE_GNT is set to 1b. When software completes the access, it must then write a 0b. |
| EE_GNT (RO field) | 7 | 0b | Grant EEPROM Access.<br>When this bit is set to 1b, software can access the EEPROM using the EE_SK, EE_CS, EE_DI, and EE_DO bits. |
| EE_PRES (RO field) | 8 | (see desc.) | EEPROM Present.<br>Setting this bit to 1b indicates that an EEPROM is present and has the correct signature field. This bit is read-only. |
| Auto_RD (RO field) | 9 | 0b | EEPROM Auto-Read Done.<br>When set to 1b, this bit indicates that the auto-read by hardware from the EEPROM is done. This bit is also set when the EEPROM is not present or when its signature field is not valid. |
| Reserved | 10 | 1b | Reserved. |
| EE_Size (RO field) | 14:11 | 0010b[1] | EEPROM Size.<br>This field defines the size of the EEPROM (see Table 8-3). |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PCI _ANA_done (RO field) | 15 | 0b | PCIe Analog Done.<br>When set to 1b, indicates that the PCIe analog section read from EEPROM is done. This bit is cleared when auto-read starts. This bit is also set when the EEPROM is not present or when its signature field is not valid. |
| PCI _Core_done (RO field) | 16 | 0b | PCIe Core Done.<br>When set to 1b, indicates that the Core analog section read from EEPROM is done. This bit is cleared when auto-read starts. This bit is also set when the EEPROM is not present or when its signature field is not valid.<br>*Note:* This bit returns the relevant done indication for the function that reads the register. |
| PCI _genarl_done (RO field) | 17 | 0b | PCIe General Done.<br>When set to 1b, indicates that the PCIe general section read from the EEPROM is done. This bit is cleared when auto-read starts. This bit is also set when the EEPROM is not present or when its signature field is not valid. |
| PCI_FUNC_DONE (RO field) | 18 | 0b | PCIe Function Done.<br>When set to 1b, indicates that the PCIe function section read from EEPROM is done. This bit is cleared when auto-read starts. This bit is also set when the EEPROM is not present or when its signature field is not valid.<br>*Note:* This bit returns the relevant done indication for the function that reads the register. |
| CORE_DONE (RO field) | 19 | 0b | Core Done.<br>When set to 1b, indicates that the Core analog section read from the EEPROM is done. This bit is cleared when auto-read starts. This bit is also set when the EEPROM is not present or when its signature field is not valid.<br>*Note:* This bit returns the relevant done indication for the function that reads the register. |
| CORE_CSR_DONE (RO field) | 20 | 0b | Core CSR Done.<br>When set to 1b, indicates that the Core CSR section read from the EEPROM is done. This bit is cleared when auto-read starts. This bit is also set when the EEPROM is not present or when its signature field is not valid.<br>*Note:* This bit returns the relevant done indication for the function that reads the register. |
| MAC_DONE (RO field) | 21 | 0b | MAC Done.<br>When set to 1b, indicates that the MAC section read from the EEPROM is done. This bit is cleared when auto-read starts. This bit is also set when the EEPROM is not present or when its signature field is not valid.<br>*Note:* This bit returns the relevant done indication for the function that reads the register. |
| Reserved | 31:22 | 0x0 | Reserved.<br>Reads as 0b. |

1. These bits are read from the EEPROM.

**Table 8-3 EEPROM Sizes (Bits 14:11)**

| Field Value | EEPROM Size | EEPROM Address Size |
|---|---|---|
| 0100b | 16 Kb | 2 bytes |
| 0101b | 32 Kb | 2 bytes |
| 0110b | 64 Kb | 2 bytes |
| 0111b | 128 Kb | 2 bytes |
| 1000b | 256 Kb | 2 bytes |
| 1001b:1111b | Reserved | Reserved |

This register provides software-direct access to the EEPROM. Software can control the EEPROM by successive writes to this register. Data and address information is clocked into the EEPROM by software toggling the EESK bit (2) of this register. Data output from the EEPROM is latched into bit 3 of this register via the internal 62.5 MHz clock and can be accessed by software via reads of this register.

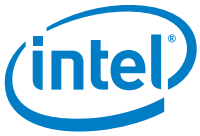
**Note:** Attempts to write to the Flash device when writes are disabled (FWE = 01b) should not be attempted. Behavior after such an operation is undefined, and might result in component and/or system hangs.

### 8.2.3.2.2 EEPROM Read Register — EERD (0x10014; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| START | 0 | 0b | Start Read.<br>Writing a 1b to this bit causes the EEPROM to read a 16-bit word at the address stored in the EE_ADDR field and then stores the result in the EE_DATA field. This bit is self-clearing. |
| DONE | 1 | 0b | Read Done.<br>Set this bit to 1b when the EEPROM read completes.<br>Set this bit to 0b when the EEPROM read is in progress.<br>*Note:* Writes by software are ignored. |
| ADDR | 15:2 | 0x0 | Read Address.<br>This field is written by software along with *Start Read* to indicate that the address of the word to read. |
| DATA | 31:16 | 0x0 | Read Data.<br>Data returned from the EEPROM read. |

This register is used by software to cause the 82599 to read individual words in the EEPROM. To read a word, software writes the address to the *Read Address* field and simultaneously writes a 1b to the Start Read field. The 82599 reads the word from the EEPROM and places it in the *Read Data* field, setting the *Read Done* field to 1b. Software can poll this register, looking for a 1b in the *Read Done* field and then using the value in the *Read Data* field.

When this register is used to read a word from the EEPROM, that word is not written to any of the 82599's internal registers even if it is normally a hardware-accessed word.



### 8.2.3.2.3 Flash Access Register — FLA (0x1001C; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| FL_SCK | 0 | 0b | Clock input to the Flash.<br>When FL_GNT is set to 1b, the FL_SCK output signal is mapped to this bit and provides the serial clock input to the Flash. Software clocks the Flash via toggling this bit with successive writes. |
| FL_CE | 1 | 0b | Chip select input to the Flash.<br>When FL_GNT is set to 1b, the FL_CE output signal is mapped to the chip select of the Flash device. Software enables the Flash by writing a 0b to this bit. |
| FL_SI | 2 | 0b | Data input to the Flash.<br>When FL_GNT is set to 1b, the FL_SI output signal is mapped directly to this bit. Software provides data input to the Flash via writes to this bit. |
| FL_SO | 3 | X | Data output bit from the Flash.<br>The FL_SO input signal is mapped directly to this bit in the register and contains the Flash serial data output. This bit is read-only from a software perspective.<br>*Note:* Writes to this bit have no effect. |
| FL_REQ | 4 | 0b | Request Flash Access.<br>Software must write a 1b to this bit to get direct Flash access. It has access when FL_GNT is set to 1b. When software completes the access, it must then write a 0b. |
| FL_GNT | 5 | 0b | Grant Flash Access.<br>When this bit is set to 1b, software can access the Flash using the FL_SCK, FL_CE, FL_SI, and FL_SO bits. |
| Reserved | 29:6 | 0b | Reserved.<br>Reads as 0b. |
| FL_BUSY | 30 | 0b | Flash Busy.<br>This bit is set to 1b while a write or an erase to the Flash is in progress, While this bit is cleared (reads as 0b), software can access to write a new byte to the Flash.<br>*Note:* This bit is read-only from a software perspective. |
| FL_ER | 31 | 0b | Flash Erase Command.<br>This command is sent to the Flash only if bits 5:4 of register EEC are also set to 00b. This bit is auto-cleared and reads as 0b. |

This register provides software direct access to the Flash. Software can control the Flash by successive writes to this register. Data and address information is clocked into the EEPROM by software toggling FL_SCK in this register. Data output from the Flash is latched into bit 3 of this register via the internal 125 MHz clock and can be accessed by software via reads of this register.

**Note:** In the 82599, the FLA register is only reset at LAN_PWR_GOOD as opposed to legacy devices at software reset.

### 8.2.3.2.4 Manageability EEPROM Control Register — EEMNGCTL (0x10110; RW)

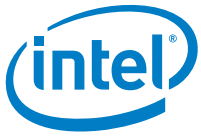
**Note:** This register can be read/write by manageability firmware and is read-only to host software.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| ADDR | 14:0 | 0x0 | Address.<br>This field is written by manageability along with *Start* bit and the *Write* bit to indicate which EEPROM address to read or write. |
| START | 15 | 0b | Start.<br>Writing a 1b to this bit causes the EEPROM to start the read or write operation according to the write bit. This bit is self cleared by hardware. |
| WRITE | 16 | 0b | Write.<br>This bit signals the EEPROM if the current operation is read or write.<br>0b = Read.<br>1b = Write. |
| EEBUSY | 17 | 0b | EPROM Busy.<br>This bit indicates that the EEPROM is busy processing an EEPROM transaction and should not be accessed. |
| Reserved | 30:18 | 0x0 | Reserved. |
| DONE | 31 | 1b | Transaction Done.<br>This bit is cleared after the *Start* bit and *Write* bit are set by manageability and is set back again when the EEPROM write or read transaction completes. |

### 8.2.3.2.5 Manageability EEPROM Read/Write Data — EEMNGDATA (0x10114; RW)

**Note:** This register can be read/write by manageability firmware and is read-only to host software.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| WRDATA | 15:0 | 0x0 | Write Data.<br>Data to be written to the EEPROM. |
| RDDATA | 31:16 | X | Read Data.<br>Data returned from the EEPROM read.<br>*Note:* This field is read only. |



### 8.2.3.2.6 Manageability Flash Control Register — FLMNGCTL (0x10118; RW)

**Note:** This register can be read/write by manageability firmware and is read-only to host software.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| ADDR | 23:0 | 0x0 | Address.<br>This field is written by manageability along with CMD and CMDV to indicate which Flash address to read or write. |
| CMD | 25:24 | 00b | Command.<br>Indicates which command should be executed. Valid only when the *CMDV* bit is set.<br>00b = Read command.<br>01b = Write command (single byte).<br>10b = Sector erase. *Note:* Sector erase is applicable only for Atmel Flashes.<br>11b = Erase. |
| CMDV | 26 | 0b | Command Valid.<br>When set, indicates that the manageability firmware issues a new command and is cleared by hardware at the end of the command. |
| FLBUSY | 27 | 0b | Flash Busy.<br>This bit indicates that the Flash is busy processing a Flash transaction and should not be accessed. |
| Reserved | 29:28 | 00b | Reserved. |
| DONE | 30 | 1b | Read Done.<br>This bit is cleared by firmware when it sets the *CMDV* bit. It is set by hardware for each Dword read that completes. This bit is read/clear by hardware enabling the multiple Dword read flow. |
| WRDONE | 31 | 1b | Global Done.<br>This bit clears after the *CMDV* bit is set by manageability and is set back again after all Flash transactions complete. For example, the Flash device finished reading all the requested read or other accesses (write and erase). |

### 8.2.3.2.7 Manageability Flash Read Data — FLMNGDATA (0x1011C; RW)

**Note:** This register can be read/write by manageability firmware and is read-only to host software.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| DATA | 31:0 | 0x0 | Read/Write Data<br>On a read transaction, this register contains the data returned from the Flash read.<br>On write transactions, bits 7:0 are written to the Flash. |

#### 8.2.3.2.8 Flash Opcode Register — FLOP (0x01013C; RW)

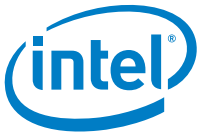
This register enables the host or firmware to define the op-code used in order to erase a sector of the Flash or erase the entire Flash. This register is reset only at power on or during LAN_PWR_GOOD assertion.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SERASE | 7:0 | 0x52 | Flash Block Erase Instruction.<br>The op-code for the Flash block erase instruction and is relevant only to Flash access by manageability. |
| DERASE | 15:8 | 0x62 | Flash Device Erase Instruction.<br>The op-code for the Flash erase instruction. |
| Reserved | 31:16 | 0x0 | Reserved. |

#### 8.2.3.2.9 General Receive Control — GRC (0x10200; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MNG_EN | 0 | 1b[1] | Manageability Enable.<br>This read-only bit indicates whether or not manageability functionality is enabled. |
| APME | 1 | 0b[1] | Advance Power Management Enable.<br>If set to 1b, APM wake up is enabled. When APM wake up is enabled and The 82599 receives a matching magic packet, it sets the *PME_Status* bit in the Power Management Control/Status register (PMCSR) and asserts the PE_WAKE_N pin. It is a single read/write bit in a single register, but has two values depending on the function that accesses the register. |
| Reserved | 31:2 | 0x0 | Reserved. |

1. Loaded from the EEPROM.



## 8.2.3.3 Flow Control Registers

### 8.2.3.3.1 Priority Flow Control Type Opcode — PFCTOP (0x0431C / 0x03008; RW)

This register is also mapped to address 0x0431C to maintain compatibility with the 82598.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| FCT | 15:0 | 0x8808 | Priority Flow Control EtherType.<br>*Note:* This field appears in little endian (MS byte first on the wire). |
| FCOP | 31:16 | 0x0101 | Priority Flow Control Opcode.<br>*Note:* This field appears in big endian (LS byte first on the wire). |

This register contains the *Type* and *Opcode* fields that are matched against a recognized priority flow control packet.

### 8.2.3.3.2 Flow Control Transmit Timer Value n — FCTTVn (0x03200 + 4*n, n=0...3; RW)

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.).

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TTV(2n) | 15:0 | 0x0 | Transmit Timer Value 2n.<br>Timer value included in XOFF frames as Timer (2n). The same value shall be set to User Priorities attached to the same TC, as defined in RTTUP2TC register. For legacy 802.3X flow control packets, TTV0 is the only timer that is used. |
| TTV(2n+1) | 31:16 | 0x0 | Transmit Timer Value 2n+1.<br>Timer value included in XOFF frames as Timer 2n+1. The same value shall be set to User Priorities attached to the same TC, as defined in RTTUP2TC register. |

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).

**Note:** The 82599 uses a fixed slot time value of 64 byte times.

#### 8.2.3.3.3 Flow Control Receive Threshold Low — FCRTL[n] (0x03220 + 4*n, n=0...7; RW)

Each 32-bit register (n=0… 7) refers to a different receive packet buffer.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 4:0 | 0x0 | Reserved. |
| RTL | 18:5 | 0x0 | Receive Threshold Low n.<br>Receive packet buffer n FIFO low water mark for flow control transmission (32 bytes granularity). |
| Reserved | 30:19 | 0x0 | Reserved. |
| XONE | 31 | 0b | XON Enable n. Per the receive packet buffer XON enable.<br>0b = Disabled.<br>1b = Enabled. |

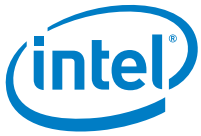
This register contains the receive threshold used to determine when to send an XON packet and counts in units of bytes. The lower four bits must be programmed to 0x0 (16-byte granularity). Software must set XONE to enable the transmission of XON frames. Each time incoming packets cross the receive high threshold (become more full), and then crosses the receive low threshold, with XONE enabled (1b), hardware transmits an XON frame.

#### 8.2.3.3.4 Flow Control Receive Threshold High — FCRTH[n] (0x03260 + 4*n, n=0...7; RW)

Each 32-bit register (n=0…7) refers to a different receive packet buffer.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 4:0 | 0x0 | Reserved. |
| RTH | 18:5 | 0x0 | Receive Threshold High n.<br>Receive packet buffer n FIFO high water mark for flow control transmission (32 bytes granularity). |
| Reserved | 30:19 | 0x0 | Reserved. |
| FCEN | 31 | 0b | Transmit flow control enable for packet buffer n. |

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 four bits must be programmed to 0x0 (16-byte granularity). Each time the receive FIFO reaches the fullness indicated by RTH, hardware transmits a pause frame if the transmission of flow control frames is enabled.



#### 8.2.3.3.5 Flow Control Refresh Threshold Value — FCRTV (0x032A0; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| FC_refresh_th | 15:0 | 0x0 | Flow Control Refresh Threshold.<br>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<br>*Note:* The FC_refresh_th must be smaller than TTV of the TC and larger than the max packet size in the TC + FC packet size + link latency and Tx latency and Rx latency in 64 byte units. |
| Reserved | 31:16 | 0x0 | Reserved. |

#### 8.2.3.3.6 Transmit Flow Control Status — TFCS (0x0CE00; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TC_XON | 7:0 | 0xFF | Set if flow control is in XON state.<br>If in link flow control mode, only bit 0 should be used. In case of priority flow control mode, each bit represents a TC. |
| Reserved | 31:9 | 0x0 | Reserved. |

#### 8.2.3.3.7 Flow Control Configuration — FCCFG (0x03D00; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 2:0 | 0x0 | Reserved. |
| TFCE | 4:3 | 0x0 | Transmit Flow Control Enable.<br>These bits Indicate that the 82599 transmits flow control packets (XON/XOFF frames) based on receive fullness. If auto-negotiation is enabled, then this bit should be set by software to the negotiated flow control value.<br>00b = Transmit flow control disabled.<br>01b = Link flow control enabled.<br>10b = Priority flow control enabled.<br>11b = Reserved.<br>*Note:* Priority flow control should be enabled in DCB mode only. |
| Reserved | 31:5 | 0x0 | Reserved. |

## 8.2.3.4 PCIe Registers

### 8.2.3.4.1 PCIe Control Register — GCR (0x11000; RW)

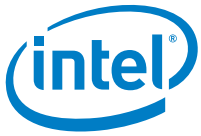
This register is shared for both LAN ports.

| Field | Bit (s) | Init Val | Description |
|---|---|---|---|
| Reserved | 2:0 | 100b | Reserved |
| Reserved | 8:3 | X | Reserved. |
| Completion Timeout resend enable | 9 | 1b | When set, enables a resend request after the completion timeout expires. This field is loaded from the *Completion Timeout Resend* bit in the EEPROM (PCIe General Config word 5 bit 15). |
| Reserved | 10 | 0b | Reserved. |
| Number of resends | 12:11 | 11b | The number of resends in case of timeout or poisoned. |
| Reserved | 17:13 | 0x0 | Reserved. |
| PCIe Capability Version | 18 | 1b[1] | Read only field reporting supported PCIe capability version.<br>0b = Capability version: 0x1.<br>1b = Capability version: 0x2. |
| Reserved | 20:19 | 0b | Reserved. |
| hdr_log inversion | 21 | 0b | If set, the header log in error reporting is written as 31:0 to log1, 63:32 in log2, etc. If not, the header is written as 127:96 in log1, 95:64 in log 2, etc. |
| Reserved | 31:22 | 0 | Reserved. |

### 8.2.3.4.2 PCIe Statistic Control Register #1 — GSCL_1 (0x11010; RW)

**Note:** This register is shared for both LAN ports.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GIO_COUNT_EN_0 | 0 | 0b | Enables PCIe statistic counter number 0. |
| GIO_COUNT_EN_1 | 1 | 0b | Enables PCIe statistic counter number 1. |
| GIO_COUNT_EN_2 | 2 | 0b | Enables PCIe statistic counter number 2. |
| GIO_COUNT_EN_3 | 3 | 0b | Enables PCIe statistic counter number 3. |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LBC Enable 0 | 4 | 0b | When set, statistics counter 0 operates in leaky bucket mode. In this mode there is an internal counter that is incremented by one for each event and is decremented by one each time the LBC timer n (n=0) expires. When the internal counter reaches the value of LBC threshold n (n=0) the internal counter is cleared and the visible associated statistic counter GSCN_0_3[0] is incremented by one.<br>When cleared, Leaky Bucket mode is disabled and the counter is incremented by one for each event. |
| LBC Enable 1 | 5 | 0b | When set, statistics counter 1 operates in leaky bucket mode. See detailed description for LBC Enable 0. |
| LBC Enable 2 | 6 | 0b | When set, statistics counter 2 operates in leaky bucket mode. See detailed description for LBC Enable 0. |
| LBC Enable 3 | 7 | 0b | When set, statistics counter 3 operates in leaky bucket mode. See detailed description for LBC Enable 0. |
| Reserved | 26:8 | 0x0 | Reserved. |
| GIO_COUNT_TEST | 27 | 0b | Reserved. |
| GIO_64_BIT_EN | 28 | 0b | Enables two 64-bit counters instead of four 32-bit counters. |
| GIO_COUNT_RESET | 29 | 0b | Reset indication of PCIe statistic counters. |
| GIO_COUNT_STOP | 30 | 0b | Stop indication of PCIe statistic counters. |
| GIO_COUNT_START | 31 | 0b | Start indication of PCIe statistic counters. |

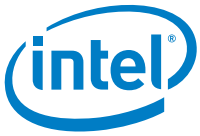
#### 8.2.3.4.3 PCIe Statistic Control Registers #2- GSCL_2 (0x11014; RW)

**Note:** This register is shared for both LAN ports.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GIO_EVENT_NUM_0 | 7:0 | 0x0 | Event number that counter 0 counts (GSCN_0). |
| GIO_EVENT_NUM_1 | 15:8 | 0x0 | Event number that counter 1 counts (GSCN_1). |
| GIO_EVENT_NUM_2 | 23:16 | 0x0 | Event number that counter 2 counts (GSCN_2). |
| GIO_EVENT_NUM_3 | 31:24 | 0x0 | Event number that counter 3 counts (GSCN_3). |

## Table 8-4 PCIe Statistic Events Encoding

| Transaction layer Events | Event Mapping (Hex) | Description |
|---|---|---|
| Bad TLP from LL | 00 | For each cycle, the counter increases by one, if a bad TLP is received (bad CRC, error reported by AL, misplaced special character, reset in thI of received tlp). |
| Requests that reached timeout | 10 | Number of requests that reached time out. |
| NACK DLLP received | 20 | For each cycle, the counter increases by one, if a message was transmitted. |
| Replay happened in retry buffer | 21 | Occurs when a replay happened due to timeout (not asserted when replay initiated due to NACK. |
| Receive error | 22 | Set when one of the following occurs:<br>1. Decoder error occurred during training in the PHY. It is reported only when training ends.<br>2. Decoder error occurred during link-up or until the end of the current packet (in case the link failed). This error is masked when entering/ exiting Electrical Idle (EI). |
| Replay roll over | 23 | Occurs when replay was initiated for more than three times (threshold is configurable by the PHY CSRs). |
| Re-sending packets | 24 | Occurs when TLP is resent in case of completion timeout. |
| Surprise link down | 25 | Occurs when link is unpredictably down (not because of reset or DFT). |
| LTSSM in L0s in both Rx and Tx | 30 | Occurs when LTSSM enters L0s state in both Tx & Rx. |
| LTSSM in L0s in Rx | 31 | Occurs when LTSSM enters L0s state in Rx. |
| LTSSM in L0s in Tx | 32 | Occurs when LTSSM enters L0s state in Tx. |
| LTSSM in L1 active | 33 | Occurs when LTSSM enters L1-active state (requested from host side). |
| LTSSM in L1 software | 34 | Occurs when LTSSM enters L1-switch (requested from switch side). |
| LTSSM in recovery | 35 | Occurs when LTSSM enters recovery state. |



#### 8.2.3.4.4 PCIe Statistic Control Register #5...#8 — GSCL_5_8 (0x011030 + 4*n, n=0...3; RW)

**Note:** These registers are shared for both LAN ports.

These registers control the operation of the leaky bucket counter n. While it is GSCL_5 for n=0. GSCL_6 for n=1, GSCL_7 for n=2 and GSCL_8 for n=3. Note that there are no GSCL_3 and GSCL_4 registers.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LBC threshold n | 15:0 | 0x0 | Threshold for the leaky bucket counter n. |
| LBC timer n | 31:16 | 0x0 | Time period between decrementing the value in leaky bucket Counter n.<br>The time period is defined in µS units. |

#### 8.2.3.4.5 PCIe Statistic Counter Registers #0...#3 — GSCN_0_3 (0x11020 + 4*n, n=0...3; RO)

**Note:** This register is shared for both LAN ports.

While it is GSCN_0 for n=0. GSCN_1 for n=1, GSCN_2 for n=2 and GSCN_3 for n=3.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Event Counter | 31:0 | 0x0 | Event counter as defined in GSCL_2.GIO_EVENT_NUM fields.<br>These registers are stuck at their maximum value of 0xFF...F and cleared on read. |

#### 8.2.3.4.6 Function Active and Power State to Manageability — FACTPS (0x10150; RO)

Register for use by the device firmware for configuration.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Func0 Power State | 1:0 | 00b | Power state indication of function 0.<br>00b = DR<br>01b = D0u<br>10b = D0a<br>11b = D3 |
| LAN0 Valid | 2 | 0b | LAN 0 Enable.<br>When this bit is set to 0b, it indicates that the LAN 0 function is disabled. When the function is enabled, the bit is set to 1b.<br>This bit is reflected if the function is disabled through the external pad. |
| Func0 Aux_En | 3 | 0b | Function 0 *Auxiliary (AUX) Power PM Enable* bit shadow from the configuration space. |
| Reserved | 5:4 | 00b | Reserved. |

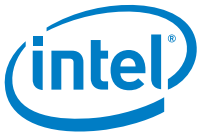
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Func1 Power State | 7:6 | 00b | Power state indication of function 1.<br>00b = DR<br>01b = D0u<br>10b = D0a<br>11b = D3 |
| LAN1 Valid | 8 | 0b | LAN 1 Enable.<br>When this bit is set to 0b, it indicates that the LAN 1 function is disabled. When the function is enabled, the bit is set to 1b.<br>This bit is reflected if the function is disabled through the external pad. |
| Func1 Aux_En | 9 | 0b | Function 1 *Auxiliary (AUX) Power PM Enable* bit shadow from the configuration space. |
| Reserved | 28:10 | 0x0 | Reserved. |
| MNGCG | 29 | 0b | Manageability Clock Gated.<br>When set, indicates that the manageability clock is gated. |
| LAN Function Sel | 30 | 0b[1] | When both LAN ports are enabled and *LAN Function Sel* equals 0b, LAN 0 is routed to PCI function 0 and LAN 1 is routed to PCI function 1. If *LAN Function Sel* equals 1b, LAN 0 is routed to PCI function 1 and LAN 1 is routed to PCI function 0. This bit is loaded from the *LAN Function Select* bit in the PCIe Control 2 EEPROM word at offset 0x05. |
| PM State changed | 31 | 0b | Indication that one or more of the functions power states had changed.<br>This bit is also a signal to the manageability unit to create an interrupt. This bit is cleared on read, and is not set for at least eight cycles after it was cleared. |

1. Loaded from the EEPROM.

### 8.2.3.4.7 PCIe Analog Configuration Register — PCIEPHYADR (0x11040; RW)

**Note:** This register is shared for both LAN ports.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Address | 11:0 | 0x0 | The indirect access' address. |
| Reserved | 24:12 | 0x0 | Reserved |
| Byte Enable | 28:25 | 0x0 | The indirect access' byte enable (4-bit). |
| Read enable | 29 | 0b | The indirect access is read transaction. |
| Write enable | 30 | 0b | The indirect access is write transaction. |
| Done indication | 31 | 0b | Acknowledge for the indirect access to the CSR. |



### 8.2.3.4.8 PCIe PHY Data Register — PCIEPHYDAT (0x11044; RW)

**Note:** This register is shared for both LAN ports.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Data | 31:0 | 0x0 | The data to write in the indirect access or the returned data of the indirect read. |

### 8.2.3.4.9 Software Semaphore Register — SWSM (0x10140; RW)

**Note:** This register is shared for both LAN ports.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SMBI | 0 | 0b | Semaphore Bit.<br>This bit is set by hardware, when this register is read by the device driver (one of two PCI functions) and cleared when the host driver writes 0b to it.<br>The first time this register is read, the value is 0b. In the next read the value is 1b (hardware mechanism). The value remains 1b until the device driver clears it.<br>This bit can be used as a semaphore between the two device's drivers.<br>This bit is cleared on PCIe reset. |
| SWESMBI | 1 | 0b | Software Semaphore bit.<br>This bit is set by the device driver (read only to the firmware) before accessing the SW_FW_SYNC register. This bit can be read as 1b only if the FWSM.FWSMBI bit is cleared.<br>The device driver should clear this bit after accessing the SW_FW_SYNC register as described in Section 10.5.4. Hardware clears this bit on PCIe reset. |
| RSV | 31:2 | 0x0 | Reserved. |

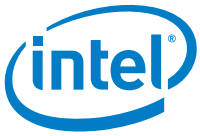
### 8.2.3.4.10 Firmware Semaphore Register — FWSM (0x10148; RW)

**Note:** This register is shared for both LAN ports.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| FWSMBI | 0 | 0b | Firmware Semaphore.<br>Firmware should set this bit to 1b before accessing the SW_FW_SYNC register. This bit can be read as 1b only if the SWSM.SMBI is cleared.<br>Firmware should set it back to 0b after modifying the SW_FW_SYNC register as described in Section 10.5.4. |
| FW_mode | 3:1 | 000b | Firmware Mode. Indicates the firmware mode as follows:<br>0x0 = None (manageability off).<br>0x1 = Reserved.<br>0x2 = PT mode.<br>0x3 = Reserved.<br>0x4 = Host interface enable only.<br>Else = Reserved. |
| Reserved | 5:4 | 00b | Reserved. |

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| EEP_reload_ ind | 6 | 0b | EEPROM Reloaded Indication.<br>Set to 1b after firmware re-loads the EEPROM.<br>Cleared by firmware once the *Clear Bit* host command is received from host software. |
| Reserved | 14:7 | 0x0 | Reserved. |
| FW_Val_bit | 15 | 0b | Firmware Valid Bit.<br>Hardware clears this bit in reset de-assertion so software can know firmware mode (bits 1-5) is invalid. firmware should set this bit to 1b when it is ready (end of boot sequence). |
| Reset_cnt | 18:16 | 000b | Reset Counter.<br>Firmware increments this counter after every reset. |
| Ext_err_ind | 24:19 | 0x0 | External Error Indication.<br>Firmware uses this register to store the reason that the firmware has reset / clock gated (such as EEPROM, Flash, patch corruption, etc.).<br>Possible values:<br>0x00 = No error.<br>0x01 = Invalid EEPROM checksum.<br>0x02 = Unlocked secured EEPROM.<br>0x03 = Clock off host command.<br>0x04 = Invalid Flash checksum.<br>0x05 = C0 checksum failed.<br>0x06 = C1 checksum failed.<br>0x07 = C2 checksum failed.<br>0x08 = C3 checksum failed.<br>0x09 = TLB table exceeded.<br>0x0A = DMA load failed.<br>0x0B = Bad hardware version in patch load.<br>0x0C = Flash device not supported in the 82599.<br>0x0D = Unspecified error.<br>0x3F = Reserved (maximum error value). |
| PCIe_config_ err_ind | 25 | 0b | PCIe Configuration Error Indication.<br>Set to 1b by firmware when it fails to configure PCIe interface.<br>Cleared by firmware upon successful configuration of PCIe interface. |
| PHY_SERDES0_config_ err_ind | 26 | 0b | PHY/SERDES0 Configuration Error Indication.<br>Set to 1b by firmware when it fails to configure PHY/SERDES of LAN0.<br>Cleared by firmware upon successful configuration of PHY/SERDES of LAN0. |
| PHY_SERDES1_config_ err_ind | 27 | 0b | PHY/SERDES1 Configuration Error Indication.<br>Set to 1b by firmware when it fails to configure PHY/SERDES of LAN1.<br>Cleared by firmware upon successful configuration of PHY/SERDES of LAN1. |
| Reserved | 31:28 | 0000b | Reserved. |

**Notes:** This register should be written only by the manageability firmware. The device driver should only read this register.



The firmware ignores the EEPROM semaphore in operating system hung states.

Bits 15:0 are cleared on firmware reset.

### 8.2.3.4.11 Software–Firmware Synchronization — SW_FW_SYNC (0x10160; RW)

**Note:** This register is shared for both LAN ports.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SMBITS | 9:0 | 0x0 | Semaphore Bits.<br>Each bit represents a different software semaphore agreed between software and firmware as listed. Bits 4:0 are owned by software while bits 9:5 are owned by firmware.<br>*Note:* Hardware does not lock access to these bits.<br>Bit 0 = SW_EEP_SM - at 1b, EEPROM access is owned by software.<br>Bit 1 = SW_PHY_SM0 - at 1b, PHY 0 access is owned by software.<br>Bit 2 = SW_PHY_SM1 - at 1b, PHY 1 access is owned by software.<br>Bit 3 = SW_MAC_CSR_SM - at 1b, Software owns access to shared CSRs.<br>Bit 4 = SW_FLASH_SM - Software Flash semaphore.<br>Bit 5 = FW_EEP_SM - at 1b, EEPROM access is owned by firmware.<br>Bit 6 = FW_PHY_SM0 - at 1b, PHY 0 access is owned by firmware.<br>Bit 7 = FW_PHY_SM1 - at 1b, PHY 1 access is owned by firmware.<br>Bit 8 = FW_MAC_CSR_SM - at 1b, firmware owns access to shared CSRs.<br>Bit 9 = FW_FLASH_SM - at 1b, firmware owns access to the Flash.<br>*Note:* Currently the FW does not access the FLASH. |
| Reserved | 30:10 | 0x0 | Reserved for future use. |
| Reserved | 31 | 0b | Reserved. |

See Section 10.5.4 for more details on software and firmware synchronization.

### 8.2.3.4.12 PCIe Control Extended Register — GCR_EXT (0x11050; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| VT_Mode | 1:0 | 00b | VT mode of operation defines the allocation of physical registers to the VFs. Software must set this field the same as GPIE.VT_Mode.<br>00b = No VT - Reserved for the case that STSTUS.IOV Ena is not set.<br>01b = VT16 - Resources are allocated to 16 VFs.<br>10b = VT32 - Resources are allocated to 32 VFs.<br>11b = VT64 - Resources are allocated to 64 VFs. |
| Reserved | 3:2 | 00b | Reserved. |

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| APBACD | 4 | 0b | Auto PBA Clear Disable.<br>When set to 1b, Software can clear the PBA only by direct write to clear access to the *PBA* bit. When set to 0b, any active PBA entry is cleared on the falling edge of the appropriate interrupt request to the PCIe block. The appropriate interrupt request is cleared when software sets the associated interrupt mask bit in the EIMS (re-enabling the interrupt) or by direct write to clear to the PBA. |
| Reserved | 29:5 | 0x0 | Reserved. |
| Buffers Clear Func | 30 | 0 | Initiate a cleaning flow for the buffers in the transaction layer for both the read & write flows.<br>*Note:* Should be only used during Master disable flow. See Section 5.2.5.3.2, Master Disable. |
| Reserved | 31 | 0 | Reserved. |

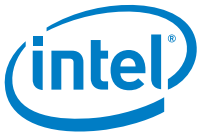
### 8.2.3.4.13 Mirrored Revision ID- MREVID (0x11064; RO)

**Note:** This register is shared for both LAN ports.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| EEPROM_RevID | 7:0 | 0x0 | Mirroring of rev ID loaded from EEPROM. |
| DEFAULT_RevID | 15:8 | 0x0 | Mirroring of default rev ID, before EEPROM load (0x0 for the 82599 A0). |
| Reserved | 31:16 | 0x0 | Reserved. |

### 8.2.3.4.14 PCIe Interrupt Cause — PICAUSE (0x110B0; RW1/C)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| CA | 0 | 0b | PCI completion abort exception. |
| UA | 1 | 0b | Unsupported I/O address exception. |
| BE | 2 | 0b | Wrong byte-enable exception in the FUNC unit. |
| TO | 3 | 0b | PCI timeout exception in the FUNC unit. |
| BMEF | 4 | 0b | Asserted when bus master enable of the PF or one of the VFs is de-asserted. |
| Reserved | 31:5 | 0x0 | Reserved |



### 8.2.3.4.15 PCIe Interrupt Enable — PIENA (0x110B8; RW)

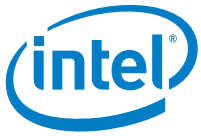
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| CA | 0 | 0b | When set to 1b, the PCI completion abort interrupt is enabled. |
| UA | 1 | 0b | When set to 1b, the unsupported I/O address interrupt is enabled. |
| BE | 2 | 0b | When set to 1b, the wrong byte-enable interrupt is enabled. |
| TO | 3 | 0b | When set to 1b, the PCI timeout interrupt is enabled. |
| BMEF | 4 | 0b | When set to 1b, the bus master enable interrupt is enabled. |
| Reserved | 31:5 | 0x0 | Reserved |

## 8.2.3.5 Interrupt Registers

### 8.2.3.5.1 Extended Interrupt Cause Register- EICR (0x00800; RW1C)

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 Val | Description |
|---|---|---|---|
| RTxQ | 15:0 | 0x0 | Receive/Transmit Queue Interrupts.<br>One bit per queue or a bundle of queues, activated on receive/transmit events.<br>The mapping of the queue to the RTxQ bits is done by the IVAR registers. |
| Flow Director | 16 | 0b | Flow director exception is activated by one of the following events:<br>1. Filter Removal failed (no matched filter to be removed).<br>2. The number of remaining free filters in the flexible filter table exceeds (goes below) the FDIRCTRL.Full-Thresh.<br>3. |
| Rx Miss | 17 | 0b | Missed packet interrupt is activated for each received packet that overflows the Rx packet buffer (overrun).<br>*Note:* The packet is dropped and also increments the associated RXMPC[n] counter. |
| PCI Exception | 18 | 0b | The PCI timeout exception is activated by one of the following events while the specific PCI event is reported in the INTRPT_CSR register:<br>1. I/O completion abort (write to Flash when Flash is write-disabled).<br>2. Unsupported I/O request (wrong address).<br>3. Byte-Enable Error. Access to a client that does not support partial byte-enable access (all but Flash, MSI-X and PCIe target).<br>4. Timeout occurred in the FUNC block. |
| MailBox | 19 | 0b | VF to PF MailBox Interrupt.<br>Cause by a VF write access to the PF mailbox. |
| LSC | 20 | 0b | Link Status Change.<br>This bit is set each time the link status changes (either from up to down, or from down to up). |
| LinkSec | 21 | 0b | Indicates that the Tx LinkSec packet counter reached the threshold requiring key exchange. |
| MNG | 22 | 0b | Manageability Event Detected.<br>Indicates that a manageability event happened. When the device is at power down mode, the MC might generate a PME for the same events that would cause an interrupt when the device is at the D0 state. |
| Reserved | 23 | 0b | Reserved. |
| GPI_SDP0 | 24 | 0b | General Purpose Interrupt on SDP0.<br>If GPI interrupt detection is enabled on this pin (via GPIE), this interrupt cause is set when the SDP0 is sampled high. |
| GPI_SDP1 | 25 | 0b | General Purpose Interrupt on SDP1.<br>If GPI interrupt detection is enabled on this pin (via GPIE), this interrupt cause is set when the SDP1 is sampled high. |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GPI_SDP2 | 26 | 0b | General Purpose Interrupt on SDP2.<br>If GPI interrupt detection is enabled on this pin (via GPIE), this interrupt cause is set when the SDP2 is sampled high. |
| GPI_SDP3 | 27 | 0b | General Purpose Interrupt on SDP3.<br>If GPI interrupt detection is enabled on this pin (via GPIE), this interrupt cause is set when the SDP3 is sampled high. |
| ECC | 28 | 0b | Unrecoverable ECC Error.<br>This bit is set when an unrecoverable error is detected in one of the device memories. Software should issue a software reset following this error. |
| Reserved | 29 | 0b | Reserved. |
| TCP Timer | 30 | 0b | TCP Timer Expired.<br>This bit is set when the timer expires. |
| Other Cause Interrupt | 31 | 0b | Activated when any bit (29:20) in the Extended Interrupt Cause Register (EICR) is set and its relevant mask bit in the EIMS is enabled. |

#### 8.2.3.5.2 Extended Interrupt Cause Set Register- EICS (0x00808; WO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Interrupt Cause Set | 30:0 | 0x0 | Setting any bit in this field, sets its corresponding bit in the EICR register and generates an interrupt if enabled by the EIMS register. |
| Reserved | 31 | 0b | Reserved. |

#### 8.2.3.5.3 Extended Interrupt Mask Set/Read Register- EIMS (0x00880; RWS)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Interrupt Enable | 30:0 | 0x0 | Each bit set to 1b enables its corresponding interrupt in the EICR.<br>Writing 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 | Reserved. |

### 8.2.3.5.4 Extended Interrupt Mask Clear Register- EIMC (0x00888; WO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Interrupt Mask | 30:0 | 0x0 | Writing a 1b to any bit clears its corresponding bit in the EIMS register disabling the corresponding interrupt in the EICR register. Writing 0b has no impact.<br>Reading this register provides no meaningful data. |
| Reserved | 31 | 0b | Reserved. |

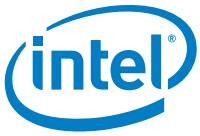
### 8.2.3.5.5 Extended Interrupt Auto Clear Register — EIAC (0x00810; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RTxQ Auto Clear | 15:0 | 0x0 | At 1b, each bit enables auto clear of the corresponding RTxQ bits in the EICR register following interrupt assertion.<br>At 0b, the corresponding bits in the EICR register are not auto cleared. |
| Reserved | 29:16 | 0x0 | Reserved. |
| TCP Timer Auto Clear | 30 | 0b | At 1b, this bit enables auto clear of the TCP timer interrupt cause in the EICR register following interrupt assertion.<br>At 0b auto clear is not enabled. |
| Reserved | 31 | 0b | Reserved. |

**Note:** Bits 29:20 should never set auto clear since they share the same MSI-X vector.

### 8.2.3.5.6 Extended Interrupt Auto Mask Enable Register — EIAM (0x00890; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Auto Mask | 30:0 | 0x0 | At 1b, each bit enables auto set and clear of its corresponding bits in the EIMS register.<br>In MSI-X mode, if any of the Auto Mask enable bits is set, the GPIE.EIAME bit must be set as well. |
| Reserved | 31 | 0b | Reserved. |



### 8.2.3.5.7 Extended Interrupt Cause Set Registers — EICS[n] (0x00A90 + 4*(n-1), n=1...2; WO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Interrupt Cause Set | 31:0 | 0x0 | Setting any bit in these registers sets its corresponding bit in the EICR[n] register and generates an interrupt if enabled by EIMS[n] register.<br>Reading this register provides no meaningful data. |

### 8.2.3.5.8 Extended Interrupt Mask Set/Read Registers — EIMS[n] (0x00AA0 + 4*(n-1), n=1...2; RWS)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Interrupt Enable | 31:0 | 0 | Each bit set at 1b enables its corresponding interrupt in the EICR[n] register.<br>Writing 1b to any bit sets it. Writing 0b has no impact. Reading this register provides a map of those interrupts that are enabled.<br>Bits 15:0 of EIMS1 are mirrored in EIMS bits 15:0. |

### 8.2.3.5.9 Extended Interrupt Mask Clear Registers — EIMC[n] (0x00AB0 + 4*(n-1), n=1...2; WO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Interrupt Mask | 31:0 | 0x0 | Writing a 1b to any bit clears its corresponding bit in the EIMS[n] register disabling the corresponding interrupt in the EICR[n] register.<br>Writing 0b has no impact. Reading this register provides no meaningful data. |

### 8.2.3.5.10 Extended Interrupt Auto Mask Enable registers — EIAM[n] (0x00AD0 + 4*(n-1), n=1...2; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Auto Mask | 31:0 | 0x0 | At 1b, each bit enables auto set and clear of its corresponding bits in the EIMS[n] register.<br>Bits 15:0 of EIAM1 are mirrored in EIAM bits 15:0.<br>In MSI-X mode, if any of the Auto Mask enable bits is set, the GPIE.EIAME bit must be set as well. |

### 8.2.3.5.11 MSIX to EITR Select — EITRSEL (0x00894; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| VFSelect | 31:0 | 0x0 | 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).<br>At 0b, it selects the PF and at 1b it selects the VF. |

### 8.2.3.5.12 Extended Interrupt Throttle Registers — EITR[n] (0x00820 + 4*n, n=0...23 and 0x012300 + 4*(n-24), n=24...128; RW)

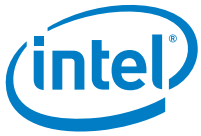
Mapping of the EITR registers to the MSI-X vectors is described in Section 7.3.4.3.3.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 2:0 | 000b | Reserved. |
| ITR Interval | 11:3 | 0x0 | Minimum inter-interrupt interval specified in 2 µs units at 1 Gb/2 and 10 Gb/s link.<br>At 100 Mb/s link, the interval is specified in 20 µs units.<br>At 0x0 interrupt throttling is disabled while any "event" causes an immediate interrupt. |
| Reserved | 14:12 | 000b | Reserved. |
| LLI Moderation | 15 | 0b | When set, LLI moderation is enabled. Otherwise, any LLI packet generates an immediate interrupt.<br>LLI moderation might 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 | Reflects the current credits for associated interrupt.<br>When CNT_WDIS is not set on a write cycle, this field must be set to 0x0. |
| ITR Counter | 27:21 | 0x0 | This field represents the seven MS bits (out of nine bits) of the ITR counter.<br>It is a down counter that is loaded with an ITR interval value each time the associated interrupt is asserted. When the ITR counter reaches zero it stops counting and triggers an interrupt.<br>On a write cycle, software must set this field to 0 if CNT_WDIS in this register is cleared (write enable to the ITR counter). |
| Reserved | 30:28 | 000b | Reserved. |
| CNT_WDIS | 31 | 0b | Write disable to the LLI credit and ITR counter.<br>When set, the LLI credit and ITR counter are not overwritten by the write access. When cleared, software must set the LLI credit and ITR counter to zero, which enables an immediate interrupt on packet reception. This bit is write only. Always read as 0b. |

### 8.2.3.5.13 L3 L4 Tuples Immediate Interrupt Rx — L34TIMIR[n] (0x0E800 + 4*n, n=0...127; RW)

This register must be initialized by software.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 11:0 | X | Reserved. |
| Size_BP | 12 | X | Size Bypass.<br>0b = Size check is performed.<br>1b = Size check is bypassed. |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 19:13 | X | Reserved.<br>Must be set to 1000000b on any programmed filter. |
| Low Latency Interrupt | 20 | X | Enables issuing a LLI when the following conditions are met:<br>• The 5-tuple filter associated with this register matches.<br>• If enabled by the Size_BP bit, the packet length is smaller than the length defined by LLITHRESH.SizeThresh. |
| Rx Queue | 27:21 | X | Identifies the Rx queue associated with this 5-tuple filter. |
| Reserved | 31:28 | X | Reserved. |

### 8.2.3.5.14 LLI Size Threshold — LLITHRESH (0x0EC90; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SizeThresh | 11:0 | 0x000 | Size Threshold.<br>A packet with length below this threshold that matches one of the 5-tuple filters with an active *Low Latency Interrupt* flag in the L34TIMIR[n] registers might trigger an LLI. |
| Reserved | 25:12 | 0x0 | Reserved. |
| Reserved | 31:26 | 000101b | Reserved. |

### 8.2.3.5.15 Immediate Interrupt Rx VLAN Priority Register- IMIRVP (0x0EC60 / 0x05AC0; RW)

IMIRVP is also mapped to address 0x05AC0 to maintain compatibility with the 82598.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Vlan_Pri | 2:0 | 000b | VLAN Priority.<br>This field includes the VLAN priority threshold. When Vlan_pri_en is set to 1b, then an incoming packet with VLAN tag with a priority equal or higher to VlanPri must trigger a LLI, regardless of the ITR moderation. |
| Vlan_pri_en | 3 | 0 | VLAN Priority Enable.<br>When 1b, an incoming packet with VLAN tag with a priority equal or higher to Vlan_Pri must trigger a LLI, regardless of the ITR moderation.<br>When 0b, the interrupt is moderated by ITR. |
| Reserved | 31:4 | | Reserved. |

### 8.2.3.5.16 Interrupt Vector Allocation Registers — IVAR[n] (0x00900 + 4*n, n=0...63; RW)

These registers map interrupt causes into EICR entries (legacy/MSI modes) or into MSI-X vectors (MSI-X modes). See Section 7.3.4 for mapping and use of these registers.

Transmit and receive queues mapping to IVAR registers is shown in Figure 8-1:

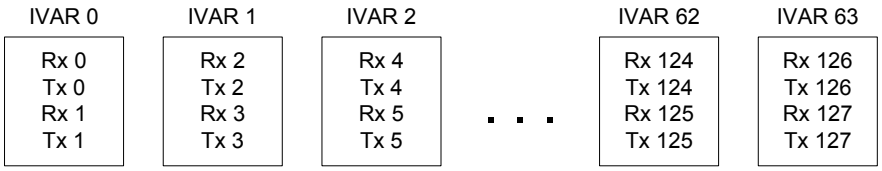
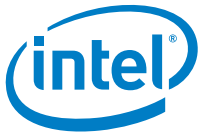



**Figure 8-1 Transmit and Receive Queues Mapping to IVAR Registers**

Fields of the IVAR registers are described in Table 8-5.

**Table 8-5 Fields of IVAR Register**

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| INT_Alloc[0] | 5:0 | X | The interrupt allocation for Rx queue (‘2xN’ for IVAR register ‘N’). |
| Reserved | 6 | 0b | Reserved. |
| INT_Alloc_val[0] | 7 | 0b | Interrupt allocation valid indication for INT_Alloc[0]. |
| INT_Alloc[1] | 13:8 | X | The interrupt allocation for Tx queue (‘2xN’ for IVAR register ‘N’). |
| Reserved | 14 | 0b | Reserved. |
| INT_Alloc_val[1] | 15 | 0b | Interrupt allocation valid indication for INT_Alloc[1]. |
| INT_Alloc[2] | 21:16 | X | The interrupt allocation for Rx queue (‘2xN’+1 for IVAR register ‘N’). |
| Reserved | 22 | 0b | Reserved. |
| INT_Alloc_val[2] | 23 | 0b | Interrupt allocation valid indication for INT_Alloc[2]. |
| INT_Alloc[3] | 29:24 | X | The interrupt allocation for Tx queue (‘2xN’+1 for IVAR register ‘N’). |
| Reserved | 30 | 0b | Reserved. |
| INT_Alloc_val[3] | 31 | 0b | Interrupt allocation valid indication for INT_Alloc[3]. |



### 8.2.3.5.17 Miscellaneous Interrupt Vector Allocation — IVAR_MISC (0x00A00; RW)

These register maps interrupt causes into MSI-X vectors (MSI-X modes). See Section 7.3.4 for mapping and use of these registers.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| INT_Alloc[0] | 6:0 | X | Defines the MSI-X vector assigned to the TCP timer interrupt cause. |
| INT_Alloc_val[0] | 7 | 0b | Valid bit for INT_Alloc[0]. |
| INT_Alloc[1] | 14:8 | X | Defines the MSI-X vector assigned to the other interrupt cause. |
| INT_Alloc_val[1] | 15 | 1b | Valid bit for INT_Alloc[1]. |
| Reserved | 31:16 | 0b | Reserved. |

**Note:** The INT_ALLOC_VAL[1] bit default value is one — to enable legacy driver functionality.

### 8.2.3.5.18 General Purpose Interrupt Enable — GPIE (0x00898; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SDP0_GPIEN | 0 | 0b | General Purpose Interrupt Detection Enable for SDP0.<br>If software-controllable I/O pin SDP0 is configured as an input, this bit (when 1b) enables use for GPI interrupt detection. |
| SDP1_GPIEN | 1 | 0b | General Purpose Interrupt Detection Enable for SDP1.<br>If software-controllable I/O pin SDP1 is configured as an input, this bit (when 1b) enables use for GPI interrupt detection. |
| SDP2_GPIEN | 2 | 0b | General Purpose Interrupt Detection Enable for SDP2.<br>If software-controllable I/O pin SDP2 is configured as an input, this bit (when 1b) enables use for GPI interrupt detection. |
| SDP3_GPIEN | 3 | 0b | General Purpose Interrupt Detection Enable for SDP3.<br>If software-controllable I/O pin SDP3 is configured as an input, this bit (when 1b) enables use for GPI interrupt detection. |
| Multiple_MSIX | 4 | 0b | MSI-X Mode. Selects between MSI-X interrupts and other interrupt modes.<br>0b = Legacy and MSI mode (non-MSI-X mode).<br>1b = MSI-X mode. |
| OCD | 5 | 0b | Other Clear Disable.<br>0b = The entire EICR is cleared on read.<br>1b = Indicates that only bits 29:16 of the EICR register are cleared on read. |
| EIMEN | 6 | 0b | EICS Immediate Interrupt Enable.<br>When set, setting this bit in the EICS register causes a LLI. If not set, the EICS interrupt waits for EITR expiration. |

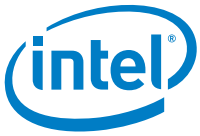
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LL Interval | 10:7 | 0x0 | Low latency Credits Increment Rate.<br>The interval is specified in 4 µs increments at 1 Gb/s and 10 Gb/s link. It is defined as 40 µs at 100 Mb/s link. A value of 0x0 disables moderation of LLI for all interrupt vectors. When LLI is disabled by the *LL Interval* bit, the *LLI Moderation* bit in all EITR registers must not be set. |
| RSC Delay | 13:11 | 000b | Delay from RSC completion triggered by ITR and interrupt assertion.<br>The delay = (RSC Delay + 1) x 4 µs = 4, 8, 12... 32 µs. |
| VT_Mode | 15:14 | 00b | Specify the number of active VFs.<br>Software must set this field the same as GCR_Ext.VT_Mode.<br>00b = Non-IOV mode.<br>10b = 32 VF mode.<br>01b = 16 VF mode.<br>11b = 64 VF mode. |
| Reserved | 29:16 | 0x0 | Reserved. |
| EIAME | 30 | 0b | Extended Interrupt Auto Mask Enable.<br>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.<br>Software may set the EIAME only in MSI-X mode. |
| PBA_support | 31 | 0b | PBA Support.<br>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 82599 behaves in a way supporting legacy INT-x interrupts.<br>*Note:* Should be cleared when working in INT-x or MSI mode and set in MSI-X mode. |

The 82599 allows for up to four externally controlled interrupts. The lower four software-definable pins, SDP[3:0], can be mapped for use as GPI interrupt bits. These mappings are enabled by the SDPx_GPIEN bits only when these signals are also configured as inputs via SDPx_IODIR. When configured to function as external interrupt pins, a GPI interrupt is generated when the corresponding pin is sampled in an active-high state.

The bit mappings are listed in Table 8-6 for clarity.

### Table 8-6 GPI-to-SDP Bit Mappings

| SDP (pin to be used as GPI) | ESDP Field Settings | | Resulting EICR Bit (GPI) |
|---|---|---|---|
| | Directionality | Enable as GPI interrupt | |
| 3 | SDP3_IODIR | SDP3_GPIEN | 27 |
| 2 | SDP2_IODIR | SDP2_GPIEN | 26 |
| 1 | SDP1_IODIR | SDP1_GPIEN | 25 |
| 0 | SDP0_IODIR | SDP0_GPIEN | 24 |



## 8.2.3.6 MSI-X Table Registers

MSI-X capability is described in section 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.

### 8.2.3.6.1 MSI-X PBA Clear — PBACL[n] (0x110C0 + 4*n, n=0...7 / 0x11068 [n=0]; RW)

PBACL[0] is also mapped to address 0x11068 to maintain compatibility with the 82598.

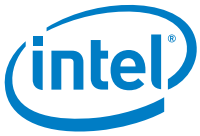
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PENBITCLR | 31:0 | 0x0 | MSI-X Pending Bits Clear.<br>Writing 1b to any bit clears it's content; writing 0b has no effect.<br>Reading this register returns the MSIPBA.PENBIT value. |

## 8.2.3.7 Receive Registers

### 8.2.3.7.1 Filter Control Register — FCTRL (0x05080; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 0 | 0b | Reserved. |
| SBP | 1 | 0b | Store Bad Packets.<br>0b = Do not store.<br>1b = Store.<br>*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) in order to be recognized by the device (even bad packets).<br>*Note:* Packets with errors are not routed to manageability even if this bit is set.<br>*Note:* Erroneous packets can be routed to the default queue rather than the originally intended queue.<br>*Note:* In packets shorter than 64 bytes, the checksum errors can be hidden while MAC errors are reported.<br>*Note:* A packet with a valid error (caused by byte error or illegal error) might have data contamination in the last eight bytes when stored in the host memory if the *Store Bad Packet* bit is set. |
| Reserved | 7:2 | 0x0 | Reserved. |
| MPE | 8 | 0b | Multicast Promiscuous Enable.<br>0b = Disabled.<br>1b = Enabled. When set, all received multicast and broadcast packets pass L2 filtering and can be directed to manageability or the host by a one of the decision filters. |
| UPE | 9 | 0b | Unicast Promiscuous Enable.<br>0b = Disabled.<br>1b = Enabled. |
| BAM | 10 | 0b | Broadcast Accept Mode.<br>0b = Ignore broadcast packets to host.<br>1b = Accept broadcast packets to host. |
| Reserved | 31:11 | 0x0 | Reserved. |

**Note:** Before receive filters are updated/modified the RXCTRL.RXEN bit should be set to 0b. After the proper filters have been set the RXCTRL.RXEN bit can be set to 1b to re-enable the receiver.



### 8.2.3.7.2 VLAN Control Register — VLNCTRL (0x05088; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| VET | 15:0 | 0x8100 | VLAN Ether Type (the VLAN Tag Protocol Identifier — TPID).<br>This register contains the type field hardware matches against to recognize an 802.1Q (VLAN) Ethernet packet.<br>For proper operation, software must not change the default setting of this field (802.3ac standard defines it as 0x8100).<br>This field must be set to the same value as the VT field in the DMATXCTL register.<br>*Note:* This field appears in little endian order (the upper byte is first on the wire (VLNCTRL.VET[15:8]). |
| Reserved | 27:16 | | Reserved. |
| CFI | 28 | 0b | Canonical Form Indicator Bit Value.<br>If CFIEN is set to 1b, then 802.1q packets with CFI equal to this field are accepted; otherwise, the 802.1q packet is discarded. |
| CFIEN | 29 | 0b | Canonical Form Indicator Enable.<br>0b = Disabled (CFI bit not compared to decide packet acceptance).<br>1b = Enabled (CFI from packet must match next CFI field to accept 802.1q packet). |
| VFE | 30 | 0b | VLAN Filter Enable.<br>0b = Disabled (filter table does not decide packet acceptance).<br>1b = Enabled (filter table decides packet acceptance for 802.1q packets). |
| Reserved | 31 | 0b | Reserved. |

### 8.2.3.7.3 Multicast Control Register — MCSTCTRL (0x05090; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MO | 1:0 | 00b | Multicast Offset. This determines which bits of the incoming multicast address are used in looking up the bit vector.<br>00b = [47:36].<br>01b = [46:35].<br>10b = [45:34].<br>11b = [43:32]. |
| MFE | 2 | 0b | Multicast Filter Enable.<br>0b = The multicast table array filter (MTA[n]) is disabled.<br>1b = The multicast table array (MTA[n]) is enabled. |
| Reserved | 31:3 | 0x0 | Reserved. |

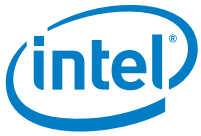
### 8.2.3.7.4 Packet Split Receive Type Register — PSRTYPE[n] (0x0EA00 + 4*n, n=0...63 / 0x05480 + 4*n, n=0...15; RW)

Registers 0...15 are also mapped to 0x05480 to maintain compatibility with the 82598.

**Note:**

- This register must be initialized by software.
- Packets are split according to the lowest-indexed entry that applies to the packet and that is enabled. For example, if bits 4 and 8 are set, then an IPv4 packet that is not TCP is split after the IPv4 header.
- This bit mask table enables or disables each type of header to be split. A value of 1b enables an entry.
- In virtualization mode, a separate PSRTYPE register is provided per pool up to the number of pools enabled. In non-virtualization mode, only PSRTYPE[0] is used.
- PSR_type4 should be set to enable RSC, regardless header split mode.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PSR_type0 | 0 | x | Reserved. |
| PSR_type1 | 1 | x | Split received NFS packets after NFS header. |
| PSR_type2 | 2 | x | Reserved. |
| PSR_type3 | 3 | x | Reserved. |
| PSR_type4 | 4 | x | Split received TCP packets after TCP header. |
| PSR_type5 | 5 | x | Split received UDP packets after UDP header. |
| PSR_type6 | 6 | x | Reserved. |
| PSR_type7 | 7 | x | Reserved. |
| PSR_type8 | 8 | x | Split received IPv4 packets after IPv4 header. |
| PSR_type9 | 9 | x | Split received IPv6 packets after IPv6 header. |
| PSR_type10 | 10 | x | Reserved. |
| PSR_type11 | 11 | x | Reserved. |
| PSR_type12 | 12 | x | Split received L2 packets after L2 header. |
| PSR_type13 | 13 | x | Reserved. |
| PSR_type14 | 14 | x | Reserved. |
| PSR_type15 | 15 | x | Reserved. |
| PSR_type16 | 16 | x | Reserved. |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PSR_type17 | 17 | x | Reserved. |
| PSR_type18 | 18 | x | Reserved. |
| Reserved | 28:19 | x | Reserved. |
| RQPL | 31:29 | x | Defines the number of bits to use for RSS redirection of packets dedicated to this pool.<br>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.<br>A value of zero means that all the traffic of the pool is sent to queue 0 of the pool. This field is used only if MRQC.MRQE equals 1010b or 1011b. |

### 8.2.3.7.5 Receive Checksum Control — RXCSUM (0x05000; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 11:0 | 0x0 | Reserved. |
| IPPCSE | 12 | 0b | IP Payload Checksum Enable. |
| PCSD | 13 | 0b | RSS/Fragment Checksum Status Selection.<br>When set to 1b, the extended descriptor write back has the *RSS* field. When set to 0b, it contains the fragment checksum. |
| Reserved | 31:14 | 0x0 | Reserved. |

The Receive Checksum Control register controls the receive checksum offloading features of the 82599. The 82599 supports the offloading of three receive checksum calculations: the fragment checksum, the IP header checksum, and the TCP/UDP checksum.

PCSD: The Fragment Checksum and IP Identification fields are mutually exclusive with the RSS hash. Only one of the two options is reported in the Rx descriptor. The RXCSUM.PCSD affect is shown in Table 8-7.

**Table 8-7 Checksum Enable/Disable**

| RXCSUM.PCSD | 0b (Checksum Enable) | 1b (Checksum Disable) |
|---|---|---|
| | Fragment checksum and IP identification are reported in the Rx descriptor. | RSS hash value is reported in the Rx descriptor. |

IPPCSE: IPPCSE controls the fragment checksum calculation. As previously noted, the fragment checksum shares the same location as the *RSS* field. The fragment checksum is reported in the receive descriptor when the RXCSUM.PCSD bit is cleared.

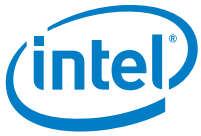
If RXCSUM.IPPCSE is cleared (the default value), the checksum calculation is not done and the value that is reported in the Rx fragment checksum field is 0b.

If RXCSUM.IPPCSE is set, the fragment checksum is aimed to accelerate checksum calculation of fragmented UDP packets. See Section 7.1.13 for a detailed explanation.

This register should only be initialized (written) when the receiver is not enabled (for example, only write this register when RXCTRL.RXEN = 0b).

### 8.2.3.7.6 Receive Filter Control Register — RFCTL (0x05008; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 5:0 | 0x0 | Reserved. |
| RSC_DIS | 5 | 0 | RSC Disable.<br>When set, disable RSC for the port by the Rx filter unit. The default value is 0b (RSC feature is enabled). |
| NFSW_DIS | 6 | 0b | NFS Write disable.<br>Disable filtering of NFS write request headers. |
| NFSR_DIS | 7 | 0b | NFS Read disable.<br>Disable filtering of NFS read reply headers. |
| NFS_VER | 9:8 | 00b | NFS version recognized by the hardware.<br>00b = NFS version 2<br>01b = NFS version 3<br>10b = NFS version 4<br>11b = Reserved for future use |
| IPv6_dis | 10 | 0b | IPv6 Disable.<br>Disable IPv6 packet filtering<br>Internal use only – should not be set to 1b. |
| Reserved | 11 | 0b | Reserved, always set to 0b. |
| Reserved | 13:12 | 00b | Reserved. |
| IPFRSP_DIS | 14 | 0b | IP Fragment Split Disable<br>When this bit is set the header of IP fragmented packets are not set.<br>Internal use only. Should not be set to 1b. |
| Reserved | 15 | 0b | Reserved. |
| Reserved | 17:16 | 00b | Reserved. |
| Reserved | 31:18 | 0x0 | Reserved.<br>Should be written with 0x0 to ensure future compatibility. |



### 8.2.3.7.7 Multicast Table Array — MTA[n] (0x05200 + 4*n, n=0...127; RW)

This table should be initialized by software before transmit and receive are enabled.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Bit Vector | 31:0 | X | Word wide bit vector specifying 32 bits in the multicast address filter table.<br>The 82599 provides multicast filtering for 4096 multicast addresses by providing single-bit entry per multicast address. Those 4096 address locations are organized in the Multicast Table Array (MTA); 128 registers of 32 bits for each one. Only 12 bits out of the 48-bit destination address are considered as a multicast address. Those 12 bits can be selected by the *MO* field of MCSTCTRL register. The 7 MS bits of the Ethernet MAC address (out of the 12 bits) selects the register index while the 5 LS bits (out of the 12 bits) selects the bit within a register. |

### 8.2.3.7.8 Receive Address Low — RAL[n] (0x0A200 + 8*n, n=0...127; RW)

While "n" is the exact unicast/multicast address entry and it is equals to 0,1,...127.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RAL | 31:0 | X | Receive Address Low.<br>The lower 32 bits of the 48-bit Ethernet MAC address.<br>*Note:* Field is defined in big endian (LS byte of RAL is first on the wire). |

These registers contain the lower bits of the 48-bit Ethernet MAC address. All 32 bits are valid.

If the EEPROM is present, the first register (RAL0) is loaded from the EEPROM.

### 8.2.3.7.9 Receive Address High — RAH[n] (0x0A204 + 8*n, n=0...127; RW)

While "n" is the exact unicast/multicast address entry and it is equals to 0,1,...127.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RAH | 15:0 | X | Receive Address High.<br>The upper 16 bits of the 48 bit Ethernet MAC Address.<br>*Note:* Field is defined in Big Endian (MS byte of RAH is Last on the wire). |
| Reserved | 21:16 | 0x0 | Reserved. |

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 30:22 | 0x0 | Reserved.<br>Reads as 0. Ignored on write. |
| AV | 31 | X<br>(see desc.) | Address Valid.<br>All receive addresses are not initialized by hardware and software should initialize them before receive is enabled. If the EEPROM is present, Receive Address[0] is loaded from the EEPROM and its *Address Valid* field is set to 1b after a software, PCI reset or EEPROM read. |

These registers contain the upper bits of the 48-bit Ethernet MAC address. The complete address is {RAH, RAL}. AV determines whether this address is compared against the incoming packet. AV is cleared by a master reset.

**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 EEPROM is present, the first register (Receive Address Register 0) is loaded from the *IA* field in the EEPROM, its *Address Select* field is 00b, and its *Address Valid* field is 1b. If no EEPROM is present, the *Address Valid* field is 0b. The *Address Valid* field for all of the other registers are 0b.

### 8.2.3.7.10 MAC Pool Select Array — MPSAR[n] (0x0A600 + 4*n, n=0...255; RW)

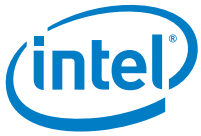
Software should initialize these registers before transmit and receive are enabled.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| POOL_ENA | 31:0 | X | Pool Enable Bit Array.<br>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 follows:<br>Bit ‘i’ in register ‘2*n’ is associated with POOL ‘i’.<br>Bit ‘i’ in register ‘2*n+1’ is associated with POOL ‘32+i’. |

### 8.2.3.7.11 VLAN Filter Table Array — VFTA[n] (0x0A000 + 4*n, n=0...127; RW)

This table should be initialized by software before transmit and receive are enabled.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| VLAN_FLT | 31:0 | X | VLAN Filter. Each bit ‘i’ in register ‘n’ affects packets with VLAN tags equal to 32*n+i. 128 VLAN Filter registers compose a table of 4096 bits that cover all possible VLAN tags.<br>Each bit when set, enables packets with the associated VLAN tags to pass. Each bit when cleared, blocks packets with this VLAN tag. |



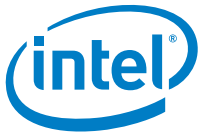
### 8.2.3.7.12 Multiple Receive Queues Command Register- MRQC (0x0EC80 / 0x05818; RW)

MRQC is also mapped to address 0x05818 to maintain compatibility with the 82598.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MRQE | 3:0 | 0x0 | Multiple Receive Queues Enable.<br>Defines the allocation of the Rx queues per RSS, virtualization and DCB.<br>0000b = RSS disabled.<br>0001b = RSS only — Single set of RSS 16 queues.<br>0010b = DCB enabled and RSS disabled — 8 TCs, each allocated 1 queue.<br>0011b = DCB enabled and RSS disabled — 4 TCs, each allocated 1 queue.<br>0100b = DCB and RSS — 8 TCs, each allocated 16 RSS queues.<br>0101b = DCB and RSS — 4 TCs, each allocated 16 RSS queues.<br>0110b = Reserved<br>0111b = Reserved<br>1000b = Virtualization only — 64 pools, no RSS, each pool allocated 2 queues.<br>1001b = Reserved<br>1010b = Virtualization and RSS — 32 pools, each allocated 4 RSS queues.<br>1011b = Virtualization and RSS — 64 pools, each allocated 2 RSS queues.<br>1100b = Virtualization and DCB — 16 pools, each allocated 8 TCs.<br>1101b = Virtualization and DCB — 32 pools, each allocated 4 TCs.<br>1110b = Reserved<br>1111b = Reserved |
| Reserved | 14:4 | 0x0 | Reserved. |
| Reserved | 15 | 0x0 | Reserved. |
| RSS Field Enable | 31:16 | 0x0 | 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.<br>Bit[16] = Enable TcpIPv4 hash function.<br>Bit[17] = Enable IPv4 hash function.<br>Bit[19:18] = Reserved<br>Bit[20] = Enable IPv6 hash function.<br>Bit[21] = Enable TcpIPv6 hash function.<br>Bit[22] = Enable UdpIPV4.<br>Bit[23] = Enable UdpIPV6.<br>Bits[31:24] = Reserved<br>*Note:* On Tunnel packets IPv4-IPv6 only the IPv4 header can be used for the RSS filtering. |

### 8.2.3.7.13 RSS Queues Per Traffic Class Register — RQTC (0x0EC70; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RQTC0 | 2:0 | 0x4 | Defines the number of bits to use for RSS redirection of packets dedicated to Traffic Class (TC) 0.<br>A value of zero means that all the traffic of TC0 is sent to queue 0 of the TC.<br>This field is used only if MRQC.MRQE equals 0100b or 0101b. |
| Reserved | 3 | 0b | Reserved. |
| RQTC1 | 6:4 | 0x4 | Defines the number of bits to use for RSS redirection of packets dedicated to TC 1.<br>A value of zero means that all the traffic of TC1 is sent to queue 0 of the TC.<br>This field is used only if MRQC.MRQE equals 0100b or 0101b. |
| Reserved | 7 | 0b | Reserved. |
| RQTC2 | 10:8 | 0x4 | Defines the number of bits to use for RSS redirection of packets dedicated to TC 2.<br>A value of zero means that all the traffic of TC2 is sent to queue 0 of the TC.<br>This field is used only if MRQC.MRQE equals 0100b or 0101b. |
| Reserved | 11 | 0b | Reserved. |
| RQTC3 | 14:12 | 0x4 | Defines the number of bits to use for RSS redirection of packets dedicated to TC 3.<br>A value of zero means that all the traffic of TC3 is sent to queue 0 of the TC.<br>This field is used only if MRQC.MRQE equals 0100b or 0101b. |
| Reserved | 15 | 0b | Reserved. |
| RQTC4 | 18:16 | 0x4 | Defines the number of bits to use for RSS redirection of packets dedicated to TC 4.<br>A value of zero means that all the traffic of TC4 is sent to queue 0 of the TC.<br>This field is used only if MRQC.MRQE equals 0100b or 0101b. |
| Reserved | 19 | 0b | Reserved. |
| RQTC5 | 22:20 | 0x4 | Defines the number of bits to use for RSS redirection of packets dedicated to TC 5.<br>A value of zero means that all the traffic of TC5 is sent to queue 0 of the TC.<br>This field is used only if MRQC.MRQE equals 0100b or 0101b. |
| Reserved | 23 | 0b | Reserved. |
| RQTC6 | 26:24 | 0x4 | Defines the number of bits to use for RSS redirection of packets dedicated to TC 6.<br>A value of zero means that all the traffic of TC6 is sent to queue 0 of the TC.<br>This field is used only if MRQC.MRQE equals 0100b or 0101b. |
| Reserved | 27 | 0b | Reserved. |
| RQTC7 | 30:28 | 0x4 | Defines the number of bits to use for RSS redirection of packets dedicated to TC 7.<br>A value of zero means that all the traffic of TC7 is sent to queue 0 of the TC.<br>This field is used only if MRQC.MRQE equals 0100b or 0101b. |
| Reserved | 31 | 0b | Reserved. |



### 8.2.3.7.14 RSS Random Key Register — RSSRK (0x0EB80 + 4*n, n=0...9 / 0x05C80 + 4*n, n=0...9; RW)

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).

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| K0 | 7:0 | 0x0 | RSS Key Byte '4*n+0' of the RSS random key, for each register 'n'. |
| K1 | 15:8 | 0x0 | RSS Key Byte '4*n+1' of the RSS random key, for each register 'n'. |
| K2 | 23:16 | 0x0 | RSS Key Byte '4*n+2' of the RSS random key, for each register 'n'. |
| K3 | 31:24 | 0x0 | RSS Key Byte '4*n+3' of the RSS random key, for each register 'n'. |

### 8.2.3.7.15 Redirection Table — RETA[n] (0x0EB00 + 4*n, n=0...31 / 0x05C00 + 4*n, n=0...31; RW)

RETA is also mapped to addresses 0x05C00-0x05C7C to maintain compatibility with the 82598. The redirection table has 128-entries in 32 registers.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Entry0 | 3:0 | X | Entry0 defines the RSS output index for hash value '4*n+0'. While 'n' is the register index, equals to 0...31. |
| Reserved | 7:4 | 0x0 | Reserved. |
| Entry1 | 11:8 | X | Entry1 defines the RSS output index for hash value '4*n+1'. While 'n' is the register index, equals to 0...31. |
| Reserved | 15:12 | 0x0 | Reserved. |
| Entry2 | 19:16 | X | Entry2 defines the RSS output index for hash value '4*n+2'. While 'n' is the register index, equals to 0...31. |
| Reserved | 23:20 | 0x0 | Reserved. |
| Entry3 | 27:24 | X | Entry3 defines the RSS output index for hash value '4*n+3'. While 'n' is the register index, equals to 0...31. |
| Reserved | 31:28 | 0x0 | Reserved. |

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 can 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.

### 8.2.3.7.16 Source Address Queue Filter — SAQF[n] (0x0E000 + 4*n, n=0...127; RW)

This register must be initialized by software

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Source Address | 31:0 | X | IP Source Address.<br>Part of the 5-tuple queue filters.<br>*Note:* Field is defined in big endian (LS byte is first on the wire). |

### 8.2.3.7.17 Destination Address Queue Filter — DAQF[n] (0x0E200 + 4*n, n=0...127; RW)

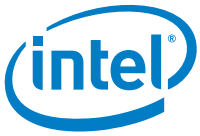
This register must be initialized by software.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Destination Address | 31:0 | X | IP Destination Address.<br>Part of the 5-tuple queue filters.<br>*Note:* Field is defined in big endian (LS byte is first on the wire). |

### 8.2.3.7.18 Source Destination Port Queue Filter — SDPQF[n] (0x0E400 + 4*n, n=0...127; RW)

This register must be initialized by software.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Source Port | 15:0 | X | TCP/UDP Source Port.<br>Part of the 5-tuple queue filters.<br>*Note:* Field is defined in Big Endian (LS byte is first on the wire). |
| Destination Port | 31:16 | X | TCP/UDP Destination Port.<br>Part of the 5-tuple queue filters. |



### 8.2.3.7.19 Five tuple Queue Filter — FTQF[n] (0x0E600 + 4*n, n=0...127; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Protocol | 1:0 | X | IP L4 protocol, part of the 5-tuple queue filters.<br>00b = TCP.<br>01b = UDP.<br>10b = SCTP.<br>11b = Other.<br>*Note:* Encoding of the protocol type for the 128 x 5-tuple filters is defined differently than the L4TYPE encoding for the flow director filters. |
| Priority | 4:2 | X | Priority value in case more than one 5-tuple filter matches.<br>000b = Reserved<br>001b = Lowest priority.<br>...<br>111b = Highest priority. |
| Reserved | 7:5 | X | Reserved. |
| Pool | 13:8 | X | The pool Index of the pool associated with this filter. |
| Reserved | 24:14 | X | Reserved for extension of the *Pool* field. |
| Mask | 29:25 | X | Mask bits for the 5-tuple fields (1b = don't compare).<br>The corresponding field participates in the match if the following bit is cleared:<br>Bit 25 = Mask source address comparison.<br>Bit 26 = Mask destination address comparison.<br>Bit 27 = Mask source port comparison.<br>Bit 28 = Mask destination port comparison.<br>Bit 29 = Mask protocol comparison. |
| Pool Mask | 30 | X | Mask bit for the *Pool* field.<br>When set to 1b, the *Pool* field is not compared as part of the 5-tuple filter. Software can clear (activate) the *Pool Mask* bit only when operating in IOV mode. |
| Queue Enable | 31 | X | When set, enables filtering of Rx packets by the 5-tuple defined in this filter to the queue indicated in register L34TIMIR.<br>*Note:* There are 128 different 5-tuple filter configuration registers sets, with indexes [0] to [127]. The mapping to a specific Rx queue is done by the Rx Queue field in the L34TIMIR register, and not by the index of the register set. |

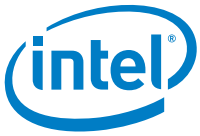
### 8.2.3.7.20 SYN Packet Queue Filter — SYNQF (0x0EC30; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Queue Enable | 0 | 0b | When set, enables routing of Rx packets to the queue indicated in this register. |
| Rx Queue | 7:1 | 0x0 | Identifies an Rx queue associated with SYN packets. |
| Reserved | 9:8 | 00b | Reserved for extension of the *Rx Queue* field. |
| Reserved | 30:10 | 0x0 | Reserved. |
| SYNQFP | 31 | 0b | Defines the priority between SYNQF and 5-tuples filter.<br>0b = 5-tuple filter priority<br>1b = SYN filter priority. |

### 8.2.3.7.21 EType Queue Filter — ETQF[n] (0x05128 + 4*n, n=0...7; RW)

See Section 7.1.2.3 for more details on the use of this register.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| EType | 15:0 | 0x0 | Identifies the protocol running on top of IEEE 802.<br>Used to route Rx packets containing this EtherType to a specific Rx queue.<br>*Note:* Field is defined in little endian (MS byte is first on the wire). |
| UP | 18:16 | 0x0 | User Priority.<br>A 802.1Q UP value to be compared against the *User Priority* field in the Rx packet. Enabled by the *UP Enable* bit. |
| UP Enable | 19 | 0b | User Priority Enable.<br>Enables comparison of the *User Priority* field in the received packet. |
| Pool | 25:20 | 0x0 | In virtualization modes, determines the target pool for the packet. |
| Pool Enable | 26 | 0b | In virtualization modes, enables the *Pool* field. |
| FCoE | 27 | 0b | When set, packets that match this filter are identified as FCoE packets. |
| Reserved | 28 | 0b | Reserved. |
| Reserved | 29 | 0b | Reserved. |
| 1588 time stamp | 30 | 0b | When set, packets with this EType are time stamped according to the IEEE 1588 specification. |
| Filter Enable | 31 | 0b | 0b = The filter is disabled for any functionality.<br>1b = The filter is enabled. Exact actions are determined by separate bits. |



### 8.2.3.7.22 EType Queue Select — ETQS[n] (0x0EC00 + 4*n, n=0...7; RW)

See Section 7.1.2.3 for more details on the use of this register.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 15:0 | 0x0 | Reserved. |
| Rx Queue | 22:16 | 0x0 | Identifies the Rx queue associated with this EType. |
| Reserved | 24:23 | 0x0 | Reserved for future extension of the *Rx Queue* field. |
| Reserved | 28:25 | 0x0 | Reserved. |
| Low Latency Interrupt | 29 | 0b | When set, packets that match this filter generate a LLI. |
| Reserved | 30 | 0x0 | Reserved. |
| Queue Enable | 31 | 0b | When set, enables filtering of Rx packets by the EType defined in this register to the queue indicated in this register. |

### 8.2.3.7.23 Rx Filter ECC Err Insertion 0 — RXFECCERR0 (0x051B8; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 8:0 | 0x1FF | Reserved. |
| ECCFLT_EN | 9 | 0b | Filter ECC Error indication Enablement.<br>When set to 1b, enables the ECC-INT + the RXF-blocking during ECC-ERR in one of the Rx filter memories. At 0b, the ECC logic can still function overcoming only single errors while dual or multiple errors can be ignored silently. |
| Reserved | 31:10 | 0x0 | Reserved. |

## 8.2.3.8 Receive DMA Registers

### 8.2.3.8.1 Receive Descriptor Base Address Low — RDBAL[n] (0x01000 + 0x40*n, n=0...63 and 0x0D000 + 0x40*(n-64), n=64...127; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| 0 | 6:0 | 0x0 | Ignored on writes. Returns 0x0 on reads. |
| RDBAL | 31:7 | X | Receive Descriptor Base Address Low. |

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.

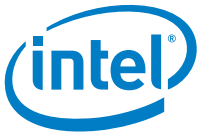
### 8.2.3.8.2 Receive Descriptor Base Address High — RDBAH[n] (0x01004 + 0x40*n, n=0...63 and 0x0D004 + 0x40*(n-64), n=64...127; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RDBAH | 31:0 | X | Receive Descriptor Base Address [63:32]. |

This register contains the upper 32 bits of the 64-bit descriptor base address.

### 8.2.3.8.3 Receive Descriptor Length — RDLEN[n] (0x01008 + 0x40*n, n=0...63 and 0x0D008 + 0x40*(n-64), n=64...127; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LEN | 19:0 | 0x0 | Descriptor Ring Length.<br>This register sets the number of bytes allocated for descriptors in the circular descriptor buffer. It must be 128-byte aligned (7 LS bit must be set to zero).<br>*Note:* Validated lengths up to 128 K (8 K descriptors). |
| Reserved | 31:20 | 0x0 | Reads as 0x0.<br>Should be written to 0x0 for future compatibility. |



### 8.2.3.8.4 Receive Descriptor Head — RDH[n] (0x01010 + 0x40*n, n=0...63 and 0x0D010 + 0x40*(n-64), n=64...127; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RDH | 15:0 | 0x0 | Receive Descriptor Head.<br>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 | Reserved.<br>Should be written with 0x0. |

### 8.2.3.8.5 Receive Descriptor Tail — RDT[n] (0x01018 + 0x40*n, n=0...63 and 0x0D018 + 0x40*(n-64), n=64...127; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RDT | 15:0 | 0x0 | Receive Descriptor Tail. |
| Reserved | 31:16 | 0x0 | Reads as 0x0.<br>Should be written to 0x0 for future compatibility. |

This register contains the tail pointer for the receive descriptor buffer. The register points to a 16-byte datum. Software writes the tail register to add receive descriptors to the hardware free list for the ring.

**Note:** The tail pointer should be set to one descriptor beyond the last empty descriptor in host descriptor ring.

### 8.2.3.8.6 Receive Descriptor Control — RXDCTL[n] (0x01028 + 0x40*n, n=0...63 and 0x0D028 + 0x40*(n-64), n=64...127; RW)

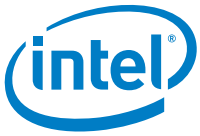
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 13:0 | 0x0 | Reserved. |
| Reserved | 14 | 0b | Reserved (softwaremight read and write in order to maintain backward compatibility.) |
| Reserved | 15 | 0b | Reserved. |
| Reserved | 22:16 | 0x0 | Reserved (software might read and write in order to maintain backward compatibility). |
| Reserved | 24:23 | 00b | Reserved. |
| ENABLE | 25 | 0b | Receive Queue Enable.<br>When set, the *ENABLE* bit enables the operation of the specific receive queue. Upon read it gets the actual status of the queue (internal indication that the queue is actually enabled/disabled). |

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 26 | 0b | Reserved (software can read and write in order to maintain backward compatibility). |
| Reserved | 29:27 | 0x0 | Reserved. |
| VME | 30 | 0b | VLAN Mode Enable.<br>0b = Do not strip VLAN tag.<br>1b = Strip VLAN tag from received 802.1Q packets destined to this queue. |
| Reserved | 31 | 0b | Reserved. |

### 8.2.3.8.7 Split Receive Control Registers — SRRCTL[n] (0x01014 + 0x40*n, n=0...63 and 0x0D014 + 0x40*(n-64), n=64...127 / 0x02100 + 4*n, [n=0...15]; RW)

SRRCTL[0...15] are also mapped to address 0x02100... to maintain compatibility with the 82598.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| BSIZEPACKET | 4:0 | 0x2 | Receive Buffer Size for Packet Buffer.<br>The value is in 1 KB resolution. Value can be from 1 KB to 16 KB. Default buffer size is 2 KB. This field should not be set to 0x0. This field should be greater or equal to 0x2 in queues where RSC is enabled. |
| Rsv | 7:5 | 000b | Reserved.<br>Should be written with 000b to ensure future compatibility. |
| BSIZEHEADER | 13:8 | 0x4 | Receive Buffer Size for Header Buffer.<br>The value is in 64 bytes resolution. Value can be from 64 bytes to 1024 bytes.<br>*Note:* The maximum supported header size is limited to 1023. Default buffer size is 256 bytes. This field must be greater than zero if the value of DESCTYPE is greater or equal to two.<br>Values above 1024 bytes are reserved for internal use only. |
| Reserved | 21:14 | 0x0 | Reserved. |
| RDMTS | 24:22 | 000b | Receive Descriptor Minimum Threshold Size.<br>A LLI associated with this queue is asserted each time the number of free descriptors is decreased to RDMTS * 64 (this event is considered as Rx ring buffer almost empty). |
| DESCTYPE | 27:25 | 000b | Define the descriptor type in Rx:<br>000b = Legacy.<br>001b = Advanced descriptor one buffer.<br>010b = Advanced descriptor header splitting.<br>011b = Reserved.<br>100b = Reserved.<br>101b = Advanced descriptor header splitting always use header buffer.<br>110b = Reserved.<br>111b = Reserved. |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Drop_En | 28 | 0b | Drop Enabled.<br>If set to 1b, packets received to the queue when no descriptors are available to store them are dropped. |
| Rsv | 31:29 | 000b | Reserved.<br>Should be written with 000b to ensure future compatibility. |

**Note:** BSIZEHEADER must be bigger than zero if DESCTYPE is equal to 010b, 011b, 100b or 101b.

### 8.2.3.8.8 Receive DMA Control Register — RDRXCTL (0x02F00; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| CRCStrip | 1 | 0 | Rx CRC Strip indication to the Rx DMA unit. This bit must be set the same as HLREG0.RXCRCSTRP.<br>0b = No CRC Strip by HW (Default).<br>1b = Strip CRC by HW. |
| Reserved | 2 | 0 | Reserved. |
| DMAIDONE | 3 | 0b | DMA Init Done.<br>When read as 1b, indicates that the DMA initialization cycle is done (RO). |
| Reserved | 16:4 | 0x0880 | Reserved. |
| RSCFRSTSIZE | 21:17 | 0x8 | Defines a minimum packet size (after VLAN stripping, if applicable) for a packet with a payload that can open a new RSC (in units of 16 byte.).<br>See RSCDBU.RSCACKDIS for packets without payload.<br>*Note:* RSCFRSTSIZE is reserved for internal use. Software should set this field to 0x0. |
| Reserved | 24:22 | 000b | Reserved. |
| RSCACKC | 25 | 0b | RSC Coalescing on ACK Change.<br>When set, an active RSC completes when the *ACK* bit in the Rx packet is different than the *ACK* bit in the RSC context. When cleared, an active RSC completes only when the *ACK* bit in the Rx packet is cleared while the *ACK* bit in the RSC context is set.<br>*Note:* RSCACKC is reserved for internal use. Software should set this bit to 1b. |
| FCOE_WRFIX | 26 | 0b | FCoE Write Exchange Fix.<br>When set, DDP context of FC write exchange is closed following a reception of a last packet in a sequence with an active *Sequence Initiative* bit in the F_CTL field. When cleared, the DDP context is not closed.<br>*Note:* FCOE_WRFIX is reserved for internal use. Software should set this bit to 1b. |
| Reserved | 31:27 | 0 | Reserved. |

### 8.2.3.8.9 Receive Packet Buffer Size — RXPBSIZE[n] (0x03C00 + 4*n, n=0...7; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 9:0 | 0x0 | Reserved. |
| SIZE | 19:10 | 0x200 | Receive Packet Buffer Size for traffic class 'n' while 'n' is the register index.<br>The size is defined in KB units and the default size of PB[0] is 512 KB. The default size of PB[1-7] is also 512 KB but it is meaningless in non-DCB mode.<br>When DCB mode is enabled the size of PB[1-7] must be set to meaningful values. The total meaningful allocated PB sizes plus the size allocated to the flow director filters should be less or equal to 512 KB.<br>Possible PB allocation in DCB mode for 8 x TCs is 0x40 (64 KB) for all PBs. Other possible setting of 4 x TCs is 0x80 (128 KB) for all PB[0-3] and 0x0 for PB[4-7].<br>See Section 3.7.7.3.5 for other optional settings with/without the flow director filters<br>*Note:* In any setting the RXPBSIZE[0] must always be enabled (greater than zero). |
| Reserved | 31:20 | 0x0 | Reserved. |

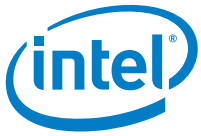
### 8.2.3.8.10 Receive Control Register — RXCTRL (0x03000; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RXEN | 0 | 0b | Receive Enable.<br>When set to 0b, filter inputs to the packet buffer are ignored. |
| Reserved | 31:1 | 0x0 | Reserved |

### 8.2.3.8.11 Rx Packet Buffer Flush Detect — RXMEMWRAP (0x03190; RO)

This register is used by software as part of a queue disable procedure (described in Section 4.6.7.1)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TC0Wrap | 2:0 | 000b | Packet Buffer 0 Wrap Around Counter.<br>A 3-bit counter that increments on each full cycle through the buffer. Once reaching 111b, the counter warps around to 000b on the next count. |
| TC0Empty | 3 | 1b | Packet Buffer 0 Empty<br>0b = Packet buffer is not empty.<br>1b = Packet buffer is empty. |
| TC1Wrap | 6:4 | 000b | Packet Buffer 1 Wrap Around Counter.<br>A 3-bit counter that increments on each full cycle through the buffer. Once reaching 111b, the counter warps around to 000b on the next count. |
| TC1Empty | 7 | 1b | Packet Buffer 1 Empty<br>0b = Packet buffer is not empty.<br>1b = Packet buffer is empty. |



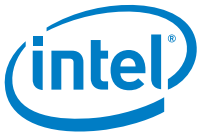
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TC2Wrap | 10:8 | 000b | Packet Buffer 2 Wrap Around Counter.<br>A 3-bit counter that increments on each full cycle through the buffer. Once reaching 111b, the counter warps around to 000b on the next count. |
| TC2Empty | 11 | 1b | Packet Buffer 2 Empty<br>0b = Packet buffer is not empty.<br>1b = Packet buffer is empty. |
| TC3Wrap | 14:12 | 000b | Packet Buffer 3 Wrap Around Counter.<br>A 3-bit counter that increments on each full cycle through the buffer. Once reaching 111b, the counter warps around to 000b on the next count. |
| TC3Empty | 15 | 1b | Packet Buffer 3 Empty<br>0b = Packet buffer is not empty.<br>1b = Packet buffer is empty. |
| TC4Wrap | 18:16 | 000b | Packet Buffer 4 Wrap Around Counter.<br>A 3-bit counter that increments on each full cycle through the buffer. Once reaching 111b, the counter warps around to 000b on the next count. |
| TC4Empty | 19 | 1b | Packet Buffer 4 Empty<br>0b = Packet buffer is not empty.<br>1b = Packet buffer is empty. |
| TC5Wrap | 22:20 | 000b | Packet Buffer 5 Wrap Around Counter.<br>A 3-bit counter that increments on each full cycle through the buffer. Once reaching 111b, the counter warps around to 000b on the next count. |
| TC5Empty | 23 | 1b | Packet Buffer 5 Empty<br>0b = Packet buffer is not empty.<br>1b = Packet buffer is empty. |
| TC6Wrap | 26:24 | 000b | Packet Buffer 6 Wrap Around Counter.<br>A 3-bit counter that increments on each full cycle through the buffer. Once reaching 111b, the counter warps around to 000b on the next count. |
| TC6Empty | 27 | 1b | Packet Buffer 6 Empty<br>0b = Packet buffer is not empty.<br>1b = Packet buffer is empty. |
| TC7Wrap | 30:28 | 000b | Packet Buffer 7 Wrap Around Counter.<br>A 3-bit counter that increments on each full cycle through the buffer. Once reaching 111b, the counter warps around to 000b on the next count. |
| TC7Empty | 31 | 1b | Packet Buffer 7 Empty<br>0b = Packet buffer is not empty.<br>1b = Packet buffer is empty. |

### 8.2.3.8.12 RSC Data Buffer Control Register — RSCDBU (0x03028; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 6:0 | 0x20 | Reserved. |
| RSCACKDIS | 7 | 0b | Disable RSC for ACK Packets.<br>disables the coalescing of TCP packets without TCP payload. This bit should be set if performance problems are found. |
| Reserved | 31:8 | 0x0 | Reserved. |

### 8.2.3.8.13 RSC Control — RSCCTL[n] (0x0102C + 0x40*n, n=0...63 and 0x0D02C + 0x40*(n-64), n=64...127; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RSCEN | 0 | 0b | RSC Enable.<br>When the RSCEN bit is set, RSC is enabled on this queue. |
| Reserved | 1 | 0b | Reserved |
| MAXDESC | 3:2 | 00b | Maximum descriptors per Large receive as follow:<br>00b = Maximum of 1 descriptor per large receive.<br>01b = Maximum of 4 descriptors per large receive.<br>10b = Maximum of 8 descriptors per large receive.<br>11b = Maximum of 16 descriptors per large receive.<br>*Note:* MAXDESC * SRRCTL.BSIZEPKT must not exceed 64 KB minus one, which is the maximum total length in the IP header and must be larger than the expected received MSS. |
| Reserved | 31:4 | 0x0 | Reserved. |



## 8.2.3.9 Transmit Registers

### 8.2.3.9.1 DMA Tx TCP Max Allow Size Requests — DTXMXSZRQ (0x08100; RW)

This register limits the total number of data bytes that may 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 will be serviced right after the current outstanding requests are completed.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Max_bytes_num_req | 11:0 | 0x10 | Max allowed number of bytes requests.<br>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 | Reserved. |

### 8.2.3.9.2 DMA Tx Control — DMATXCTL (0x04A80; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TE | 0 | 0b | Transmit Enable.<br>When set, this bit enables the transmit operation of the DMA-Tx. |
| Reserved | 1 | 0b | Reserved. |
| Reserved | 2 | 1b | Reserved. |
| GDV | 3 | 0b | Global Double VLAN Mode.<br>When set, this bit enables the Double VLAN mode. |
| Reserved | 15:4 | 0x0 | Reserved. |
| VT | 31:16 | 0x8100 | VLAN Ether-Type (the VLAN Tag Protocol Identifier — TPID).<br>For proper operation, software must not change the default setting of this field (802.3ac standard defines it as 0x8100).<br>This field must be set to the same value as the VET field in the VLNCTRL register. |

### 8.2.3.9.3 DMA Tx TCP Flags Control Low — DTXTCPFLGL (0x04A88; RW)

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 Val | Description |
|---|---|---|---|
| TCP_flg_first_seg | 11:0 | 0xFF6 | TCP Flags First Segment.<br>Bits that make AND operation with the TCP flags at TCP header in the first segment. |
| Reserved | 15:12 | 0x0 | Reserved. |
| TCP_Flg_mid_seg | 27:16 | 0xFF6 | TCP Flags Middle Segments.<br>The low bits that make AND operation with the TCP flags at TCP header in the middle segments. |
| Reserved | 31:28 | 0x0 | Reserved. |

### 8.2.3.9.4 DMA Tx TCP Flags Control High- DTXTCPFLGH (0x04A8C; RW)

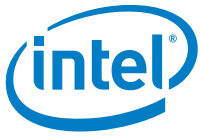
This register holds the mask bits for the TCP flags in Tx segmentation (described in Section 7.2.4.7.3).

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TCP_Flg_lst_seg | 11:0 | 0xF7F | TCP Flags Last Segment.<br>Bits that make AND operation with the TCP flags at TCP header in the last segment. |
| Reserved | 31:12 | 0x0 | Reserved. |

### 8.2.3.9.5 Transmit Descriptor Base Address Low — TDBAL[n] (0x06000+0x40*n, n=0...127; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| 0 | 6:0 | 0b | Ignored on writes. Returns 0b on reads. |
| TDBAL | 31:7 | X | Transmit Descriptor Base Address Low. |

This register contains the lower bits of the 64-bit descriptor base address. The lower seven bits are ignored. The Transmit Descriptor Base Address must point to a 128 byte-aligned block of data.



### 8.2.3.9.6 Transmit Descriptor Base Address High — TDBAH[n] (0x06004+0x40*n, n=0...127; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TDBAH | 31:0 | X | Transmit Descriptor Base Address [63:32]. |

This register contains the upper 32 bits of the 64 bit Descriptor base address.

### 8.2.3.9.7 Transmit Descriptor Length — TDLEN[n] (0x06008+0x40*n, n=0...127; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LEN | 19:0 | 0x0 | Descriptor Ring Length.<br>This register sets the number of bytes allocated for descriptors in the circular descriptor buffer. It must be 128byte-aligned (7 LS bit must be set to zero).<br>*Note:* Validated Lengths up to 128K (8K descriptors). |
| Reserved | 31:20 | 0x0 | Reads as 0x0. Should be written to 0x0. |

### 8.2.3.9.8 Transmit Descriptor Head — TDH[n] (0x06010+0x40*n, n=0...127; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TDH | 15:0 | 0x0 | Transmit Descriptor Head. |
| Reserved | 31:16 | 0x0 | Reserved. Should be written with 0x0. |

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 the hardware could become confused.

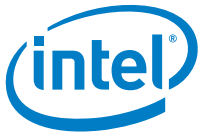
### 8.2.3.9.9 Transmit Descriptor Tail — TDT[n] (0x06018+0x40*n, n=0...127; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TDT | 15:0 | 0x0 | Transmit Descriptor Tail. |
| Reserved | 31:16 | 0x0 | Reads as 0x0. Should be written to 0x0for future compatibility. |

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.

### 8.2.3.9.10 Transmit Descriptor Control — TXDCTL[n] (0x06028+0x40*n, n=0...127; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PTHRESH | 6:0 | 0x0 | Pre-Fetch Threshold<br>Controls when a prefetch of descriptors is considered. This threshold refers to the number of valid, unprocessed transmit descriptors the 82599 has in its on-chip buffer. If this number drops below PTHRESH, the algorithm considers pre-fetching descriptors from host memory. However, this fetch does not happen unless there are at least HTHRESH valid descriptors in host memory to fetch.<br>*Note:* HTHRESH should be given a non-zero value each time PTHRESH is used. |
| Rsv | 7 | 0x0 | Reserved. |
| HTHRESH | 14:8 | 0x0 | Host Threshold. |
| Rsv | 15 | 0x0 | Reserved. |
| WTHRESH | 22:16 | 0x0 | Write-Back Threshold.<br>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.<br>*Note:* Since the default value for write-back threshold is 0b, descriptors are normally written back as soon as they are processed. WTHRESH must be written to a non-zero value to take advantage of the write-back bursting capabilities of the 82599.<br>*Note:* When WTHRESH is set to a non-zero value, the software 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 Tx descriptors with the EOP bit set and at least once in the 40 descriptors.<br>*Note:* When Head write-back is enabled (TDWBAL[n].Head_WB_En = 1b), the WTHRESH must be set to zero. |
| Reserved | 24:23 | 0x0 | Reserved. |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| ENABLE | 25 | 0b | Transmit Queue Enable.<br>When set, this bit enables the operation of a specific transmit queue:<br>Default value for all queues is 0b.<br>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.<br>When disabling a queue, this bit is cleared only after all activity at the queue stopped.<br>*Note:* This bit is set only when the queue is enabled.<br>Upon read – get the actual status of the queue (internal indication that the queue is actually enabled/disabled)<br>*Note:* When setting the global Tx enable DMATXCTL.TE the ENABLE bit of Tx queue zero is enabled as well. |
| SWFLSH | 26 | 0b | Transmit Software Flush. This bit enables software to trigger descriptor write-back flushing, independently of other conditions.<br>This bit is self cleared by hardware. |
| Reserved | 27 | 0b | Reserved. |
| Reserved | 28 | 0b | Reserved. |
| Reserved | 29 | 0b | Reserved. |
| Reserved | 31:30 | 0x0 | Reserved. |

This register controls the fetching and write-back of transmit descriptors. The three threshold values are used to determine when descriptors is read from and written to host memory.

**Note:** When WTHRESH = 0b only descriptors with the *RS* bit set are written back.

For PTHRESH and HTHRESH recommended setting please refer to Section 7.2.3.4.

## 8.2.3.9.11 Tx Descriptor Completion Write Back Address Low — TDWBAL[n] (0x06038+0x40*n, n=0...127; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Head_WB_En | 0 | 0b | Head Write-Back Enable.<br>0b = Head write-back is disabled.<br>1b = Head write-back is enabled.<br>When head_WB_en is set, the 82599 does not write-back Tx descriptors. |
| Reserved | 1 | 0 | Reserved. |
| HeadWB_Low | 31:2 | 0x0 | Lowest 32 bits of the head write-back memory location (Dword aligned).<br>Last 2 bits of this field are ignored and are always read as 0.0, meaning that the actual address is Qword aligned. |

### 8.2.3.9.12 Tx Descriptor Completion Write Back Address High — TDWBAH[n] (0x0603C+0x40*n, n=0...127; RW)

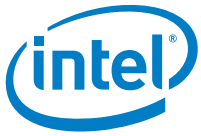
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| HeadWB_High | 31:0 | 0x0 | Highest 32 bits of head write-back memory location (for 64-bit addressing). |

### 8.2.3.9.13 Transmit Packet Buffer Size — TXPBSIZE[n] (0x0CC00 + 0x4*n, n=0...7; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 9:0 | 0x0 | Reserved. |
| SIZE | 19:10 | 0xA0 (160 KB) | Transmit packet buffer size of TCn.<br>At default setting (no DCB) only packet buffer 0 is enabled and TXPBSIZE values for TC 1-7 are meaningless.<br>Other than the default configuration the 82599 supports partitioned configurations when DCB is enabled.<br>Symmetrical 8 TCs partitioning: 0x14 (20KB) for TXPBSIZE[0...7].<br>Symmetrical 4 TCs partitioning: 0x28 (40KB) for TXPBSIZE[0...3] and 0x0 (0KB) for TXPBSIZE[4...7].<br>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) |
| Reserved | 31:20 | 0x0 | Reserved. |

### 8.2.3.9.14 Manageability Transmit TC Mapping — MNGTXMAP (0x0CD10; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MAP | 2:0 | 0x0 | Map value indicates the TC that the transmit manageability traffic is routed to. |
| Reserved | 31:3 | 0x0 | Reserved. |



### 8.2.3.9.15 Multiple Transmit Queues Command Register — MTQC (0x08120; RW)

This register can be modified only as part of the init phase.

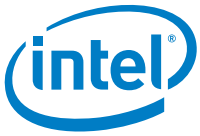
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RT_Ena | 0 | 0b | DCB Enabled Mode.<br>See functionality in the following table. |
| VT_Ena | 1 | 0b | Virtualization Enabled Mode.<br>When set, the 82599 supports either 16, 32, or 64 pools. See functionality in the following table.<br>This bit should be set the same as PFVTCTL.VT_Ena. |
| NUM_TC_OR_Q | 3:2 | 00b | Number of TCs or Number of Tx Queues per Pools.<br>See functionality in the following table. |
| Reserved | 31:4 | 0x0 | Reserved. |

Permitted value and functionality of: RT_Ena; VT_Ena; NUM_TC_OR_Q. For Tx queue assignment in DCB and VT modes refer to Table 7-25 in Section 7.2.1.2.1.

| Device Setting | | | Device Functionality | |
|---|---|---|---|---|
| RT_Ena | VT_Ena | NUM_TC_OR_Q | Tx Queues | TC & VT |
| 0b | 0b | 00b | 0 — 63 | - |
| <> 0b | <> 0b | 00b | Reserved | |
| 0b | 0b | <> 00 | Reserved | |
| 1b | 0b | 01b | Reserved | |
| 1b | 0b | 10b | 0 — 127 | TC0 — TC3 |
| 1b | 0b | 11b | 0 — 127 | TC0 — TC7 |
| 0b | 1b | 01b | 0 — 127 | 64 VMs |
| 0b | 1b | 10b | 0 — 127 | 32 VMs |
| 0b | 1b | 11b | Reserved | |
| 1b | 1b | 01b | Reserved | |
| 1b | 1b | 10b | 0 — 127 | TC0 — TC3 & 32 VMs |
| 1b | 1b | 11b | 0 — 127 | TC0 — TC7 & 16 VMs |

### 8.2.3.9.16 Tx Packet Buffer Threshold — TXPBTHRESH (0x04950 +0x4*n, n=0...7; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| THRESH | 9:0 | 0x96/0x0 | Threshold used for checking room place in Tx packet buffer of TCn.<br>Threshold in KB units, when the packet buffer is filled up with payload over that threshold, no more data read request is sent.<br>Default values:<br>0x96 (150 KB) for TXPBSIZE0.<br>0x0 (0 KB) for TXPBSIZE1-7.<br>It should be set to: (packet buffer size) — MSS.<br>For instance, if packet buffer size is set to 20 KB in corresponding TXPBSIZE.SIZE, if MSS of 9.5 KB (9728-byte) jumbo frames is supported for TCn, it is set to: 0xA (10 KB). |
| Reserved | 31:10 | 0x0 | Reserved. |



## 8.2.3.10 DCB Registers

DCB registers are owned by the PF in an IOV mode.

### 8.2.3.10.1 DCB Receive Packet Plane Control and Status — RTRPCS (0x02430; RW)

RTRPCS is equivalent to the 82598's RMCS.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 0 | 0b | Reserved. |
| RRM | 1 | 0b | Receive Recycle Mode defines the recycle mode within a BWG.<br>0b = No recycle.<br>1b = Recycle within the BWG. It is the only supported mode when DCB is enabled. |
| RAC | 2 | 0b | Receive Arbitration Control.<br>0b = Round Robin (RR).<br>1b = Weighted Strict Priority (WSP). |
| Reserved | 5:3 | 0x0 | Reserved. |
| Reserved | 15:6 | 0x0 | Reserved. |
| LRPB | 18:16 | 0x0 | Last Received Packet Buffer Status Indication.<br>Indicates the last packet buffer that was used in Rx arbiter. |
| Reserved | 26:19 | 0x0 | Reserved. |
| Reserved | 27 | 0b | Reserved |
| Reserved | 31:28 | 0x6 | Reserved |

### 8.2.3.10.2 DCB Transmit Descriptor Plane Control and Status — RTTDCS (0x04900; RW) DMA-Tx

RTTDCS was DPMCS mapped to 0x07F40 in the 82598.

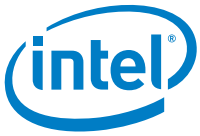
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TDPAC | 0 | 0b | TC Transmit Descriptor Plane Arbitration Control.<br>0b = RR<br>1b = WSP |
| VMPAC | 1 | 0b | VM Transmit Descriptor Plane Arbitration Control.<br>0b = RR<br>1b = Weighted Round Robin (WRR). |
| Reserved | 3:2 | 00b | Reserved. |

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TDRM | 4 | 0b | TC Transmit descriptor plane recycle mode defines the recycle mode within a BWG.<br>0b = No recycle.<br>1b = Recycle within the BWG. It is the only supported mode. |
| Reserved | 5 | 0b | Reserved. |
| ARBDIS | 6 | 0 | DCB Arbiters Disable.<br>When set to 1 this bit pauses the Tx Descriptor plane arbitration state-machine. Therefore, during nominal operation this bit should be set to 0. |
| Reserved | 16:7 | 0 | Reserved. |
| LTTDESC (RO) | 19:17 | 0x0 | Last Transmitted TC (RO).<br>This field indicates the last transmitted TC in XMIT descriptor arbiter DMA. |
| Reserved | 21:20 | 00b | Reserved. |
| BDPM | 22 | 1b | Bypass Data_Pipe Monitor. In order to enable bypassing the above limit.<br>In DCB mode, this bit must be cleared. |
| BPBFSM | 23 | 1b | Bypass Packet Buffer Free Space Monitor.<br>In order to enable bypassing the packet buffer free space monitor (not checking if there is enough free space in the packet buffer before requesting the data).<br>This bit must be cleared in DCB mode or SR-IOV mode. |
| Reserved | 30:24 | 0x0 | Reserved. |
| SPEED_CHG | 31 | 0b | Link speed has changed. Read and clear flag.<br>Set by hardware to indicate that the link speed has changed.<br>Cleared by software at the end of the link speed change procedure. Refer to Section 4.6.11.2. |

### 8.2.3.10.3 DCB Transmit Packet Plane Control and Status- RTTPCS (0x0CD00; RW)

RTTPCS is mapped to 0x0CD00 for compatibility with the 82598's PDPMCS.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 4:0 | 0x0 | Reserved |
| TPPAC | 5 | 0b | Transmit Packet Plane Arbitration Control<br>0b = RR (with respect to stop markers).<br>1b = Strict Priority (SP), with respect to stop markers) |
| Reserved | 7:6 | 00b | Reserved. |
| TPRM | 8 | 0b | Transmit packet plane recycle mode defines the recycle mode within a BWG.<br>0b = No recycle.<br>1b = Recycle within the BWG. |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 21:9 | 0x0 | Reserved. |
| ARBD | 31:22 | 0x224 | ARB_delay. Minimum cycles delay between a packet's arbitration.<br>When RTTPCS.TPPAC is set to 1b the arbitration delay is according to ARBD, otherwise the arbitration delay is 0x0. Should be kept at default in non-DCB mode. In DCB mode, should be set to 0x004. |

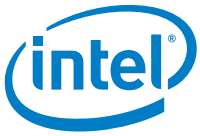
### 8.2.3.10.4 DCB Receive User Priority to Traffic Class — RTRUP2TC (0x03020; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| UP0MAP | 2:0 | 0x0 | Receive UP 0 to TC Mapping.<br>When set to n, UP 0 is bound to TC n.<br>Used for two purposes:<br>• Define into which Rx packet buffer incoming traffic carrying 802.1p field set to 0 is routed.<br>• Define according to the filling status of which Rx packet puffer a Priority Flow Control (PFC) frame with the *Timer 0* field and Class Enable Vector bit 0 set is sent. |
| UP1MAP | 5:3 | 0x0 | Receive UP 1 to TC Mapping.<br>When set to n, UP 1 is bound to TC n.<br>Used for two purposes:<br>• Define into which Rx packet buffer incoming traffic carrying 802.1p field set to 1 is routed.<br>• Define according to the filling status of which Rx Packet Buffer a P FC frame with the *Timer 1* field and *Class Enable Vector* bit 1 set is sent. |
| UP2MAP | 8:6 | 0x0 | Receive UP 2 to TC Mapping.<br>When set to n, UP 2 is bound to TC n.<br>Used for two purposes:<br>• Define into which Rx packet buffer incoming traffic carrying 802.1p field set to 2 is routed.<br>• Define according to the filling status of which Rx Packet Buffer a PFC frame with the *Timer 2* field and *Class Enable Vector* bit 2 set is sent. |
| UP3MAP | 11:9 | 0x0 | Receive UP 3 to TC Mapping.<br>When set to n, UP 3 is bound to TC n.<br>Used for two purposes:<br>• Define into which Rx packet buffer incoming traffic carrying 802.1p field set to 3 is routed.<br>• Define according to the filling status of which Rx packet buffer a PFC frame with the *Timer 3* field and *Class Enable Vector* bit 3 set is sent. |
| UP4MAP | 14:12 | 0x0 | Receive UP 4 to TC Mapping.<br>When set to n, UP 4 is bound to TC n.<br>Used for two purposes:<br>• Define into which Rx packet buffer incoming traffic carrying 802.1p field set to 4 is routed.<br>• Define according to the filling status of which Rx packet buffer a PFC frame with the *Timer 4* field and *Class Enable Vector* bit 4 set is sent. |

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| UP5MAP | 17:15 | 0x0 | Receive UP 5 to TC Mapping.<br>When set to n, UP 5 is bound to TC n.<br>Used for two purposes:<br>• Define into which Rx packet buffer incoming traffic carrying 802.1p field set to 5 is routed.<br>• Define according to the filling status of which Rx packet buffer a PFC frame with the *Timer 5* field and *Class Enable Vector* bit 5 set is sent. |
| UP6MAP | 20:18 | 0x0 | Receive UP 6 to TC Mapping.<br>When set to n, UP 6 is bound to TC n.<br>Used for two purposes:<br>• Define into which Rx packet buffer incoming traffic carrying 802.1p field set to 6 is routed.<br>• Define according to the filling status of which Rx packet buffer a PFC frame with the *Timer 6* field and *Class Enable Vector* bit 6 set is sent. |
| UP7MAP | 23:21 | 0x0 | Receive UP 7 to TC Mapping.<br>When set to n, UP 7 is bound to TC n.<br>Used for two purposes:<br>• Define into which Rx packet buffer incoming traffic carrying 802.1p field set to 7 is routed.<br>• Define according to the filling status of which Rx packet buffer a PFC frame with the *Timer 7* field and *Class Enable Vector* bit 7 set is sent. |
| Reserved | 31:24 | 0x0 | Reserved. |

### 8.2.3.10.5 DCB Transmit User Priority to Traffic Class — RTTUP2TC (0x0C800; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| UP0MAP | 2:0 | 0x0 | Transmit UP 0 to TC Mapping.<br>When set to n, UP 0 is bound to TC n.<br>Used when receiving a PFC frame with the *Timer 0* field and *Class Enable Vector* bit 0 set, to determine which TC must be paused. |
| UP1MAP | 5:3 | 0x0 | Transmit UP 1 to TC Mapping.<br>When set to n, UP 1 is bound to TC n.<br>Used when receiving a PFC frame with the *Timer 1* field and *Class Enable Vector* bit 1 set, to determine which TC must be paused. |
| UP2MAP | 8:6 | 0x0 | Transmit UP 2 to TC Mapping.<br>When set to n, UP 2 is bound to TC n.<br>Used when receiving a PFC frame with the *Timer 2* field and *Class Enable Vector* bit 2 set, to determine which TC must be paused. |
| UP3MAP | 11:9 | 0x0 | Transmit UP 3 to TC Mapping.<br>When set to n, UP 3 is bound to TC n.<br>Used when receiving a PFC frame with the *Timer 3* field and *Class Enable Vector* bit 3 set, to determine which TC must be paused. |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| UP4MAP | 14:12 | 0x0 | Transmit UP 4 to TC Mapping.<br>When set to n, UP 4 is bound to TC n.<br>Used when receiving a PFC frame with the *Timer 4* field and *Class Enable Vector* bit 4 set, to determine which traffic class must be paused. |
| UP5MAP | 17:15 | 0x0 | Transmit UP 5 to TC Mapping.<br>When set to n, UP 5 is bound to TC n.<br>Used when receiving a PFC frame with the *Timer 5* field and *Class Enable Vector* bit 5 set, to determine which traffic class must be paused. |
| UP6MAP | 20:18 | 0x0 | Transmit UP 6 to TC Mapping.<br>When set to n, UP 6 is bound to V n.<br>Used when receiving a PFC frame with the *Timer 6* field and *Class Enable Vector* bit 6 set, to determine which traffic class must be paused. |
| UP7MAP | 23:21 | 0x0 | Transmit UP 7 to TC Mapping.<br>When set to n, UP 7 is bound to TC n.<br>Used when receiving a PFC frame with the *Timer 7* field and *Class Enable Vector* bit 7 set, to determine which TC must be paused. |
| Reserved | 31:24 | 0x0 | Reserved. |

### 8.2.3.10.6 DCB Receive Packet Plane T4 Config — RTRPT4C[n] (0x02140 + 4*n, n=0...7; RW)

RTRPT4C is equivalent to the 82598's RT2CR.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| CRQ | 8:0 | 0x0 | Credit Refill Quantum.<br>Amount of credits to refill in 64-byte granularity.<br>Possible values 0x000:0x1FF (0 to 32,704 bytes). |
| BWG | 11:9 | 0x0 | Bandwidth Group Index.<br>Bandwidth Group (BWG). |
| MCL | 23:12 | 0x0 | Max Credit Limit.<br>Maximum amount of credits for a configured packet buffer in 64-byte granularity.<br>Possible values 0x000:0xFFF (0to 262,080bytes). |
| Reserved | 29:24 | 0x0 | Reserved. |
| GSP | 30 | 0b | Group Strict Priority.<br>When set to 1b enables strict priority to the appropriate packet buffer over any traffic of other packet buffers within the group. |
| LSP | 31 | 0b | Link Strict Priority.<br>If set to 1b enables strict priority to the appropriate packet buffer over any traffic of other packet buffers. |

### 8.2.3.10.7 Strict Low Latency Tx Queues — TXLLQ[n] (0x082E0 + 4*n, n=0...3; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Strict Low latency | 31:0 | 0x0 | Strict Low Latency Enable.<br>When set, defines the relevant Tx queue as strict low latency. All queues belong to a LSP TC must be set as strict low latency queues.<br>Bit 'm' in register 'n' correspond to Tx queue 32 x 'n' + 'm'. |

### 8.2.3.10.8 DCB Receive Packet Plane T4 Status — RTRPT4S[n] (0x02160 + 4*n, n=0...7; RO)

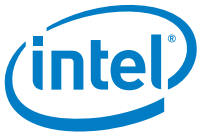
RTRPT4S is equivalent to the 82598's RT2SR.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 31:0 | 0x0 | Reserved. |

### 8.2.3.10.9 DCB Transmit Descriptor Plane T2 Config - RTTDT2C[n] (0x04910 + 4*n, n=0...7; RW) DMA-Tx

RTTDT2C was TDTQ2TCCR in the 82598 at 0x0602C + 0x40*n, n=0...7.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| CRQ | 8:0 | 0x0 | Credit Refill Quantum.<br>Amount of credits to refill the TC in 64-byte granularity<br>Possible values 0x000 – 0x1FF (0 – 32,704 bytes) |
| BWG | 11:9 | 0x0 | Bandwidth Group Index.<br>Assignment of this TC to a bandwidth group. |
| MCL | 23:12 | 0x0 | Max Credit Limit.<br>Max amount of credits for a configured TC in 64-byte granularity<br>Possible values 0x000 – 0xFFF (0 – 262,080 bytes) |
| Reserved | 29:24 | 0x0 | Reserved. |
| GSP | 30 | 0b | Group Strict Priority.<br>When set to 1b enables strict priority to the appropriate TC over any traffic of other TCs within the group. |
| LSP | 31 | 0b | Link Strict Priority.<br>When set to 1b enables strict priority to the appropriate TC over any traffic of other TCs. |



### 8.2.3.10.10 DCB Transmit Packet Plane T2 Config — RTTPT2C[n] (0x0CD20 + 4*n, n=0...7; RW)

RTTPT2C is mapped to 0x0CD20 + 4*n [n=0...7] for compatibility with the 82598's TDPT2TCCR.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| CRQ | 8:0 | 0x0 | Credit Refill Quantum.<br>Amount of credits to refill the TC in 64-byte granularity.<br>Possible values 0x000: 0x1FF (0 to 32,704 bytes). |
| BWG | 11:9 | 0x0 | Bandwidth Group.<br>Assignment of this TC to a BWG. |
| MCL | 23:12 | 0x0 | Max Credit Limit.<br>Max amount of credits for a configured TC in 64-byte granularity.<br>Possible values 0x000:0xFFF (0 – 262,080 bytes). |
| Reserved | 29:24 | 0x0 | Reserved. |
| GSP | 30 | 0b | Group Strict Priority.<br>When set to 1b enables strict priority to the appropriate TC over any traffic of other TCs within the group. |
| LSP | 31 | 0b | Link Strict Priority.<br>When set to 1b enables strict priority to the appropriate TC over any traffic of other TCs. |

### 8.2.3.10.11 DCB Transmit Packet plane T2 Status — RTTPT2S[n] (0x0CD40 + 4*n, n=0...7; RO)

RTTPT2S is mapped to 0x0CD40 + 4*n [n=0...7] for compatibility with the 82598's TDPT2TCSR.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 31:0 | 0x0 | Reserved. |

### 8.2.3.10.12 DCB Transmit Rate–Scheduler MMW — RTTBCNRM (0x04980; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MMW_SIZE | 10:0 | 0x0 | Maximum memory window size for the rate-scheduler (for all Tx queues).<br>This is the maximum amount of 1 KB units of payload compensation time that can be accumulated for Tx queues attached to TCn. This number must be multiplied by the rate-factor of the Tx queue before performing the MMW saturation check for that queue. |
| Reserved | 31:11 | 0x0 | Reserved. |

### 8.2.3.10.13 DCB Transmit Descriptor Plane Queue Select — RTTDQSEL (0x04904; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TXDQ_IDX | 6:0 | 0x0 | Tx Descriptor Queue Index or TX Pool of Queues Index<br>This register is used to set VM and Transmit Scheduler parameters that are configured per Tx queue or per Tx pool of queues via indirect access. It means that prior to read or write access such registers, software has to make sure this field contains the index of the Tx queue or Tx pool of queue to be accessed.<br>When DCB is disabled, VM parameters include a pool of Tx queues. As a result, this field points to the index of the pool (and not a queue index).<br>When DCB is enabled, and/or when programming rate limiters, this field points to a Tx queue index.<br>The registers that are affected by this index are:<br>RTTDT1C, RTTDT1S, RTTBCNRC, RTTBCNRS |
| Reserved | 31:7 | 0x0 | Reserved. |

### 8.2.3.10.14 DCB Transmit Descriptor Plane T1 Config — RTTDT1C (0x04908; RW)

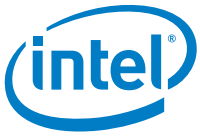
128 internal registers indirectly addressed via RTTDQSEL.TXDQ_IDX. When DCB is disabled, configure the pool index with the credits allocated to the entire pool.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| CRQ | 13:0 | X | Credit Refill Quantum.<br>Amount of credits to refill the VM in 64-byte granularity.<br>Possible values 0x000:0x3FFF (0 to 1,048,512 bytes). |
| Reserved | 31:14 | 0x0 | Reserved. |

### 8.2.3.10.15 DCB Transmit Descriptor Plane T1 Status — RTTDT1S (0x0490C; RO)

128 internal registers indirectly addressed via RTTDQSEL.TXDQ_IDX.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 31:0 | 0x0 | Reserved. |



### 8.2.3.10.16 DCB Transmit Rate-Scheduler Config — RTTBCNRC (0x04984; RW)

128 internal registers indirectly addressed via RTTDQSEL.TXDQ_IDX.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RF_DEC | 13:0 | X | Tx rate-scheduler rate factor hexadecimal part, for the Tx queue indexed by TXDQ_IDX field in the RTTDQSEL register.<br>Rate factor bits that come after the hexadecimal point.<br>Meaningful only if the *RS_ENA* bit is set.<br>When RTTBCNRD.DRIFT_ENA is set, this field is periodically modified by hardware as well. |
| RF_INT | 23:14 | X | Tx rate-scheduler rate factor integral part, for the Tx queue indexed by TXDQ_IDX field in the RTTDQSEL register<br>Rate factor bits that come before the hexadecimal point.<br>Rate factor is defined as the ratio between the nominal link rate (such as 1 GbE) and the maximum rate allowed to that queue .<br>Minimum allowed bandwidth share for a queue is 0.1% of the link rate. For example, 10 Mb/s for the 82599 operated at 10 GbE, leading to a maximum allowed rate factor of 1000.<br>Meaningful only if the *RS_ENA* bit is set.<br>When RTTBCNRD.DRIFT_ENA is set, this field is periodically modified by hardware as well. |
| Reserved | 30:24 | 0x0 | Reserved. |
| RS_ENA (SC) | 31b | 0 | Tx rate-scheduler enable, for the Tx queue indexed by TXDQ_IDX field in the RTTDQSEL register<br>When set, the rate programmed in this register is enforced (the queue is rate controlled). At the time it is set, the current timer value is loaded into the time stamp stored for that entry. The bit can be self-cleared internally if the full line rate is recovered via the rate-drift mechanism.<br>When cleared, the rate factor programmed in this register is meaningless, the switch for that queue is always forced to on. The queue is not rate-controlled . Bandwidth group assignment of this TC to a BWG.<br>Each TC must be assigned to a different BWG number, unless the TC is a member of a BWG. No more than two TCs can share the same BWG. |

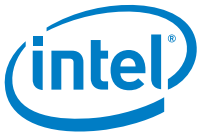
### 8.2.3.10.17 DCB Transmit Rate-Scheduler Status — RTTBCNRS (0x04988; RW)

128 internal registers indirectly addressed via RTTDQSEL.TXDQ_IDX.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MIFS | 31:0 | | Tx rate-scheduler current Minimum Inter-Frame Spacing (MIFS), for the Tx queue indexed by TXDQ_IDX field in the RTTDQSEL register.<br>When read, it is the current algebraic value of the MIFS interval for the queue, expressed in byte units (31 LS-bits taken), relative to the rate-scheduler. It is obtained by hardware subtracting the current value of the timer associated to that rate-scheduler from the time stamp stored for that queue. A strict positive value means a switch in off state. It is expressed in 2's complement format. |

### 8.2.3.10.18 DCB Transmit BCN Rate Drift — RTTBCNRD (0x0498C; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 0 | 0b | Reserved |
| BCN_CLEAR_ALL | 1 | 0b/SC | Clear all BCN rate-limiters.<br>When set, the 128 RTTBCNRC.RS_ENA bits are cleared — releasing any active BCN rate-limiter. This bit must be set by software each time the link speed has changed.<br>This bit is self cleared by hardware. |
| DRIFT_FAC | 15:2 | 0b | BCN Rate Drift Factor.<br>Rate drift factor bits that come after the hexadecimal point, while a zero is always assumed before the hexadecimal point (because rate drift factor must be smaller than unity). The rate drift factor ranges from 0.00006 to 0.99994.<br>Rate drift factor is a decreasing factor by which every rate-factor of BCN rate-controlled queues must be multiplied periodically, once every DRIFT_INT µs. Each time the rate-factor of a queue reaches unity, the *RS_ENA* bit in its corresponding RTTBCNRC register is internally cleared.<br>Meaningful only when the *DRIFT_ENA* bit is set. |
| DRIFT_INT | 30:16 | 0b | BCN Rate Drift Interval Timer.<br>Interval in µs used internally to periodically increase the rate of BCN rate-controlled queues (namely the rate-drift mechanism).<br>Meaningful only when the *DRIFT_ENA* bit is set. |
| DRIFT_ENA | 31 | 0b | BCN Rate Drift Enable bit.<br>When cleared, the rate-drift mechanism performed by hardware is disabled. It is assumed software handles it.<br>When set, the rate-drift mechanism performed by hardware is enabled. Rate of BCN rate-controlled queues are periodically increased in a multiplicative manner. Relevant only for Tx queues for which the *RX_ENA* bit is set in the RTBCNRC register. |



## 8.2.3.11 DCA Registers

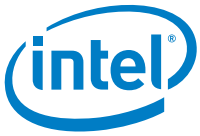
### 8.2.3.11.1 Rx DCA Control Register — DCA_RXCTRL[n] (0x0100C + 0x40*n, n=0...63 and 0x0D00C + 0x40*(n-64), n=64...127 / 0x02200 + 4*n, [n=0...15]; RW)

DCA_RXCTRL[0...15] are also mapped to address 0x02200... to maintain compatibility with the 82598.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 4:0 | 00x | Reserved. |
| Rx Descriptor DCA EN | 5 | 0b | Descriptor DCA EN.<br>When set, hardware enables DCA for all Rx descriptors written back into memory. When cleared, hardware does not enable DCA for descriptor write-backs. |
| Rx Header DCA EN | 6 | 0b | Rx Header DCA EN.<br>When set, hardware enables DCA for all received header buffers. When cleared, hardware does not enable DCA for Rx Headers.[1] |
| Rx Payload DCA EN | 7 | 0b | Payload DCA EN.<br>*Note:* When set, hardware enables DCA for all Ethernet payloads written into memory. When cleared, hardware does not enable DCA for Ethernet payloads. Default cleared. |
| Reserved | 8 | 0b | Reserved. |
| RXdescReadROEn | 9 | 1b | Rx Descriptor Read Relax Order Enable |
| Reserved | 10 | 0b | Reserved. |
| RXdescWBROen | 11 | 0b (RO) | Rx Descriptor Write Back Relax Order Enable.<br>This bit must be 0b to enable correct functionality of the descriptors write back. |
| Reserved | 12 | 1b | Reserved. Must be set to 0. |
| RXdataWriteROEn | 13 | 1b | Rx data Write Relax Order Enable |
| Reserved | 14 | 0b | Reserved. |
| RxRepHeaderROEn | 15 | 1b | Rx Split Header Relax Order Enable |
| Reserved | 23:16 | 0x0 | Reserved. |
| CPUID | 31:24 | 0x0 | Physical ID (see complete description in Section 3.1.3.1.2).<br>Legacy DCA capable platforms — The device driver, upon discovery of the physical CPU ID and CPU bus ID, programs the *CPUID* field with the physical CPU and bus ID associated with this Rx queue.<br>DCA 1.0 capable platforms — The device driver programs a value, based on the relevant APIC ID, associated with this Rx queue. |

### 8.2.3.11.2 Tx DCA Control Registers — DCA_TXCTRL[n] (0x0600C + 0x40*n, n=0...127; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 4:0 | 0x0 | Reserved. |
| Tx Descriptor DCA EN | 5 | 0b | Descriptor DCA Enable. When set, hardware enables DCA for all Tx descriptors written back into memory. When cleared, hardware does not enable DCA for descriptor write-backs. This bit is cleared as a default and also applies to head write back when enabled. |
| Reserved | 7:6 | 00b | Reserved. |
| Reserved | 8 | 0b | Reserved. |
| TXdescRDROEn | 9 | 1b | Tx Descriptor Read Relax Order Enable. |
| Reserved | 10 | 0b | Reserved. |
| TXdescWBROen | 11 | 1b | Relax Order Enable of Tx Descriptor as well as head pointer write back (when set). |
| Reserved | 12 | 0b | Reserved. |
| TXDataReadROEn | 13 | 1b | Tx Data Read Relax Order Enable. |
| Reserved | 23:14 | 0x0 | Reserved. |
| CPUID | 31:24 | 0x0 | Physical ID (see complete description in Section 3.1.3.1.2)<br>Legacy DCA capable platforms — the device driver, upon discovery of the physical CPU ID and CPU bus ID, programs the CPUID field with the physical CPU and bus ID associated with this Tx queue.<br>DCA 1.0 capable platforms — the device driver programs a value, based on the relevant APIC ID, associated with this Tx queue. |



### 8.2.3.11.3 DCA Requester ID Information Register — DCA_ID (0x11070; RO)

To ease software implementation, a DCA requester ID field, composed of device ID, bus # and function # is set up in MMIO space for software to program the chipset DCA Requester ID Authentication register.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Function Number | 2:0 | 0x0 | Function Number.<br>Function number assigned to the function based on BIOS/OS enumeration. |
| Device Number | 7:3 | 0x0 | Device Number.<br>Device number assigned to the function based on BIOS/OS enumeration. |
| Bus Number | 15:8 | 0x0 | Bus Number.<br>Bus Number assigned to the function based on BIOS/OS enumeration. |
| Reserved | 31:16 | 0x0 | Reserved. |

### 8.2.3.11.4 DCA Control Register — DCA_CTRL (0x11074; RW)

**Note:** This register is shared by both LAN functions.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| DCA_DIS | 0 | 1b | DCA Disable.<br>0b = DCA tagging is enabled for this device.<br>1b = DCA tagging is disabled for this device. |
| DCA_MODE | 4:1 | 0x0 | DCA Mode.<br>0000b = Legacy DCA is supported. The *TAG* field in the TLP header is based on the following coding: bit 0 is DCA enable; bits 3:1 are CPU ID).<br>0001b = DCA 1.0 is supported. When DCA is disabled for a given message, the *TAG* field is 0000b,0000b. If DCA is enabled, the TAG is set per queue as programmed in the relevant DCA Control register.<br>All other values are undefined. |
| Reserved | 31:5 | 0x0 | Reserved. |

## 8.2.3.12 Security Registers

Security registers are mainly concerned with the internal settings of the AES crypto engine shared by LinkSec and IPsec. They are owned by the PF in an IOV mode.

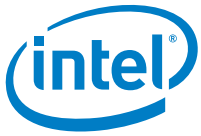
Refer to Section 4.6.12 for the way to modify these registers prior to enabling or disabling a security offload. Note that only one security offload, either LinkSec or IPsec, can be enabled at a time.

Security offload can be disabled via internal security fuses. In this case, the following security related fields are not writable:

- SECTXCTRL.SECTX_DIS is read as 0x1.
- SECRXCTRL.SECRX_DIS is read as 0x1.
- IPSTXIDX.IPS_TX_EN is read as 0x0.
- IPSRXIDX.IPS_RX_EN is read as 0x0.
- LSECTXCTRL bits 1:0 are read as 00b.
- LSECRXCTRL bits 3:2 are read as 00b.

### 8.2.3.12.1 Security Tx Control — SECTXCTRL (0x08800; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SECTX_DIS | 0 | 1b<br>RW / RO if fused-off | Tx Security Offload Disable Bit.<br>When set, the AES crypto engine used in Tx by LinkSec and IPsec off loads is disabled. This mode must be used to save the 82599's power consumption when no security offload is enabled.<br>When cleared, the AES crypto engine used in Tx by LinkSec or IPsec off load is enabled. Normal operating mode when a security offload is enabled. |
| TX_DIS | 1 | 0b | Disable Sec Tx Path.<br>When set, no new packet is fetched out from the Tx packet buffers, so that the Tx security block can be internally emptied prior to changing the security mode. SECTXSTAT.SECTX_RDY bit is deasserted until the path is emptied by hardware.<br>When cleared, Tx data path is enabled. Normal operating mode. |
| STORE_FORWARD | 2 | 0b | Tx Sec Buffer Mode.<br>When set, a complete frame is stored in the internal security Tx buffer prior to being forwarded to the MAC. Operating mode when IPsec offload is enabled (as requested to overwrite ICV field in AH frames).<br>*Note:* It increases the Tx internal latencies (for all TCs).<br>When cleared, Tx sec buffer is operated in pass-through mode. Operating mode when LinkSec is enabled or when no security offload is enabled. |
| Reserved | 31:3 | 0x0 | Reserved. |



### 8.2.3.12.2 Security Tx Status — SECTXSTAT (0x08804; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SECTX_RDY | 0 | 0b | Tx security block ready for mode change.<br>When set, it indicates that the internal data path from the Tx packet buffers to the Tx security block has been emptied, and thus the security mode can be changed by software.<br>When cleared, it indicates that the internal data path from the Tx packet buffers to the Tx security block is not empty, and thus software cannot change the security mode.<br>This bit is polled by software once the SECTXCTRL.TX_DIS bit was set. |
| SECTX_OFF_DIS | 1 | 0b | Security offload is disabled by fuse or strapping pin. |
| ECC_TXERR | 2 | 0b | Unrecoverable ECC error in the Tx SA table or SEC Tx FIFO occurred.<br>When set, it indicates that an unrecoverable ECC error occurred when accessing internally the Tx SA table. The ECC interrupt is set as well, until the device is reset by software.<br>When cleared, no ECC error occurred on the Tx SA table from the last time the device has reset. |
| Reserved | 31:3 | 0x0 | Reserved. |

### 8.2.3.12.3 Security Tx Buffer Almost Full — SECTXBUFFAF (0x08808; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| FULLTHRESH | 9:0 | 0x250 | Tx Security Buffer Almost Full Threshold (relatively to full capacity).<br>The size of the security buffer is 0x274 lines of 16 bytes. In LinkSec offload, the buffer operates in pass-through mode and the recommended threshold is 0x250. It means that the almost full indication is generated very soon while only a fraction of a packet is stored in the buffer. In IPSec mode, the buffer operates in a store and forward mode and the recommended threshold is 0x15. It means that the almost full indication is generated only after the buffer contains at least an entire jumbo packet. |
| Reserved | 31:10 | 0x0 | Reserved. |

### 8.2.3.12.4 Security Tx Buffer Minimum IFG — SECTXMINIFG (0x08810; RW) SEC-Tx

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MINSECIFG | 3:0 | 0x1 | Minimum IFG between packets.<br>It is the minimum gap between consecutive frames from the DBU-Tx required for the security block. The MINSECIFG is measured in Wake DMA clock units (equal to 6.4 ns in 10 GbE). |
| Reserved | 7:4 | 0 | Reserved. |
| SECTXDCB | 12:8 | 0x10 | This field is used to configure the Security Tx Buffer.<br>• If PFC is enabled, then the *SECTXDCB* field should be set to 0x1F.<br>• If PFC is not enabled, then the default value should be used (0x10). |

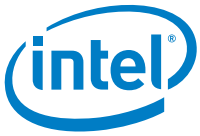
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 31:13 | 0 | Reserved. |
| Reserved | 31:4 | 0x100 | Reserved. |

### 8.2.3.12.5 Security Rx Control — SECRXCTRL (0x08D00; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SECRX_DIS | 0 | 1b<br>RW / RO if fused-off | Rx Security Offload Disable Bit.<br>When set, the AES crypto engine used in Rx by LinkSec and IPsec offloads is disabled. This mode must be used to save the 82599's power consumption when no security offload is enabled.<br>When cleared, the AES crypto engine used in Rx by LinkSec or IPsec offload is enabled. Normal operating mode when a security offload is enabled. |
| RX_DIS | 1 | 0b | Disable Sec Rx Path.<br>When set, any new packet received from the Rx MAC is filtered out, so that the Rx security block can be internally emptied prior to changing the security mode. SECRXSTAT.SECRX_RDY bit is deasserted until the path is emptied by hardware.<br>When cleared, Rx data path is enabled. Normal operating mode. |
| Reserved | 31:2 | 0x0 | Reserved. |

### 8.2.3.12.6 Security Rx Status — SECRXSTAT (0x08D04; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SECRX_RDY | 0 | 0b | Rx security block ready for mode change.<br>When set, it indicates that the internal data path from the Rx MAC to the Rx security block has been emptied, and thus the security mode can be changed by software.<br>When cleared, it indicates that the internal data path from the Rx MAC to the Rx security block is not empty, and thus software cannot change the security mode.<br>This bit is polled by software once the SECRXCTRL.RX_DIS bit was set. |
| SECRX_OFF_DIS | 1 | 0b | Security offload is disabled by fuse or strapping pin. |
| ECC_RXERR | 2 | 0b | Unrecoverable ECC error in an Rx SA table occurred.<br>When set, it indicates that an unrecoverable ECC error occurred when accessing internally one Rx SA table. The ECC interrupt is set as well, until the device is reset by software.<br>When cleared, no ECC error occurred on the Rx SA table from the last time device has reset. |
| Reserved | 31:3 | 0x0 | Reserved. |



## 8.2.3.13 LinkSec Registers

The LinkSec registers are initialized at software reset. When LinkSec is disabled, the LinkSec statistic registers are meaningless and their values are unpredictable.

### 8.2.3.13.1 LinkSec Tx Capabilities Register — LSECTXCAP (0x08A00; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| NCA | 2:0 | 1b | Tx CA-Supported.<br>Number of CA's supported by the device. |
| NSC | 6:3 | 1b | Tx SC Capable.<br>Number of SC's supported by the device on the transmit data path. The 82599 supports twice the number of SA's as the Tx SC for seamless re-keying, such as 2 SA's. |
| Reserved | 15:7 | 0x0 | Reserved. |
| LSECTXSUM | 23:16 | 0x0 | Tx LSEC Key SUM.<br>A bit wise XOR of the LSECTXKEY 0 bytes and LSECTXKEY 1 bytes. This register can be used by KaY (the programming entity) to validate key programming. |
| Reserved | 31:24 | 0x0 | Reserved. |

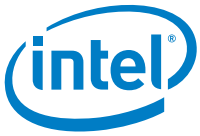
### 8.2.3.13.2 LinkSec Rx Capabilities Register — LSECRXCAP (0x08F00; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| NCA | 2:0 | 1b | Rx CA-supported.<br>Number of CA's supported by the device. |
| NSC | 6:3 | 1b | Rx SC Capable.<br>Number of SC's supported by the device on the receive data path. The 82599 supports twice the number SA's as the Rx SC for seamless re-keying, such as 2 SA's. |
| Reserved | 15:7 | 0x0 | Reserved. |
| RXLKM | 23:16 | 0x0 | Rx LSEC Key SUM.<br>A byte wise XOR of all bytes of the Rx LinkSec keys 0…1 as defined in registers LSECRXKEY [n, m]. This register can be used by KaY (the programming entity) to validate key programming. |
| Reserved | 31:24 | 0x0 | Reserved. |

### 8.2.3.13.3 LinkSec Tx Control Register — LSECTXCTRL (0x08A04; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LSTXEN | 1:0 | 00b (see Table Note) | Enable Tx LinkSec.<br>Enable Tx LinkSec offloading.<br>00b = Disable Tx LinkSec (Tx all packets without LinkSec offload).<br>01b = Add integrity signature.<br>10b = Encrypt and add integrity signature.<br>11b = Reserved.<br>When this field equals 00b (LinkSec offload is disabled).<br>The Tx Untagged Packet register is not incremented for transmitted packets when Enable Tx LinkSec equals 00b. |
| Reserved | 2 | 0b | 0b = Reserved. |
| Reserved | 3 | 0 | Reserved. |
| Reserved | 4 | 00b | Reserved. |
| AISCI | 5 | 1b | Always Include SCI.<br>This field controls whether SCI is explicitly included in the transmitted SecTag. Since the ES bit in the SecTag is fixed at Zero, the AISCI must always be set to 1b.<br>0b = False<br>1b = True |
| Reserved | 6 | 0b | Reserved. |
| Reserved | 7 | 0b | Reserved. |
| PNTRH | 31:8 | 11..1b | PN Exhaustion Threshold.<br>MSB of the threshold over which hardware needs to interrupt KaY to warn Tx SA PN exhaustion and triggers a new SA re-negotiation. Bits 7:0 of the threshold are all 1's. |

**Note:** Bits 1:0 are RW, but they are RO if fused-off and/or if SECTXCTRL.SECTX_DIS is set to 1b, and/or if IPSTXIDX.IPS_TX_EN is set to 1b.



### 8.2.3.13.4 LinkSec Rx Control register — LSECRXCTRL (0x08F04; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 1:0 | 00b | Reserved. |
| LSRXEN | 3:2 | 00b (see Table Note) | Enable Rx LinkSec.<br>Controls the level of LinkSec packet filtering.<br>00b = Disable Rx LinkSec (pass all packets to host without LinkSec processing and no LinkSec header strip).<br>01b = Check (execute LinkSec offload and post frame to host and ME even when it fails LinkSec operation unless failed ICV and C bit was set).<br>10b = Strict (execute LinkSec offload and post frame to host and ME only if it does not fail LinkSec operation).<br>11b = Rx LinkSec Drop (drop all packets that include LinkSec header). |
| Reserved | 5:4 | 00b | Reserved. |
| PLSH | 6 | 0b | Post LinkSec Header.<br>When set, the device posts the LinkSec header and signature (ICV) to host memory. During normal operation this bit should be cleared. |
| RP | 7 | 1b | Replay Protect. Enable replay protection. |
| Reserved | 31:8 | 0x0 | Reserved. |

**Note:** Bits 3:2 are RW, but they are RO if fused-off and/or if SECRXCTRL.SECRX_DIS is set to 1b, and/or if IPSRXIDX.IPS_RX_EN is set to 1b.

### 8.2.3.13.5 LinkSec Tx SCI Low — LSECTXSCL (0x08A08; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SecYL | 31:0 | 0x0 | Ethernet MAC Address SecY Low.<br>The 4 LS bytes of the Ethernet MAC address copied to the SCI field in the LinkSec header.<br>*Note:* Field is defined in big endian (LS byte is first on the wire). |

### 8.2.3.13.6 LinkSec Tx SCI High — LSECTXSCH (0x08A0C; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SecYH | 15:0 | 0x0 | Ethernet MAC Address SecY High.<br>The 2 MS bytes of the Ethernet MAC address copied to the SCI field in the LinkSec header.<br>*Note:* Field is defined in big endian (LS byte is first on the wire). |
| PI | 31:16 | 0x0 | Port Identifier.<br>Always zero for transmitted packets. This field is RO. |

### 8.2.3.13.7 LinkSec Tx SA — LSECTXSA (0x08A10; RW)

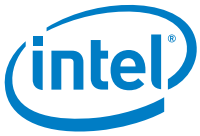
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| AN0 | 1:0 | 0b | AN0 – Association Number 0.<br>This 2-bit field is posted to the AN field in the transmitted LinkSec header when SA 0 is active. |
| AN1 | 3:2 | 0b | AN1 – Association Number 1.<br>This 2-bit field is posted to the AN field in the transmitted LinkSec header when SA 1 is active. |
| SelSA | 4 | 0b | SA Select (SelSA).<br>This bit selects between SA 0 or SA 1 smoothly, such as on a packet boundary. A value of 0b selects SA 0 and a value of 1b selects SA 1. |
| ActSA (RO) | 5 | 0b | Active SA (ActSA).<br>This bit indicates the active SA. The ActSA follows the value of the SelSA on a packet boundary. The KaY (the programming entity) can use this indication to retire the old SA. |
| Reserved | 31:6 | 0x0 | Reserved. |

### 8.2.3.13.8 LinkSec Tx SA PN 0 — LSECTXPN0 (0x08A14; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PN | 31:0 | 0x0 | PN – Packet Number.<br>This field is posted to the PN field in the transmitted LinkSec header when SA 0 is active. It is initialized by the KaY at SA creation and then increments by 1 for each transmitted packet using this SA.<br>Packets should never be transmitted if the PN repeats itself. In order to protect against such an event hardware generates an LSECPN interrupt to KaY when the PN reaches the exhaustion threshold as defined in the LSECTXCTRL register. There is an additional level of defense against repeating the PN. Hardware never transmits packets after the PN reaches a value of 0xFF...FF. In order to guarantee it, hardware clears the *Enable Tx LinkSec* field in the LSECTXCTRL register to 00b once a packet is transmitted with a PN that equals to 0xFF...F0.<br>*Note:* Field is defined in big endian (LS byte is first on the wire). |

### 8.2.3.13.9 LinkSec Tx SA PN 1 — LSECTXPN1 (0x08A18; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PN | 31:0 | 0x0 | PN – Packet Number.<br>This field is posted to the PN field in the transmitted LinkSec header when SA 1 is active. It is initialized by the KaY at SA creation and then increments by 1 for each transmitted packet using this SA.<br>Packets should never be transmitted if the PN repeats itself. In order to protect against such an event hardware generates an LSECPN interrupt to KaY when the PN reaches the exhaustion threshold as defined in the LSECTXCTRL register. There is additional level of defense against repeating the PN. hardware never transmits packets after the PN reaches a value of 0xFF...FF. In order to guarantee it, hardware clears the *Enable Tx LinkSec* field in the LSECTXCTRL register to 00b once a packet is transmitted with a PN that equals to 0xFF...F0.<br>*Note:* Field is defined in big endian (LS byte is first on the wire). |



### 8.2.3.13.10 LinkSec Tx Key 0 — LSECTXKEY0[n] (0x08A1C + 4*n, n=0...3; WO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LSECK0 | 31:0 | 0x0 | LSEC Key 0.<br>Transmit LinkSec key of SA 0.<br>n=0 LSEC Key defines bits 31:0 of the Tx LinkSec key.<br>n=1 LSEC Key defines bits 63:32 of the Tx LinkSec key.<br>n=2 LSEC Key defines bits 95:64 of the Tx LinkSec key.<br>n=3 LSEC Key defines bits 127:96 of the Tx LinkSec key.<br>This field is WO for confidentiality protection. For data integrity check, hash value is accessible by the LSECTXSUM field in the LSECCAP register. If for some reason a read request is aimed to this register a value of all zeros are returned. |

### 8.2.3.13.11 LinkSec Tx Key 1 — LSECTXKEY1[n] (0x08A2C + 4*n, n=0...3; WO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LSECK1 | 31:0 | 0x0 | LSEC Key 1.<br>Transmit LinkSec key of SA 1.<br>n=0 LSEC Key defines bits 31:0 of the Tx LinkSec key.<br>n=1 LSEC Key defines bits 63:32 of the Tx LinkSec key.<br>n=2 LSEC Key defines bits 95:64 of the Tx LinkSec key.<br>n=3 LSEC Key defines bits 127:96 of the Tx LinkSec key.<br>This field is WO for confidentiality protection. For data integrity check, hash value is accessible by the LSECTXSUM field in the LSECCAP register. If for some reason a read request is aimed to this register a value of all zeros are returned. |

### 8.2.3.13.12 LinkSec Rx SCI Low — LSECRXSCL (0x08F08; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MAL | 31:0 | 0x0 | Ethernet MAC Address SecY low.<br>The 4 LS bytes of the Ethernet MAC address in the *SCI* field in the incoming packet that are compared with this field for SCI matching. Comparison result is meaningful only if the *SC* bit in the TCI header is set.<br>*Note:* Field is defined in big endian (LS byte is first on the wire). |

### 8.2.3.13.13 LinkSec Rx SCI High — LSECRXSCH (0x08F0C; RW)

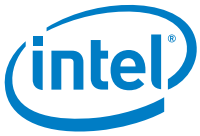
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MAH | 15:0 | 0x0 | Ethernet MAC Address SecY High.<br>The 2 MS bytes of the Ethernet MAC address in the *SCI* field in the incoming packet that are compared with this field for SCI matching. Comparison result is meaningful only if the *SC* bit in the TCI header is set.<br>*Note:* Field is defined in Big Endian (MS byte is last on the wire). |
| PI | 31:16 | 0x0 | Port Identifier.<br>The port number in the *SCI* field in the incoming packet that is compared with this field for SCI matching. Comparison result is meaningful only if the *SC* bit in the TCI header is set.<br>*Note:* Field is defined in big endian (LS byte is first on the wire). |

### 8.2.3.13.14 LinkSec Rx SA Registers

The registers in this section relate to the LinkSec receive SA context. There are 2 SA(s) in the receive data path defined as SA0 and SA1. The following registers with index n relates to the SA index.

### 8.2.3.13.15 LinkSec Rx SA — LSECRXSA[n] (0x08F10 + 4*n, n=0...1; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| AN | 1:0 | 00b | AN – Association Number.<br>This field is compared with the *AN* field in the *TCI* field in the incoming packet for match. |
| SAV | 2 | 0b | SA Valid.<br>This bit is set or cleared by the KaY to validate or invalidate the SA. |
| FRR (RO) | 3 | 0b | Frame Received.<br>This bit is cleared when the *SA Valid* (bit 2) transitions from 0b to 1b, and is set when a frame is received with this SA. When the *Frame Received* bit is set the *Retired* bit of the other SA of the same SC is set.<br>*Note:* A single frame reception with the new SA is sufficient to retire the old SA since it is assumed that the replay window is zero. |
| Retired (RO) | 4 | 0b | Retired.<br>When this bit is set, the SA is invalid (retired). This bit is cleared when a new SA is configured by the KaY (SA Valid transition to 1b). It is set to 1b when a packet is received with the other SA of the same SC.<br>*Note:* A single frame reception with the new SA is sufficient to retire the old SA since it is assumed that the replay window is zero. |
| Reserved | 31:5 | 0x0 | Reserved. |



### 8.2.3.13.16 LinkSec Rx SA PN — LSECRXPN[n] (0x08F18 + 4*n, n=0...1; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PN | 31:0 | 0x0 | PN – Packet Number.<br>Register 'n' holds the *PN* field of the next incoming packet that uses SA 'n', 'n' = 0, 1. The *PN* field in the incoming packet must be greater or equal to the PN register. The PN register is set by KaY at SA creation. It is updated by hardware for each received packet using this SA to be received PN + 1.<br>*Note:* Field is defined in Big Endian (LS byte is first on the wire). |

### 8.2.3.13.17 LinkSec Rx Key — LSECRXKEY[n,m] (0x08F20 + 0x10*n + 4*m, n=0...1, m=0...3; WO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LSECK | 31:0 | 0x0 | LSEC Key.<br>Receive LinkSec key of SA n, while n=0...1.<br>m=0 LSEC Key defines bits 31:0 of the Rx LinkSec key.<br>m=1 LSEC Key defines bits 63:32 of the Rx LinkSec key.<br>m=2 LSEC Key defines bits 95:64 of the Rx LinkSec key.<br>m=3 LSEC Key defines bits 127:96 of the Rx LinkSec key.<br>This field is WO for confidentiality protection. For data integrity check, KaY hash value is accessible by the LSECRXSUM field in the LSECCAP registers. If for some reason a read request is aimed to this register a value of all zeros are returned. |

## 8.2.3.14 LinkSec Tx Port Statistics

These counters are defined by the specification as 64 bits while implementing only 32 bits in hardware. The KaY must implement the 64-bit counter in software by polling regularly the hardware statistic counters. Hardware counters wrap around from 0xFF..F to 0x0 and cleared on read.

Note that 82599 includes a 10 KB FIFO between the security block output and the MAC block. In the case of a pause event, packets stored in this FIFO are dropped for instant response to the pause request. When it is time to resume transmission, the packets are re-transmitted from the transmit packet buffer to the security block. These re-transmitted packets are counted twice in all the relevant security transmit counters.

### 8.2.3.14.1 Tx Untagged Packet Counter — LSECTXUT (0x08A3C; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| UPC | 31:0 | 0x0 | Untagged Packet CNT.<br>Increments for each transmitted packet that is transmitted with the ILSec bit cleared in the packet descriptor while *Enable Tx LinkSec* field in the LSECTXCTRL register is either 01b or 10b. The KaY must implement a 64-bit counter. It can do that by reading the LSECTXUT register regularly. |

### 8.2.3.14.2 Encrypted Tx Packets — LSECTXPKTE (0x08A40; RW)

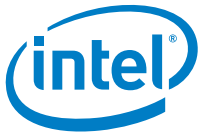
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| EPC | 31:0 | 0x0 | Encrypted Packet CNT.<br>Increments for each transmitted packet through the controlled port with the *E* bit set (such as confidentiality was prescribed for this packet by software/firmware). |

### 8.2.3.14.3 Protected Tx Packets — LSECTXPKTP (0x08A44; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PPC | 31:0 | 0x0 | Protected Packet CNT.<br>Increments for each transmitted packet through the controlled port with the *E* bit cleared (such as integrity only was prescribed for this packet by software/firmware). |

### 8.2.3.14.4 Encrypted Tx Octets — LSECTXOCTE (0x08A48; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| EOC | 31:0 | 0x0 | Encrypted Octet CNT.<br>Increments for each byte of user data through the controlled port with the *E* bit set (such as confidentiality prescribed for this packet by software/firmware). |



### 8.2.3.14.5 Protected Tx Octets — LSECTXOCTP (0x08A4C; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| POC | 31:0 | 0x0 | Protected Octet CNT.<br>Increments for each byte of user data through the controlled port with the *E* bit cleared such as integrity only was prescribed for this packet by software/firmware). |

## 8.2.3.15 LinkSec Rx Port Statistic Counters

These counters are defined by the specification as 64 bits while implementing only 32 bits in hardware. The KaY must implement the 64-bit counter in software by polling regularly the hardware statistic counters.

### 8.2.3.15.1 LinkSec Untagged Rx Packet — LSECRXUT (0x08F40; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| UPC | 31:0 | 0x0 | Untagged Packet CNT.<br>Increments for each packet received having no tag. Also increments for any KaY packets regardless of the LinkSec tag. Increments only when the *Enable Rx LinkSec* field in the LSECRXCTRL register is either 01b or 10b.<br>*Note:* Flow control frames are also counted by this counter. |

### 8.2.3.15.2 LinkSec Rx Octets Decrypted — LSECRXOCTE (0x08F44; RC)

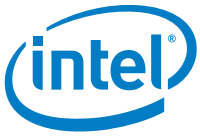
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| DROC | 31:0 | 0x0 | Decrypted Rx Octet CNT.<br>The number of octets of user data recovered from received frames that were both integrity protected and encrypted. This includes the octets from SecTag to ICV not inclusive. These counts are incremented even if the user data recovered failed the integrity check or could not be recovered. |

### 8.2.3.15.3 LinkSec Rx Octets Validated — LSECRXOCTP (0x08F48; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| VOC | 31:0 | 0b | Validated Rx Octet CNT.<br>The number of octets of user data recovered from received frames that were integrity protected but not encrypted.This includes the octets from SecTag to ICV not inclusive. These counts are incremented even if the user data recovered failed the integrity check or could not be recovered. |

### 8.2.3.15.4 LinkSec Rx Packet with Bad Tag — LSECRXBAD (0x08F4C; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| BRPC | 31:0 | 0b | Bad Rx Packet CNT.<br>Number of packets received having a invalid tag. |



### 8.2.3.15.5 LinkSec Rx Packet No SCI — LSECRXNOSCI (0x08F50; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| USRPC | 31:0 | 0b | No SCI Rx Packet CNT.<br>Number of packets received with unrecognizable SCI and dropped due to that condition. |

### 8.2.3.15.6 LinkSec Rx Packet Unknown SCI — LSECRXUNSCI (0x08F54; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| USRPC | 31:0 | 0b | Unknown SCI Rx Packet CNT.<br>Number of packets received with an unrecognized SCI but still forwarded to the host. |

## 8.2.3.16 LinkSec Rx SC Statistic Counters

These counters are defined by the specification as 64 bits while implementing only 32 bits in hardware. The KaY must implement the 64-bit counter in software by polling regularly the hardware statistic counters. Hardware counters wrap around from 0xFF..F to 0x0 and cleared on read.

### 8.2.3.16.1 LinkSec Rx Unchecked Packets — LSECRXUC (0x08F58; RC)

Software/firmware needs to maintain the full-sized register.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| URPC | 31:0 | 0x0 | Unchecked Rx Packet CNT.<br>Number of packets received with LinkSec encapsulation (SecTag) while Validate Frames is disabled (LSECRXCTRL bits 3:2 equal 00b). |

### 8.2.3.16.2 LinkSec Rx Delayed Packets — LSECRXDELAY (0x08F5C; RC)

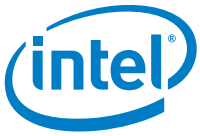
Software/firmware needs to maintain the full-sized register.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| DRPC | 31:0 | 0x0 | Delayed Rx Packet CNT.<br>Number of packets received and accepted for validation having failed replay protection and Replay Protect is false (LSECRXCTRL bit 7 is 0b). |

### 8.2.3.16.3 LinkSec Rx Late Packets — LSECRXLATE (0x08F60; RC)

Software/firmware needs to maintain the full-sized register.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LRPC | 31:0 | 0x0 | Late Rx Packet CNT.<br>Number of packets received and accepted for validation having failed replay-protection and Replay Protect is true (LSECRXCTRL bit 7 is 1b). |



## 8.2.3.17 LinkSec Rx SA Statistic Counters

These counters are defined by the specification as 64 bits while implementing only 32 bits in hardware. The KaY must implement the 64-bit counter in software by polling regularly the hardware statistic counters. Hardware counters wrap around from 0xFF..F to 0x0 and cleared on read.

### 8.2.3.17.1 LinkSec Rx Packet OK — LSECRXOK[n] (0x08F64 + 4*n, n=0...1; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| ORPC | 31:0 | 0x0 | OK Rx Packet CNT.<br>Number of packets received that were valid (authenticated) and passed replay protection. |

### 8.2.3.17.2 LinkSec Rx Invalid — LSECRXINV[n] (0x08F6C + 4*n, n=0...1; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| ICRPC | 31:0 | 0x0 | Invalid Rx Packet CNT.<br>Number of packets received that were not valid (authentication failed) and were forwarded to host. |

### 8.2.3.17.3 LinkSec Rx Not valid count — LSECRXNV[n] (0x08F74 + 4*n, n=0...1; RC)

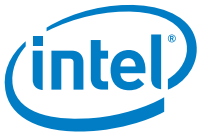
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| ICRPC | 31:0 | 0x0 | Not valid Rx Packet CNT.<br>Number of packets received that were not valid (authentication failed) and were dropped. |

### 8.2.3.17.4 LinkSec Rx Unused SA Count — LSECRXUNSA (0x08F7C; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| ISSRPC | 31:0 | 0x0 | Invalid SA Rx Packet CNT.<br>Number of packets received that were associated with an SA that is not in use (no match on auto-negotiation or not valid or retired) and were forwarded to host. |

### 8.2.3.17.5 LinkSec Rx Not Using SA Count — LSECRXNUSA (0x08F80; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| ISSRPC | 31:0 | 0x0 | Invalid SA Rx Packet CNT.<br>Number of packets received that were associated with an SA that is not in use (No match on auto-negotiation or not valid or retired) and were dropped. |



## 8.2.3.18 IPsec Registers

IPsec registers are owned by the PF in an IOV mode.

Unlike LinkSec, there is no added value here to encrypt the SA contents when being read by software because the SA contents is available in clear text from system memory like for any IPsec flow handled in software.

### 8.2.3.18.1 IPsec Tx Index — IPSTXIDX (0x08900; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| IPS_TX_EN | 0 | 0b<br>(see table note) | IPsec Tx offload enable bit.<br>0b = IPsec offload ability is disabled for the Tx path, regardless of the contents of the Tx SA table.<br>1b = IPsec offload ability is enabled for the Tx path. |
| Reserved | 2:1 | 00b | Reserved. |
| SA_IDX | 12:3 | 0x0 | SA index for indirect access into the Tx SA table. |
| Reserved | 29:13 | 0x0 | Reserved. |
| READ | 30 | 0b<br>SC by HW | Read Command.<br>When set, the contents of the Tx SA table entry pointed by the SA_IDX field is loaded into the IPSTXKEY 0...3 and IPSTXSALT registers.<br>Immediately self cleared by hardware once the entry contents has been loaded into the registers. |
| WRITE | 31 | 0b<br>SC by HW | Write Command.<br>When set, the contents of the IPSTXKEY 0...3 and IPSTXSALT registers are loaded into the Tx SA table entry pointed to by the SA_IDX field.<br>Immediately self cleared by hardware once the entry contents have been loaded into the memory. |

**Notes:** Write and Read bits must not be set at the same time by software.

IPS_TX_EN is RW, but it is RO if fused-off and/or if SECTXCTRL.SECTX_DIS is set to 1b.

### 8.2.3.18.2 IPsec Tx Key Registers — IPSTXKEY[n] (0x08908 + 4*n, n=0...3; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| AES-128 KEY | 31:0 | 0x0 | 4 bytes of a 16-byte key that has been read/written from/into the Tx SA entry pointed to by SA_IDX.<br>n=0 Contains the LSB of the key.<br>n=3 Contains the MSB of the key. |

### 8.2.3.18.3 IPsec Tx Salt Register — IPSTXSALT (0x08904; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| AES-128 SALT | 31:0 | 0x0 | 4-byte salt that has been read/written from/into the Tx SA entry pointed to by SA_IDX. |

### 8.2.3.18.4 IPsec Rx Index — IPSRXIDX (0x08E00; RW)

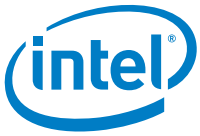
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| IPS_RX_EN | 0 | 0b<br>(see table note) | IPsec Rx offload enable bit.<br>0b = IPsec offload ability is disabled for the Rx path, regardless of the contents of Rx SA tables.<br>1b = IPsec offload ability is enabled for the Rx path. |
| TABLE | 2:1 | 00b | Table select bits.<br>00b = No Rx SA table is accessed.<br>01b = IP address table is accessed.<br>10b = SPI table is accessed.<br>11b = Key table is accessed. |
| TB_IDX | 12:3 | 0x0 | Table index bits for indirect access into the Rx SA table selected by the *Table* bits.<br>When accessing the IP address table, only the seven least significant bits of this field are meaningful. |
| Reserved | 29:13 | 0x0 | Reserved. |
| READ | 30 | 0b<br>SC by HW | Read Command.<br>When set, the contents of the Rx SA table entry as pointed to by the [TABLE, TB_IDX] fields is loaded into the corresponding registers.<br>Immediately self cleared by hardware once the entry contents have been loaded into the corresponding registers.<br>For instance, if this bit is set together with Table=10b and TB_IDX=0x9, then the SPI value stored in entry nine is loaded into the IPSRXSPI 0...3 registers. Rx SA registers related to another Rx SA table (like IPSRXKEY 0...3 registers) must not be read when Table=01b. |
| WRITE | 31 | 0b<br>SC by HW | Write command.<br>When set, the contents of the registers affected by the Rx SA table pointed to by the *Table* field is loaded into the table entry pointed to by the TB_IDX field.<br>Immediately self cleared by hardware once the entry contents have been loaded into the memory.<br>For instance, if this bit is set together with *Table*=10b and TB_IDX=0x9, then the value written in IPSRXSPI 0...3 registers is loaded into the SPI table entry nine. |

**Notes:**

*Write* and *Read* bits must not be set at the same time by software.

IPS_RX_EN is RW, but it is RO if fused-off and/or if SECRXCTRL.SECRX_DIS is set to 1b.

Software is not allowed to write/read access registers that belong to different Rx SA tables without writing the IPSRXIDX register in between for setting the *WriteRead* bit. Refer to Rx SA tables access rules described in Section 7.12.9.2.



Software should not make changes in the Rx SA tables while changing the IPSEC_EN bit.

### 8.2.3.18.5 IPsec Rx IP address Register — IPSRXIPADDR (0x08E04 + 4*n, n=0...3; RW)

These registers are related to the IP Address table.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| IPADDR | 31:0 | 0x0 | 4 bytes of a16-byte destination IP address for the associated Rx SA(s).<br>n=0 Contains the MSB for an IPv6 IP address.<br>n=3 Contains an IPv4 IP address or the LSB for an IPv6 IP address.<br>For an IPv4 address, IPSRXIPADDR 0...2 must be written with zeros.<br>*Note:* Field is defined in big endian (LS byte is first on the wire). |

### 8.2.3.18.6 IPsec Rx SPI Register — IPSRXSPI (0x08E14; RW)

This register is related to the Rx SPI table.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SPI | 31:0 | 0x0 | SPI field for the SPI entry.<br>*Note:* Field is defined in big endian (LS byte is first on the wire). |

### 8.2.3.18.7 IPsec Rx SPI Register — IPSRXIPIDX (0x08E18; RW)

This register is related to the Rx SPI table.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| IP_IDX | 6:0 | 0x0 | IP Index.<br>Index in the IP address table where the destination IP address associated to that SPI entry is found. |
| Reserved | 31:7 | 0x0 | Reserved. |

### 8.2.3.18.8 IPsec Rx Key Register — IPSRXKEY[n] (0x08E1C + 4*n, n=0...3; RW)

These registers are related to the Rx KEY table.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| AES-128 KEY | 31:0 | 0x0 | 4 bytes of a16-byte key of the KEY entry.<br>n=0 Contains the LSB of the key.<br>n=3 Contains the MSB of the key. |

### 8.2.3.18.9 IPsec Rx Salt Register — IPSRXSALT (0x08E2C; RW)

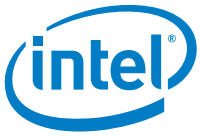
This register is related to the Rx KEY table.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| AES-128 SALT | 31:0 | 0x0 | 4-byte salt associated to the KEY entry. |

### 8.2.3.18.10 IPsec Rx Mode Register — IPSRXMOD (0x08E30; RW)

This register is related to the Rx KEY table.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| VALID | 0 | 0b | Valid Bit.<br>0b = The KEY entry is not valid.<br>1b = The KEY entry is valid. |
| Reserved | 1 | 0b | Reserved. |
| PROTO | 2 | 0b | IPsec Protocol Select.<br>0b = The KEY entry offloads AH packets.<br>1b = The KEY entry offloads ESP packets. |
| DECRYPT | 3 | 0b | Decryption Bit.<br>When set, hardware performs decryption offload for this KEY entry.<br>Meaningful only if the *Proto* bit is set (like ESP mode). |
| IPv6 | 4 | 0b | IPv6 Type.<br>0b = Only matched IPv4 packets are offloaded for that KEY entry.<br>1b = Only matched IPv6 packets are offloaded for that KEY entry. |
| Reserved | 31:5 | 0x0 | Reserved. |



# 8.2.3.19 Timers Registers

## 8.2.3.19.1 TCP Timer — TCPTIMER (0x0004C; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Duration | 7:0 | 0x0 | Duration.<br>Duration of the TCP interrupt interval, in ms. |
| KickStart | 8 | 0b | Counter kick-start.<br>Writing a 1b to this bit kick-starts the counter down-count from the initial value defined in the *Duration* field. Writing 0b has no effect (WS). |
| TCPCountEn | 9 | 0b | TCP Count Enable.<br>0b = TCP timer counting is disabled.<br>1b = TCP timer counting is enabled.<br>Upon enabling, TCP counter must count 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 | TCP Count Finish.<br>This bit enables software to trigger a TCP timer interrupt, regardless of the internal state.<br>0b = No effect (WS).<br>1b = Triggers an interrupt and resets the internal counter to its initial value. Down-count does not restart until either *KickStart* is activated or *Loop* is set. |
| Loop | 11 | 0b | TCP Loop.<br>0b = TCP counter must stop at a zero value, and must not re-start until *KickStart* is activated.<br>1b = TCP counter must reload duration each time it reaches zero, and must go on down-counting from this point without kick-starting. |
| Reserved | 31:12 | 0x0 | Reserved. |

# 8.2.3.20 FCoE Registers

## 8.2.3.20.1 Tx FC SOF Flags Register - TSOFF (0x04A98; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SOF0 | 7:0 | 0x2D | Start Of Frame 0.<br>Class 2 Start of Frame used in the first packet of FC sequence. Default setting of SOFi2. |
| SOF1 | 15:8 | 0x2E | Start Of Frame 1.<br>Class 3 Start of Frame used in the first packet of FC sequence. Default setting of SOFi3. |
| SOF2 | 23:16 | 0x35 | Start Of Frame 2.<br>Class 2 Start of Frame used in all packets but the first one of FC sequence. Default setting of SOFn2. |
| SOF3 | 31:24 | 0x36 | Start Of Frame 3.<br>Class 3 Start of Frame used in all packets but the first one of FC sequence. Default setting of SOFn3. |

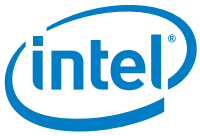
## 8.2.3.20.2 Tx FC EOF Flags Register - TEOFF (0x04A94; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| EOF0 | 7:0 | 0x41 | End Of Frame 0.<br>By default it is set to EOFn code used in all packets but the last one on a sequence. |
| EOF1 | 15:8 | 0x42 | End Of Frame 1.<br>By default it is set to EOFt code used to close a sequence. |
| EOF2 | 23:16 | 0x49 | End Of Frame 2.<br>By default it is set to EOFni code. |
| EOF3 | 31:24 | 0x50 | End Of Frame 3.<br>By default it is set to EOFa code. |

FCoE Rx registers

## 8.2.3.20.3 Rx FC SOF Flags Register - RSOFF (0x051F8; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SOF0 | 7:0 | 0x2D | Start Of Frame 0.<br>Class 2 Start of Frame used in the first packet of FC sequence. Default setting of SOFi2. |
| SOF1 | 15:8 | 0x2E | Start Of Frame 1.<br>Class 3 Start of Frame used in the first packet of FC sequence. Default setting of SOFi3. |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SOF2 | 23:16 | 0x35 | Start Of Frame 2.<br>Class 2 Start of Frame used in all packets but the first one of FC sequence. Default setting of SOFn2. |
| SOF3 | 31:24 | 0x36 | Start Of Frame 3.<br>Class 3 Start of Frame used in all packets but the first one of FC sequence. Default setting of SOFn3. |

### 8.2.3.20.4 Rx FC EOF Flags Register - REOFF (0x05158; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| EOF0 | 7:0 | 0x41 | End Of Frame 0.<br>By default it is set to EOFn code used in all packets but the last one on a sequence. |
| EOF1 | 15:8 | 0x42 | End Of Frame 1.<br>By default it is set to EOFt code used to close a sequence. |
| EOF2 | 23:16 | 0x49 | End Of Frame 2.<br>By default it is set to EOFni code. |
| EOF3 | 31:24 | 0x50 | End Of Frame 3.<br>By default it is set to EOFa code. |

### 8.2.3.20.5 FC Receive Control — FCRXCTRL (0x05100; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| FCOELLI | 0 | 0b | Low Latency Interrupt by FCoE Frame.<br>When set to 1b any FCP-RSP frame or last data packet in a sequence with the *Sequence Initiative* bit set, generates a Low Latency Interrupt (LLI). |
| SavBad | 1 | 0 | Enable Save Bad Frame.<br>Whenset to 1b, frames with good Ethernet CRC and bad FC CRC are posted to the legacy receive queues. If the *SavBad* bit is set to 0b, such frames are discarded. In both cases frames with bad FC CRC increment the FCCRC statistic counter. |
| FRSTRDH | 2 | 0 | Enable First Read Packet Header.<br>This field impacts received packets that are off-loaded by Large FC receive while their FC payload is posted directly to the user buffers. When set, headers of the first frame that matches an FC DDP context are posted to the legacy receive queues. |
| LASTSEQH | 3 | 0 | Enable Headers of Last Frame in a Sequence.<br>This field impacts received packets that are off-loaded by Large FC receive while their FC payload is posted directly to the user buffers. When set, headers of Last Frame in a Sequence are posted to the legacy receive queues. |
| ALLH | 4 | 0 | Enable All Headers.<br>This field impacts received packets that are off-loaded by Large FC receive while their FC payload is posted directly to the user buffers. When set, headers of any received packet are posted to the legacy receive queues. |

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| FRSTSEQH | 5 | 0 | Enable First Sequence Packet Header.<br>This field impacts received packets that are off-loaded by Large FC receive while their FC payload is posted directly to the user buffers. When set, headers of the first frame in any sequence are posted to the legacy receive queues. |
| ICRC | 6 | 0 | Ignore Bad FC CRC.<br>When set, the 82599 ignores bad FC CRC. In this case packets might be processed by the Large FC receive even if they carry bad FC CRC. |
| FCCRCBO | 7 | 1 | FC CRC Byte Ordering.<br>When set to 1b, the FC CRC bytes are treated in Rx as big Endian. Whenset to 0b, the FC CRC are treated as little endian (as Ethernet CRC). This bit should be set to the same value as DMATXCTL.FCCRCBO. |
| FCOEVER | 11:8 | 0 | Supported FCoE Version Number.<br>FCoE frames that carry higher version number than *FCOEVER* are not processed by the FCoE Rx offload logic. |
| Reserved | 31:12 | 0x0 | Reserved. |

### 8.2.3.20.6 FCoE Redirection Control — FCRECTL (0x0ED00; RW)

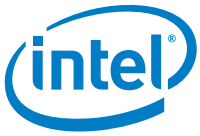
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| ENA | 0 | 0b | FC Redirection Enable.<br>When cleared, the redirection table is not active. When set to 1b the FC redirection is enabled.<br>*Software Note:* When FC redirection is enabled, the Pool Enable and the Queue Enable bits in the ETQF and ETQS registers must be cleared for FCoE data packets. |
| Reserved | 31:1 | 0x0 | Reserved. |

### 8.2.3.20.7 FCoE Redirection Table — FCRETA[n] (0x0ED10 + 4*n, n=0...7; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Table Entry | 6:0 | 0x0 | Table Entry.<br>Defines the redirection output queue number. Register 'n' is the table entry index 'n' which is the matched value to the 3 LS bits of the FC exchange ID. |
| Reserved | 31:7 | 0x0 | Reserved. |

### 8.2.3.20.8 FC User Descriptor PTR Low — FCPTRL (0x02410; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PTR_LOW | 31:0 | N/A | User Descriptor PTR Low.<br>Four least significant bytes of the physical pointer to the user descriptor list. The pointer must be 16-byte aligned so the four LS bits are read only as zeros. |



### 8.2.3.20.9 FC User Descriptor PTR High — FCPTRH (0x02414; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PTR_HI | 31:0 | N/A | User Descriptor PTR High.<br>Four most significant bytes of the physical pointer to the user descriptor list. |

### 8.2.3.20.10 FC Buffer Control — FCBUFF (0x02418; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Valid | 0 | 0b | DMA Context Valid.<br>When set to 1b indicates that the context is valid. If software clears the *Context Valid* bit, software should poll it until it is actually cleared by hardware before unlocking the user buffers. |
| First | 1 | 0b | DMA First.<br>This bit is a status indication. Software should clear it during FC context programming. The DMA unit sets this bit when it receives a frame that matches the context and marked by the filter unit as first. |
| Last | 2 | 0b | DMA Last.<br>This bit is a status indication. Software should clear it during FC context programming.Hardware sets this bit when it exhausts the last user buffer. |
| BUFFSIZE | 4:3 | 00b | Buffer Size.<br>This field defines the user buffer size used in this context as follows:<br>00b = 4 KB.<br>10b = 16 KB.<br>01b = 8 KB.<br>11b = 64 KB. |
| Reserved | 6:5 | 00b | Reserved. |
| WRCONTX | 7 | 0b | Write DDP Context.<br>This bit should be set to 1b for write exchange context aimed for target (responder) usage. This bit should be set to 0b for read exchange context aimed for initiator (originator) usage. |
| BUFFCNT | 15:8 | 0x0 | Buffer Count.<br>Defines the number of the user buffers while 0x0 equals 256. It is programmed by software and updated by hardware during reception. |
| Offset | 31:16 | 0x0 | User Buffer Offset.<br>Byte offset within the user buffer to which the FC data of large FC receive should be posted. |

### 8.2.3.20.11 FC Receive DMA RW — FCDMARW (0x02420; RW)

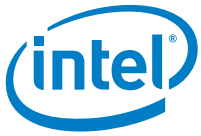
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| FCoESEL | 8:0 | 0x0 | FCoE context Select.<br>This field defines the FCoE Rx context index (equals the OX_ID for that context). |
| Reserved | 12:9 | 0x0 | Reserved. |
| Reserved | 13 | 0 | Reserved. |
| WE | 14 | 0b | Write Enable.<br>When this bit is set, the content of FCPTRL, FCPTRH and FCBUFF registers are programmed to the FCoE DMA context of index FCoESEL. This bit should never be set together with the *RE* bit in this register. |
| RE | 15 | 0b | Read Enable.<br>When this bit is set, the internal FCoE DMA context of index FCoESEL is fetched to the FCPTRL, FCPTRH and FCBUFF registers. This bit should never be set together with the *WE* bit in this register. |
| LASTSIZE | 31:16 | 0x0 | Last User Buffer Size.<br>Defines the size in bytes of the last user buffer. |

### 8.2.3.20.12 FC FLT Context — FCFLT (0x05108; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Valid | 0 | N/A | Filter Context Valid.<br>When set to 1b indicates that the context is valid. |
| First | 1 | N/A | Filter First.<br>This bit is a status indication. Software should clear it during FC context programming. The filter unit sets this bit when it receives a first frame that matches the context. |
| Reserved | 7:2 | N/A | Reserved. |
| SEQ_ID | 15:8 | N/A | Sequence ID.<br>The sequence ID of the last received frame. Initialized to 0x0 by the driver at context programming. |
| SEQ_CNT | 31:16 | N/A | Sequence Count.<br>The sequence count of the expected received frame. Initialized to 0x0 by the driver at context programming. |

### 8.2.3.20.13 FC Offset Parameter — FCPARAM (0x051D8; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PARAM | 31:0 | 0x0 | FC Parameter.<br>This field contains the expected FC parameter in the next received frame. Initialized to 0x0 by the driver at context programming.<br>*Note:* Field is defined in big endian (LS byte is first on the wire). |



## 8.2.3.20.14 FC Filter RW Control — FCFLTRW (0x05110; WO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| FCoESEL | 8:0 | 0x0 | FCoE context Select.<br>This field defines the FCoE Rx context index (equals the OX_ID for that context). |
| Reserve | 12:9 | 0x0 | Reserved. |
| Re-Validate | 13 | 0b | Fast re-validation of the filter context.<br>Setting this bit together with the *WE* bit in this register validates the selected filter context. Hardware sets the *Valid* bit and clears the *First* bit (described in the FCFLT register) while keeping all other filter parameters intact. |
| WE | 14 | 0b | Write Enable.<br>When this bit is set, the content of the FCFLT register is programmed to the filter of index FCoESEL. This bit should never be set together with the *RE* bit in this register. |
| RE | 15 | 0b | Read Enable.<br>When this bit is set, the internal filter context of index FCoESEL is fetched to the FCFLT register. This bit should never be set together with the *WE* bit in this register. |
| Reserve | 31:16 | 0x0 | Reserved. |

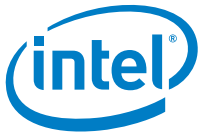
## 8.2.3.21 Flow Director Registers

Global settings registers.

### 8.2.3.21.1 Flow Director Filters Control Register — FDIRCTRL (0x0EE00; RW)

**Note:** This register should be configured ONLY as part of the flow director initialization flow or clearing the flow director table. Programming of this register with non-zero value PBALLOC initializes the flow director table.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PBALLOC | 1:0 | 00b | Memory allocation for the flow director filters.<br>00b = No memory allocation — Flow Director Filters are disabled<br>01b = 64 KB (8 K minus 2 signature filters or 2 K minus 2 perfect match filters).<br>10b = 128 KB (16 K minus 2 signature filters or 4 K minus 2 perfect match filters).<br>11b = 256 KB (32 K minus 2 signature filters or 8 K minus 2 perfect match filters). |
| Reserved | 2 | 0b | Reserved. |
| INIT-Done | 3 | 0b | Flow director initialization completion indication (read only status).<br>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), software must wait for the *INIT-Done* indication before Rx is enabled. |
| Perfect-Match | 4 | 0b | Flow director filters mode of operation.<br>When set to 1b, hardware supports perfect match filters according to PBALLOC. When cleared to 0b, hardware supports signature filters according to PBALLOC. |
| Report-Status | 5 | 0b | Report flow director filter's status in the *RSS* field of the Rx descriptor for packets that matches a flow director filter.<br>Enabling the flow director filter's status, the RXCSUM.PCSD bit should be set as well (disabling the fragment checksum).<br>*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 | Reserved. |
| Report-Status always | 7 | 0b | Report flow director status in the *RSS* field of the Rx descriptor on any packet that can be candidates for the flow director filters.<br>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 | Absolute Rx queue index used for the dropped packets.<br>Software can set this queue to an empty one by setting RDLEN[n] to 0x0. |
| Reserved | 15 | 0b | Reserved. |
| Flex-Offset | 20:16 | 0x0 | Offset within the first 64 bytes of the packet of a flexible 2-byte tuple.<br>The offset is defined in word units counted from the first byte of the destination Ethernet MAC address. |
| Reserved | 23:21 | 0x0 | Reserved. |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Max-Length | 27:24 | 0x0 | Maximum linked list length.<br>This field defines the maximum recommended linked list associated to any hash value (defined in units of two filters). Packets that match filters that exceed the *Max-Length* are reported with an active *Length* bit in the *Extended Error* field. In addition, drop filters that exceed the *Max-Length* are posted to the Rx queue defined in the filter context rather than the *Drop-Queue* defined in this register.<br>*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 can be a good choice. |
| Full-Thresh | 31:28 | 0x0 | Full threshold is a recommended minimum number of flows that should remain unused (defined in units of 16 filters).<br>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.<br>*Note:* When the flow director filters are used completely, hardware discards silently further filters programming. |

### 8.2.3.21.2 Flow Director Filters Lookup Table HASH Key — FDIRHKEY (0x0EE68; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Key | 31:0 | 0x80000001 | Programmable hash lookup table key. |

### 8.2.3.21.3 Flow Director Filters Signature Hash Key — FDIRSKEY (0x0EE6C; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Key | 31:0 | 0x80800101 | Programmable Signature Key. |

### 8.2.3.21.4 Flow Director Filters DIPv4 Mask — FDIRDIP4M (0x0EE3C; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| IPM | 31:0 | 0x0 | Mask Destination IPv4 Address. Each cleared bit means that the associated bit of the destination IPv4 address is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the destination IPv4 address is ignored (masked out).<br>The LS bit of this register matches the first byte on the wire. |

### 8.2.3.21.5 Flow Director Filters Source IPv4 Mask — FDIRSIP4M (0x0EE40; RW)

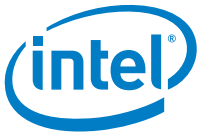
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| IPM | 31:0 | 0x0 | Mask Source IPv4 Address.<br>Each cleared bit means that the associated bit of the source IPv4 address is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the source IPv4 address is ignored (masked out).<br>The LS bit of this register matches the first byte on the wire. |

### 8.2.3.21.6 Flow Director Filters TCP Mask — FDIRTCPM (0x0EE44; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SPortM | 15:0 | 0x0 | Mask TCP Source Port.<br>Each cleared bit means that the associated bit of the TCP source port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the TCP source port is ignored (masked out).<br>*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 | Mask TCP Destination Port.<br>Each cleared bit means that the associated bit of the TCP destination port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the TCP destination port is ignored (masked out).<br>*Note:* This register is swizzle the same as the FDIRTCPM.SPortM. |

### 8.2.3.21.7 Flow Director Filters UDP Mask — FDIRUDPM (0x0EE48; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SPortM | 15:0 | 0x0 | Mask UDP Source Port.<br>Each cleared bit means that the associated bit of the UDP source port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the UDP source port is ignored (masked out).<br>*Note:* This register is swizzle the same as the FDIRTCPM.SPortM. |
| DPortM | 31:16 | 0x0 | Mask UDP Destination Port.<br>Each cleared bit means that the associated bit of the UDP destination port is meaningful for the filtering functionality. Each bit set to 1b means that the associated bit of the UDP destination port is ignored (masked out).<br>*Note:* This register is swizzle the same as the FDIRTCPM.SPortM. |



### 8.2.3.21.8 Flow Director Filters IPv6 Mask — FDIRIP6M (0x0EE74; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SIPM | 15:0 | 0x0 | Mask Source IPv6 address.<br>Each cleared bit means that the associated byte of the source IPv6 address is meaningful for the filtering functionality. Each bit set to 1b means that the associated byte of the source IPv6 address is ignored (masked out).<br>The LS bit of this register matches the first byte on the wire. |
| DIPM | 31:16 | 0x0 | Mask Destination IPv6 address.<br>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).<br>The entire 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.<br>The LS bit of this register matches the first byte on the wire. |

### 8.2.3.21.9 Flow Director Filters Other Mask — FDIRM (0x0EE70; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| VLANID | 0 | 0b | Mask VLAN ID tag.<br>When cleared the 12 bits of the VLAN ID tag are meaningful for the filtering functionality. |
| VLANP | 1 | 0b | Mask VLAN Priority tag.<br>When cleared the 3 bits of the VLAN Priority are meaningful for the filtering functionality. |
| POOL | 2 | 0b | Mask Pool.<br>When cleared the target pool number is meaningful for the filtering functionality. |
| L4P | 3 | 0b | Mask L4 Protocol.<br>When cleared the UDP/TCP/SCTP protocol type is meaningful for the filtering functionality.<br>*Note:* For the flow director filtering aspects, SCTP is treated as if it is TCP. |
| FLEX | 4 | 0b | Mask Flexible Tuple.<br>When cleared the 2 bytes of the flexible tuple are meaningful for the filtering functionality. |
| DIPv6 | 5 | 0b | Mask Destination IPv6.<br>When cleared the compare against the IP6AT filter is meaningful for IPv6 packets. |
| Reserved | 31:6 | 0x0 | Reserved. |

Global Status / Statistics Registers

### 8.2.3.21.10 Flow Director Filters Free — FDIRFREE (0x0EE38; RW)

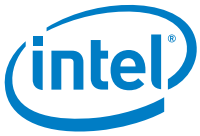
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| FREE | 15:0 | 0x8000 | Number of free (non programmed) filters in the flow director Filters logic. |
| Reserved | 30:16 | 0x0 | Reserved. |
| Reserved | 31 | 0b | Reserved. |

### 8.2.3.21.11 Flow Director Filters Length — FDIRLEN (0x0EE4C; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MAXLEN | 5:0 | 0x0 | Longest linked list of filters in the table.<br>This field records the length of the longest linked list that is updated since the last time this register was read by software. The longest bucket reported by this field includes MAXLEN + 1 filters. |
| Reserved | 7:6 | 00b | Reserved. |
| Bucket Length | 13:8 | 0x0 | The length of the linked list indicated by a query command.<br>This field is valid following a query command completion. |
| Reserved | 15:14 | 00b | Reserved. |
| Reserved | 30:16 | 0x0 | Reserved. |
| Reserved | 31 | 0b | Reserved. |

### 8.2.3.21.12 Flow Director Filters Usage Statistics — FDIRUSTAT (0x0EE50; RW/RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| ADD | 15:0 | 0x0 | Number of added filters.<br>This field counts the number of added filters to the flow director filters logic. The counter is stacked at 0xFFFF and cleared on read. |
| REMOVE | 31:16 | 0x0 | Number of removed filters.<br>This field counts the number of removed filters to the flow director filters logic. The counter is stacked at 0xFFFF and cleared on read. |



### 8.2.3.21.13 Flow Director Filters Failed Usage Statistics — FDIRFSTAT (0x0EE54; RW/RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| FADD | 7:0 | 0x0 | Number of filters addition events that do not change the number of free (non programmed) filters in the flow director filters logic (FDIRFREE.FREE).<br>These events can be either filters update, filters collision, or tentative of filter additions when there is no sufficient space remaining in the filter table. The counter is stacked at 0xFF and cleared on read. |
| FREMOVE | 15:8 | 0x0 | Number of failed removed filters.<br>The counter is stacked at 0xFF and cleared on read. |
| Reserved | 31:16 | 0x0 | Reserved. |

### 8.2.3.21.14 Flow Director Filters Match Statistics — FDIRMATCH (0x0EE58; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PCNT | 31:0 | 0x0 | Number of packets that matched any flow director filter.<br>The counter is stacked at 0xFF…F and cleared on read.<br>*Note:* This counter can include packets that match the L2 filters or 5 tuple filters or Syn filters even if they are enabled for queue assignment. |

### 8.2.3.21.15 Flow Director Filters Miss Match Statistics — FDIRMISS (0x0EE5C; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PCNT | 31:0 | 0x0 | Number of packets that missed matched any flow director filter.<br>The counter is stacked at 0xFF…F and cleared on read. |

Flow Programming Registers

### 8.2.3.21.16 Flow Director Filters Source IPv6 — FDIRSIPv6[n] (0x0EE0C + 4*n, n=0…2; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| IP6SA | 31:0 | 0x0 | Three MS DWords of the source IPv6 address.<br>While the LS byte of FDIRSIPv6[0] is first on the wire. The FDIRIPSA contains the LS Dword of the IP6 address while its MS byte is last on the wire. |

### 8.2.3.21.17 Flow Director Filters IP SA — FDIRIPSA (0x0EE18; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| IP4SA | 31:0 | 0x0 | Source IPv4 address or LS Dword of the Source IPv6 address.<br>While the field is defined in big endian (LS byte is first on the wire). |

### 8.2.3.21.18 Flow Director Filters IP DA — FDIRIPDA (0x0EE1C; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| IP4DA | 31:0 | 0x0 | Destination IPv4 address. While the field is defined in big endian (LS byte is first on the wire). |

### 8.2.3.21.19 Flow Director Filters Port — FDIRPORT (0x0EE20; RW)

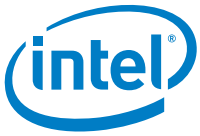
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Source | 15:0 | 0x0 | Source Port number while the field is defined in Little Endian (MS byte is first on the wire).<br>*Note:* For SCTP filter the Source and Destination port numbers must be set to zero (while the HW does not check it). |
| Destination | 31:16 | 0x0 | Destination Port number while the field is defined in Little Endian (MS byte is first on the wire).<br>*Note:* For SCTP filter the Source and Destination port numbers must be set to zero (while the HW does not check it). |

### 8.2.3.21.20 Flow Director Filters VLAN and FLEX Bytes — FDIRVLAN (0x0EE24; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Vlan | 15:0 | 0x0 | Vlan Tag while the field is defined in Little Endian (MS byte is first on the wire).<br>The CFI bit must be set to Zero while it is not checked by hardware. |
| Flex | 31:16 | 0x0 | 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). |

### 8.2.3.21.21 Flow Director Filters Hash Signature — FDIRHASH (0x0EE28; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Hash | 14:0 | 0x0 | Bucket hash value that identifies a filter's linked list. |
| Bucket Valid | 15 | 0b | The *Valid* bit is set by hardware each time there is at least one filter assigned to this hash. |

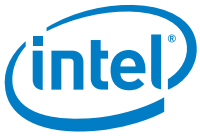


| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Signature / SW-Index | 30:16 | 0x0 | Flow director filter signature for signature filters and software-index for perfect match filters. |
| Reserved | 31 | 0b | Reserved. |

## 8.2.3.21.22 Flow Director Filters Command Register — FDIRCMD (0x0EE2C; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| CMD | 1:0 | 00b | Flow Director Filter Programming Command.<br>00b = No Action<br>01b = Add Flow<br>10b = Remove Flow<br>11b = Query Command<br>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 | Valid filter is found by the query command.<br>This bit is set by the 82599 following a query command completion. |
| Filter-Update | 3 | 0b | Filter Update Command.<br>This bit is relevant only for Add Flow command and must be set to zero in any other commands. When cleared, the filter parameters do not override existing ones if exist while setting only the collision bit. When set to 1b the new filter parameters override existing ones if exist keeping the collision bit as is. |
| IPv6DMatch | 4 | 0b | IP Destination match to IP6AT filter.<br>This bit is meaningful only for perfect match IPv6 filters. Otherwise it should be cleared by software at programming time. When set to 1b the destination IPv6 address should match the IP6AT. When cleared, the destination IPv6 address should not match the IP6AT.<br>This field can never match local VM to VM traffic. |
| L4TYPE | 6:5 | 0b | L4 Packet Type. Defines the packet as one of the following L4 types:<br>00b = Reserved<br>01b = UDP<br>10b = TCP<br>11b = SCTP<br>*Note:* Encoding of the L4TYPE for the flow director filters is defined differently than the protocol type encoding in the FTQF registers for the 128 x 5 tuple filters. |
| IPV6 | 7 | 0b | IPv6 packet type when set to 1b and IPv4 packet type at 0b.<br>*Note:* The IP type is checked always even if the filters do not check for IP address match. |
| CLEARHT | 8 | 0b | Clear Internal Flow Director Head and Tail Registers.<br>This bit is set only as part of Flow Director init. During nominal Operation it must be kept at 0b. |

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Drop | 9 | 0b | Packet drop action:<br>Receive packets that match a filter with active Drop bit and do not exceed the maximum recommended linked list length defined in FDIRCTRL.Max-Length field are posted to the global queue defined by FDIRCTRL.Drop-Queue.<br>Receive packets that match a filter with active Drop bit and exceeds the maximum recommended linked list length defined in FDIRCTRL.Max-Length field are posted to the queue defined by RX-Queue field in this register. The receive descriptor of such packets is reported with active FDIRErr(0) flag indicating that the Max-Length was exceeded.<br>The Drop Flag is useful only for perfect match filters and it should be cleared by software for Signature filters.<br>When the Drop bit is set, the Queue-EN flag must be set and Rx-Queue in this register must be valid as well. Otherwise, the result is unexpected. |
| INT | 10 | 0b | Matched packet generates a LLI. |
| Last | 11 | 0b | Last filter indication in the linked list.<br>At flow programming, software should set the last bit to 1b. Hardware can modify this bit when adding or removing flows from the same linked list. |
| Collision | 12 | 0b | Collision Indication.<br>This field is set to 1b when software programs the same multiple times. In signature based filtering, it is set when software programs a filter with the same hash and signature multiple times. It should be cleared by software when it adds a flow. It can also be set by hardware when two flows collide with the same hash and signature. During reception, this bit is reported on the Rx descriptor of packets that match the filter.<br>See bit 7 for description of the query- type. |
| Reserved | 14:13 | 00b | Reserved. |
| Queue-EN | 15 | 0b | Enable routing matched packet to the queue defined by the Rx-Queue.<br>*Note:* Packets redirection to the FDIRCTRL.Drop-Queue is not gated by the Queue-EN bit. |
| Rx-Queue | 22:16 | 0x0 | Rx Queue Index.<br>This field defines the absolute Rx queue index in all modes of operation (regardless of DCB and VT enablement). |
| Reserved | 23 | 0b | Reserved. |
| Pool | 29:24 | 0x0 | Pool number is meaningful when VT mode is enabled.<br>When both VT is not enabled, this field must be set by software to 0x0. |
| Reserved | 31:30 | 0x0 | Reserved. |



# 8.2.3.22 MAC Registers

## 8.2.3.22.1 PCS_1G Global Config Register 1 — PCS1GCFIG (0x04200; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 29:0 | 0x8 | Reserved. |
| PCS_isolate | 30 | 0b | PCS Isolate.<br>Setting this bit isolates the 1 GbE PCS logic from the MAC's data path. PCS control codes are still sent and received. |
| Reserved | 31 | 1b | Reserved. |

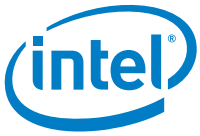
## 8.2.3.22.2 PCG_1G link Control Register — PCS1GLCTL (0x04208; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| FLV | 0 | 0 | Forced Link 1 GbE Value.<br>This bit denotes the link condition when *Force Link* is set.<br>0b = Forced link down.<br>1b = Forced 1 GbE link up. |
| Reserved | 4:1 | 0x7 | Reserved. |
| FORCE 1G LINK | 5 | 0 | Force 1 GbE Link.<br>If this bit is set then the internal LINK_OK variable is forced to *Forced Link Value*, bit 0 of this register. Else LINK_OK is decided by internal AN/SYNC state machines. This bit is only valid when the link mode is 1 GbE mode. |
| LINK LATCH LOW | 6 | 0 | Link Latch Low Enable.<br>If this bit is set then *Link OK* going LOW; negedge) is latched until a CPU read happens. Once a CPU read happens *Link OK* is continuously updated until *Link OK* again goes LOW (negedge is seen). |
| Reserved | 17:7 | 0 | Reserved. |
| AN 1G TIMEOUT EN | 18 | 1b | Auto Negotiation 1 GbE Timeout Enable.<br>This bit enables the 1 GbE auto- negotiation timeout feature. During 1 GbE auto-negotiation, if the link partner doesn't respond with auto-negotiation pages but continues to send good idle symbols then linkup is assumed. (This enables a link-up condition when a link partner is not auto-negotiation capable and does not affect otherwise). |
| Reserved | 19 | 0b | Reserved. |
| Reserved | 20 | 0b | Reserved, must be set to 0b. |
| Reserved | 24:21 | 0x0 | Reserved. |

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LINK OK FIX EN | 25 | 1b | Link OK Fix En.<br>Control for enabling/disabling LinkOK-SyncOK fix. This bit should be set to 1b for nominal operation. |
| Reserved | 31:26 | 0x0 | Reserved. |

### 8.2.3.22.3 PCS_1G Link Status Register — PCS1GLSTA (0x0420C; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 3:0 | 1110b | Reserved. |
| SYNC OK 1G | 4 | 0b | Sync OK 1 GbE.<br>This bit indicates the current value of SYN OK from the 1G PCS Sync state machine, |
| Reserved | 15:5 | 0x0 | Reserved. |
| AN 1G COMPLETE | 16 | 0b | Auto Negotiation1 GbE Complete.<br>This bit indicates that the 1 GbE auto-negotiation process completed. |
| AN PAGE RECEIVED | 17 | 0b | Auto-Negotiation Page Received.<br>This bit indicates that a link partner's page was received during auto-negotiation process. Clear on read. |
| AN 1G TIMEDOUT | 18 | 0b | Auto Negotiation1 GbE Timed Out.<br>This bit indicates 1 GbE auto-negotiation process was timed out. Valid after *AN 1G Complete* bit is set. |
| AN REMOTE FAULT | 19 | 0b | Auto Negotiation Remote Fault.<br>This bit indicates that a 1 GbE auto-negotiation page was received with remote fault indication during 1 GbE auto-negotiation process. Clear on read. |
| AN ERROR (RW) | 20 | 0b | Auto Negotiation Error.<br>This bit indicates that an auto-negotiation error condition was detected in 1 GbE auto-negotiation mode. Valid after the *AN 1G Complete* bit is set.<br>Auto-negotiation error conditions:<br>Both nodes not full duplex or remote fault indicated or received.<br>Software can also force an auto-negotiation error condition by writing to this bit (or can clear an existing auto-negotiation error condition). Cleared at the start of auto-negotiation. |
| Reserved | 31:21 | 0x0 | Reserved. |



### 8.2.3.22.4 PCS_1 Gb/s Auto Negotiation Advanced Register — PCS1GANA (0x04218; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 4:0 | 0x0 | Reserved. |
| FDC | 5 | 1b | FD: Full-Duplex.<br>Setting this bit means the local device is capable of full-duplex operation. This bit should be set to 1b for normal operation. |
| Reserved | 6 | 0b | Reserved. |
| ASM | 8:7 | 11b | ASM_DIR/PAUSE: Local PAUSE Capabilities.<br>The local device's PAUSE capability is encoded in this field.<br>00b = No PAUSE.<br>01b = Symmetric PAUSE.<br>10b = Asymmetric PAUSE toward link partner.<br>11b = Both symmetric and asymmetric PAUSE toward local device. |
| Reserved | 11:9 | 0x0 | Reserved. |
| RFLT | 13:12 | 00b | Remote Fault. The local device's remote fault condition is encoded in this field. Local device can indicate a fault by setting a non-zero remote fault encoding and re-negotiating.<br>00b = No error, link good.<br>01b = Link failure.<br>10b = Offline.<br>11b = Auto-negotiation error. |
| Reserved | 14 | 0b | Reserved. |
| NEXTP | 15 | 0b | NEXTP: Next Page Capable.<br>The local device asserts this bit to request next page transmission. Clear this bit when local device has no subsequent next pages. |
| Reserved | 31:16 | 0x0 | Reserved. |

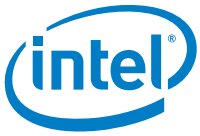
### 8.2.3.22.5 PCS_1GAN LP Ability Register — PCS1GANLP (0x0421C; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 4:0 | 0x0 | Reserved. |
| LPFD | 5 | 0b | LP Full-Duplex (SerDes).<br>When 1b, link partner is capable of full-duplex operation. When 0b, link partner is incapable of full-duplex mode. |
| LPHD | 6 | 0b | LP Half-Duplex (SerDes).<br>When 1b, link partner is capable of half-duplex operation. When 0b, link partner is incapable of half-duplex mode. |

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LPASM | 8:7 | 00b | LPASMDR/LPPAUSE(SERDES).<br>The link partner's PAUSE capability is encoded in this field.<br>00b = No PAUSE.<br>01b = Symmetric PAUSE.<br>10b = Asymmetric PAUSE toward link partner.<br>11b = Both symmetric and asymmetric PAUSE toward local device. |
| Reserved | 11:9 | 0x0 | Reserved. |
| PRF | 13:12 | 00b | LP Remote Fault (SerDes)[13:12].<br>The link partner's remote fault condition is encoded in this field.<br>00b = No error, link good.<br>10b = Link failure.<br>01b = Offline.<br>11b = Auto-negotiation error. |
| ACK | 14 | 0b | Acknowledge (SerDes).<br>The link partner has acknowledged page reception. |
| LPNEXTP | 15 | 0b | LP Next Page Capable (SerDes).<br>The link partner asserts this bit to indicate its ability to accept next pages. |
| Reserved | 31:16 | 0x0 | Reserved. |

### 8.2.3.22.6 PCS_1G Auto Negotiation Next Page Transmit Register — PCS1GANNP (0x04220; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| CODE | 10:0 | 0x0 | Message/Unformatted Code Field.<br>The message field is an 11-bit wide field that encodes 2048 possible messages. Unformatted code field is an 11-bit wide field, which can contain an arbitrary value. |
| TOGGLE | 11 | 0b | Toggle.<br>This bit is used to ensure synchronization with the link partner during a next page exchange. This bit always takes the opposite value of the *Toggle* bit in the previously exchanged link code word. The initial value of the *Toggle* bit in the first next page transmitted is the inverse of bit 11 in the base link code word and, therefore, can assume a value of 0b or 1b. The Toggle bit must be set as follows:<br>0b Previous value of the transmitted Link Code Word equaled 1b.<br>1b Previous value of the transmitted Link Code Word equaled 0b. |
| ACK2 | 12 | 0b | Acknowledge2.<br>Acknowledge is used to indicate that a device has successfully received its link partner’s link code word. |
| PGTYPE | 13 | 0b | Message/ Unformatted Page.<br>This bit is used to differentiate a message page from an unformatted page. The encodings are:<br>0b = Unformatted page.<br>1b = Message page. |
| Reserved | 14 | 0b | Reserved. |



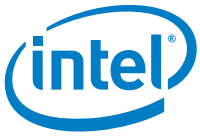
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| NXTPG | 15 | 0b | Next Page.<br>This bit is used to indicate whether or not this is the last next page to be transmitted. The encodings are:<br>0b = Last page.<br>1b = Additional next pages follow. |
| Reserved | 31:16 | 0x0 | Reserved. |

### 8.2.3.22.7 PCS_1G Auto Negotiation LP's Next Page Register — PCS1GANLPNP (0x04224; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| CODE | 10:0 | 0x0 | Message/Unformatted Code Field.<br>The message field is an 11-bit wide field that encodes 2048 possible messages. Unformatted code field is an 11-bit wide field, which can contain an arbitrary value. |
| TOGGLE | 11 | 0bb | Toggle.<br>This bit is used to ensure synchronization with the link partner during a next page exchange. This bit always takes the opposite value of the *Toggle* bit in the previously exchanged link code word. The initial value of the *Toggle* bit in the first next page transmitted is the inverse of bit 11 in the base link code word and, therefore, can assume a value of 0b or 1b. The Toggle bit must be set as follows:<br>0b = Previous value of the transmitted link code word equalled 1b.<br>1b = Previous value of the transmitted link code word equalled 0b. |
| ACK2 | 12 | 0 | Acknowledge2.<br>Acknowledge is used to indicate that a device has successfully received its link partner's link code word. |
| MSGPG | 13 | 0bb | Message Page.<br>This bit is used to differentiate a message page from an unformatted page. The encodings are:<br>0b = Unformatted page.<br>1b = Message page. |
| ACK | 14 | 0 | Acknowledge.<br>The link partner has acknowledge next page reception. |
| NXTPG | 15 | 0b | Next Page.<br>This bit is used to indicate whether or not this is the last next page to be transmitted. The encodings are:<br>0b = Last page.<br>1b = Additional next pages follow. |
| Reserved | 31:16 | 0x0 | Reserved. |

### 8.2.3.22.8 MAC Core Control 0 Register — HLREG0 (0x04240; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TXCRCEN | 0 | 1b | Tx CRC Enable.<br>Enables a CRC to be appended by hardware to a Tx packet if requested by user.<br>0b = No CRC appended, packets always passed unchanged.<br>1b = Enable CRC by hardware (default). |
| Reserved | 1 | 1b | Reserved. |
| RXCRCSTRP | 1 | 1 | Rx CRC STRIP.<br>Causes the CRC to be stripped by HW from all packets The RDRXCTL.CRCStrip must be set the same as this bit.<br>0b = No CRC Strip by HW.<br>1b = Strip CRC by HW (Default). |
| JUMBOEN | 2 | 0b | Jumbo Frame Enable.<br>Enables frames up to the size specified in Reg MAXFRS (31:16).<br>0b = Disable jumbo frames (default).<br>1b = Enable jumbo frames. |
| Reserved | 9:3 | 0x1 | Reserved. Must be set to 0x1. |
| TXPADEN | 10 | 1b | Tx Pad Frame Enable.<br>Pad short Tx frames to 64 bytes if requested by user.<br>0b = Transmit short frames with no padding.<br>1b = Pad frames (default). |
| Reserved | 14:11 | 0101b | Reserved. |
| LPBK | 15 | 0b | LOOPBACK.<br>Turn On Loopback Where Transmit Data Is Sent Back Through Receiver.<br>0b = Loopback disabled (Default).<br>1b = Loopback enabled. |
| MDCSPD | 16 | 1b | MDC SPEED.<br>High or Low Speed MDC Clock Frequency To PCS, XGXS, WIS, etc.<br>**MDCSPD / Freq at 10 GbE / Freq at 1 GbEs / Freq at 100 Mb/s**<br>0b / 2.4 MHz / 240 KHz / 240 KHz<br>1b / 24 MHz / 2.4 MHz / 240 KHz<br>*Note:* 1b = default. |
| CONTMDC | 17 | 0b | Continuous MDC.<br>Turn Off MDC Between MDIO Packets<br>0b = MDC Off Between Packets (default)<br>1b = Continuous MDC |
| Reserved | 19:18 | 00b | Reserved. |
| PREPEND | 23:20 | 0x0 | Prepend Value.<br>Number of 32-bit words starting after the preamble and SFD, to exclude from the CRC generator and checker (default – 0x0). |
| Reserved | 24 | 0b | Reserved. |
| Reserved | 26:25 | 00b | Reserved. |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RXLNGTHERREN | 27 | 1b | Rx Length Error Reporting.<br>0b = Disable reporting of all rx_length_err events.<br>1b = Enable reporting of rx_length_err events if length field < 0x0600. |
| RXPADSTRIPEN | 28 | 0b | Rx Padding Strip Enable.<br>0b = Do not strip padding from Rx packets with length field < 64 (default).<br>1b = Strip padding from Rx packets with length field < 64 (debug only).<br>*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 | Reserved. |

## 8.2.3.22.9 MAC Core Status 1 Register- HLREG1 (0x04244; RO)

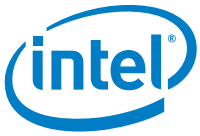
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 3:0 | 0001b | Reserved. |
| Reserved | 4 | 0b | Reserved. |
| RXERRSYM | 5 | 0b | Rx Error Symbol.<br>Error Symbol During Rx Packet (Latch High, Clear On Read).<br>0b = No error symbol (default).<br>1b = Error symbol received. |
| RXILLSYM | 6 | 0b | Rx Illegal Symbol.<br>Illegal Symbol During Rx Packet (Latch High, Clear On Read).<br>0b = No illegal symbol received (default).<br>1b = Illegal symbol received. |
| RXIDLERR | 7 | 0b | Rx Idle Error.<br>Non Idle Symbol During Idle Period (Latch High, Clear On Read).<br>0b = No idle errors received (default).<br>1b = Idle error received. |
| RXLCLFLT | 8 | 0b | Rx Local Fault.<br>Fault reported from PMD, PMA, or PCS (Latch High, Clear On Read).<br>0b = No local fault (default).<br>1b = Local fault is or was active. |
| RXRMTFLT | 9 | 0b | Rx Remote Fault.<br>Link Partner Reported Remote Fault (Latch High, Clear On Read).<br>0b = No remote fault (default).<br>1b = Remote fault is or was active. |
| Reserved | 31:10 | 0x0 | Reserved. |

### 8.2.3.22.10 Pause and Pace Register — PAP (0x04248; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 15:0 | 0xFFFF | Reserved. |
| PACE | 19:16 | 0x0 | 0000b = 10 GbE (LAN)<br>0001b = 1 GbE<br>0010b = 2 GbE<br>0011b = 3 GbE<br>0100b = 4 GbE<br>0101b = 5 GbE<br>0110b = 6 GbE<br>0111b = 7 GbE<br>1000b = 8 GbE<br>1001b = 9 GbE<br>1111b = 9.294196 GbE (WAN)<br>All other values are reserved. |
| Reserved | 31:20 | 0x0 | Reserved |

### 8.2.3.22.11 MDI Single Command and Address — MSCA (0x0425C; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MDIADD | 15:0 | 0x0000 | MDI Address.<br>Address used for new protocol MDI accesses (default – 0x0000). |
| DEVADD | 20:16 | 0x0 | DeviceType/Register Address.<br>Five bits representing either device type if STCODE = 00b or register address if STCODE = 01b. |
| PHYADD | 25:21 | 0x0b | PHY Address.<br>The address of the external device. |
| OPCODE | 27:26 | 00 | OP Code.<br>Two bits identifying operation to be performed (default – 00b).<br>00b = Address cycle (new protocol only).<br>01b = Write operation.<br>10b = Read increment address (new protocol only) or read operation (old protocol only).<br>11b = Read operation (new protocol only). |
| STCODE | 29:28 | 01b | ST Code.<br>Two Bits Identifying Start Of Frame And Old Or New Protocol (Default – 01).<br>00b = New protocol.<br>01b = Old protocol.<br>1Xb = Illegal. |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MDICMD | 30 | 0b | MDI Command. Perform the MDIO Operation in this register, cleared when done.<br>0b = MDI Ready, operation complete (default).<br>1b = Perform operation, operation in progress. |
| Reserved | 31 | 0b | Reserved. |

### 8.2.3.22.12 MDI Single Read and Write Data — MSRWD (0x04260; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MDIWRDATA | 15:0 | 0x0 | MDI Write Data.<br>Write data For MDI writes to the external device. |
| MDIRDDATA | 31:16 | 0x0 | MDI Read Data.<br>Read data from the external device (RO). |

### 8.2.3.22.13 Max Frame Size — MAXFRS (0x04268; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 15:0 | 0x0 | Reserved. |
| MFS | 31:16 | 0x5EE | 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 = 1b). When jumbo frames are not enabled the 82599 uses a hardwired value of 1518 for this field.<br>The MFS does not include the 4 bytes of the VLAN header. Packets with VLAN header can be as large as MFS + 4. When double VLAN is enabled, the device adds 8 to the MFS for any packets.<br>This value has no effect on transmit frames; it is the responsibility of software to limit the size of transmit frames. |

### 8.2.3.22.14 XGXS Status 1 — PCSS1 (0x4288; RO)

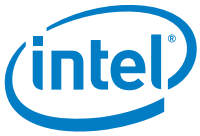
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 1:0 | 00b | Reserved. |
| PCS Receive Link Status | 2 | 0b | 0b = PCS receive link down.The receive link status remains cleared until it is read (latching low).<br>1b = PCS receive link up. For 10BASE-X ->lanes de-skewed. |
| Reserved | 6:3 | 0x0 | Reserved. |
| Local Fault | 7 | 1b | 0b = No LF detected on receive path.<br>1b = LF detected on transmit or receive path. The *LF* bit is set to one when either of the local fault bits located in PCS Status 2 register are set to a 1b. |
| Reserved | 31:8 | 0x0 | Reserved. |

### 8.2.3.22.15 XGXS Status 2 — PCSS2 (0x0428C; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| 10GBASE-R Capable | 0 | 0b | 0b = PCS is not able to support 10GBASE-R port type.<br>1b = PCS is able to support 10GBASE-R port type. |
| 10GBASE-X capable | 1 | 1b | 0b = PCS is not able to support 10GBASE-X port type.<br>1b = PCS is able to support 10GBASE-X port type. |
| 10GBASE-W capable | 2 | 0b | 0b = PCS is not able to support 10GBASE-W port type.<br>1b = PCS is able to support 10GBASE-W port type. |
| Reserved | 9:3 | 0x0 | Reserved. |
| Receive local fault | 10 | 1b | 0b = No local fault condition on the receive path (latch high)<br>1b = Local fault condition on the receive path. |
| Transmit local fault | 11 | 0b | 0b = No local fault condition on the transmit path (latch high)<br>1b = Local fault condition on the transmit path. |
| Reserved | 13:12 | 00b | Reserved. |
| Device present | 15:14 | 10b | 00b = No device responding at this address.<br>01b = No device responding at this address.<br>10b = Device responding at this address.<br>11b = No device responding at this address. |
| Reserved | 31:16 | 0x0 | Reserved. |

### 8.2.3.22.16 10GBASE-X PCS Status — XPCSS (0x04290; RO)

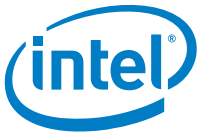
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Lane 0 sync | 0 | 0b | 0b = Lane 0 is not synchronized.<br>1b = Lane 0 is synchronized. |
| Lane 1 sync | 1 | 0b | 0b = Lane 1 is not synchronized.<br>1b = Lane 1 is synchronized. |
| Lane 2 sync | 2 | 0b | 0b = Lane 2 is not synchronized.<br>1b = Lane 2 is synchronized. |
| Lane 3 sync | 3 | 0b | 0b = Lane 3 is not synchronized.<br>1b = Lane 3 is synchronized. |
| Reserved | 11: 4 | 0b | Ignore when read. |
| 10GBASE-X lane alignment status | 12 | 0b | 0b = 10GBASE-X PCS receive lanes not aligned.<br>1b = 10GBASE-X PCS receive lanes aligned (align_status – good). |
| Reserved | 15:13 | 0x0 | Reserved, ignore when read. |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| De-skew error | 16 | 0b | 0b = Indicates no de-skew error was detected (latch high).<br>1b = Indicates a de-skew error was detected. |
| Align column count 4 | 17 | 0b | 0b = Indicates the align column count is less than four (latch high).<br>1b = Indicates the align column count has reached four. |
| Lane 0 invalid code | 18 | 0b | 0b = Indicates no invalid code was detected (latch high).<br>1b = Indicates an invalid code was detected for that lane. |
| Lane 1 invalid code | 19 | 0b | 0b = Indicates no invalid code was detected (latch high).<br>1b = Indicates an invalid code was detected for that lane. |
| Lane 2 invalid code | 20 | 0b | 0b = Indicates no invalid code was detected (latch high).<br>1b = Indicates an invalid code was detected for that lane. |
| Lane 3 invalid code | 21 | 0b | 0b = Indicates no invalid code was detected (latch high).<br>1b = Indicates an invalid code was detected for that lane. |
| Lane 0 comma count4 | 22 | 0b | 0b = Indicates the comma count for that lane is less than four (latch high).<br>1b = Indicates the comma count for that lane has reached four. |
| Lane 1 comma count 4 | 23 | 0b | 0b = Indicates the comma count for that lane is less than four (latch high).<br>1b = Indicates the comma count for that lane has reached four. |
| Lane 2 comma count 4 | 24 | 0b | 0b = Indicates the comma count for that lane is less than four (latch high).<br>1b = Indicates the comma count for that lane has reached four. |
| Lane 3 comma count 4 | 25 | 0b | 0b = Indicates the comma count for that lane is less than four (latch high).<br>1b = Indicates the comma count for that lane has reached four. |
| Lane 0 Signal Detect | 26 | 0b | 0b = Indicates noise, no signal is detected.<br>1b = Indicates a signal is detected. |
| Lane 1 Signal Detect | 27 | 0b | 0b = Indicates noise, no signal is detected.<br>1b = Indicates a signal is detected. |
| Lane 2 Signal Detect | 28 | 0b | 0b = Indicates noise, no signal is detected.<br>1b = Indicates a signal is detected. |
| Lane 3 Signal Detect | 29 | 0b | 0b = Indicates noise, no signal is detected.<br>1b = Indicates a signal is detected. |
| Reserved | 31:30 | 0b | Reserved. |

### 8.2.3.22.17 SerDes Interface Control Register — SERDESC (0x04298; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Tx_lanes_polarity | 3:0 | 0* | Bit 3 = Changes bits polarity of MAC Tx lane 3.<br>Bit 2 = Changes bits polarity of MAC Tx lane 2.<br>Bit 1 = Changes bits polarity of MAC Tx lane 1.<br>Bit 0 = Changes bits polarity of MAC Tx lane 0.<br>Changes bits polarity if set to 1b. |
| Rx_lanes_polarity | 7:4 | 0* | Bit 7 = Changes bits polarity of MAC Rx lane 3.<br>Bit 6 = Changes bits polarity of MAC Rx lane 2.<br>Bit 5 = Changes bits polarity of MAC Rx lane 1.<br>Bit 4 = Changes bits polarity of MAC Rx lane 0.<br>Changes bits polarity if set to 1b. |
| swizzle_tx_lanes | 11:8 | 0* | Bit 11 = Swizzles bits of MAC Tx lane 3.<br>Bit 10 = Swizzles bits of MAC Tx lane 2.<br>Bit 9 = Swizzles bits of MAC Tx lane 1.<br>Bit 8 = Swizzles bits of MAC Tx lane 0.<br>Swizzles bits if set to 1b.<br>These bits are for debug only – software should not change the default EEPROM value. |
| swizzle_rx_lanes | 15:12 | 0* | Bit 15 = Swizzles bits of MAC Rx lane 3.<br>Bit 14 = Swizzles bits of MAC Rx lane 2.<br>Bit 13 = Swizzles bits of MAC Rx lane 1.<br>Bit 12 = Swizzles bits of MAC Rx lane 0.<br>Swizzles bits if set to 1b.<br>These bits are for debug only – software should not change the default EEPROM value. |
| swap_tx_lane_3 | 17:16 | 11b* | Determines Core destination Tx lane for MAC Tx lane 3. |
| swap_tx_lane_2 | 19:18 | 10b* | Determines Core destination Tx lane for MAC Tx lane 2. |
| swap_tx_lane_1 | 21:20 | 01b* | Determines Core destination Tx lane for MAC Tx lane 1. |
| swap_tx_lane_0 | 23:22 | 00b* | Determines Core destination Tx lane for MAC tx lane 0.<br>00b = MAC Tx lane 0 to Core Tx lane 0.<br>01b = MAC Tx lane 0 to Core Tx lane 1.<br>10b = MAC Tx lane 0 to Core Tx lane 2.<br>11b = MAC Tx lane 0 to Core Tx lane 3. |
| swap_rx_lane_3 | 25:24 | 11b* | Determines which Core lane is mapped to MAC Rx lane 3. |
| swap_rx_lane_2 | 27:26 | 10b* | Determines which Core lane is mapped to MAC Rx lane 2. |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| swap_rx_lane_1 | 29:28 | 01b* | Determines which Core lane is mapped to MAC Rx lane 1. |
| swap_rx_lane_0 | 31:30 | 00b* | Determines which Core lane is mapped to MAC Rx lane 0.<br>00b = Core Rx lane 0 to MAC Rx lane 0.<br>01b = Core Rx lane 1 to MAC Rx lane 0.<br>10b = Core Rx lane 2 to MAC Rx lane 0.<br>11b = Core Rx lane 3 to MAC Rx lane 0. |

* Also programmable via EEPROM.

### 8.2.3.22.18 FIFO Status/CNTL Report Register — MACS (0x0429C; RW)

This register reports FIFO status in xgmii_mux.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| XGXS SYNC Fix Disable | 0 | 0b | Use shift-fsm control for the XGXS sync process.<br>0b = Normal functionality (default).<br>1b = Use shift-fsm control, disable fix (debug only). |
| XGMII-GMII Tx END Fix Disable | 1 | 0b | Disable tx_end on link-down.<br>0b = Normal functionality, link down causes tx_end (default).<br>1b = Disable tx_end on link-down (debug only). |
| XGXS Deskew Fix Disable | 2 | 0b | Disable align on invalid fix.<br>0b = Normal functionality (default).<br>1b = Disable align on invalid fix (debug only). |
| Nonce Match Disable | 3 | 0b | Disable nonce match.<br>0b = Normal functionality (default).<br>1b = Disable nonce match (debug only). |
| Reserved | 15:4 | 0x0 | Reserved.<br>On a write access to this field the SW should maintain the value of this field. |
| Config fault length | 23:16 | 0x1F | Sets the length in clock cycles of LF stream. |
| Config FIFO threshold | 27:24 | 0x6 | Determines threshold for asynchronous FIFO (generation of data_available signal is determined by cfg_fifo_th[3:0]). |
| tx FIFO underrun | 28 | 0b | Indicates FIFO under run in xgmii_mux_tx_fifo. |
| tx FIFO overrun | 29 | 0b | Indicates FIFO overrun in xgmii_mux_tx_fifo. |
| rx FIFO underrun | 30 | 0b | Indicates FIFO under run in xgmii_mux_rx_fifo. |
| rx FIFO overrun | 31 | 0b | Indicates FIFO overrun in xgmii_mux_rx_fifo. |

### 8.2.3.22.19 Auto Negotiation Control Register — AUTOC (0x042A0; RW)

**Note:** The 82599 Device Firmware may access AUTOC register in parallel to software driver and a synchronization between them is needed. For more information see Section 10.5.4.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| FLU | 0 | 0b | Force Link Up.<br>0b = Normal mode.<br>1b = MAC forced to link_up. Link is active in the speed configured in AUTOC.LMS.<br>This setting forces the auto-negotiation arbitration state machine to AN_GOOD and sets the link_up indication regardless of the XGXS/ PCS_1G status. |
| ANACK2 | 1 | 0b* | Auto-Negotiation Ack2 field.<br>This value is transmitted in the *Achnowledge2* field of the null next page that is transmitted during a next page handshake. |
| ANSF | 6:2 | 00001b* | Auto-Negotiation Selector Field.<br>This value will be used as the Selector Field in the Link Control Word during Clause 73 Backplane Auto–Negotiation process. (Default value set according to 802.3ap-2007). |
| 10G_PMA_PMD_PARALLEL | 8:7 | 01b* | Define 10 GbE PMA/PMD over four differential pairs (Tx and Rx each).<br>00b = XAUI PMA/PMD.<br>01 = KX4 PMA/PMD.<br>10 = CX4 PMA/PMD.<br>11 = Reserved. |
| 1G_PMA_PMD | 9 | 1b* | PMA/PMD used for 1 GbE.<br>0b = SFI PMA/PMD (the AUTOC.LMS should be set to 000b).<br>1b = KX or BX PMA/PMD. |
| D10GMP | 10 | 0b* | Disables 10 GbE Parallel Detect on Dx (Dr/D3) without main–power.<br>0b = No specific action.<br>1b = Disables 10 GbE Parallel Detect when main power is removed. This bit is valid only if RATD is set.<br>*Note:* If MNG_VETO bit is set, any low-power link mode changes will be hold off. |
| RATD | 11 | 0b* | Restarts auto-negotiation on transition to Dx.<br>This bit enables the functionality to restart KX/KX4/KR backplane auto-negotiation on transition to Dx (Dr/D3), targeting an 1GbE link.<br>0b = Does not restart auto-negotiation when the 82599 moves to the Dx state.<br>1b = Restarts auto-negotiation to reach a low–power link mode (1 GbE link) when the 82599 transitions to the Dx state. Only 1GbE will be advertised by auto-negotiation. In this case, if a partner doesn't have 1GbE capabilities, a link will not be established. 10GbE can still be achieved by Parallel Detect of a XAUI partner if D10GMP bit is clear.<br>*Note:* If MNG_VETO bit is set, any low-power link mode changes will be hold off. |

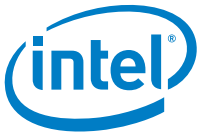
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Restart_AN | 12 | 0b* | Applies new link settings and restarts relative auto-negotiation process (self–clearing bit).<br>0b = No action needed.<br>1b = Applies new link settings and restarts auto-negotiation.<br>*Note:* This bit must be set to make any new link settings affective as indicated in Section 3.7.4.2. |
| LMS | 15:13 | 100b* | Link Mode Select. Selects the active link mode:<br>000b = 1 GbE link (no backplane auto-negotiation).<br>001b = 10 GbE parallel link (KX4 – no backplane auto-negotiation).<br>010b = 1 GbE link with clause 37 auto-negotiation enable (BX interface).<br>011b = 10 GbE serial link (SFI – no backplane auto-negotiation).<br>100b = KX/KX4/KR backplane auto-negotiation enable. 1 GbE (Clause 37) auto-negotiation disabled.<br>101b = SGMII 100M/1 GbE link.<br>110b = KX/KX4/KR backplane auto-negotiation enable. 1 GbE (Clause 37) auto-negotiation enable.<br>111b = KX/KX4/KR auto-negotiation enable. SGMII 100 Mb/s and 1GbE (in KX) enable. |
| KR_support | 16 | 1b* | Configures the *A2* bit of the *Technology Ability Field* in the auto-negotiation word while *A0:A1* fields are configured in the KX_support field (bits 31:30):<br>0b = KR not supported. Value is Illegal if KX and KX4 are also not supported (AUTOC.KX_support – 00b).<br>1b = KR supported.<br>*Note:* This bit is not relevant to the parallel detect process. |
| FECR | 17 | 0b* | FEC Requested.<br>Configures the *F1* bit in the backplane auto-negotiation base link code word. Should be set to 1b only if KR ability is set to 1b (AUTOC.KR = 1b).<br>0b = FEC not requested from link partner.<br>1b = FEC requested from link partner. |
| FECA | 18 | 1b* | FEC Ability.<br>Configures the *F0* bit in the backplane auto-negotiation base link code word. Should be set to 1b only if KR ability is set to 1b (AUTOC.KR = 1b).<br>0b = FEC not supported.<br>1b = FEC supported. |
| ANRXAT | 22:19 | 0011b* | Backplane Auto-Negotiation Rx Align Threshold.<br>Sets threshold to determine alignment is stable. |
| ANRXDM | 23 | 1b* | Auto-Negotiation Rx Drift Mode.<br>Enables following the drift caused by PPM in the Rx data.<br>0b = Disables drift mode.<br>1b = Enables drift mode. |
| ANRXLM | 24 | 1b* | Auto-Negotiation Rx Loose Mode.<br>Enables less restricted functionality (allow 9/11 bit symbols).<br>0b = Disables loose mode.<br>1b = Enables loose mode. |

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| ANPDT | 26:25 | 00b* | Auto-Negotiation Parallel Detect Timer.<br>Configures the parallel detect counters.<br>00b = 1 ms<br>01b = 2 ms<br>10b = 5 ms<br>11b = 8 ms |
| RF | 27 | 0b* | This bit is loaded to the RF of the auto-negotiation word. |
| PB | 29:28 | 00b* | Pause Bits.<br>The value of these bits is loaded to bits D11–D10 of the Link code word (pause data).<br>Bit 29 is loaded to D11. |
| KX_support | 31:30 | 11b* | Configures the A0:A1 bits of the *Technology Ability Field* of the backplane auto-negotiation word while A2 field is configured in the KR_support bit (bit 16):<br>00b= A0 – 0; A1 – 0. KX not supported. KX4 not supported. Value is Illegal if KR is also not supported (AUTOC.KR_support – 0b).<br>01b = A0 – 1; A1 – 0. KX supported. KX4 not supported.<br>10b = A0 – 0; A1 – 1. KX not supported. KX4 supported.<br>11b = A0 – 1; A1 – 1. KX supported. KX4 supported. |

* Also programmable via EEPROM.

## 8.2.3.22.20 Link Status Register — LINKS (0x042A4; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| KX_SIG_DET | 0 | 0b | Signal Detect of 1 GbE and 100 Mb/s.<br>0b = A signal is not present (Fail).<br>1b = A signal is present (OK). |
| FEC_SIG_DET | 1 | 0b | Signal detect of FEC<br>0b = FEC reports signal not detected (failed).<br>1b = FEC reports signal detected (good). |
| FEC_BLOCK_LOCK | 2 | 0b | 10 GbE serial PCS FEC block lock.<br>0b = No FEC block lock.<br>1b = FEC reached block lock. |
| KR_HI_BERR | 3 | 0b | 10GbE serial KR_PCS high error rate (greater than $10^{-4}$).<br>0b = Low BERR.<br>1b = High BERR. |
| KR_PCS_BLOCK_LOCK | 4 | 0b | 10 GbE serial PCS block lock.<br>0b = No KR_PCS block lock.<br>1b = KR_PCS reached block lock. |
| KX/KX4/KR Backplane AN Next Page received | 5 | 0b | KX/KX4/KR AN Next Page Received.<br>A new link partner next page was received during the backplane auto-negotiation process.<br>Latch high, clear on read. |

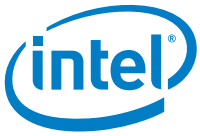


| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| KX/KX4/KR Backplane AN Page received | 6 | 0b | KX/KX4/KR Backplane Auto Negotiation Page Received.<br>A new link partner page was received during the auto-negotiation process.<br>Latch high, clear on read. |
| Link Status | 7 | 0b | 0b = Link is currently down or link was down since last time read.<br>1b = Link is Up and there was no link down from last time read.<br>Self cleared upon read if the link is low and set if the link is up. |
| KX4_SIG_DET | 11:8 | | Signal Detect of 10 GbE Parallel (KX4, CX4 or XAUI).<br>Bit[11, 10, 9, 8] shows lane <3,2,1,0> status, respectively.<br>For each bit:<br>0b = A signal is not present (failed).<br>1b = A signal is present (good). |
| KR_SIG_DET | 12 | | Signal Detect of 10 GbE serial (KR or SFI).<br>0b = Signal not detected (failed).<br>1b = Signal detected (good). |
| 10G lane sync_status | 16:13 | | 10G Parallel lane sync status.<br>bit[16,15,14,13] show lane <3,2,1,0> status accordingly.<br>per each bit:<br>0b = sync_status is FAILED (not synchronized to code-group).<br>1b = sync_status is OK (synchronized to code-group). |
| 10G Align Status | 17 | | 10 GbE align_status.<br>0b = Align_status failed (deskew process not complete).<br>1b = Align_status good (all lanes are synchronized and aligned). |
| 1G Sync Status | 18 | | 1G sync_status.<br>0b = Sync_status failed (not synchronized to code-group).<br>1b = Sync_status is good (synchronized to code-group). |
| KX/KX4/KR Backplane AN Receiver Idle | 19 | | KX/KX4/KR Backplane Auto Negotiation Rx Idle.<br>0b = Receiver good.<br>1b = Receiver is in idle-waiting to align and sync on DME. |
| 1G AN enabled (clause 37 AN) | 20 | | PCS_1 GbE auto-negotiation is enabled (clause 37). |
| 1G link Enabled PCS_1G | 21 | | 1 GbE PCS enabled for 1 GbE and SGMII operation. |
| 10G link Enabled (XGXS) | 22 | | XGXS Enabled for 10 GbE operation. |
| FEC_EN | 23 | | Status of forward–error–correction in 10 GbE serial link (KR operating mode).<br>0b = FEC disabled.<br>1b = FEC enabled. |
| 10G_SER_EN | 24 | | Status of 10 GbE serial PCS (KR PCS) for KR or SFI operation.<br>0b = KR PCS disabled.<br>1b = KR PCS enabled. |
| SGMII_EN | 25 | | Status of SGMII operation.<br>0b = SGMII disabled.<br>1b = SGMII enabled. |

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MLINK_MODE | 27:26 | | MAC link mode status.<br>00b = 1 GbE<br>01b = 10 GbE parallel<br>10b = 10 GbE serial<br>11b = Auto-negotiation |
| LINK_SPEED | 29:28 | | MAC link speed status.<br>00b = Reserved<br>01b = 100 Mb/s<br>10b = 1 GbE<br>11b = 10 GbE |
| Link Up | 30 | 0b | link up<br>0b = Link is down.<br>1b = Link is up. |
| KX/KX4/KR Backplane AN Completed | 31 | 0b | Indicates KX/KX4/KR backplane auto-negotiation has completed successfully. |

## 8.2.3.22.21 Link Status Register 2 — LINKS2 (0x04324; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MAC Rx Link Mode | 1:0 | 00b | MAC link mode in the Core Rx path.<br>00b = 1 GbE<br>01b = 10 GbE parallel<br>10b = 10GbE serial<br>11b = Auto-negotiation |
| Reserved | 2 | 0b | Reserved. |
| MAC Tx Link Mode | 4:3 | 00b | MAC link mode in the Core Tx path.<br>00b = 1 GbE<br>01b = 10 GbE parallel<br>10b = 10GbE serial<br>11b = Auto-negotiation |
| Reserved | 5 | 0b | Reserved. |
| Link-Partner AN | 6 | 0b | Link Partner KX/KX4/KR Backplane Auto-Negotiation Ability.<br>0b = Link partner is not KX/KX4/KR backplane auto-negotiation capable.<br>1b = Link partner is KX/KX4/KR backplane auto-negotiation capable. |
| Reserved | 31:7 | 0x0 | Reserved. |



### 8.2.3.22.22 Auto Negotiation Control 2 Register — AUTOC2 (0x042A8; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 15:0 | 0b | Reserved. |
| 10G_PMA_PMD_Serial | 17:16 | 00b* | PMAPMD used for 10 GbE serial link operation:<br>00b = KR<br>01b = Reserved<br>10b = SFI<br>11b = Reserved |
| DDPT | 18 | 0* | Disable DME Pages Transmit.<br>Setting this bit disables the DME pages transmitting while the device in auto–negotiation mode (it transmits 0bs instead). |
| Reserved | 27:19 | 0b | Reserved |
| FASM | 28 | 0b | Force the auto-negotiation arbitration state machine to idle<br>0b = No Force (normal operation).<br>1b = Force state and keep constant forced state |
| Reserved | 29 | 0b | Reserved. |
| PDD | 30 | 0b* | Disable the parallel detect part in the KX/KX4/KR backplane auto-negotiation.<br>When set to 1b the auto-negotiation process avoids any parallel detect activity, and relies only on the DME pages received and transmitted.<br>0b = Enable the parallel detect (normal operation).<br>1b = Disable the parallel detect (debug only). |
| Reserved | 31 | 0b | Reserved. |

* Loaded from the AUTOC2 word in the MAC EEPROM section

### 8.2.3.22.23 Auto Negotiation Link Partner Link Control Word 1 Register — ANLP1 (0x042B0; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LP AN page D low | 15:0 | 0x0 | LP auto-negotiation advertisement page fields D[15:0].<br>[15] = NP<br>[14] = Acknowledge<br>[13] = RF<br>[12] = Reserved<br>[11:10] = Pause<br>[9:5] = Echoed Nonce field<br>[4:0] = Selector field |
| ANAS | 19:16 | 0x0 | Auto-Negotiation state machine status. If zero, it indicates that state machine is in idle state. |
| Reserved | 31:20 | 0x0 | Reserved. |

To ensure that software has the ability to read the same Link Partner Link Control Word (located across two registers), once ANLP1 is read, ANLP2 is locked until the ANLP2 register is read. ANLP2 does not hold valid data before ANLP1 is read.

### 8.2.3.22.24 Auto Negotiation Link Partner Link Control Word 2 Register — ANLP2 (0x042B4; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LP Transmitted Nonce Field | 4:0 | 0x0 | LP auto-negotiation advertisement page fields T[4:0]. |
| LP Technology Ability Field Low | 15:5 | 0x0 | LP auto-negotiation advertisement page fields A[10:0]. |
| LP Technology Ability Field High | 31:16 | 0x0 | LP auto-negotiation advertisement page fields A[26:11]. |

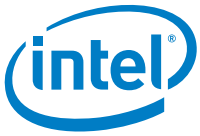
To ensure that software has the ability to read the same Link Partner Link Control Word (located across two registers), once ANLP1 is read, ANLP2 is locked until the ANLP2 register is read. ANLP2 does not hold valid data before ANLP1 is read.

### 8.2.3.22.25 MAC Manageability Control Register — MMNGC (0x042D0; Host-RO/MNG-RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MNG_VETO | 0 | 0b | MNG_VETO (default 0b).<br>Access read/write by manageability, read only to the host.<br>0b = No specific constraints on link from manageability.<br>1b = Hold off any low-power link mode changes. This is done to avoid link loss and interrupting manageability activity. |
| Reserved | 31:1 | 0x0 | Reserved. |

### 8.2.3.22.26 Auto Negotiation Link Partner Next Page 1 Register — ANLPNP1 (0x042D4; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LP AN Next Page Low | 31:0 | 0x0 | LP Auto-Negotiation Next Page Fields D[31:0].<br>[31:16] = Unformatted Code<br>[15] = NP<br>[14] = Acknowledge<br>[13] = MP<br>[12] = Acknowledge2<br>[11] = Toggle<br>[10:0] = Message/Unformatted Code |



To ensure that software has the ability to read the same Link Partner Link Control Word (located across two registers), once ANLP1 is read, ANLP2 is locked until the ANLP2 register is read. ANLP2 does not hold valid data before ANLP1 is read.

### 8.2.3.22.27 Auto Negotiation Link Partner Next Page 2 Register — ANLPNP2 (0x042D8; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LP AN Next Page high | 15:0 | 0x0 | LP AN Next Page Fields D[47:32]. [15:0] = Unformatted Code. |
| Reserved | 31:16 | 0x0 | Reserved. |

To ensure that software has the ability to read the same Link Partner Link Control Word (located across two registers), once ANLPNP1 is read, ANLPNP2 is locked until the ANLPNP2 register is read. ANLPNP2 does not hold valid data before ANLPNP1 is read.

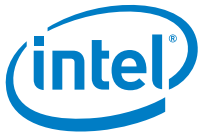
### 8.2.3.22.28 KR PCS and FEC Control Register — KRPCSFC (0x042EO; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 10:0 | 0x0 | Reserved.<br>Bits should be written as 0x0 and ignored on read. |
| Reserved | 15:11 | 0x0 | Reserved. |
| FEC_ENABLE_ERR | 16 | 1b | FEC Enable Error Indication to KR-PCS.<br>0b = Disabled.<br>1b = The FEC decoder indicates error to the KR-PCS by means of setting both sync bits to the value 11 in the 1st, 9th, 17th, 25th, and 32nd of the 32 decoded 64b/66b blocks from the corresponding erred FEC block. |
| Reserved | 17 | 0b | Reserved. |
| FEC_N_CNT | 19:18 | 00b | Good Parity Block Count.<br>Indicates the number of good parity blocks required for block lock.<br>00b = 4 good blocks<br>01b = 2 good blocks<br>10b = 5 good blocks<br>11b = 7 good blocks |
| FEC_M_CNT | 21:20 | 00b | Bad Parity Block Count.<br>Indicates the number of bad parity blocks required for loss of block lock.<br>00b = 8 errors<br>01b = 4 errors<br>10b = 12 errors<br>11b =-15 errors |
| FEC_LOOSE_MODE | 22 | 0b | Enables FEC Loose Mode<br>0b = All errors counted.<br>1b = Correctable errors are not counted. |

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| FEC_RX_SWAP | 23 | 0b | FEC Rx Bit Order Swap.<br>Swaps the bit order of the FEC_RX inputs. Swaps both *din* and sync_in bit order. |
| FEC_TX_SWAP | 24 | 0b | FEC Tx Bit Order Swap.<br>Swaps the bit order of the FEC_TX inputs. Swaps both *din* and *sync_in* bit order. |
| Reserved | 25 | 0b | Reserved. |
| SLIPASS | 26 | 0b | Loss of Sync (frame_align) Idle Pass-Through Select.<br>0b = LF passed to XGMII output when loss of sync.<br>1b = Decoder output data passed to XGMII output when loss of sync. |
| SSYNC | 27 | 0b | Rx Block Lock Override.<br>Once block lock sync has been acquired on the Rx input, Rx input remains in block lock sync regardless of Rx input data. |
| Reserved | 28 | 0b | Reserved. |
| Reserved | 31:29 | 0x0 | Reserved. |

## 8.2.3.22.29 KR PCS Status Register — KRPCSS (0x042E4; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 2:0 | 0x0 | Reserved.<br>Write 0 ignore read. |
| ERRCNT_BLK | 10:3 | 0x0 | Rx Decoder Error Counter.<br>This counter does not rollover and holds its value until it is read if it reaches its maximum value. This count is cleared when this register is read. |
| BERBAD_CNTR | 16:11 | 0x0 | BER Bad Counter (count cleared on register read)<br>Field indicates number of times BER_BAD state was entered. |
| RX_FIFO_ERR_LH | 17 | 0b | Elastic Buffer Error (latched high, clear on read).<br>0b = No Rx elastic buffer overflow or underflow condition since last read.<br>1b = Indicates that Rx elastic buffer overflow or underflow condition occurred since last read. |
| RX_LF_DET | 18 | 0b | RX_LF Detect (latched high, clear on read).<br>0b = No local fault message was detected in the Rx path since last read.<br>1b = Indicates that the local fault message was detected in the Rx path since last read. |
| RX_FRM_ALIGN_ERR | 19 | 0b | Frame Align Error (latched high, clear on read).<br>0b = No hi_ber or miss of frame_lock occurred since last read.<br>1b = Indicates that the hi_ber or miss of frame_lock occurred since last time the register was read. |
| BLKLCK | 20 | 0b | Rx Block Lock Status bit (latched low, set on read).<br>0b = Link lost block lock since last register read.<br>1b = Indicates that the link has remained in the block lock state since the last read of this register. |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| HBER_STS | 21 | 0b | Rx High Bit Error Rate Status bit (latched high, clear on read).<br>0b = Link has not been in high BER state since previous read.<br>1b = Indicates that the link has been in the high BER state since the last time the register was read. |
| RX_LF_DET | 22 | 0b | RX_LF Detect (latched high, clear on read).<br>0b = No local fault message was detected in the Rx path.<br>1b = Indicates that the local fault message was detected in the Rx path. |
| LNK_STS | 23 | 0b | Rx Link Status (latched low, set on read).<br>0b = Indicates that the link has been lost since the last time the bit was read.<br>1b = No loss of link since last time bit was read. |
| RX_UNDERFLOW | 24 | 0b | Rx Underflow Status (latched high, clear on read).<br>0b = No underflow condition in rx_fifo.<br>1b = Indicates that the rx_fifo has reached the underflow condition and data might have been lost/corrupted. |
| RX_OVERFLOW | 25 | 0b | Rx Overflow Status (latched high, clear on read).<br>0b = No overflow condition in rx_fifo.<br>1b = Indicates that the rx_fifo has reached the overflow condition and data might have been lost or corrupted. |
| RX_FIFO_ERR | 26 | 0b | Rx Elastic Buffer Error<br>0b = No elastic buffer error.<br>1b = Indicates that Rx elastic buffer is currently in the overflow or underflow condition. This bit is not latched and is asserted only when the FIFO is in the overflow or underflow condition. |
| RX_DATA_VALID | 27 | 1b | Data Valid Status (latched low, set on read).<br>This bit indicates that the rx_fifo has not experienced an overflow or underflow since the last time this register was read. |
| TX_UNDERFLOW | 28 | 0b | Tx Underflow Status (latched high, clear on read).<br>This bit indicates that the tx_fifo has reached the underflow condition and data might have been lost/corrupted. |
| TX_OVERFLOW | 29 | 0b | Tx Overflow Status (latched high, clear on read).<br>This bit indicates that the tx_fifo has reached the overflow condition and data might have been lost/corrupted. |
| TX_FIFO_ERR | 30 | 0b | Unlatched FIFO Error Status.<br>This bit indicates that the tx_fifo has reached the overflow or underflow condition and data might have been lost / corrupted. This bit is not latched and is only asserted while the FIFO is in the overflow or underflow condition. |
| TX_DATA_VALID | 31 | 1b | Data Valid Status (latched low, set on read).<br>0b = tx_fifo has experienced an overflow or underflow since the last time this register was read.<br>1b = tx_fifo has not experienced an overflow or underflow since the last time this register was read. |

### 8.2.3.22.30 FEC Status 1 Register — FECS1 (0x042E8; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| FEC_CR_OUT | 31:0 | 0x0 | FEC Correctable Error Counter.<br>The FECS1 counts the correctable FEC data blocks, detected and corrected by the FEC Rx logic. |

### 8.2.3.22.31 FEC Status 2 Register — FECS2 (0x042EC; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| FEC_UNCR_OUT | 31:0 | 0x0 | FEC Uncorrectable Error Counter.<br>The FECS2 counts the bad uncorrectable FEC data blocks, detected by the FEC Rx logic. |

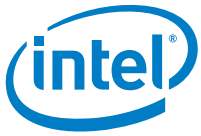
### 8.2.3.22.32 Core Analog Configuration Register — CoreCTL (0x014F00; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Data | 7:0 | 0x0 | Data to Core Analog Registers.<br>Data is ignored when bit 16 is set. |
| Address | 15:8 | 0x0 | Address to Core Analog Registers. |
| Latch address | 16 | 0b | 0b = Normal write operation.<br>1b = Latch this address for the next read transaction. Data is ignored and is not written on this transaction. |
| Reserved | 31:17 | 0x0 | Reserved. |

Reading the Core registers must be done in two steps:

1. Send a Write command with bit 16 set, and the desired reading offset in the *Address* field (bits [15:8]).
2. Send a Read command to CoreCTL. The returned data is from the indirect address in the Core register space, which was provided in step (1).

To configure (write) registers in the Core block, the driver should write the proper address to CoreCTL.Address and data written to CoreCTL.Data.



### 8.2.3.22.33 Core Common Configuration Register — SMADARCTL (0x014F10; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Data | 7:0 | 0b | Data to Core Analog Registers.<br>Data is ignored when bit 16 is set. |
| Address | 15:8 | 0x0 | Address to Core Analog Registers. |
| Latch address | 16 | 0b | 0b = Normal write operation.<br>1b = Latch this address for the next read transaction. Data is ignored and is not written on this transaction. |
| Reserved | 31:17 | 0x0 | Reserved. |

Reading the Smadar registers must be done in two steps:

1. Send a Write command with bit 16 set, and the desired reading offset in the *Address* field (bits [15:8]).
2. Send a Read command to SMADARCTL. The returned data is from the indirect address in the Core register space, which was provided in step (1).

To configure (write) registers in the Smadar block, the driver should write the proper address to SMADARCTL.Address and data written to SMADARCTL.Data.

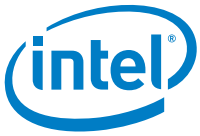
### 8.2.3.22.34 MAC Flow Control Register — MFLCN (0x04294; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PMCF | 0 | 0b | Pass MAC Control Frames.<br>Filter out unrecognizable pause (flow control opcode doesn't match) and other control frames.<br>0b = Filter unrecognizable pause frames.<br>1b = Pass/forward unrecognizable pause frames. |
| DPF | 1 | 0b | Discard Pause Frame<br>When set to 0b, pause frames are sent to the host. Setting this bit to 1b causes pause frames to be discarded only when *RFCE* or *RPFCE* are set to 1b. If both *RFCE* and *RPFCE* are set to 0b, this bit has no effect on incoming pause frames. |
| RPFCE | 2 | 0b | Receive Priority Flow Control Enable**.**<br>Indicates that the 82599 responds to receiving PFC packets. If auto-negotiation is enabled, this bit should be set by software to the negotiated flow control value.<br>*Note:* PFC should be enabled in DCB mode only.<br>*Note:* Receive PFC and receive link flow control are mutually exclusive and programmers should not configure both of them to be enabled at the same time.<br>*Note:* This bit should not be set if bit 3 is set. |

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RFCE | 3 | 0b | Receive Flow Control Enable<br>Indicates that the 82599 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.<br>*Note:* This bit should not be set if bit 2 is set. |
| Reserved | 31:4 | 0x0 | Reserved. |

## 8.2.3.22.35 SGMII Control Register — SGMIIC (0x04314; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SRXRASSMP | 3:0 | 0x0 | Shift Rx Rate-Adapt Single Data Sampling.<br>This value determines the sampling point of the sampled received data in the fast domain by the fast clock relative to the slow clock. |
| SRXRARSMP | 7:4 | 0x0 | Shift Rx Rate-Adapt Replicated Data Sampling.<br>This value determines the sampling point of the received replicated data in the fast domain by the fast clock. |
| STXRASMP | 11:8 | 0x0 | Shift Tx Rate-Adapt Sampling.<br>This value determines the sampling point of the transmitted data in the slow domain by the fast clock. |
| ANSFLU100 | 12 | 0b | AN SGMII Force Link Up 100 Mb/s.<br>0b = Normal mode.<br>1b = PGS_1G forced to 100 Mb/s link. This setting forces the PCS_1G Link_Ok indication and forced the 100 Mb/s speed indication toward the auto-negotiation ARB state machine regardless of the PCS_1G status. |
| ANSBYP | 13 | 0b | AN SGMII Bypass. I<br>f this bit is set, the IDLE detect state is bypassed during auto-negotiation (Clause 37) in SGMII mode. This reduces the acknowledge time in SGMII mode. |
| ANSTRIG | 14 | 0b | AN SGMII Trigger.<br>If this bit is set, auto-negotiation (Clause 37) is not automatically triggered in SGMII mode even if SYNC fails. Auto-negotiation is triggered only in response to PHY messages or by a manual setting like changing auto-negotiation *Enable*/*Restart* bits. |
| ANSLNKTMR | 15 | 0b | AN SGMII Link-Timer (Configure the SGMII Link Timer).<br>0b = 1.6 ms<br>1b = 3.2 ms |
| Reserved | 16 | 0b | Reserved. |
| ANIGNRRXRF | 17 | 0b | Auto-Negotiation Ignore Received RF Field.<br>0b = If the received page contains RF != 00b, don't set link_up indication.<br>1b = Ignore from the received RF field content. |
| Reserved | 31:18 | 0x0 | Reserved. |



## 8.2.3.23 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 µs; a small time-delay prior to reading the statistics is required to avoid a potential mismatch between an interrupt and its cause.
- If RSC is enabled, statistics are collected before RSC is applied to the packets.
- If TSO is enabled, statistics are collected after segmentation.
- All byte (octet) counters composed of two registers can be fetched by two consecutive 32-bit accesses while reading the low 32-bit register first or a single 64-bit access.
- All receive statistic counters count the packets and bytes before coalescing by the RSC logic or FCoE DDP logic.
- All receive statistic counters in the filter unit (listed as follows) include also packets that might be dropped by the packet buffer or receive DMA. Same comment is valid for the byte counters associated with these packet counters: PRC64, PRC127, PRC255, PRC511, PRC1023, PRC1522, BPRC, MPRC, GPRC, RXNFGPC, RUC and ROC

### 8.2.3.23.1 CRC Error Count — CRCERRS (0x04000; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| CEC | 31:0 | 0x0 | CRC Error Count.<br>Counts the number of receive packets with CRC errors. In order for a packet to be counted in this register, it must be 64 bytes or greater (from <Destination Address> through <CRC>, inclusively) in length. This register counts all packets received, regardless of L2 filtering and receive enablement. |

### 8.2.3.23.2 Illegal Byte Error Count — ILLERRC (0x04004; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| IBEC | 31:0 | 0x0 | Illegal Byte Error Count.<br>Counts the number of receive packets with illegal bytes errors (such as there is an illegal symbol in the packet). This registers counts all packets received, regardless of L2 filtering and receive enablement. |

### 8.2.3.23.3 Error Byte Packet Count — ERRBC (0x04008; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| EBC | 31:0 | 0x0 | Error Byte Count.<br>Counts the number of receive packets with error bytes (such as there is an error symbol in the packet). This registers counts all packets received, regardless of L2 filtering and receive enablement. |

### 8.2.3.23.4 Rx Missed Packets Count — RXMPC[n] (0x03FA0 + 4*n, n=0...7; RC) DBU-Rx

**Note:** This is a RO register only.

### 8.2.3.23.5 MAC Local Fault Count — MLFC (0x04034; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MLFC | 31:0 | 0x0 | Number of faults in the local MAC.<br>This register is valid only when the link speed is 10 Gb/s. |

### 8.2.3.23.6 MAC Remote Fault Count — MRFC (0x04038; RC)

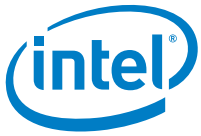
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MRFC | 31:0 | 0x0 | Number of faults in the remote MAC.<br>This register is valid only when the link speed is 10 Gb/s. |

### 8.2.3.23.7 Receive Length Error Count — RLEC (0x04040; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RLEC | 31:0 | 0x0 | Number of packets with receive length errors.<br>A length error occurs if an incoming packet length field in the MAC header doesn't match the packet length. To enable the receive length error count, the HLREG.RXLNGTHERREN bit needs to be set to 1b. This registers counts all packets received, regardless of L2 filtering and receive enablement. |

### 8.2.3.23.8 Switch Security Violation Packet Count — SSVPC (0x08780; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SSVPC | 31:0 | 0x0 | Switch Security Violation Packet Count.<br>This register counts Tx packets dropped 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. |



### 8.2.3.23.9 Link XON Transmitted Count — LXONTXC (0x03F60; RC)

**Note:** This is a RO register only.

### 8.2.3.23.10 Link XON Received Count — LXONRXCNT (0x041A4; RC)

**Note:** This counter is similar to LXONRXC in the 82598 that was in address 0x0CF60.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| XONRXC | 15:0 | 0 | Number of XON packets received.<br>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 is designated to a few priorities. If a priority FC packet contains both XOFF and XON, only the LXOFFRXCNT counter is incremented. |
| Reserved | 31:16 | 0 | Reserved. |

### 8.2.3.23.11 Link XOFF Transmitted Count — LXOFFTXC (0x03F68; RC)

**Note:** This is a RO register only.

### 8.2.3.23.12 Link XOFF Received Count — LXOFFRXCNT (0x041A8; RC)

**Note:** This counter is similar to LXOFFRXC in the 82598 that was in address 0x0CF68.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| XOFFRXC | 15:0 | 0x0 | Number of XOFF packets received.<br>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 is designated to a few priorities. If a priority FC packet contains both XOFF and XON, only this counter is incremented. |
| Reserved | 31:16 | 0x0 | Reserved. |

### 8.2.3.23.13 Priority XON Transmitted Count — PXONTXC[n] (0x03F00 + 4*n, n=0...7; RC)

**Note:** This is a RO register only.

### 8.2.3.23.14 Priority XON Received Count — PXONRXCNT[n] (0x04140 + 4*n, n=0...7; RC)

**Note:** These counters are similar to PXONRXC[n] in the 82598 that were in address 0x0CF00 + 4*n, n=0...7.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| XONRXC | 15:0 | 0x0 | Number of XON packets received per UP.<br>Sticks to 0xFFFF. |
| Reserved | 31:16 | 0x0 | Reserved. |

### 8.2.3.23.15 Priority XOFF Transmitted Count — PXOFFTXCNT[n] (0x03F20 + 4*n, n=0...7; RC)

**Note:** This is a RO register only.

### 8.2.3.23.16 Priority XOFF Received Count — PXOFFRXCNT[n] (0x04160 + 4*n, n=0...7; RC)

**Note:** These counters are similar to PXOFFRXC[n] in the 82598 that were in address 0x0CF20 + 4*n, n=0...7.

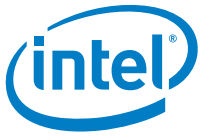
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| XOFFRXC | 15:0 | 0x0 | Number of XOFF packets received per UP.<br>Sticks to 0xFFFF. |
| Reserved | 31:16 | 0x0 | Reserved. |

### 8.2.3.23.17 Priority XON to XOFF Count — PXON2OFFCNT[n] (0x03240 + 4*n, n=0...7; RC)

**Note:** This is a RO register only.

### 8.2.3.23.18 Packets Received [64 Bytes] Count — PRC64 (0x0405C; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PRC64 | 31:0 | 0x0 | Number of good packets received that are 64 bytes in length (from <Destination Address> through <CRC>, inclusively).<br>This registers counts packets that pass L2 filtering regardless on receive enablement and does not include received flow control packets. |



### 8.2.3.23.19 Packets Received [65–127 Bytes] Count — PRC127 (0x04060; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PRC127 | 31:0 | 0x0 | Number of packets received that are 65-127 bytes in length (from <Destination Address> through <CRC>, inclusively).<br>This registers counts packets that pass L2 filtering regardless on receive enablement and does not include received flow control packets. |

### 8.2.3.23.20 Packets Received [128–255 Bytes] Count — PRC255 (0x04064; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PRC255 | 31:0 | 0x0 | Number of packets received that are 128-255 bytes in length (from <Destination Address> through <CRC>, inclusively).<br>This registers counts packets that pass L2 filtering regardless on receive enablement and does not include received flow control packets. |

### 8.2.3.23.21 Packets Received [256–511 Bytes] Count — PRC511 (0x04068; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PRC511 | 31:0 | 0x0 | Number of packets received that are 256-511 bytes in length (from <Destination Address> through <CRC>, inclusively).<br>This registers counts packets that pass L2 filtering regardless on receive enablement and does not include received flow control packets. |

### 8.2.3.23.22 Packets Received [512–1023 Bytes] Count — PRC1023 (0x0406C; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PRC1023 | 31:0 | 0x0 | Number of packets received that are 512-1023 bytes in length (from <Destination Address> through <CRC>, inclusively).<br>This registers counts packets that pass L2 filtering regardless on receive enablement and does not include received flow control packets. |

### 8.2.3.23.23 Packets Received [1024 to Max Bytes] Count — PRC1522 (0x04070; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PRC1522 | 31:0 | 0x0 | Number of packets received that are 1024-max bytes in length (from <Destination Address> through <CRC>, inclusively).<br>This registers counts packets that pass L2 filtering regardless on receive enablement and does not include received flow control packets.<br>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 8.2.3.23.52). Due to changes in the standard for maximum frame size for VLAN tagged frames in 802.3, this device accepts packets that have a maximum length of 1522 bytes. The RMON statistics associated with this range has been extended to count 1522 byte long packets. |

### 8.2.3.23.24 Broadcast Packets Received Count — BPRC (0x04078; RO)

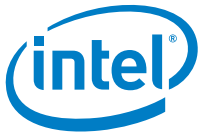
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| BPRC | 31:0 | 0x0 | Number of good (non-erred) broadcast packets received.<br>This register does not count received broadcast packets when the broadcast address filter is disabled. The counter counts packets regardless on receive enablement. |

### 8.2.3.23.25 Multicast Packets Received Count — MPRC (0x0407C; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MPRC | 31:0 | 0x0 | Number of good (non-erred) multicast packets received that pass L2 filtering (excluding broadcast packets).<br>This register does not count received flow control packets. This registers counts packets regardless on receive enablement. |

### 8.2.3.23.26 Good Packets Received Count — GPRC (0x04074; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GPRC | 31:0 | 0x0 | Number of good (non-erred) Rx packets (from the network) that pass L2 filtering and has a legal length as defined by *LongPacketEnable*.<br>This registers counts packets regardless on receive enablement. |



### 8.2.3.23.27 Good Octets Received Count Low — GORCL (0x04088; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| CNT_L | 31:0 | 0x0 | Lower 32 bits of the good octets received counter.<br>The GORCL and GORCH registers make up a logical 36-bit octet counter of the packets counted by GPRC. This register includes bytes received in a packet from the <Destination Address> field through the <CRC> field, inclusively. |

### 8.2.3.23.28 Good Octets Received Count High — GORCH (0x0408C; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| CNT_H | 3:0 | 0x0 | Higher four bits of the good octets received counter. |
| Reserved | 31:4 | 0x0 | Reserved |

### 8.2.3.23.29 Good Rx Non-Filtered Packet Counter — RXNFGPC (0x041B0; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GPC | 31:0 | 0x0 | Number of good (non-erred with legal length) Rx packets (from the network) regardless of packet filtering and receive enablement. |

### 8.2.3.23.30 Good Rx Non-Filter Byte Counter Low — RXNFGBCL (0x041B4; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| BCL | 31:0 | 0x0 | Low 32 bits of the 36-bit byte counter of good (non-erred) Rx packets that match RXNFGPC.<br>The counter counts all bytes from <Destination Address> field through the <CRC> field, inclusively. |

### 8.2.3.23.31 Good Rx Non-Filter Byte Counter High — RXNFGBCH (0x041B8; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| BCH | 3:0 | 0x0 | Higher four bits of the 36-bit byte counter associated with RXFGBCL. |
| Reserved | 31:4 | 0x0 | Reserved |

### 8.2.3.23.32 DMA Good Rx Packet Counter — RXDGPC (0x02F50; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GPC | 31:0 | 0x0 | Number of good (non-erred) Rx packets from the network posted to the host memory.<br>In case of packet replication (or mirrored), the counter counts each packet only once. The counter counts packets directed to ALL Rx queues or specific Rx queues as defined by the RXDSTATCTRL register. |

### 8.2.3.23.33 DMA Good Rx Byte Counter Low — RXDGBCL (0x02F54; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GBCL | 31:0 | 0x0 | Lower 32 bits of the 36-bit byte counter of good (non-erred) Rx packets that match RXDGPC.<br>The counter counts all bytes posted to the host before VLAN strip. Furthermore, bytes of RSC and FCoE are counted before coalescing or DDP. |

### 8.2.3.23.34 DMA Good Rx Byte Counter High — RXDGBCH (0x02F58; RC)

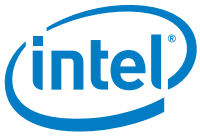
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GBCH | 3:0 | 0x0 | Higher four bits of the 36-bit byte counter associated with RXDGBCL. |
| Reserved | 31:4 | 0x0 | Reserved. |

### 8.2.3.23.35 DMA Duplicated Good Rx Packet Counter — RXDDPC (0x02F5C; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GPC | 31:0 | 0x0 | Number of replicated or mirrored packets that meet the RXDGPC conditions.<br>The sum of RXDDPC and RXDGPC is the total good (non-erred) Rx packets from the network that are posted to the host.<br>*Note:* The counter counts packets directed to ALL Rx queues or specific Rx queues as defined by the RXDSTATCTRL register. |

### 8.2.3.23.36 DMA Duplicated Good Rx Byte Counter Low — RXDDBCL (0x02F60; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GBCL | 31:0 | 0x0 | Lower 32 bits of the 36-bit byte counter of good (non-erred) Rx packets that match RXDDPC.<br>The counter counts all bytes posted to the host before VLAN strip. Furthermore, bytes of RSC and FCoE are counted before coalescing or DDP. |



### 8.2.3.23.37 DMA Duplicated Good Rx Byte Counter High — RXDDBCH (0x02F64; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GBCH | 3:0 | 0x0 | Higher four bits of the 36-bit byte counter associated with RXDDBCL. |
| Reserved | 31:4 | 0x0 | Reserved. |

### 8.2.3.23.38 DMA Good Rx LPBK Packet Counter — RXLPBKPC (0x02F68; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GPC | 31:0 | 0x0 | Number of good (non-erred) Rx packets from a local VM posted to the host memory.<br>In case of packet replication (or mirrored), the counter counts each packet only once. The counter counts packets directed to ALL Rx queues or specific Rx queues as defined by the RXDSTATCTRL register. The counter is not affected by RSC and FCoE DDP since both functions are not supported for LPBK traffic. |

### 8.2.3.23.39 DMA Good Rx LPBK Byte Counter Low — RXLPBKBCL (0x02F6C; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GBCL | 31:0 | 0x0 | Lower 32 bits of the 36-bit byte counter of good (non-erred) Rx packets that match RXLPBKPC.<br>The counter counts all bytes posted to the host before VLAN strip. Furthermore, bytes of RSC and FCoE are counted before coalescing or DDP. |

### 8.2.3.23.40 DMA Good Rx LPBK Byte Counter High — RXLPBKBCH (0x02F70; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GBCH | 3:0 | 0x0 | Higher four bits of the 36-bit byte counter associated with RXLPBKBCL. |
| Reserved | 31:4 | 0x0 | Reserved. |

### 8.2.3.23.41 DMA Duplicated Good Rx LPBK Packet Counter — RXDLPBKPC (0x02F74; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GPC | 31:0 | 0x0 | Number of replicated or mirrored packets that meet the RXLPBKPC conditions.<br>The sum of RXDLPBKPC and RXLPBKPC is the total good (non-erred) Rx packets from a local VM posted to the host.<br>*Note:* The counter counts packets directed to ALL Rx queues or specific Rx queues as defined by the RXDSTATCTRL register. |

### 8.2.3.23.42 DMA Duplicated Good Rx LPBK Byte Counter Low — RXDLPBKBCL (0x02F78; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GBCL | 31:0 | 0x0 | Low 32 bits of the 36-bit byte counter of good (non-erred) Rx packets that match RXDLPBKPC.<br>The counter counts all bytes posted to the host before VLAN strip. Furthermore, bytes of RSC and FCoE are counted before coalescing or DDP. |

### 8.2.3.23.43 DMA Duplicated Good Rx LPBK Byte Counter High — RXDLPBKBCH (0x02F7C; RC)

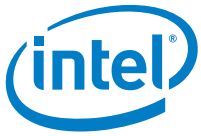
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GBCH | 3:0 | 0x0 | Higher four bits of the 36-bit byte counter associated with RXDLPBKBCL. |
| Reserved | 31:4 | 0x0 | Reserved. |

### 8.2.3.23.44 Good Packets Transmitted Count — GPTC (0x04080; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GPTC | 31:0 | 0x0 | Number of good packets transmitted.<br>This register counts good (non-erred) transmitted packets. A good transmit packet is considered one that is 64 or more bytes (from <Destination Address> through <CRC>, inclusively) in length. The register counts transmitted clear packets, secure packets and FC packets. |

### 8.2.3.23.45 Good Octets Transmitted Count Low — GOTCL (0x04090; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| CNT_L | 31:0 | 0x0 | Lower 32 bits of the good octets transmitted counter.<br>See complete description in Section 8.2.3.23.46. |



### 8.2.3.23.46 Good Octets Transmitted Count High — GOTCH (0x04094; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| CNT_H | 3:0 | 0x0 | Higher four bits of the good octets transmitted counter.<br>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 | Reserved. |

### 8.2.3.23.47 DMA Good Tx Packet Counter – TXDGPC (0x087A0; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GPTC | 31:0 | 0x0 | Number of Tx packets from the host memory.<br>This counter includes packets that are transmitted to the external network as well as packets that are transmitted only to local VMs. The later case can happen only in VT mode when the local switch is enabled. Packets dropped due to anti-spoofing filtering or VLAN tag validation (as described in Section 7.10.3.9.2) are not counted. |

### 8.2.3.23.48 DMA Good Tx Byte Counter Low – TXDGBCL (0x087A4; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| BCL | 31:0 | 0x0 | Lower 32 bits of the 36-bit byte counter of the Tx packets that match TXDGPC.<br>The counter counts all bytes posted by the host AND the VLAN (if bytes were added by hardware). |

### 8.2.3.23.49 DMA Good Tx Byte Counter High – TXDGBCH (0x087A8; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| BCH | 3:0 | 0x0 | Higher four bits of the 36-bit byte counter associated to TXDGBCL. |
| Reserved | 31:4 | 0x0 | Reserved. |

### 8.2.3.23.50 Receive Undersize Count — RUC (0x040A4; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RUC | 31:0 | 0x0 | Receive Undersize Error.<br>This register counts the number of received frames that are shorter than minimum size (64 bytes from <Destination Address> through <CRC>, inclusively), and had a valid CRC. This register counts packets regardless of L2 filtering and receive enablement. |

### 8.2.3.23.51 Receive Fragment Count — RFC (0x040A8; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RFC | 31:0 | 0x0 | 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).<br>This register counts packets regardless of L2 filtering and receive enablement. |

### 8.2.3.23.52 Receive Oversize Count — ROC (0x040AC; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| ROC | 31:0 | 0x0 | Receive Oversize Error. This register counts the number of received frames that are longer than maximum size as defined by MAXFRS.MFS (from <Destination Address> through <CRC>, inclusively) and have valid CRC. This register counts packets regardless of L2 filtering and receive enablement. |

### 8.2.3.23.53 Receive Jabber Count — RJC (0x040B0; RC)

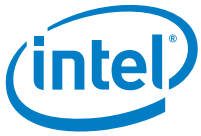
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RJC | 31:0 | 0x0 | Number of receive jabber errors.<br>This register counts the number of received packets that are greater than maximum size and have bad CRC (this is slightly different from the Receive Oversize Count register). The packets length is counted from <Destination Address> through <CRC>, inclusively. This register counts packets regardless of L2 filtering and receive enablement. |

### 8.2.3.23.54 Management Packets Received Count — MNGPRC (0x040B4; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MNGPRC | 31:0 | 0x0 | Number of management packets received.<br>This register counts the total number of packets received that pass the management filters. Management packets include RMCP and ARP packets. Any packets with errors are not counted, except for the packets that are dropped because the management receive FIFO is full are counted. |

### 8.2.3.23.55 Management Packets Dropped Count — MNGPDC (0x040B8; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MPDC | 31:0 | 0 | Number of management packets dropped.<br>This register counts the total number of packets received that pass the management filters and then are dropped because the management receive FIFO is full. Management packets include any packet directed to the manageability console (such as RMCP and ARP packets). |



### 8.2.3.23.56 Management Packets Transmitted Count — MNGPTC (0x0CF90; RO)

**Note:** This is a RO register only.

### 8.2.3.23.57 Total Octets Received Low — TORL (0x040C0; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| CNT_L | 31:0 | 0x0 | Lower 32 bits of the total octets received counter.<br>See complete description in Section 8.2.3.23.58. |

### 8.2.3.23.58 Total Octets Received High — TORH (0x040C4; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| CNT_H | 3:0 | 0x0 | Higher four bits of the total octets received counter.<br>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 | Reserved. |

### 8.2.3.23.59 Total Packets Received — TPR (0x040D0; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TPR | 31:0 | 0x0 | Number of all packets received.<br>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.<br>TPR can count packets interrupted by link disconnect although they have a CRC error. |

### 8.2.3.23.60 Total Packets Transmitted — TPT (0x040D4; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TPT | 31:0 | 0x0 | Number of all packets transmitted.<br>This register counts the total number of all packets transmitted. This register counts all packets, including standard packets, secure packets, FC packets, and manageability packets. |

### 8.2.3.23.61 Packets Transmitted (64 Bytes) Count — PTC64 (0x040D8; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PTC64 | 31:0 | 0x0 | Number of packets transmitted that are 64 bytes in length (from <Destination Address> through <CRC>, inclusively).<br>This register counts all packets, including standard packets, secure packets, FC packets, and manageability packets. |

### 8.2.3.23.62 Packets Transmitted [65–127 Bytes] Count — PTC127 (0x040DC; RC)

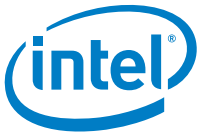
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PTC127 | 31:0 | 0x0 | Number of packets transmitted that are 65-127 bytes in length (from <Destination Address> through <CRC>, inclusively).<br>This register counts all packets, including standard packets, secure packets, and manageability packets. |

### 8.2.3.23.63 Packets Transmitted [128–255 Bytes] Count — PTC255 (0x040E0; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PTC255 | 31:0 | 0x0 | Number of packets transmitted that are 128-255 bytes in length (from <Destination Address> through <CRC>, inclusively).<br>This register counts all packets, including standard packets, secure packets, and manageability packets. |

### 8.2.3.23.64 Packets Transmitted [256–511 Bytes] Count — PTC511 (0x040E4; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PTC511 | 31:0 | 0x0 | Number of packets transmitted that are 256-511 bytes in length (from <Destination Address> through <CRC>, inclusively).<br>This register counts all packets, including standard packets, secure packets, and manageability packets. |



### 8.2.3.23.65 Packets Transmitted [512–1023 Bytes] Count — PTC1023 (0x040E8; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PTC1023 | 31:0 | 0x0 | Number of packets transmitted that are 512-1023 bytes in length (from <Destination Address> through <CRC>, inclusively).<br>This register counts all packets, including standard packets, secure packets, and manageability packets. |

### 8.2.3.23.66 Packets Transmitted [Greater Than 1024 Bytes] Count — PTC1522 (0x040EC; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PTC1522 | 31:0 | 0x0 | Number of packets transmitted that are 1024 or more bytes in length (from <Destination Address> through <CRC>, inclusively).<br>This register counts all packets, including standard packets, secure packets, and manageability packets.<br>Due to changes in the standard for maximum frame size for VLAN tagged frames in 802.3, this device transmits packets that have a maximum length of 1522 bytes. The RMON statistics associated with this range has been extended to count 1522 byte long packets. This register counts all packets, including standard and secure packets. |

### 8.2.3.23.67 Multicast Packets Transmitted Count — MPTC (0x040F0; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MPTC | 31:0 | 0x0 | Number of multicast packets transmitted.<br>This register counts the number of multicast packets transmitted. This register counts all packets, including standard packets, secure packets, FC packets and manageability packets. |

### 8.2.3.23.68 Broadcast Packets Transmitted Count — BPTC (0x040F4; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| BPTC | 31:0 | 0x0 | Number of broadcast packets transmitted count.<br>This register counts all packets, including standard packets, secure packets, FC packets and manageability packets |

### 8.2.3.23.69 MAC Short Packet Discard Count — MSPDC (0x04010; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MSPDC | 31:0 | 0x0 | Number of MAC short packet discard packets received. |

### 8.2.3.23.70 XSUM Error Count — XEC (0x04120; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| XEC | 31:0 | 0x0 | Number of receive IPv4, TCP, UDP or SCTP XSUM errors. |

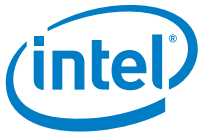
XSUM errors are not counted when a packet has any MAC error (CRC, length, under-size, over-size, byte error or symbol error).

### 8.2.3.23.71 Receive Queue Statistic Mapping Registers — RQSMR[n] (0x02300 + 4*n, n=0...31; RW)

These registers define the mapping of the receive queues to the per queue statistics. Several queues can be mapped to a single statistic register. Each statistic register counts the number of packets and bytes of all the queues that are mapped to that statistics. The registers counting Rx queue statistics are: QPRC, QBRC, and QPRDC.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Q_MAP[0] | 3:0 | 0x0 | For each register 'n', Q_MAP[0] defines the per queue statistic registers that are mapped to Rx queue '4*n+0'. (see examples that follow). |
| Reserved | 7:4 | 0x0 | Reserved. |
| Q_MAP[1] | 11:8 | 0x0 | For each register 'n', Q_MAP[1] defines the per queue statistic registers that are mapped to Rx queue '4*n+1'. (see examples that follow). |
| Reserved | 15:12 | 0x0 | Reserved. |
| Q_MAP[2] | 19:16 | 0x0 | For each register 'n', Q_MAP[2] defines the per queue statistic registers that are mapped to Rx queue '4*n+2'. (see examples that follow). |
| Reserved | 23:20 | 0x0 | Reserved. |
| Q_MAP[3] | 27:24 | 0x0 | For each register 'n', Q_MAP[3] defines the per queue statistic registers that are mapped to Rx queue '4*n+3'. (see examples that follow). |
| Reserved | 31:28 | 0x0 | Reserved. |

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].



### 8.2.3.23.72 Rx DMA Statistic Counter Control — RXDSTATCTRL (0x02F40; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| QSEL | 4:0 | 0x0 | The *Queue Select* field controls which Rx queues are considered for the DMA good Rx and DMA duplicated counters as follows:<br>00000b ... 01111b = The counters relates to the same queues that are directed to the QPRC[QSEL] counter as defined by the RQSMR[n] registers.<br>10000b = The counters relates to all Rx queues.<br>All other values are reserved. |
| Reserved | 31:5 | 0x0 | Reserved. |

### 8.2.3.23.73 Transmit Queue Statistic Mapping Registers — TQSM[n] (0x08600 + 4*n, n=0...31; RW)

These registers define the mapping of the transmit queues to the per queue statistics. Several queues can be mapped to a single statistic register. Each statistic register counts the number of packets and bytes of all the queues that are mapped to that statistics. The registers counting Tx queue statistics are: QPTC and QBTC.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Q_MAP[0] | 3:0 | 0x0 | 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 | Reserved. |
| Q_MAP[1] | 11:8 | 0x0 | 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 | Reserved. |
| Q_MAP[2] | 19:16 | 0x0 | 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 | Reserved. |
| Q_MAP[3] | 27:24 | 0x0 | 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 | Reserved. |

### 8.2.3.23.74 Queue Packets Received Count — QPRC[n] (0x01030 + 0x40*n, n=0...15; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PRC | 31:0 | 0x0 | Number of packets received for the queue.<br>FCoE packets are counted in QRPC even if they are posted only to the DDP queue (with no traces in the legacy queue). |

### 8.2.3.23.75 Queue Packets Received Drop Count — QPRDC[n] (0x01430 + 0x40*n, n=0...15; RC)

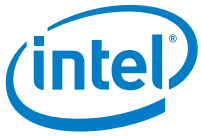
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PRDC | 31:0 | 0x0 | Total number of receive packets dropped for the queue.<br>Packets can be dropped for the following reasons:<br>1. Rx queue is disabled in the RXDCTL[n] register.<br>2. No free descriptors in the Rx queue while hardware is set to *Drop En* in the SRRCTL[n] register or in the PFQDE register. |

### 8.2.3.23.76 Queue Bytes Received Count Low — QBRC_L[n] (0x01034 + 0x40*n, n=0...15; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| BRC_L | 31:0 | 0x0 | Lower 32 bits of the statistic counter.<br>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 QPRC[n].<br>The counter counts all bytes posted to the host before VLAN strip. Furthermore, bytes of RSC and FCoE are counted before coalescing or DDP. |

### 8.2.3.23.77 Queue Bytes Received Count High — QBRC_H[n] (0x01038 + 0x40*n, n=0...15; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| BRC_H | 3:0 | 0x0 | Higher four bits of the statistic counter described in QBRC_L. |
| Reserved | 31:4 | 0x0 | Reserved. |



### 8.2.3.23.78 Queue Packets Transmitted Count — QPTC[n] (0x08680 + 0x4*n, n=0...15 / 0x06030 + 0x40*n, n=0...15; RC)

These registers are also mapped to 0x06030 to maintain compatibility with the 82598.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PTC | 31:0 | 0x0 | Number of packets transmitted for the queue.<br>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 VLAN tag validation (as described in Section 7.10.3.9.2) are not counted. |

### 8.2.3.23.79 Queue Bytes Transmitted Count Low — QBTC_L[n] (0x08700 + 0x8*n, n=0...15; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| BTC_L | 31:0 | 0x0 | Lower 32 bits of the statistic counter.<br>The QBTC_L and QBTC_H registers make up a logical 36-bit counter of transmitted bytes of the packets counted by the matched QPTC counter. These registers count all bytes in the packets from the <Destination Address> field through the <CRC> field, inclusively. These registers must be accessed as two consecutive 32-bit entities while the QBTC_L register is read first, or a single 64-bit read cycle. Each register is read cleared. In addition, it sticks at 0xFF...F to avoid overflow. |

### 8.2.3.23.80 Queue Bytes Transmitted Count High — QBTC_H[n] (0x08704 + 0x8*n, n=0...15; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| BTC_H | 3:0 | 0x0 | Higher four bits of the statistic counter described in QBTC_L. |
| Reserved | 31:4 | 0x0 | Reserved. |

### 8.2.3.23.81 FC CRC Error Count — FCCRC (0x05118; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| CRC_CNT | 15:0 | 0x0 | FC CRC Count.<br>Count the number of packets with good Ethernet CRC and bad FC CRC. |
| Reserve | 31:16 | N/A | Reserved. |

### 8.2.3.23.82 FCoE Rx Packets Dropped Count — FCOERPDC (0x0241C; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RPDC | 31:0 | 0x0 | Number of Rx packets dropped due to lack of descriptors. |

### 8.2.3.23.83 FC Last Error Count — FCLAST (0x02424; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Last_CNT | 15:0 | 0x0 | Number of packets received to valid FCoE contexts while their user buffers are exhausted. |
| Reserve | 31:16 | N/A | Reserved. |

### 8.2.3.23.84 FCoE Packets Received Count — FCOEPRC (0x02428; RC)

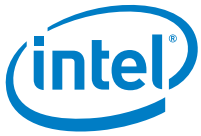
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PRC | 31:0 | 0x0 | Number of FCoE packets posted to the host.<br>In normal operation (no save bad frames) it equals to the number of good packets. |

### 8.2.3.23.85 FCOE DWord Received Count — FCOEDWRC (0x0242C; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| DWRC | 31:0 | 0x0 | Number of DWords count in good received packets with no Ethernet CRC or FC CRC errors.<br>The counter relates to FCoE packets starting at the FC header up to and including the FC CRC (it excludes Ethernet encapsulation). |

### 8.2.3.23.86 FCoE Packets Transmitted Count — FCOEPTC (0x08784; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PTC | 31:0 | 0x0 | Number of FCoE packets transmitted.<br>*Note:* The counter does not include packets dropped due to anti-spoofing filtering or VLAN tag validation as described in Section 7.10.3.9.2. This rule is applicable if FCoE traffic is sent by a VF. |



### 8.2.3.23.87 FCoE DWord Transmitted Count — FCOEDWTC (0x08788; RC)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| DWTC | 31:0 | 0x0 | Number of DWords count in transmitted packets.<br>The counter relates to FCoE packets starting at the FC header up to and including the FC CRC (it excludes Ethernet encapsulation). |

# 8.2.3.24 Wake Up Control Registers

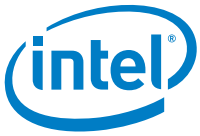
## 8.2.3.24.1 Wake Up Control Register — WUC (0x05800; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 0 | 0b | Reserved . |
| PME_En | 1 | 0b | PME_En.<br>This bit is used by the driver to enable wakeup capabilities as programmed by the WUFC register. Wakeup is further gated by the PME_En bit of the PMCS register.<br>*Note:* Setting the PME_En bit in the PMCSR register also sets this bit. |
| PME_Status (RO) | 2 | 0b | PME_Status.<br>This bit is set when the 82599 receives a wake-up event. It is the same as the PME_Status bit in the PMCSR. Writing a 1b to this bit clears it. The *PME_Status* bit in the PMCSR is also cleared. |
| Reserved | 3 | 0b | Reserved. |
| WKEN | 4 | 1b | WKEN<br>This bit can be cleared to disable PE_WAKE_N pin de-assertion even if APM is enabled in the EEPROM. In this case, PMCSR wake-up status will be invalid. Refer to Section 5.3 for the correct way to enable APM with valid status.<br>*Note:* This bit should not be cleared while in ACPI mode. |
| Reserved | 31:5 | 0x0 | Reserved. |

The *PME_En* and *PME_Status* bits are reset when LAN_PWR_GOOD is 0b. When AUX_PWR=0b these bits are also reset by the assertion of PE_RST_N.

## 8.2.3.24.2 Wake Up Filter Control Register — WUFC (0x05808; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LNKC | 0 | 0b | Link Status Change Wake Up Enable. |
| MAG | 1 | 0b | Magic Packet Wake Up Enable. |
| EX | 2 | 0b | Directed Exact Wake Up Enable. |
| MC | 3 | 0b | Directed Multicast Wake Up Enable.<br>Setting this bit does not enable broadcast packets that are enabled by the *BC* bit in this register. |
| BC | 4 | 0b | Broadcast Wake Up Enable. |
| ARP | 5 | 0b | ARP/IPv4 Request Packet Wake Up Enable. |
| IPV4 | 6 | 0b | Directed IPv4 Packet Wake Up Enable. |
| IPV6 | 7 | 0b | Directed IPv6 Packet Wake Up Enable. |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 14:8 | 0x0 | Reserved. |
| NoTCO | 15 | 0b | Ignore TCO Packets for TCO. |
| FLX0 | 16 | 0b | Flexible Filter 0 Enable. |
| FLX1 | 17 | 0b | Flexible Filter 1 Enable. |
| FLX2 | 18 | 0b | Flexible Filter 2 Enable. |
| FLX3 | 19 | 0b | Flexible Filter 3 Enable. |
| FLX4 | 20 | 0b | Flexible Filter 4 Enable. |
| FLX5 | 21 | 0b | Flexible Filter 5 Enable. |
| Reserved | 31:22 | 0b | Reserved. |

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.

If the *NoTCO* bit is set, then any packet that passes the manageability packet filtering does not cause a wake up event even if it passes one of the wake up filters.

### 8.2.3.24.3 Wake Up Status Register — WUS (0x05810; RW1C)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 31:0 | 0x0 | Reserved. |

**Note:** This register is de-featured and software should not read it. To enable ACPI, this register must be cleared by writing 0x3F01FF.

### 8.2.3.24.4 IP Address Valid — IPAV (0x5838; RW)

The IP address valid indicates whether the IP addresses in the IP address table are valid:

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| V40 | 0 | 0b[1] | IPv4 Address 0 Valid. |
| V41 | 1 | 0b | IPv4 Address 1 Valid. |
| V42 | 2 | 0b | IPv4 Address 2 Valid. |
| V43 | 3 | 0b | IPv4 Address 3 Valid. |
| Reserved | 15:4 | 0x0 | Reserved |

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| V60 | 16 | 0b | IPv6 Address 0 Valid. |
| Reserved | 31:17 | 0x0 | Reserved |

1. Loaded from EEPROM

### 8.2.3.24.5 IPv4 Address Table — IP4AT[n] (0x05840 + 8*n, n = 0...3; RW)

4 x IPv4 addresses for ARP/IPv4 request packet and directed IPv4 packet wake up. IPv4[0] is loaded from MIPAF words in the EEPROM.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| IPV4ADDR | 31:0 | X | IPv4 Address ‘n’, ‘n’ = 0...3. |

### 8.2.3.24.6 IPv6 Address Table — IP6AT[n] (0x05880 + 4*n, n = 0...3; RW)

1 x IPv6 addresses for a neighbor discovery packet filtering and directed IPv6 packet wake up. According to the power management section; one Ipv6 address is supported and it is programmed in the Ipv6 Address Table (IP6AT)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| IPV6ADDR | 31:0 | X | 4 x Register IPv6 filter.<br>Register ‘n’ contains bytes ‘4*n’ up to ‘4*n+3’ of the IPv6 address. LS byte of register ‘0’ is first on the wire. |

### 8.2.3.24.7 Wake Up Packet Length — WUPL (0x05900; RO)

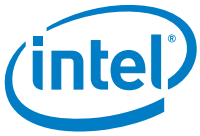
This register is de-featured and software should not access it (not read nor write)

### 8.2.3.24.8 Wake Up Packet Memory (128 Bytes) — WUPM[n] (0x05A00 + 4*n, n=0...31; RO)

This register is de-featured and software should not access it (not read nor write)

### 8.2.3.24.9 Flexible Host Filter Table Registers — FHFT (0x09000 — 0x093FC and 0x09800 — 0x099FC; RW)

Each of the six Flexible Host Filters Table (FHFT) registers contains a 128-byte pattern and a corresponding 128-bit mask array. If enabled, the first 128 bytes of the received packet are compared against the non-masked bytes in the FHFT register.



Each 128-byte filter is composed of 32 Dword entries, where each two Dwords are accompanied by an 8-bit mask, one bit per filter byte.

**Note:** The *Length* field must be eight byte-aligned. For filtering packets shorter than eight byte-aligned the values should be rounded up to the next eight byte-aligned value, hardware implementation compares eight bytes at a time so it should get extra zero masks (if needed) until the end of the length value.

If the actual length, which is defined by the Length Field register and the mask bits is not eight byte-aligned, there might be a case that a packet, which is shorter then the actual required length pass the flexible filter. This can happen due to a comparison of up to seven bytes that come after the packet but are not a real part of the packet.

The last Dword of each filter contains a *Length* field defining the number of bytes from the beginning of the packet compared by this filter, the *Length* field should be an eight byte-aligned value. If the actual packet length is less than (length — 8) (length is the value specified by the *Length* field), the filter fails. Otherwise, it depends on the result of actual byte comparison. The value should not be greater than 128.

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

. . .

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

Each of the filters have allocated addresses as follows:

- Filter 0 — 0x09000 — 0x090FF
- Filter 1 — 0x09100 — 0x091FF
- Filter 2 — 0x09200 — 0x092FF
- Filter 3 — 0x09300 — 0x093FF
- Filter 4 — 0x09800 — 0x098FF
- Filter 5 — 0x09900 — 0x099FF

The following table 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 registers during filter operation can result in a packet being mis-classified if the write operation collides with packet reception. As a result, it is recommended that the flex filters be disabled prior to changing their setup.

## 8.2.3.25 Management Filters Registers

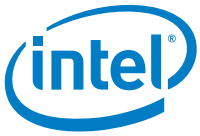
The Management Filters registers are RO for the host. These registers are initialized at LAN Power Good and can be loaded from the EEPROM by the manageability firmware.

### 8.2.3.25.1 Management VLAN TAG Value — MAVTV[n] (0x5010 +4*n, n=0...7; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| VID | 11:0 | 0x0 | 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 | Reserved. |

### 8.2.3.25.2 Management Flex UDP/TCP Ports — MFUTP[n] (0x5030 + 4*n, n=0...7; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MFUTP[2n] | 15:0 | 0x0 | (2n)-th Management Flex UDP/TCP port. |
| MFUTP[2n+1] | 31:16 | 0x0 | (2n+1)-th Management Flex UDP/TCP port. |



Each 32-bit register (n=0,…,7) refers to two port filters (register 0 refers to ports 0 and 1, register 2 refers to ports 2 and 3, etc.). Note that SCTP packets do not match the MFUTP filters.

### 8.2.3.25.3 Management Ethernet Type Filters- METF[n] (0x05190 + 4*n, n=0...3; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| EType | 15:0 | 0x0 | EtherType value to be compared against the L2 *EtherType* field in the Rx packet.<br>*Note:* Appears in little endian order (high byte first on the wire). |
| Reserved | 29:16 | 0x0 | Reserved. |
| Polarity | 30 | 0b | 0b = Positive filter. Filter enters the decision filters if a match occurred.<br>1b = Negative filter. Filter enters the decision filters if a match did not occur. |
| Reserved | 31 | 0b | Reserved. |

### 8.2.3.25.4 Management Control Register — MANC (0x05820; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 16:0 | 0x0 | Reserved. |
| RCV_TCO_EN | 17 | 0b | Receive TCO Packets Enabled.<br>When this bit is set it enables the receive flow from the wire to the manageability block. |
| Reserved | 18 | 0b | Reserved. |
| RCV_ALL | 19 | 0b | Receive All Enable.<br>When set, all packets are received from the wire and passed to the manageability block. |
| MCST_PASS_L2 | 20 | 0b | Receive All Multicast.<br>When set, all received multicast packets pass L2 filtering and can be directed to the MNG or Host by ~~a~~ one of the decision filters. Broadcast packets are not forwarded by this bit. |
| EN_MNG2HOST | 21 | 0b | Enable manageability packets to host memory.<br>This bit enables the functionality of the MANC2H register. When set the packets that are specified in the MANC2H registers are also forwarded to host memory, if they pass manageability filters. |
| Bypass VLAN | 22 | 0b | When set, VLAN filtering is bypassed for MNG packets. |
| EN_XSUM_FILTER | 23 | 0b | When set, this bit enables Xsum filtering to manageability. Meaning, only packets that pass L3, L4 checksum are sent to the manageability block.<br>*Note:* This capability is not provided for tunneled packets. |
| EN_IPv4_FILTER | 24 | 0b | Enable IPv4 address Filters.<br>When set, the last 128 bits of the MIPAF register are used to store four IPv4 addresses for IPv4 filtering. When cleared, these bits store a single IPv6 filter. |

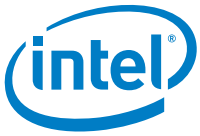
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| FIXED_NET_TYPE | 25 | 0b | Fixed Next Type.<br>If set, only packets matching the net type defined by the NET_TYPE field pass to manageability. Otherwise, both tagged and un-tagged packets can be forwarded to the manageability engine. |
| NET_TYPE | 26 | 0b | Net Type.<br>0b = Pass only un-tagged packets.<br>1b = Pass only VLAN tagged packets.<br>Valid only if FIXED_NET_TYPE is set. |
| Reserved | 31:27 | 0x0 | Reserved. |

### 8.2.3.25.5 Manageability Filters Valid — MFVAL (0x5824; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MAC | 3:0 | 0x0 | MAC.<br>Indicates if the MAC unicast filter registers (MMAH, MMAL) contain valid Ethernet MAC Addresses. Bit 0 corresponds to filter 0, etc. |
| Reserved | 7:4 | 0x0 | Reserved. |
| VLAN | 15:8 | 0x0 | VLAN.<br>Indicates if the VLAN filter registers (MAVTV) contain valid VLAN tags. Bit 8 corresponds to filter 0, etc. |
| IPv4 | 19:16 | 0x0 | IPv4.<br>Indicates if the IPv4 address filters (MIPAF) contain valid IPv4 addresses. Bit 16 corresponds to IPv4 address 0. These bits apply only when IPv4 address filters are enabled (MANC.EN_IPv4_FILTER=1). |
| Reserved | 23:20 | 0x0 | Reserved. |
| IPv6 | 27:24 | 0x0 | IPv6.<br>Indicates if the IPv6 address filter registers (MIPAF) contain valid IPv6 addresses. Bit 24 corresponds to address 0, etc. Bit 27 (filter 3), applies only when IPv4 address filters are not enabled. (MANC.EN_IPv4_FILTER=0). |
| Reserved | 31:28 | 0x0 | Reserved. |

### 8.2.3.25.6 Management Control To Host Register — MANC2H (0x5860; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Host Enable | 7:0 | 0x0 | Host Enable.<br>When set, indicates that packets routed by the manageability filters to manageability are also sent to the host. Bit 0 corresponds to decision filter (MDEF[0] and MDEF_EXT[0]), bit 1 corresponds to decision filter (MDEF[1] and MDEF_EXT[1]), etc. The MANC2H routing is further enabled by a global MANC.EN_MNG2HOST bit. |
| Reserved | 31:8 | 0x0 | Reserved. |



### 8.2.3.25.7 Manageability Decision Filters- MDEF[n] (0x5890 + 4*n, n=0...7; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Unicast (AND) | 0 | 0b | Unicast.<br>Controls the inclusion of unicast address filtering in the manageability filter decision (AND section). |
| Broadcast (AND) | 1 | 0b | Broadcast.<br>Controls the inclusion of broadcast address filtering in the manageability filter decision (AND section). |
| VLAN (AND) | 2 | 0b | VLAN.<br>Controls the inclusion of VLAN address filtering in the manageability filter decision (AND section). |
| IP Address (AND) | 3 | 0b | IP Address.<br>Controls the inclusion of IP address filtering in the manageability filter decision (AND section). |
| Unicast (OR) | 4 | 0b | Unicast.<br>Controls the inclusion of unicast address filtering in the manageability filter decision (OR section). |
| Broadcast (OR) | 5 | 0b | Broadcast.<br>Controls the inclusion of broadcast address filtering in the manageability filter decision (OR section). |
| Multicast (AND) | 6 | 0b | Multicast.<br>Controls the inclusion of multicast address filtering in the manageability filter decision (AND section). Broadcast packets are not included by this bit. The packet must pass some L2 filtering to be included by this bit – either by the MANC.MCST_PASS_L2 or by some dedicated Ethernet MAC Address. |
| ARP Request (OR) | 7 | 0b | ARP Request.<br>Controls the inclusion of ARP request filtering in the manageability filter decision (OR section). |
| ARP Response (OR) | 8 | 0b | ARP Response.<br>Controls the inclusion of ARP response filtering in the manageability filter decision (OR section). |
| Neighbor Discovery (OR) | 9 | 0b | Neighbor Discovery.<br>Controls the inclusion of neighbor discovery filtering in the manageability filter decision (OR section). The neighbor types accepted by this filter are types 0x86, 0x87, 0x88 and 0x89. |
| Port 0x298 (OR) | 10 | 0b | Port 0x298.<br>Controls the inclusion of port 0x298 filtering in the manageability filter decision (OR section). |
| Port 0x26F (OR) | 11 | 0b | Port 0x26F.<br>Controls the inclusion of port 0x26F filtering in the manageability filter decision (OR section). |

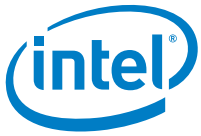
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Flex port (OR) | 27:12 | 0x0 | Flex Port.<br>Controls the inclusion of flex port filtering in the manageability filter decision (OR section). Bit 12 corresponds to flex port 0, etc. |
| Flex TCO (OR) | 31:28 | 0x0 | Flex TCO.<br>Controls the inclusion of Flex TCO filtering in the manageability filter decision (OR section). Bit 28 corresponds to Flex TCO filter 0, etc. |

### 8.2.3.25.8 Manageability Decision Filters- MDEF_EXT[n] (0x05160 + 4*n, n=0...7; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| L2 EtherType (AND) | 3:0 | 0x0 | L2 EtherType.<br>Controls the inclusion of L2 EtherType filtering in the manageability filter decision (AND section). |
| Reserved | 7:4 | 0x0 | Reserved for additional L2 EtherType AND filters. |
| L2 EtherType (OR) | 11:8 | 0x0 | L2 EtherType.<br>Controls the inclusion of L2 EtherType filtering in the manageability filter decision (OR section). |
| Reserved | 15:12 | 0x0 | Reserved for additional L2 EtherType OR filters. |
| Reserved | 31:16 | 0x0 | Reserved. |

### 8.2.3.25.9 Manageability IP Address Filter — MIPAF[m,n] (0x58B0 + 0x10*m + 4*n, m=0...3, n=0...3; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| IP_ADDR | 31:0 | X | Manageability IP Address Filters.<br>For each n, m, m=0...3, n=0...3 while MANC.EN_IPv4_FILTER = 0, MIPAF[m,n] register holds Dword 'n' of IPv6 filter 'm' (4 x IPv6 filters).<br>For each n, m, m=0...3, n=0...3 while MANC.EN_IPv4_FILTER = 1, MIPAF[m,n] registers for m=0,1,2 is the same as the previous case (3 x IPv6 filters). And MIPAF[3,n] registers holds IPv4 filter 'n' (4 x IPv4 filters).<br>*Note:* These registers appear in big endian order (LS byte, LS address is first on the wire). |



### 8.2.3.25.10 Manageability Ethernet MAC Address Low — MMAL[n] (0x5910 + 8*n, n=0...3; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MMAL | 31:0 | X | Manageability Ethernet MAC Address Low.<br>The lower 32 bits of the 48-bit Ethernet MAC address.<br>*Note:* Appears in big endian order (LS byte of MMAL is first on the wire). |

### 8.2.3.25.11 Manageability Ethernet MAC Address High — MMAH[n] (0x5914 + 8*n, n=0...3; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MMAH | 15:0 | X | Manageability Ethernet MAC Address High.<br>The upper 16 bits of the 48-bit Ethernet MAC address.<br>*Note:* Appears in big endian order (MS byte of MMAH is last on the wire). |
| Reserved | 31:16 | 0x0 | Reserved.<br>Reads as 0x0. Ignored on write. |

### 8.2.3.25.12 Flexible TCO Filter Table Registers — FTFT (0x09400-0x097FC; RW)

Each of the four Flexible TCO Filters Table (FTFT) registers contains a 128-byte pattern and a corresponding 128-bit mask array. If enabled, the first 128 bytes of the received packet are compared against the non-masked bytes in the FTFT register.

**Note:** FTFT registers are configured by firmware. Host write/read access to these registers should be avoided.

Each 128-byte filter is composed of 32 Dword entries, where each two Dwords are accompanied by an 8-bit mask, one bit per filter byte. 15:8] etc. The *Mask* field is set so that bit 0 in the mask masks byte 0, bit 1 masks byte 1 etc. A value of one in the *Mask* field means that the appropriate byte in the filter should be compared to the appropriate byte in the incoming packet.

**Notes:** The *Mask* field must be eight byte-aligned even if the *Length* field is not eight byte-aligned as the hardware implementation compares eight bytes at a time so it should get extra masks until the end of the next Qword. Any *Mask* bit that is located after the length should be set to zero indicating no comparison should be done.

If the actual length, which is defined by the Length Field register and the mask bits is not eight byte-aligned, there might be a case where a packet, which is shorter then the actual required length passes the flexible filter. This can happen due to a comparison of up to seven bytes that come after the packet but are not a real part of the packet.

The last Dword of each filter contains a *Length* field defining the number of bytes from the beginning of the packet compared by this filter. If actual packet length is less than the 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.

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

. . .

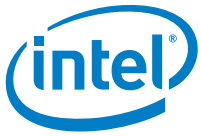
| 31 0 | 31 8 | 7 0 | 31 0 | 31 0 |
|---|---|---|---|---|
| Reserved | Reserved | Mask [127:120] | Dword 29 | Dword 28 |
| Length | Reserved | Mask [127:120] | Dword 31 | Dword 30 |

| Field | Dword | Address | Bit(s) | Initial Value |
|---|---|---|---|---|
| Filter 0 DW0 | 0 | 0x09400 | 31:0 | X |
| Filter 0 DW1 | 1 | 0x09404 | 31:0 | X |
| Filter 0 Mask[7:0] | 2 | 0x09408 | 7:0 | X |
| Reserved | 3 | 0x0940C | | X |
| Filter 0 DW2 | 4 | 0x09410 | 31:0 | X |
| … | | | | |
| Filter 0 DW30 | 60 | 0x094F0 | 31:0 | X |
| Filter 0 DW31 | 61 | 0x094F4 | 31:0 | X |
| Filter 0 Mask[127:120] | 62 | 0x094F8 | 7:0 | X |
| Length | 63 | 0x094FC | 6:0 | X |

### 8.2.3.25.13 LinkSec Software/Firmware Interface — LSWFW (0x015F14; RO)

**Note:** This register is shared for both LAN ports.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Lock LinkSec Logic | 0 | 0b | Block LinkSec<br>0b = Host can access LinkSec registers.<br>1b = Host cannot access LinkSec registers. |
| Block host traffic | 1 | 0b | When set, all host traffic (Tx and Rx) is blocked. |
| Request LinkSec (SC) | 2 | 0b | When set, a message is sent to the MC, requesting access to the LinkSec registers. |
| Release LinkSec (SC) | 3 | 0b | When set, a message is sent to the MC, releasing ownership of the LinkSec registers. |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 7:4 | 0x0 | Reserved. |
| LinkSec Ownership | 8 | 0b | Set by firmware to indicate the status of the LinkSec ownership:<br>0b = LinkSec owned by host (default).<br>1b = LinkSec owned by MC. |
| Reserved | 31:9 | 0x0 | Reserved. |

**Note:** The access rules on this register are for the driver software.

# 8.2.3.26 Time Sync (IEEE 1588) Registers

## 8.2.3.26.1 Rx Time Sync Control Register — TSYNCRXCTL (0x05188; RW)

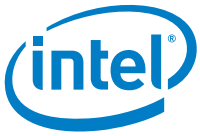
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RXTT(RO/V) | 0 | 0b | Rx Time Stamp Valid.<br>Equals 1b when a valid value for Rx time stamp is captured in the Rx Time Stamp register. Cleared by read of Rx Time Stamp (RXSTMPH) register. |
| Type | 3:1 | 0x0 | Type of packets to time stamp:<br>000b = Time stamp L2 (V2) packets only (sync or Delay_req depends on message type in Section 8.2.3.26.6 and packets with message ID 2 and 3).<br>001b = Time stamp L4 (V1) packets only (sync or Delay_req depends on message type in Section 8.2.3.26.6).<br>010b = Time stamp V2 (L2 and L4) packets (sync or Delay_req depends on message type in Section 8.2.3.26.6 and packets with message ID 2 and 3).<br>101b = Time stamp all packets in which message ID bit 3 is zero, which means time stamp all event packets. This is applicable for V2 packets only.<br>011b = Reserved<br>100b = Reserved<br>110b = Reserved<br>111b = Reserved |
| En | 4 | 0b | Enable Rx Time Stamp.<br>0x0 = Time stamping disabled.<br>0x1 = Time stamping enabled. |
| RSV | 31:5 | 0x0 | Reserved. |

## 8.2.3.26.2 Rx Time Stamp Low — RXSTMPL (0x051E8; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RXSTMPL | 31:0 | 0x0 | Rx time stamp LSB value. |

## 8.2.3.26.3 Rx Time Stamp High — RXSTMPH (0x051A4; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RXSTMPH | 31:0 | 0x0 | Rx time stamp MSB value. |



### 8.2.3.26.4 Rx Time Stamp Attributes Low — RXSATRL (0x051A0; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SourceIDL | 31:0 | 0x0 | Sourceuuid Low.<br>Captured bytes 24-27 in the PTP message as listed in Section 7.9.5 while the MS byte is last on the wire. In a V1 PTP packet it is the 4 LS bytes of the *Sourceuuid* field and in V2 PTP packet it is part of the *Source Port ID* field. |

### 8.2.3.26.5 Rx Time Stamp Attributes High- RXSATRH (0x051A8; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| SourceIDH | 15:0 | 0x0 | Sourceuuid High.<br>Captured bytes 22-23 in the PTP message as listed in Section 7.9.5 while the LS byte is first on the wire. In a V1 PTP packet it is the 2 MS bytes of the *Sourceuuid* field and in V2 PTP packet it is part of the *Source Port ID* field. |
| SequenceID | 31:16 | 0x0 | Sequence Id.<br>Captured value of the *SequenceID* field in the PTP Rx packet while LS byte first on the wire. |

### 8.2.3.26.6 Rx Message Type Register Low — RXMTRL (0x05120; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| CTRLT | 7:0 | 0x0 | V1 control to time stamp. |
| MSGT | 15:8 | 0x0 | V2 message ID to time stamp. |
| UDPT | 31:16 | 0x319 | UDP port number to time stamp. |

### 8.2.3.26.7 Tx Time Sync Control Register — TSYNCTXCTL (0x08C00; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TXTT(RO/V) | 0 | 0b | Tx Time Stamp Valid.<br>Equals 1b when a valid value for Tx time stamp is captured in the Tx Time Stamp register. Cleared by read of Tx Time Stamp (TXSTMPH) register. |
| RSV | 3:1 | 0x0 | Reserved. |

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| EN | 4 | 0x0 | Enable Tx Time Stamp.<br>0x0 Time stamping disabled.<br>0x1 Time stamping enabled. |
| RSV | 31:5 | 0x0 | Reserved. |

### 8.2.3.26.8 Tx Time Stamp Value Low — TXSTMPL (0x08C04; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TXSTMPL | 31:0 | 0x0 | Tx time stamp LSB value. |

### 8.2.3.26.9 Tx Time Stamp Value High — TXSTMPH (0x08C08; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TXSTMPH | 31:0 | 0x0 | Tx time stamp MSB value. |

### 8.2.3.26.10 System Time Register Low — SYSTIML (0x08C0C; RW)

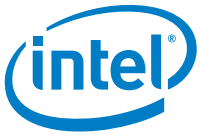
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| STL | 31:0 | 0x0 | System time LSB register. |

### 8.2.3.26.11 System Time Register High — SYSTIMH (0x08C10; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| STH | 31:0 | 0x0 | System time MSB register. |

### 8.2.3.26.12 Increment Attributes Register — TIMINCA (0x08C14; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| IV | 23:0 | 0x0 | Increment Value (*incvalue).* |
| IP | 31:24 | 0x0 | Increment Period (*incperiod)*.<br>*Note:* The minimum permitted functional value is two. |



### 8.2.3.26.13 Time Adjustment Offset Register Low — TIMADJL (0x08C18; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TADJL | 31:0 | 0x0 | Time Adjustment Value Low. |

### 8.2.3.26.14 Time Adjustment Offset Register High — TIMADJH (0x08C1C; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TADJH | 30:0 | 0x0 | Time Adjustment Value High. |
| Sign | 31 | 0x0 | Sign (“0”=”+”, “1”=”-“). |

### 8.2.3.26.15 TimeSync Auxiliary Control Register — TSAUXC (0x08C20; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| EN_TT0 | 0 | 0b | Enable Target Time 0. |
| EN_TT1 | 1 | 0b | Enable Target Time 1. |
| Reserved | 2 | 0b | Reserved. |
| UTT0 | 3 | 0b | Use target time 0 to clear clk_out 0 down counter. |
| ST0 | 4 | 0b | Start clock out toggle only if target of clock out occurs. |
| Reserved | 5 | 0b | Reserved. |
| UTT1 | 6 | 0b | Use target time 1 to clear clk_out 1 down counter. |
| ST1 | 7 | 0b | Start clock out toggle only on target time 1, at this point a rising edge of clock out occurs. |
| EN_TS0 | 8 | 0b | Enable Hardware Time Stamp 0. |
| AUTT0 | 9 | 0b | Auxiliary Time Stamp Taken.<br>Cleared when read after an auxiliary time stamp 0 occurred. |
| EN_TS1 | 10 | 0b | Enable Hardware Time Stamp 1. |
| AUTT1 | 11 | 0b | Auxiliary Time Stamp Taken.<br>Cleared when read after auxiliary time stamp 1 occurred. |

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Mask | 16:12 | 0b | Masking Value for Target Time.<br>The value in this field determines the masked bits in the comparison of the system time and target time (where 0 = no masking, 1 = bit 0 is masked, 2 = bit 0 and 1 are masked and so on up to 24 in which bits 0 through bit 23 are masked. Any value higher than 24 are reserved). |
| RSV | 31:17 | 0b | Reserved. |

### 8.2.3.26.16 Target Time Register 0 Low — TRGTTIML0 (0x08C24; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TTL | 31:0 | 0x0 | Target time 0 LSB register. |

### 8.2.3.26.17 Target Time Register 0 High — TRGTTIMH0 (0x08C28; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TTH | 31:0 | 0x0 | Target time 0 MSB register. |

### 8.2.3.26.18 Target Time Register 1 Low — TRGTTIML1 (0x08C2C; RW)

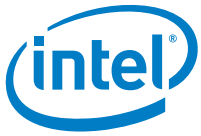
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TTL | 31:0 | 0x0 | Target time 1 LSB register. |

### 8.2.3.26.19 Target Time Register 1 High — TRGTTIMH1 (0x08C30; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TTH | 31:0 | 0x0 | Target time 1 MSB register. |

### 8.2.3.26.20 Frequency Out 0 Control Register — FREQOUT0 (0x08C34; RW) SEC-Tx

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RLV | 31:0 | 0x0 | Reload value for frequency out zero down counter. |



### 8.2.3.26.21 Frequency Out 1 Control Register — FREQOUT1 (0x08C38; RW) SEC-Tx

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RLV | 31:0 | 0x0 | Reload value for frequency out one down counter. |

### 8.2.3.26.22 Auxiliary Time Stamp 0 Register Low — AUXSTMPL0 (0x08C3C; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TST_Low | 31:0 | 0x0 | Auxiliary time stamp 0 LSB value. |

### 8.2.3.26.23 Auxiliary Time Stamp 0 Register High — AUXSTMPH0 (0x08C40; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TST_Hi | 31:0 | 0x0 | Auxiliary time stamp 0 MSB value. |

### 8.2.3.26.24 Auxiliary Time Stamp 1 Register Low — AUXSTMPL1 (0x08C44; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TST_Low | 31:0 | 0x0 | Auxiliary time stamp 1 LSB value. |

### 8.2.3.26.25 Auxiliary Time Stamp 1 Register High — AUXSTMPH1 (0x08C48; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| TST_Hi | 31:0 | 0x0 | Auxiliary time stamp 1 MSB value. |

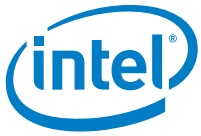
# 8.2.3.27 Virtualization PF Registers

## 8.2.3.27.1 VT Control Register — PFVTCTL (0x051B0; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| VT_Ena | 0 | 0b | Virtualization Enabled Mode.<br>When set, the 82599 supports either 16, 32, or 64 pools.<br>When cleared, Rx traffic is handled internally as if it belongs to VF zero while VF zero is enabled.<br>This bit should be set the same as MTQC.VT_Ena. |
| Reserved | 6:1 | 0x0 | Reserved. |
| DEF_PL | 12:7 | 0x0 | Default Pool.<br>Pool assignment for packets that do not pass any pool queuing decision. Enabled by the *Dis_Def_Pool* bit. |
| Reserved | 28:13 | 0x0 | Reserved. |
| Dis_Def_Pool | 29 | 0b | Disable Default Pool.<br>Determines the behavior of an Rx packet that does not match any Rx filter and is therefore not allocated a destination pool.<br>0b = Packet is assigned to the default pool (see DEF_PL).<br>1b = Packet is dropped. |
| Rpl_En | 30 | 0b | Replication Enable, when set to 1b. |
| Reserved | 31 | 0b | Reserved. |

## 8.2.3.27.2 PF Mailbox — PFMailbox[n] (0x04B00 + 4*n, n=0...63; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Sts (WO) | 0 | 0b | Status/Command from PF ready.<br>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 (WO) | 1 | 0b | VF message received.<br>Setting this bit, causes an interrupt to the relevant VF. This bit always read as zero. Setting this bit sets the *PFACK* bit in VFMailbox. |
| VFU | 2 | 0b | Buffer is taken by VF.<br>This bit is RO for the PF and is a mirror of the *VFU* bit in the VFMailbox register. |
| PFU | 3 | 0b | Buffer is taken by PF.<br>This bit can be set only if the *VFU* bit is cleared and is mirrored in the *PFU* bit of the VFMailbox register. |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| RVFU (WO) | 4 | 0b | Reset VFU.<br>Setting this bit clears the *VFU* bit in the corresponding VFMailbox register. This bit should be used only if the VF driver is not operational. Setting this bit also resets the corresponding bits in the PFMBICR *VFREQ* and *VFACK* fields. |
| Reserved | 31:5 | 0x0 | Reserved. |

### 8.2.3.27.3 PF Mailbox Interrupt Causes Register — PFMBICR[n] (0x00710 + 4*n, n=0...3; RW1C)

Each register handles 16 VFs and are defined as follows.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| VFREQ | 15:0 | 0x0 | Each bit in the *VFREQ* field is set when VF number (16*n+j) wrote a message in its mailbox.<br>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 | Each bit in the *VFACK* field is set when VF number (16*n+j) acknowledged a PF message.<br>While 'n' is the register index, n=0...3 and '16+j' is the index of the bits in the VFACK, j=0...15. |

### 8.2.3.27.4 PF Mailbox Interrupt Mask Register — PFMBIMR[n] (0x00720 + 4*n, n=0...1; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| VFIM | 31:0 | 0xFF | Mailbox interrupt enable from VF # 32*n+j, while 'n' is the register index and 'j' is the bit number. |

### 8.2.3.27.5 PF VFLR Events Indication — PFVFLRE[n] (0x00600, 0x001C0; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| VFLE | 31:0 | 0x0 | When set, bit 'i' in register 'n' reflects an FLR event on VF# 32*n+i.<br>These bits are accessible only to the PF and are cleared by writing 0x1 to the matched bit in the PFVFLREC registers. |

### 8.2.3.27.6 PF VFLR Events Clear — PFVFLREC[n] (0x00700 + 4*n, n=0...1; W1C)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Clear VFLE | 31:0 | X | Writing a 0x1 to bit 'i' in register 'n' clears the FLR event on VF# 32*n+i indicated in the PFVFLRE[n] registers. |

### 8.2.3.27.7 PF VF Receive Enable — PFVFRE[n] (0x051E0 + 4*n, n=0...1; RW)

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 Section 4.2.2.2 for more details.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| VFRE | 31:0 | 0x0 | Bit j.<br>Enables receiving packets to VF# (32*n+j).<br>Each bit is cleared by the relevant VFLR. |

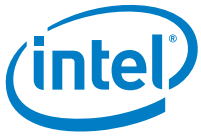
### 8.2.3.27.8 PF VF Transmit Enable — PFVFTE[n] (0x08110 + 4*n, n=0...1; RW)

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 Section 4.2.2.2 for more details.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| VFTE | 31:0 | 0x0 | Bit j.<br>Enables transmitting packets from VF# (32*n+j).<br>Each bit is cleared by the relevant VFLR. |

### 8.2.3.27.9 PF PF Queue Drop Enable Register — PFQDE (0x02F04; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| QDE | 0 | 0b | Enable drop of packets from Rx queue Queue_Index.<br>This bit overrides the SRRCTL.drop_en bit of each queue. For example, if either of the bits is set, a packet received when no descriptor is available is dropped. |
| Reserved | 3:1 | 0x0 | Reserved (see *WE* and *RE* bit descriptions). |
| Reserved | 7:4 | 0x0 | Reserved. |
| Queue Index | 14:8 | 0x0 | Indicates the queue referenced upon WE/RE commands. |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 15 | 0b | Reserved. |
| WE | 16 | 0b | Write Enable.<br>When this bit is set, the content of bits 3:0 are written into the relevant queue context. Bits 3:1 are reserved.<br>This bit should never be set together with the *RE* bit in this register. |
| RE | 17 | 0b | Read Enable.<br>When this bit is set, the content of bits 3:0 are read from the relevant queue context. Bits 3:1 are reserved.<br>This bit should never be set together with the *WE* bit in this register. |
| Reserved | 31:18 | 0x0 | Reserved. |

### 8.2.3.27.10 PF VM Tx Switch Loopback Enable — PFVMTXSW[n] (0x05180 + 4*n, n=0...1; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LLE | 31:0 | 0x0 | Local Loopback Enable.<br>For each register 'n', and bit 'i', i=0..31, enables Local loopback for pool 32*n+1.<br>When set, a packet originating from a specific pool and destined to the same pool is allowed to be looped back. If cleared, the packet is dropped. |

### 8.2.3.27.11 PF VF Anti Spoof Control — PFVFSPOOF[n] (0x08200 + 4*n, n=0...7; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MACAS | 7:0 | 0x0 | For each register 'n', and bit 'i', i=0..7, enables anti-spoofing filter on Ethernet MAC addresses for VF(8*n+1). |
| VLANAS | 15:8 | 0x0 | For each register 'n', and bit '8+i', i=0..7, enables anti-spoofing filter on VLAN tag for VF(8*n+i).<br>*Note:* If *VLANAS* is set for a specific pool, then the respective *MACAS* bit must be set as well. |
| Reserved | 31:16 | 0x0 | Reserved. |

### 8.2.3.27.12 PFDMA Tx General Switch Control — PFDTXGSWC (0x08220; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| LBE | 0 | 0b | Enables VMDQ loopback. |
| Reserved | 31:1 | 0x0 | Reserved. |

### 8.2.3.27.13 PF VM VLAN Insert Register — PFVMVIR[n] (0x08000 + 4*n, n=0...63; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Port VLAN ID | 15:0 | 0x0 | Port VLAN tag to insert if the *VLANA* field = 01b. |
| Reserved | 29:16 | 0x0 | Reserved. |
| VLANA | 31:30 | 0x0 | VLAN action.<br>00b = Use descriptor command.<br>01b = Always insert default VLAN.<br>10b = Never insert VLAN.<br>11b = Reserved. |

### 8.2.3.27.14 PF VM L2 Control Register — PFVML2FLT[n] (0x0F000 + 4*n, n=0...63; RW)

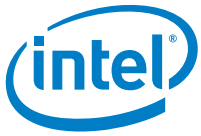
This register controls per VM Inexact L2 Filtering.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 23:0 | 0x0 | Reserved. |
| AUPE | 24 | 0b | Accept Untagged Packets Enable.<br>When set, packets without a VLAN tag can be forwarded to this queue, assuming they pass the Ethernet MAC address queuing mechanism. |
| ROMPE | 25 | 0b | Receive Overflow Multicast Packets. Accept packets that match the MTA table. |
| ROPE | 26 | 0b | Receive MAC Filters Overflow. Accept packets that match the PFUTA table. |
| BAM | 27 | 0b | Broadcast Accept. |
| MPE | 28 | 0b | Multicast Promiscuous. |
| Reserved | 31:29 | 0x0 | Reserved. |

### 8.2.3.27.15 PF VM VLAN Pool Filter — PFVLVF[n] (0x0F100 + 4*n, n=0...63; RW)

Software should initialize these registers before transmit and receive are enabled.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| VLAN_Id | 11:0 | X | Defines a VLAN tag for pool VLAN filter n.<br>The bitmap defines which pools belong to this VLAN.<br>*Note:* Appears in little endian order (LS byte last on the wire). |



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 30:12 | X | Reserved. |
| VI_En | 31 | X | VLAN Id Enable — this filter is valid. |

### 8.2.3.27.16 PF VM VLAN Pool Filter Bitmap — PFVLVFB[n] (0x0F200 + 4*n, n=0...127; RW)

Software should initialize these registers before transmit and receive are enabled.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| POOL_ENA | 31:0 | x | Pool Enable Bit Array.<br>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 follows:<br>• Bit 'i' in register '2*n' is associated with POOL 'i'.<br>• Bit 'i' in register '2*n+1' is associated with POOL '32+i'. |

### 8.2.3.27.17 PF Unicast Table Array — PFUTA[n] (0x0F400 + 4*n, n=0...127; RW)

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 seven MS bits of the Ethernet MAC address (out of the 12 bits) selects the register index while the five LS bits (out of the 12 bits) selects the bit within a register.

**Note:** All accesses to this table must be 32 bit.

The look-up algorithm is the same one used for the MTA table.

This table should be zeroed by software before start of operation.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Bit Vector | 31:0 | X | Word wide bit vector specifying 32 bits in the unicast destination address filter table. |

### 8.2.3.27.18 PF Mirror Rule Control — PFMRCTL[n] (0x0F600 + 4*n, n= 0...3; RW)

This register defines mirroring rules for each of four destination pools.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| VPME | 0 | 0b | Virtual Pool Mirroring Enable.<br>Enables mirroring of certain pools as defined in the PFMRVM registers. |
| UPME | 1 | 0b | Uplink Port Mirroring Enable.<br>Enables mirroring of all traffic received from the network. |
| DPME | 2 | 0b | Downlink Port Mirroring Enable.<br>Enables mirroring of all traffic transmitted to the network. |
| VLME | 3 | 0b | VLAN Mirroring Enable.<br>Enables mirroring of a set of given VLANs as defined in the PFMRVLAN registers. |
| Reserved | 7:4 | 0x0 | Reserved. |
| MP | 13:8 | 0x0 | Mirror Pool.<br>Defines the destination pool for this mirror rule. |
| Reserved | 31:14 | 0x0 | Reserved. |

### 8.2.3.27.19 PF Mirror Rule VLAN — PFMRVLAN[n] (0x0F610 + 4*n, n= 0...7; RW)

This register defines the VLAN values as listed in the PFVLVF table taking part in the VLAN mirror rule.

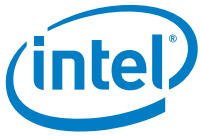
Registers 0, 4 correspond to rule 0, registers 1, 5 correspond to rule 1, etc. Registers 0-3 correspond to the LSB in the PFVLVF table. For example, register 0 corresponds to VLAN filters 31:0, while register 4 corresponds to VLAN filters 63:32.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| VLAN | 31:0 | 0x0 | Bitmap listing which VLANs participate in the mirror rule. |

### 8.2.3.27.20 PF Mirror Rule Pool — PFMRVM[n] (0x0F630 + 4*n, n= 0...7; RW)

This register defines which pools are being mirrored to the destination pool.

Registers 0, 4 correspond to rule 0, registers 1, 5 correspond to rule 1, etc. Registers 0-3 correspond to the LSB in the pool list. For example, register 0 corresponds to pools 31:0, while register 4 corresponds to pools 63:32.



| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Pool | 31:0 | 0x0 | Bitmap listing which pools participate in the mirror rule. |

# 8.3 Device Registers — VF

## 8.3.1 Registers Allocated Per Queue

Depending on configuration, each pool has 2, 4, or 8 queues allocated to it. Note that in IOV mode, any queues not allocated to a VF are allocated to the PF. The registers assigned to a queue are accessible both in its VF address space and in the PF address space. This section describes the address mapping of registers that belong to queues.

Section 7.10.2.7.2 defines the correspondence of queue indices between the PF and the VFs. For example, when in configuration for 32 VFs, queues 124-127 in the PF correspond to queues [3:0] of VF# 31.

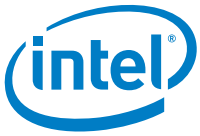
The queues are enumerated in each VF from 0 (such as [1:0], [3:0], or [7:0]). If a queue is allocated to a VF, then its corresponding registers are accessible in the VF CSR space. Each register is allocated an address in the VF (relative to its base) according to its index in the VF space. Therefore, 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).

## 8.3.2 Non-Queue 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 read/write by the VF (like CTRL) are replicated in the PF, one per VF, in adjacent addresses.

**Note:** Since the VF address space is limited to 16 KB, any register that resides above that address in the PF space cannot reside in the same address in the VF space and is therefore allocated in another location in the VF.



## 8.3.3 MSI—X Register Summary VF — BAR 3

| Virtual Address | Physical Address Base (+ VFn *0x30) | Abbreviation | Name |
|---|---|---|---|
| 0x0000 + n*0x10, n=0...2 | 0x00010 | MSIXTADD | MSIX Table Entry Lower Address |
| 0x0004 + n*0x10, n=0...2 | 0x00018 | MSIXTUADD | MSIX Table Entry upper Address |
| 0x0008 + n*0x10, n=0...2 | 0x00028 | MSIXTMSG | MSIX Table Entry Message |
| 0x000C + n*0x10, n=0...2 | N/A | MSIXTVCTRL | MSIX Table Vector Control |
| Max(Page Size, 0x2000) | N/A | MSIXPBA | MSI-X Pending Bit Array |

### 8.3.3.1 MSI—X Table Entry Lower Address — MSIXTADD (BAR3: 0x0000 + n*0x10, n=0...2; RW)

See Section 9.3.8.2 for details of this register.

### 8.3.3.2 MSI—X Table Entry Upper Address — MSIXTUADD (BAR3: 0x0004 + n*0x10, n=0...2; RW)

See Section 9.3.8.2 for details of this register.

### 8.3.3.3 MSI—X Table Entry Message — MSIXTMSG (BAR3: 0x0008 + n*0x10, n=0...2; RW)

See Section 9.3.8.2 for details of this register.

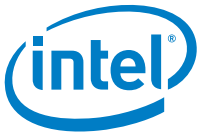
### 8.3.3.4 MSI—X Table Entry Vector Control — MSIXVCTRL (BAR3: 0x000C + n*0x10, n=0...2; RW)

See Section 9.3.8.2 for details of this register.

### 8.3.3.5 MSIXPBA (BAR3: 0x02000; RO) — MSIXPBA Bit Description

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Pending Bits | 2:0 | 0x0 | For each pending bit that is set, the function has a pending message for the associated MSI-X table entry.<br>Pending bits that have no associated MSI-X table entry are reserved. |
| Reserved | 31:3 | 0x0 | Reserved |

**Note:** If a page size larger than 8 KB is programmed in the IOV structure, the address of the MSIX PBA table moves to be page aligned.

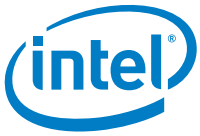


# 8.3.4 Registers Summary VF — BAR 0

## 8.3.4.1 VF Registers Table

| Virtual Address | Abbreviation | Name | Block | Reset Source | RW |
|---|---|---|---|---|---|
| **General Control Registers** | | | | | |
| 0x00000 | VFCTRL | VF Control Register | Target | | WO |
| 0x00008 | VFSTATUS | VF Status Register | Target | | RO |
| 0x00010 | VFLINKS | VF Link Status Register | MAC | | RO |
| 0x00048 | VFFRTIMER | VF Free Running Timer | Rx-Filter | | RO |
| 0x002FC | VFMailbox | VF Mailbox | Target | | RW |
| 0x00200 + 4*n, n=0...15 | VFMBMEM[n] | VF Mailbox Memory | Target | | RW |
| 0x03190 | VFRXMEMWRAP | VF Rx Packet Buffer Flush Detect | DBU-Rx | | RO |
| **Interrupt Registers** | | | | | |
| 0x00100 | VFEICR | VF Extended Interrupt Cause | Interrupt | | RC/W1C |
| 0x00104 | VFEICS | VF Extended Interrupt Cause Set | Interrupt | | WO |
| 0x00108 | VFEIMS | VF Extended Interrupt Mask Set/Read | Interrupt | | RWS |
| 0x0010C | VFEIMC | VF Extended Interrupt Mask Clear | Interrupt | | WO |
| 0x00110 | VFEIAC | VF Extended Interrupt Auto Clear | Interrupt | | RW |
| 0x00114 | VFEIAM | VF Extended Interrupt Auto Mask Enable | Interrupt | | RW |
| 0x00820 + 4*n, n=0...1 | VFEITR | VF Extended Interrupt Mask Set/Read | Interrupt | | RWS |
| 0x00120 + 4*n, n=0...3 | VFIVAR | VF Interrupt Vector Allocation Registers | Interrupt | | RW |
| 0x00140 | VFIVAR_MISC | VF Interrupt Vector Allocation Registers | Interrupt | | RW |
| 0x00180 + 4*n, n=0,1 | VFRSCINT | VF RSC Enable Interrupt | Interrupt | | RW |
| 0x00148 | VFPBACL | VF MSI—X PBA Clear | PCIe | | RW1C |
| **Receive DMA Registers** | | | | | |
| 0x01000 + 0x40*n, n=0...7 | VFRDBAL | VF Receive Descriptor Base Address Low | DMA-Rx | | RW |

| Virtual Address | Abbreviation | Name | Block | Reset Source | RW |
|---|---|---|---|---|---|
| 0x01004 + 0x40*n, n=0...7 | VFRDBAH | VF Receive Descriptor Base Address High | DMA-Rx | | RW |
| 0x01008 + 0x40*n, n=0...7 | VFRDLEN | VF Receive Descriptor Ring Length | DMA-Rx | | RW |
| 0x01010 + 0x40*n, n=0...7 | VFRDH | VF Receive Descriptor Head | DMA-Rx | | RO |
| 0x01018 + 0x40*n, n=0...7 | VFRDT | VF Receive Descriptor Tail | DMA-Rx | | RW |
| 0x01028 + 0x40*n, n=0...7 | VFRXDCTL | VF Receive Descriptor Control | DMA-Rx | | RW |
| 0x01014 + 0x40*n, n=0...7 | VFSRRCTL | VF Split and Replication Receive Control Register queue | DMA-Rx | | RW |
| 0x00300 | VFPSRTYPE | VF Replication Packet Split Receive Type | DBU-Rx | | RW |
| 0x0102C + 0x40*n, n=0...7 | VFRSCCTL | VF RSC Control | DMA-Rx | | RW |
| **Transmit DMA Registers** | | | | | |
| 0x02000 + 0x40*n, n=0...7 | VFTDBAL | VF Transmit Descriptor Base Address Low | DMA-Tx | | RW |
| 0x02004 + 0x40*n, n=0...7 | VFTDBAH | VF Transmit Descriptor Base Address High | DMA-Tx | | RW |
| 0x02008 + 0x40*n, n=0...7 | VFTDLEN | VF Transmit Descriptor Ring Length | DMA-Tx | | RW |
| 0x02010 + 0x40*n, n=0...7 | VFTDH | VF Transmit Descriptor Head | DMA-Tx | | RO |
| 0x02018 + 0x40*n, n=0...7 | VFTDT | VF Transmit Descriptor Tail | DMA-Tx | | RW |
| 0x02028 + 0x40*n, n=0...7 | VFTXDCTL | VF Transmit Descriptor Control | DMA-Tx | | RW |
| 0x02038 + 0x40*n, n=0...7 | VFTDWBAL | VF Tx Descriptor Completion Write-Back Address Low | DMA-Tx | | RW |
| 0x0203C + 0x40*n, n=0...7 | VFTDWBAH | VF Tx Descriptor Completion Write-Back Address High | DMA-Tx | | RW |
| **DCA Registers** | | | | | |
| 0x0100C + 0x40*n, n=0...7 | VFDCA_RXCTRL | VF Rx DCA Control Registers | DMA-Rx | | RW |
| 0x0200C + 0x40*n, n=0...7 | VFDCA_TXCTRL | VF Tx DCA Control Registers | DMA-Tx | | RW |
| **Statistic Register** | | | | | |
| 0x0101C | VFGPRC | VF Good Packets Received Count | DMA-Rx | | RO |
| 0x0201C | VFGPTC | VF Good Packets Transmitted Count | STAT | | RO |
| 0x01020 | VFGORC_LSB | VF Good Octets Received Count Low | DMA-Rx | | RO |
| 0x01024 | VFGORC_MSB | VF Good Octets Received Count High | DMA-Rx | | RO |



| Virtual Address | Abbreviation | Name | Block | Reset Source | RW |
|---|---|---|---|---|---|
| 0x02020 | VFGOTC_LSB | VF Good Octets Transmitted Count Low | STAT | | RO |
| 0x02024 | VFGOTC_MSB | VF Good Octets Transmitted Count High | STAT | | RO |
| 0x01034 | VFMPRC | VF Multicast Packets Received Count | DMA-Rx | | RO |

# 8.3.5 Detailed Register Descriptions —VF

All the registers in this section are replicated per VF. The addresses are relative to the beginning of each VF address space. The address relative to BAR0 as programmed in the IOV structure in the PF configuration space (offset 0x180-0x184) can be found by the following formula:

```
VF BAR0 + Max(16K, system page size)* VF# + CSR offset.
```

## 8.3.5.1 General Control Registers —VF

### 8.3.5.1.1 VF Control Register — VFCTRL (0x00000; WO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Reserved | 25:0 | 0x0 | Reserved. |
| RST | 26 | 0b | VF Reset.<br>This bit performs a reset of the queue enable and the interrupt registers of the VF. |
| Reserved | 31:27 | 0x0 | Reserved |

### 8.3.5.1.2 VF Status Register — VFSTATUS (0x00008; R)

This register is a mirror of the PF status register. See Section 8.2.3.1.2 for details of this register.

### 8.3.5.1.3 VF Link Status Register — VFLINKS (0x00010; RO)

This register is a mirror of the PF LINKS register. See Section 8.2.3.22.20 for details of this register.

#### 8.3.5.1.4 VF Free Running Timer — VFFRTIMER (0x00048; RO)

This register mirrors the value of a free running timer register in the PF — RTFRTIMER. The register is reset by a PCI reset and/or software reset. This register is a mirror of the PF register.

#### 8.3.5.1.5 VF Mailbox — VFMailbox (0x002FC; RW)

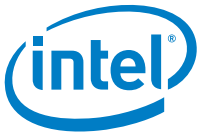
| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Req (WO) | 0 | 0b | Request for PF ready.<br>Setting this bit, causes an interrupt to the PF. This bit always reads as zero. Setting this bit sets the corresponding bit in *VFREQ* field in PFMBICR register. |
| Ack (WO) | 1 | 0b | PF message received.<br>Setting this bit, causes an interrupt to the PF. This bit always reads as zero. Setting this bit sets the corresponding bit in *VFACK* field in PFMBICR register. |
| VFU | 2 | 0b | Buffer is taken by VF.<br>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 | Buffer is taken by PF.<br>This bit is RO for the VF and is a mirror of the *PFU* bit of the PFMailbox register. |
| PFSTS (RC) | 4 | 0b | PF wrote a message in the mailbox. |
| PFACK (RC) | 5 | 0b | PF acknowledged the VF previous message. |
| RSTI (RO) | 6 | 1b | Indicates that the PF had reset the shared resources and the reset sequence is in progress. |
| RSTD (RC) | 7 | 0b | Indicates that a PF software reset completed.<br>This bit is cleared on read. |
| Reserved | 31:8 | 0x0 | Reserved. |

**Note:** *VFLR* won't clear the VFMAILBOX.VFU bit. This bit should be cleared by a direct write access or by setting PFMailbox.RVFU bit.

#### 8.3.5.1.6 VF Mailbox Memory — VFMBMEM (0x00200 + 4*n, n=0...15; RW)

Mailbox memory for PF and VF drivers communication. The mailbox size for each VM is 64 bytes accessed by32-bit registers. Locations can be accessed as 32-bit or 64-bit words.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| Mailbox Data | 31:0 | X | *Mailbox Data* field composed of 16 x 4 byte registers. |



#### 8.3.5.1.7 VF Rx Packet Buffer Flush Detect — VFRXMEMWRAP (0x03190; RO)

This register mirrors the PF RXMEMWRAP described in Section 8.2.3.8.11.

### 8.3.5.2 Interrupt Registers — VF

#### 8.3.5.2.1 VF Extended Interrupt Cause — VFEICR (0x00100; RC/W1C)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MSIX | 2:0 | 0x0 | Indicates an interrupt cause mapped to MSI-X vectors 2:0. |
| Reserved | 31:3 | 0x0 | Reserved |

#### 8.3.5.2.2 VF Extended Interrupt Cause Set — VFEICS (0x00104; WO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MSIX | 2:0 | 0x0 | Sets to corresponding *EICR* bit of MSI-X vectors 2:0. |
| Reserved | 31:3 | 0x0 | Reserved |

#### 8.3.5.2.3 VF Extended Interrupt Mask Set/Read — VFEIMS (0x00108; RWS)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MSIX | 2:0 | 0x0 | *Set Mask* bit for the corresponding *EICR* bit of MSI-X vectors 2:0. |
| Reserved | 31:3 | 0x0 | Reserved |

#### 8.3.5.2.4 VF Extended Interrupt Mask Clear — VFEIMC (0x0010C; WO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MSIX | 2:0 | 0x0 | *Clear Mask* bit for the corresponding *EICR* bit of MSI-X vectors 2:0. |
| Reserved | 31:3 | 0x0 | Reserved |

#### 8.3.5.2.5 VF Extended Interrupt Auto Mask Enable — VFEIAM (0x00114; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MSIX | 2:0 | 0x0 | *Auto Mask* bit for the corresponding *EICR* bit of MSI-X vectors 2:0. |
| Reserved | 31:3 | 0x0 | Reserved |

#### 8.3.5.2.6 VF Extended Interrupt Mask Set/Read — VFEITR[n] (0x00820 + 4*n, n=0...1; RWS)

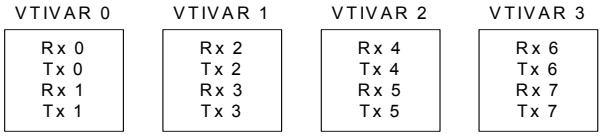
See register description in Section 8.2.3.5.12.

#### 8.3.5.2.7 VF Interrupt Vector Allocation Registers — VFIVAR[n] (0x00120 + 4*n, n=0...3; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| INT_Alloc[0] | 0 | X | Defines the MSI-X vector (0 or 1) assigned to Rx queue ‘2*N’ for IVAR ‘N’ register (N=0...3). |
| reserved | 6:1 | 0x0 | Reserved. |
| INT_Alloc_val[0] | 7 | 0b | Valid bit for INT_Alloc[0]. |
| INT_Alloc[1] | 8 | X | 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 | Reserved. |
| INT_Alloc_val[1] | 15 | 0b | Valid bit for INT_Alloc[1]. |
| INT_Alloc[2] | 16 | X | 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 | Reserved. |
| INT_Alloc_val[2] | 23 | 0b | Valid bit for INT_Alloc[2]. |
| INT_Alloc[3] | 24 | X | 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 | Reserved. |
| INT_Alloc_val[3] | 31 | 0b | Valid bit for INT_Alloc[3]. |

These registers map interrupt causes into MSI-X vectors. See additional details in Section 7.3.4.

Transmit and receive queues mapping to VFIVAR registers is as follows:




### 8.3.5.2.8 VF Interrupt Vector Allocation Registers — VFIVAR_MISC (0x00140; RW)

This register maps the mailbox interrupt into an MSI-X vector. See additional details in Section 7.3.4.

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| INT_Alloc[0] | 1:0 | X | Defines the MSI-X vector assigned to the mailbox interrupt. |
| Reserved | 6:2 | 0x0 | Reserved. |
| INT_Alloc_val[0] | 7 | 0b | Valid bit for INT_Alloc[0]. |
| Reserved | 31:8 | 0x0 | Reserved. |

### 8.3.5.2.9 VF RSC Enable Interrupt — VFRSCINT[n] (0x00180 + 4*n, n=0,1; RW)

See register description in Section 8.2.3.5.12.

### 8.3.5.2.10 VF MSI—X PBA Clear — VFPBACL (0x00148; RW1C)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PENBIT | 2:0 | 000b | MSI-X Pending Bits Clear.<br>Writing a 1b to any bit clears the corresponding MSIXPBA bit; writing a 0x0 has no effect.<br>Reading this register returns the PBA vector. |
| Reserved | 31:3 | 0x0 | Reserved. |

## 8.3.5.3 Receive DMA Registers — VF

### 8.3.5.3.1 VF Receive Descriptor Base Address Low — VFRDBAL[n] (0x01000 + 0x40*n, n=0...7; RW)

See RDBAL description in Section 8.2.3.8.1.

### 8.3.5.3.2 VF Receive Descriptor Base Address High — VFRDBAH[n] (0x01004 + 0x40*n, n=0...7; RW)

See RDBAH description in Section 8.2.3.8.2.

### 8.3.5.3.3 VF Receive Descriptor Ring Length — VFRDLEN[n] (0x01008 + 0x40*n, n=0...7; RW)

See RDLEN description in Section 8.2.3.8.3.

### 8.3.5.3.4 VF Receive Descriptor Head — VFRDH[n] (0x01010 + 0x40*n, n=0...7; RO)

See RDH description in Section 8.2.3.8.4.

### 8.3.5.3.5 VF Receive Descriptor Tail — VFRDT[n] (0x01018 + 0x40*n, n=0...7; RW)

See RDT description in Section 8.2.3.8.5.

### 8.3.5.3.6 VF Receive Descriptor Control — VFRXDCTL[n] (0x01028 + 0x40*n, n=0...7; RW)

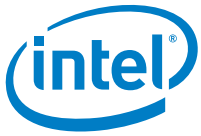
See RXDCTL description in Section 8.2.3.8.6.

### 8.3.5.3.7 VF Split and Replication Receive Control Register queue — VFSRRCTL (0x01014 + 0x40*n, n=0...7; RW)

See SRRCTL description in Section 8.2.3.8.7.

### 8.3.5.3.8 VF Replication Packet Split Receive Type — VFPSRTYPE (0x00300; RW)

See PSRTYPE description in Section 8.2.3.7.4.



#### 8.3.5.3.9 VF RSC Control — VFRSCCTL[n] (0x0102C + 0x40*n, n=0...7; RW)

See RSCCTL description in Section 8.2.3.8.13.

### 8.3.5.4 Transmit Registers — VF

#### 8.3.5.4.1 VF Transmit Descriptor Base Address Low — VFTDBAL[n] (0x02000 + n*0x40, n=0...3; RW)

See TDBAL description in Section 8.2.3.9.5.

#### 8.3.5.4.2 VF Transmit Descriptor Base Address High — VFTDBAH[n] (0x02004 + n*0x40, n=0...3; RW)

See TDBAH description in Section 8.2.3.9.6.

#### 8.3.5.4.3 VF Transmit Descriptor Ring Length — VFTDLEN[n] (0x02008 + n*0x40, n=0...3; RW)

See TDLEN description in Section 8.2.3.9.7.

#### 8.3.5.4.4 VF Transmit Descriptor Head — VFTDH[n] (0x02010 + n*0x40, n=0...3; RO)

See TDH description in Section 8.2.3.9.8.

#### 8.3.5.4.5 VF Transmit Descriptor Tail — VFTDT[n] (0x02018 + n*0x40, n=0...3; RW)

See TDT description in Section 8.2.3.9.9.

#### 8.3.5.4.6 VF Transmit Descriptor Control — VFTXDCTL[n] (0x02028 + n*0x40, n=0...3; RW)

See RSCCTL description in Section 8.2.3.9.10.

#### 8.3.5.4.7 VF Tx Descriptor Completion Write-Back Address Low — VFTDWBAL[n] (0x02038 + n*0x40, n=0...3; RW)

See RSCCTL description in Section 8.2.3.9.11.

#### 8.3.5.4.8 VF Tx Descriptor Completion Write-Back Address High — VFTDWBAH[n] (0x0203C + n*0x40, n=0...3;RW)

See RSCCTL description in Section 8.2.3.9.12.

### 8.3.5.5 DCA Registers — VF

#### 8.3.5.5.1 Rx DCA Control Registers — VFDCA_RXCTRL[n] (0x0100C + 0x40*n, n=0...7; RW)

See DCA_RXCTRL description in Section 8.2.3.11.1.

#### 8.3.5.5.2 Tx DCA Control Registers — VFDCA_TXCTRL[n] (0x0200C + 0x40*n, n=0...7; RW)

See DCA_TXCTRL description in Section 8.2.3.11.2.

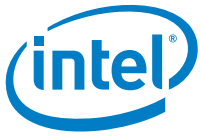
### 8.3.5.6 Statistic Register Descriptions — VF

#### 8.3.5.6.1 VF Good Packets Received Count — VFGPRC (0x0101C; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GPRC | 31:0 | 0x0 | Number of good packets received for this VF (of any length).<br>This counter includes loopback packets or replications of multicast packets.<br>The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000. |

#### 8.3.5.6.2 VF Good Packets Transmitted Count — VFGPTC (0x0201C; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GPTC | 31:0 | 0x0 | Number of good packets sent by the queues allocated to this VF.<br>This counter includes loopback packets or packets latter dropped by the switch or the MAC but does not include packet dropped by anti spoofing or VLAN tag filtering (as described in Section 7.10.3.9.2).<br>The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000. |



### 8.3.5.6.3 VF Good Octets Received Count Low — VFGORC_LSB (0x01020; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GORC-LSB | 31:0 | 0x0 | Number of good octets received (32 LS bits of a 36-bit counter) by the queues allocated to this VF.<br>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 (such as LinkSec octets not being counted).<br>Bytes of RSC are counted before coalescing.<br>The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000. |

### 8.3.5.6.4 VF Good Octets Received Count High — VFGORC_MSB (0x01024; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GORC-MSB | 3:0 | 0x0 | Number of good octets received (4 MS bits of a 36-bit counter) by the queues allocated to this VF.<br>See the complete explanation in Section 8.3.5.6.3.<br>The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xF to 0x0. |

### 8.3.5.6.5 VF Good Octets Transmitted Count — VFGOTC_LSB (0x02020; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GOTC-LSB | 31:0 | 0x0 | Number of good octets transmitted (32 LS bits of a 36-bit counter) by the queues allocated to this VF.<br>This register includes bytes transmitted in a packet from the <Destination Address> field through the <CRC> field, inclusively. This register counts octets of the packets counted by the VFGPTC register. Octets are counted on the VF interface rather than on the network interface (such as LinkSec octets not being counted).<br>The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000. |

#### 8.3.5.6.6 VF Good Octets Transmitted Count — VFGOTC_MSB (0x02024; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| GOTC-MSB | 3:0 | 0x0 | Number of good octets transmitted (4 MS bits of a 36-bit counter) by the queues allocated to this VF.<br>See the complete explanation in Section 8.3.5.6.5.<br>The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xF to 0x0. |
| Reserved | 31:4 | 0x0 | Reserved. |

#### 8.3.5.6.7 VF Multicast Packets Received Count — VFMPRC (0x01034; RO)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| MPRC | 31:0 | 0x0 | Number of multicast good packets received by this VF (of any length) that pass Ethernet MAC address filtering (excluding broadcast packets).<br>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.<br>This counter includes loopback packets or replications of multicast packets.<br>The counter is not cleared on read. Furthermore, the register is a cyclic counter incrementing from 0xFFFF to 0x0000. |

# 9.0 PCIe Programming Interface

## 9.1 PCI Compatibility

PCIe is fully compatible with existing deployed PCI software. To achieve this, PCIe hardware implementations conform to the following requirements:

- All devices are required to be supported by deployed PCI software and must be enumerable as part of a tree-through PCI device enumeration mechanisms.
- Devices must not require any resources such as address decode ranges and interrupts beyond those claimed by PCI resources for operation of software compatible and software transparent features with respect to existing deployed PCI software.
- Devices in their default operating state must confirm to PCI ordering and cache coherency rules from a software viewpoint.
- PCIe devices must conform to the PCI power management specification and must not require any register programming for PCI-compatible power management beyond those available through PCI power management capability registers. Power management is expected to conform to a standard PCI power management by existing PCI bus drivers.

PCIe devices implement all registers required by the PCI specification as well as the power management registers and capability pointers specified by the PCI power management specification. In addition, PCIe defines a PCIe capability pointer to indicate support for PCIe extensions and associated capabilities.

The 82599 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 EEPROM settings as detailed in the following sections):

- Mandatory PCI configuration registers
- Power management capabilities
- MSI / MSI-X capabilities
- Vital Product Data (VPD) capability

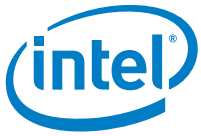
- PCIe extended capabilities:
  - Advanced Error Reporting (AER)
  - Serial ID
  - Alternate requester ID.
  - Single root IOV

## 9.2 Configuration Sharing Among PCI Functions

The 82599 contains a single physical PCIe core interface. It is designed so that each of the logical LAN devices (LAN 0, LAN 1) appears as a distinct function implementing its own PCIe device header space.

Many of the fields of the PCIe header space contain hardware default values that are either fixed or can be overridden using an EEPROM, but might not be independently specified for each logical LAN device. The following fields are considered to be common to both LAN functions:

| | |
|---|---|
| Vendor ID | The Vendor ID of the 82599 is specified to a single value 0x8086.<br>The value is reflected identically for both LAN devices. |
| Revision | The revision number of the 82599 is reflected identically for both LAN devices. |
| Header Type | This field indicates if a device is single function or multi-function. The value reflected in this field is reflected identically for both LAN devices, but the actual value reflected depends on LAN disable configuration.<br>When both the 82599 LAN ports are enabled, both PCIe headers return 0x80 in this field, acknowledging being part of a multi-function device. LAN 0 exists as device function 0, while LAN 1 exists as device function 1.<br>If function 1 is disabled, then only a single-function device is indicated (this field returns a value of 0x00) and the LAN exists as device function 0. |
| Subsystem ID | The Subsystem ID of the 82599 can be specified via an EEPROM, but only a single value can be specified. The value is reflected identically for both LAN devices. |
| Subsystem Vendor ID | The Subsystem Vendor ID of the 82599 can be specified via an EEPROM, but only a single value can be specified. The value is reflected identically for both LAN devices. |
| Cap_Ptr<br>Max Latency<br>Min Grant | These fields reflect fixed values that are constant values reflected for both LAN devices. |



The following fields are implemented as unique to each LAN functions:

| | |
|---|---|
| Device ID | The Device ID reflected for each LAN function can be independently specified via an EEPROM. |
| Command<br>Status | Each LAN function implements its own Command/Status registers. |
| Latency Timer<br>Cache Line Size | Each LAN function implements these registers independently. The system should program these fields identically for each LAN to ensure consistent behavior and performance of each device. |
| Memory BAR, IO BAR<br>Expansion ROM BAR<br>MSIX BAR | Each LAN function implements its own Base Address registers, enabling each device to claim its own address region(s). The I/O BAR is enabled by the *IO_Sup* bit in the EEPROM. |
| Interrupt Pin | Each LAN function independently indicates which interrupt pin (INTA#...INTD#) is used by that device's MAC to signal system interrupts. The value for each LAN device can be independently specified via an EEPROM, but only if both LAN devices are enabled. |
| Class Code | Each function can have its own device class. Function 0 can be dummy function, LAN or storage and Function 1 can be either LAN or storage. Both are enabled by the EEPROM. |

# 9.3 PCIe Register Map

Configuration registers are assigned one of the attributes described in the table that follows.

## 9.3.1 Register Attributes

The following table lists the register attributes used in this section.

| RD/WR | Description |
|---|---|
| RO | Read-only register: Register bits are read-only and cannot be altered by software. |
| RW | Read-write register: Register bits are read-write and can be either set or reset. |
| RW1C | Read-only status, Write-1b-to-clear status register, Writing a 0b to RW1C bits has no effect. |
| ROS | Read-only register with sticky bits: Register bits are read-only and cannot be altered by software. Bits are not cleared by reset and can only be reset with the PWRGOOD signal. Devices that consume AUX power are not allowed to reset sticky bits when AUX power consumption (either via AUX power or PME Enable) is enabled. |
| RWS | Read-write register: Register bits are read-write and can be either set or reset by software to the desired state. Bits are not cleared by reset and can only be reset with the PWRGOOD signal. Devices that consume AUX power are not allowed to reset sticky bits when AUX power consumption (either via AUX power or PME Enable) is enabled. |
| 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 EEPROM. Bits are read-only after initialization and can only be reset (for write-once by firmware) with the PWRGOOD signal. |
| RsvdP | Reserved and preserved: Reserved for future read/write implementations; software must preserve value read for writes to these bits. |
| RsvdZ | Reserved and zero: Reserved for future RW1C implementations; software must use 0b for writes to these bits. |

## 9.3.2 PCIe Configuration Space Summary

Table 9-1lists the PCIe configuration registers while their detailed description is given in the sections that follow. PCI configuration fields in the summary 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 EEPROM)
- Magenta fields indicate hard-coded values.

- Other fields contain RW attributes.

## Table 9-1 PCI Configuration Space

| Section | Byte Offset | Byte 3 | Byte 2 | Byte 1 | Byte 0 |
|---|---|---|---|---|---|
| Mandatory PCI Register | 0x0 | Device ID | | Vendor ID | |
| | 0x4 | Status Register | | Control Register | |
| | 0x8 | Class Code (0x020000/0x010000) | | | Revision ID |
| | 0xC | Reserved | Header Type (0x0/ 0x80) | Latency Timer | Cache Line Size (0x10) |
| | 0x10 | Base Address Register 0 | | | |
| | 0x14 | Base Address Register 1 | | | |
| | 0x18 | Base Address Register 2 | | | |
| | 0x1C | Base Address Register 3 | | | |
| | 0x20 | Base Address Register 4 | | | |
| | 0x24 | Base Address Register 5 | | | |
| | 0x28 | CardBus CIS pointer (0x0000) | | | |
| | 0x2C | Subsystem ID | | Subsystem Vendor ID | |
| | 0x30 | Expansion ROM Base Address | | | |
| | 0x34 | Reserved | | | Cap Ptr (0x40) |
| | 0x38 | Reserved | | | |
| | 0x3C | Max Latency (0x00) | Min Grant (0x00) | Interrupt Pin (0x01...0x04) | Interrupt Line (0x00) |
| PCI / PCIe Capabilities | 0x40...0x47 | Power management capability | | | |
| | 0x50...0x67 | MSI Capability | | | |
| | 0x70...0x7B | MSI-X Capability | | | |
| | 0xA0...0xDB | PCIe Capability | | | |
| | 0xE0...0xE7 | VPD Capability | | | |
| Extended PCIe Configuration | 0x100...0x12B | AER Capability | | | |
| | 0z140...0x14B | Serial ID Capability | | | |
| | 0x150...0x157 | ARI Capability | | | |
| | 0x160...0x19C | SR-IOV Capability | | | |

## 9.3.3 Mandatory PCI Configuration Registers — Except BARs

### 9.3.3.1 Vendor ID Register (0x0; RO)

This is a read-only register that has the same value for all PCI functions. It identifies unique Intel products.

### 9.3.3.2 Device ID Register (0x2; RO)

This is a read-only register that identifies individual the 82599 PCI functions. Both ports have the same default value equals to 0x10D8, and can be auto-loaded from the EEPROM during initialization with different values for each port as well as the dummy function (See Section 4.4 for dummy function relevance).

### 9.3.3.3 Command Register (0x4; RW)

Shaded bits are not used by this implementation and are hardwired to 0b. Each function has its own Command register. Unless explicitly specified, functionality is the same in both functions.

| Bit(s) | Init Val | Description |
|---|---|---|
| 0 | 0b | I/O Access Enable. |
| 1 | 0b | Memory Access Enable. |
| 2 | 0b | Enable Mastering, also named Bus Master Enable (BME)).<br>• LAN functions RW field<br>• Dummy function RO as zero field |
| 3 | 0b | Special Cycle Monitoring – Hardwire to 0b. |
| 4 | 0b | MWI Enable – Hardwire to 0b. |
| 5 | 0b | Palette Snoop Enable – Hardwire to 0b. |
| 6 | 0b | Parity Error Response. |
| 7 | 0b | Wait Cycle Enable – Hardwired to 0b. |
| 8 | 0b | SERR# Enable. |
| 9 | 0b | Fast Back-to-Back Enable – Hardwire to 0b. |
| 10 | 1b | Interrupt Disable.<br>When set, devices are prevented from generating legacy interrupt messages. |
| 15:11 | 0b | Reserved. |

### 9.3.3.4 Status Register (0x6; RO)

Shaded bits are not used by this implementation and are hardwired to 0b. Each function has its own Status register. Unless explicitly specified, functionality is the same in both functions.

| Bits | Init Val | RW | Description |
|---|---|---|---|
| 2:0 | 0b | | Reserved. |
| 3 | 0b | RO | Interrupt Status.[1] |
| 4 | 1b | RO | New Capabilities.<br>Indicates that a device implements extended capabilities. The 82599 sets this bit and implements a capabilities list to indicate that it supports PCI Power Management, Message Signaled Interrupts (MSI), Enhanced Message Signaled Interrupts (MSI-X), VPD and the PCIe extensions. |
| 5 | 0b | | 66 MHz Capable – Hard wire to 0b. |
| 6 | 0b | | Reserved. |
| 7 | 0b | | Fast Back-to-Back Capable – Hard wire to 0b. |
| 8 | 0b | RW1C | Data Parity Reported. |
| 10:9 | 00b | | DEVSEL Timing – Hard wire to 0b. |
| 11 | 0b | RW1C | Signaled Target Abort. |
| 12 | 0b | RW1C | Received Target Abort. |
| 13 | 0b | RW1C | Received Master Abort. |
| 14 | 0b | RW1C | Signaled System Error. |
| 15 | 0b | RW1C | Detected Parity Error. |

1. The *Interrupt Status* field is a RO field that indicates that an interrupt message is pending internally to the device.

### 9.3.3.5 Revision Register (0x8; RO)

The default revision ID of this device is 0x00. The value of the rev ID is a logic XOR between the default value and the value in EEPROM word 0x1D. Note that LAN 0 and LAN 1 functions have the same revision ID.

### 9.3.3.6 Class Code Register (0x9; RO)

The class code is a read-only value that identifies the device functionality according to the value of the *Storage Class* bit in the EEPROM PCIe Configuration (Offset 0x01).

- Class Code = 0x020000 (Ethernet Adapter) if EEPROM->*Storage Class* = 0b
- Class Code = 0x010000 (SCSI Storage device) if EEPROM->*Storage Class* = 1b

In the dummy function the class code equals to 0xFF0000.

### 9.3.3.7 Cache Line Size Register (0xC; RW)

This field is implemented by PCIe devices as a read/write field for legacy compatibility purposes but has no impact on any PCIe device functionality. Loaded from the EEPROM. All functions are initialized to the same value.

### 9.3.3.8 Latency Timer (0xD; RO), Not Supported

Not used. Hard wire to 0b.

### 9.3.3.9 Header Type Register (0xE; RO)

This indicates if a device is single- or multi-function. If a single LAN function is the only active one then this field has a value of 0x00 to indicate a single function device. If other functions are enabled then this field has a value of 0x80 to indicate a multi-function device. Table 9-2 lists the different options to set the header type field.

**Table 9-2 Header Type Settings**

| Lan 0 Enable | Lan 1 Enable | Cross-Mode Enable | Dummy Function Enable | Header Type Expected Value |
|---|---|---|---|---|
| 0 | 0 | X | X | N/A (no function) |
| 1 | 0 | 0 | X | 0x00 |
| 0 | 1 | 0 | 0 | 0x00 |
| 0 | 1 | 0 | 1 | 0x80 (dummy exist) |
| 1 | 1 | X | X | 0x80 (dual function) |
| 1 | 0 | 1 | 0 | 0x00 |
| 1 | 0 | 1 | 1 | 0x80 (dummy exist) |
| 0 | 1 | 1 | X | 0x00 |

### 9.3.3.10 Subsystem Vendor ID Register (0x2C; RO)

This value can be loaded automatically from the EEPROM at power up or reset. A value of 0x8086 is the default for this field at power up if the EEPROM does not respond or is not programmed. All functions are initialized to the same value.

## 9.3.4 Subsystem ID Register (0x2E; RO)

This value can be loaded automatically from the EEPROM at power up with a default value of 0x0000.

| PCI Function | Default Value | EEPROM Address |
|---|---|---|
| LAN Functions | 0x0000 | 0x0B |

## 9.3.5 Cap_Ptr Register (0x34; RO)

The *Capabilities Pointer* field (Cap_Ptr) is an 8-bit field that provides an offset in the 82599's PCI configuration space for the location of the first item in the capabilities linked list. The 82599 sets this bit and implements a capabilities list to indicate that it supports PCI power management, MSIs, and PCIe extended capabilities. Its value is 0x40, which is the address of the first entry: PCI power management.

### 9.3.5.1 Interrupt Line Register (0x3C; RO)

Read/write register programmed by software to indicate which of the system interrupt request lines the 82599's interrupt pin is bound to. Refer to the PCI definition for more details. Each PCI function has its own register.

Max_Lat/Min_Gnt not used. Hard wired to 0b.

### 9.3.5.2 Interrupt Pin Register (0x3D; RO)

Read-only register. LAN 0 / LAN 1[1] — A value of 0x1...0x4 indicates that this function implements a legacy interrupt on INTA#...INTD# respectively. Loaded from the EEPROM word offset 0x01 in the EEPROM PCIe Configuration Space per function. Refer to the following detailed explanation for cases in which any of the LAN port(s) are disabled.

1. If only a single device/function of the 82599 component is enabled, this value is ignored and the *Interrupt Pin* field of the enabled device reports INTA# usage.

# 9.3.6 Mandatory PCI Configuration Registers — BARs

## 9.3.6.1 Memory and IO Base Address Registers (0x10...0x27; RW)

Base Address Registers (BARs) are used to map the 82599 register space of the device functions. The 82599 has a memory BAR, I/O BAR and MSI-X BAR described in Table 9-3. The BARs location and sizes are described in the Table 9-3 and Table 9-4. The fields within each BAR are then described in Table 9-4.

**Table 9-3 the 82599 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. |
| 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 EEPROM at word PCIe Control 3 – Offset 0x07. |
| MSI-X BAR | The MSI-X vectors and Pending Bit Array (PBA) structures are accessed as direct memory mapped offsets from the MSI-X BAR. Software can access Dword entities. |

**Table 9-4 Base Address Registers' Fields**

| Field | Bits | RW | Description | |
|---|---|---|---|---|
| Memory and I/O Space Indication | 0 | RO | 0b = Indicates memory space.<br>1b = Indicates I/O. | |
| Memory Type | 2:1 | RO | 10b = 64-bit BAR | |
| Prefetch Memory | 3 | R | 0b = Non-prefetchable space.<br>1b = Prefetchable space.<br>This bit should be set only on systems that do not generate prefetchable cycles.<br>This bit is loaded from the *PREFBAR* bit in the NVM. | |
| Address Space (low register for 64-bit memory BARs) | 31:4 | RW | The length of the RW bits and RO 0b bits depend on the mapping window sizes. Initial value of the RW fields is 0x0. | |
| | | | Mapping Window | RO bits |
| | | | MSI-X space is 16 KB. | 13:4 |
| | | | I/O space size is 32 bytes (32-bit BAR). | 4:0 |
| | | | Memory CSR + Flash BAR size depends on EEPROM PCIe Control 3 word, Flash_Size and CSR_Size fields. | 16:4 for 128 KB<br>17:4 for 256 KB<br>and so on... |

### 9.3.6.2 Expansion ROM Base Address Register (0x30; RW)

This register is used to define the address and size information for boot-time access to the optional Flash memory. It is enabled by EEPROM words 0x24 and 0x14 for LAN 0 and LAN 1, respectively. This register returns a zero value for functions without an expansion ROM window.

| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| En | 0 | RW | 0b | 0b = Disables expansion ROM access.<br>1b = Enables 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.<br>The LAN Expansion ROM spaces can be either 64 KB or up to 8 MB in powers of 2. Mapping window size is set by EEPROM word 0x0F. |

## 9.3.7 PCIe Capabilities

The first entry of the PCI capabilities link list is pointed to by the Cap_Ptr register. Table 9-5 lists the capabilities supported by the 82599.

**Table 9-5 PCI Capabilities List**

| 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.

### 9.3.7.1 PCI Power Management Capability

All fields are reset at full power up. All fields except PME_En and PME_Status are reset after exiting from the D3cold state. If AUX power is not supplied, the PME_En and PME_Status fields also reset after exiting from the D3cold state. Refer to the detailed description for registers loaded from the EEPROM at initialization.

| Byte Offset | Byte 3 | Byte 2 | Byte 1 | Byte 0 |
|---|---|---|---|---|
| 0x40 | Power Management Capabilities | | Next Pointer (0x50 / 0xA0) | Capability ID (0x01) |
| 0x44 | Data | Bridge Support Extensions | Power Management Control & Status | |

#### 9.3.7.1.1 Capability ID Register (0x40; RO)

This field equals 0x01 indicating the linked list item as being the PCI Power Management registers.

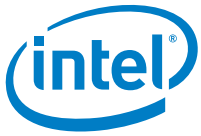
#### 9.3.7.1.2 Next Pointer Register (0x41; RO)

This field provides an offset to the next capability item in the capability list. This field equals for both LAN ports to 0x50 pointing to the MSI capability. In dummy function, it equals to 0xA0 pointing to the PCIe Capabilities.

#### 9.3.7.1.3 Power Management Capabilities — PMC Register (0x42; RO)

This field describes the device functionality during the power management states as listed in the following table. Note that each device function has its own register.

| Bits | RW | Default | Description |
|---|---|---|---|
| 15:11 | RO | 01001b | PME_Support.<br>This 5-bit field indicates the power states in which the function can assert PME#.<br>Condition Functionality Values:<br>• No AUX Pwr PME at D0 and D3hot = 01001b<br>• AUX Pwr PME at D0, D3hot, and D3cold = 11001b |
| 10 | RO | 0b | D2_Support.<br>The 82599 does not support the D2 state. |
| 9 | RO | 0b | D1_Support.<br>The 82599 does not support the D1 state. |
| 8:6 | RO | 000b | AUX Current.<br>Required current defined in the Data register. |
| 5 | RO | 1b | DSI.<br>the 82599 requires its device driver to be executed following a transition to the D0 un-initialized state. |
| 4 | RO | 0b | Reserved. |
| 3 | RO | 0b | PME_Clock<br>Disabled. Hard wire to 0b. |
| 2:0 | RO | 011b | Version.<br>The 82599 complies with the PCI PM specification revision 1.2. |



### 9.3.7.1.4 Power Management Control / Status Register — PMCSR (0x44; RW)

This register (shown in the following table) is used to control and monitor power management events in the device. Note that each device function has its own PMCSR.

| Bits | RW | Default | Description |
|---|---|---|---|
| 15 | RW1CS | 0b at power up | PME_Status.<br>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 | RO | 01b | Data_Scale.<br>This field indicates the scaling factor that's used when interpreting the value of the Data register.<br>This field equals 01b (indicating 0.1 watt/units) and the Data_Select field is set to 0, 3, 4, 7, (or 8 for function 0). Otherwise, it equals 00b. |
| 12:9 | RW | 0000b | Data_Select.<br>This 4-bit field is used to select which data is to be reported through the Data register and *Data_Scale* field. These bits are writeable only when power management is enabled via the EEPROM. |
| 8 | RWS | 0b at power up | PME_En.<br>Writing a 1b to this register enables Wakeup. |
| 7:4 | RO | 0000b | Reserved. |
| 3 | RO | 0b | No_Soft_Reset.<br>This bit is always set to 0b to indicate that the 82599 performs an internal reset upon transitioning from D3hot to D0 via software control of the *PowerState* bits. Configuration context is lost when performing the soft reset. Upon transition from the D3hot to the D0 state, a full re-initialization sequence is needed to return the 82599 to the D0 Initialized state. |
| 2 | RO | 0b | Reserved for PCIe. |
| 1:0 | RW | 00b | PowerState.<br>This field is used to set and report the power state of a function as follows:<br>00b = D0<br>01b = D1 (cycle ignored if written with this value).<br>10b = D2 (cycle ignored if written with this value).<br>11b = D3 |

### 9.3.7.1.5 PMCSR_BSE Bridge Support Extensions Register (0x46; RO)

This register is not implemented in the 82599; values set to 0x00.

#### 9.3.7.1.6 Data Register (0x47; RO)

This optional register is used to report power consumption and heat dissipation. The reported register is controlled by the *Data_Select* field in the PMCSR; the power scale is reported in the *Data_Scale* field in the PMCSR. The data for this field is loaded from the EEPROM if power management is enabled in the EEPROM or with a default value of 0x00. The values for the 82599's functions are as follows:

| Function | D0 (Consume/ Dissipate) | D3 (Consume/ Dissipate) | Common | Data_Scale/ Data_Select |
|---|---|---|---|---|
| | (0x0/0x4) | (0x3/0x7) | (0x8) | |
| 0 | EEP PCIe Control Word at offset 0x06 | EEP PCIe Control Word at offset 0x06 | Multi-function option:<br>EEP PCIe Control Word at offset 0x06<br>Single-function option:<br>0x00 | 01b |
| 1 | EEP PCIe Control Word at offset 0x06 | EEP PCIe Control Word at offset 0x06 | 0x00 | 01b |

**Note:** For other Data_Select values the Data register output is reserved (0b).

### 9.3.7.2 MSI Capability

This structure is required for PCIe devices.

| Byte Offset | Byte 3 | Byte 2 | Byte 1 | Byte 0 |
|---|---|---|---|---|
| 0x50 | Message Control (0x0080) | | Next Pointer (0x70) | Capability ID (0x05) |
| 0x54 | Message Address | | | |
| 0x58 | Message Upper Address | | | |
| 0x5C | Reserved | | Message Data | |
| 0x60 | Mask Bits | | | |
| 0x64 | Pending Bits | | | |

#### 9.3.7.2.1 Capability ID Register (0x50; RO)

This field equals 0x05 indicating that the linked list item as being the MSI registers.

#### 9.3.7.2.2 Next Pointer Register (0x51; RO)

This field provides an offset to the next capability item in the capability list. Its value of 0x70 and points to MSI-X capability.

### 9.3.7.2.3 Message Control Register (0x52; RW)

These register fields are listed in the following table. Note that there is a dedicated register (per PCI function) to separately enable its MSI.

| Bits | RW | Default | Description |
|---|---|---|---|
| 0 | RW | 0b | MSI Enable.<br>1b = Message Signaled Interrupts. The 82599 generates an MSI for interrupt assertion instead of INTx signaling. |
| 3:1 | RO | 000b | Multiple Messages Capable.<br>indicates a single requested message per function. |
| 6:4 | RO | 000b | Multiple Message Enable.<br>returns 000b to indicate that it supports a single message per function. |
| 7 | RO | 1b | 64-bit Capable.<br>A value of 1b indicates that the 82599 is capable of generating 64-bit message addresses. |
| 8 | RO | 1b* | MSI per-vector masking.<br>0b = Indicates that the 82599 is not capable of per-vector masking.<br>1b = Indicates that the 82599 is capable of per-vector masking.<br>*Note:* The value is loaded from the *MSI Mask* bit in the EEPROM. |
| 15:9 | RO | 0b | Reserved.<br>Reads as 0b |

### 9.3.7.2.4 Message Address Low Register (0x54; RW)

Written by the system to indicate the lower 32 bits of the address to use for the MSI memory write transaction. The lower two bits always return 0b regardless of the write operation.

### 9.3.7.2.5 Message Address High Register (0x58; RW)

Written by the system to indicate the upper 32 bits of the address to use for the MSI memory write transaction.

### 9.3.7.2.6 Message Data Register (0x5C; RW)

Written by the system to indicate the lower 16 bits of the data written in the MSI memory write Dword transaction. The upper 16 bits of the transaction are written as 0b.

### 9.3.7.2.7 Mask Bits Register (0x60; RW)

The Mask Bits and Pending Bits registers enable software to disable or defer message sending on a per-vector basis. As the 82599 supports only one message, only bit 0 of these registers are implemented.

| Bits | RW | Default | Description |
|---|---|---|---|
| 0 | RW | 0b | MSI Vector 0 Mask.<br>If set, the 82599 is prohibited from sending MSI messages. |
| 31:1 | RO | 0x0 | Reserved. |

### 9.3.7.2.8 Pending Bits Register (0x64; RW)

| Bits | RW | Default | Description |
|---|---|---|---|
| 0 | RO | 0b | If set, the 82599 has a pending MSI message. |
| 31:1 | RO | 0x0 | Reserved. |

## 9.3.8 MSI-X Capability

More than one MSI-X capability structure per function is prohibited while a function is permitted to have both an MSI and an MSI-X capability structure.

In contrast to the MSI capability structure, which directly contains all of the control/status information for the function's vectors, the MSI-X capability structure instead points to an MSI-X table structure and an MSI-X Pending Bit Array (PBA) structure, each residing in memory space.

Each structure is mapped by a BAR belonging to the function that begins at 0x10 in the configuration space. A BAR Indicator Register (BIR) indicates which BAR and a Qword-aligned offset indicates where the structure begins relative to the base address associated with the BAR. The BAR is 64-bit, but must map to the memory space. A function is permitted to map both structures with the same BAR or map each structure with a different BAR.

The MSI-X table structure (Section 9.3.8.2) typically contains multiple entries, each consisting of several fields: *Message Address*, *Message Upper Address*, *Message Data*, and *Vector Control*. Each entry is capable of specifying a unique vector.

The PBA structure [MSI-X PBA Register (0x78; RO)] contains the function's pending bits, one per table entry, organized as a packed array of bits within Qwords. The last Qword is not necessarily fully populated.

To request service using a given MSI-X table entry, a function performs a Dword memory write transaction using:

- The contents of the *Message Data* field entry for data
- The contents of the *Message Upper Address* field for the upper 32 bits of the address
- The contents of the *Message Address* field entry for the lower 32 bits of the address

A memory read transaction from the address targeted by the MSI-X message produces undefined results.

The MSI-X table and MSI-X PBA are permitted to co-reside within a naturally aligned 4 KB address range, though they must not overlap with each other.

MSI-X table entries and *Pending* bits are each numbered 0 through N-1, where N-1 is indicated by the *Table Size* field in the MSI-X Message Control register. For a given arbitrary MSI-X table entry K, its starting address can be calculated with the formula:

Entry starting address = Table base + K*16

For the associated *Pending* bit K, its address for Qword access and bit number within that Qword can be calculated with the formulas:

Qword address = PBA base + (K div 64)*8

Qword bit# = K mod 64

Software that chooses to read *Pending* bit K with Dword accesses can use these formulas:

Dword address = PBA base + (K div 32)*4

Dword bit# = K mod 32

### 9.3.8.1 MSI-X Capability Structure

| Byte Offset | Byte 3 | Byte 2 | Byte 1 | Byte 0 |
|---|---|---|---|---|
| 0x70 | Message Control (0x00090) | | Next Pointer (0xA0) | Capability ID (0x11) |
| 0x74 | Table Offset | | | |
| 0x78 | PBA Offset | | | |

#### 9.3.8.1.1 Capability ID Register (0x70; RO)

This field equals 0x11 indicating that the linked list item as being the MSI-X registers.

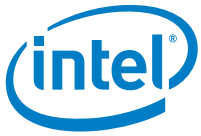
#### 9.3.8.1.2 Next Pointer Register (0x71; RO)

This field provides an offset to the next capability item in the capability list. Its value of 0xA0 points to PCIe capability.

#### 9.3.8.1.3 Message Control Register (0x72; RW)

These register fields are listed in the following table. Note that there is a dedicated register (per PCI function).

| Bits | RW | Default | Description |
|---|---|---|---|
| 10:0 | RO | 0x3F | Table Size.<br>System software reads this field to determine the MSI-X Table Size N, which is encoded as N-1. The 82599 supports up to 64 different interrupt vectors per function.<br>This field is loaded from the EEPROM MSI_X _N field. |
| 13:11 | RO | 0b | Always returns 0b on a read.<br>A write operation has no effect. |
| 14 | RW | 0b | Function Mask.<br>If 1b, all of the vectors associated with the function are masked, regardless of their per-vector *Mask* bit states.<br>If 0b, each vector's *Mask* bit determines whether the vector is masked or not.<br>Setting or clearing the MSI-X *Function Mask* bit has no effect on the state of the per-vector *Mask* bits. |
| 15 | RW | 0b | MSI-X Enable.<br>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.<br>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.<br>If 0b, the function is prohibited from using MSI-X to request service. |



#### 9.3.8.1.4 MSI-X Table Offset Register (0x74; RW)

These register fields are listed in the following table.

| Bits | RW | Default | Description |
|---|---|---|---|
| 2:0 | RO | 0x3 | Table BIR.<br>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 | RO | 0x000 | Table Offset.<br>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.<br>*Note:* This field is read only. |

#### 9.3.8.1.5 MSI-X Pending Bit Array — PBA Offset (0x78; RW)

This register fields are listed in the following table.

| Bits | RW | Default | Description |
|---|---|---|---|
| 2:0 | RO | 0x4 | PBA BIR.<br>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 | RO | 0x0400 | PBA Offset.<br>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.<br>This field is read only. |

### 9.3.8.2 MSI-X Table Structure

| Dword3 — MSIXTVCTRL | Dword2 — MSIXTMSG | Dword1 — MSIXTUADD | Dword0 — MSIXTADD | Entry Number | BAR 3 — Offset |
|---|---|---|---|---|---|
| Vector Control | Msg Data | Msg Upper Addr | Msg Lower Addr | 0 | Base (0x0000) |
| Vector Control | Msg Data | Msg Upper Addr | Msg Lower Addr | 1 | Base + 1*16 |
| Vector Control | Msg Data | Msg Upper Addr | Msg Lower Addr | 2 | Base + 2*16 |
| … | … | … | … | … | |
| Vector Control | Msg Data | Msg Upper Addr | Msg Lower Addr | 63 | Base + 63*16 |
| Vector Control | Msg Data | Msg Upper Addr | Msg Lower Addr | 64 | Base + 64*16 |
| … | … | … | … | … | |
| Vector Control | Msg Data | Msg Upper Addr | Msg Lower Addr | 255 | Base + 255*16 |

**Note:** All MSI-X vectors > MSI-X 63, are usable only by the Virtual Functions (VFs) in IOV mode. These vectors are not exposed to the operating system by the *Table Size* field in the MSI-X Message Control word.

#### 9.3.8.2.1 MSI–X Message Address Low — MSIXTADD (BAR3: 0x0 + 0x10*n, n=0...255; RW)

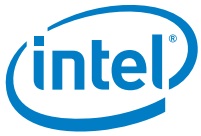
| Bits | Type | Default | Description |
|---|---|---|---|
| 1:0 | RW | 0x00 | Message Address.<br>For proper Dword alignment, software must always write zeros to these two bits; otherwise, the result is undefined. The state of these bits after reset must be 0b. These bits are permitted to be read-only or read/write. |
| 31:2 | RW | 0x00 | Message Address.<br>System-specified message lower address. For MSI-X messages, the contents of this field from an MSI-X table entry specifies the lower portion of the Dword-aligned address (AD[31:02]) for the memory write transaction. This field is read/write. |

#### 9.3.8.2.2 MSI–X Message Address High — MSIXTUADD (BAR3: 0x4 + 0x10*n, n=0...255; RW)

| Bits | Type | Default | Description |
|---|---|---|---|
| 31: 0 | RW | 0x00 | Message Upper Address.<br>System-specified message upper address bits. If this field is zero, Single Address Cycle (SAC) messages are used. If this field is non-zero, Dual Address Cycle (DAC) messages are used. This field is read/write. |

#### 9.3.8.2.3 MSI–X Message Data — MSIXTMSG (BAR3: 0x8 + 0x10*n, n=0...255; RW)

| Bits | Type | Default | Description |
|---|---|---|---|
| 31:0 | RW | 0x00 | Message Data.<br>System-specified message data. For MSI-X messages, the contents of this field from an MSI-X table entry specifies the data driven on AD[31:0] during the memory write transaction's data phase. This field is read/write. |



#### 9.3.8.2.4 MSI–X Vector Control — MSIXTVCTRL (BAR3: 0xC + 0x10*n, n=0...255; RW)

| Bits | Type | Default | Description |
|---|---|---|---|
| 0 | RW | 1b | Mask Bit.<br>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.<br>This bit's state after reset is 1b (entry is masked).<br>This bit is read/write. |
| 31:1 | RW | 0x00 | Reserved.<br>After reset, the state of these bits must be 0b.<br>However, for potential future use, software must preserve the value of these reserved bits when modifying the value of other *Vector Control* bits. If software modifies the value of these reserved bits, the result is undefined. |

### 9.3.8.3 MSI-X PBA Structure (BAR3: 0x2000 + 4*n, n=0...7; RW)

| Field | Bit(s) | Init Val | Description |
|---|---|---|---|
| PENBIT | 31:0 | 0x0 | MSI-X Pending Bits.<br>Each bit is set to 1b when the appropriate interrupt request is set and cleared to 0b when the appropriate interrupt request is cleared. Bit 'i' in register 'N' is associated to MSI-X vector 32 * 'N' + 'i', 'N' = 0...3. |

**Note:** Registers 2...7 are usable only by the VFs in IOV mode. These registers are not exposed to the operating system by the *Table Size* field in the MSI-X Message Control word.

## 9.3.9 VPD Registers

The 82599 supports access to a VPD structure stored in the EEPROM 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 EEPROM is in process on the other function can yield to unexpected results.

| Byte Offset | Byte 3 | Byte 2 | Byte 1 | Byte 0 |
|---|---|---|---|---|
| 0xE0 | VPD Address | | Next Pointer (0x00) | Capability ID (0x03) |
| 0xE4 | VPD Data | | | |

### 9.3.9.1 Capability ID Register (0xE0; RO)

This field equals 0x3 indicating the linked list item as being the VPD registers.

### 9.3.9.2 Next Pointer Register (0xE1; RO)

Offset to the next capability item in the capability list. A 0x00 value indicates that it is the last item in the capability-linked list.

### 9.3.9.3 VPD Address Register (0xE2; RW)

Word-aligned byte address of the VPD area in the EEPROM to be accessed. The register is read/write, and the initial value at power-up is indeterminate.

| Bits | RW | Default | Description |
|---|---|---|---|
| 14:0 | RW | X | Address.<br>Dword-aligned byte address of the VPD area in the EEPROM 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 | RW | 0b | F.<br>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.<br>0b = Read. Set by hardware when data is valid.<br>1b = Write. Cleared by hardware when data is written to the EEPROM.<br>The VPD address and data should not be modified before the action is done. |

### 9.3.9.4 VPD Data Register (0xE4; RW)

VPD read/write data.

| Bits | RW | Default | Description |
|---|---|---|---|
| 31:0 | RW | X | VPD Data.<br>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.<br>In a write access, the data should be set before the address and the flag is set. |

## 9.3.10 PCIe Configuration Registers

The 82599 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) | Capability ID (0x10) |
| 0xA4 | Device Capability | | | |
| 0xA8 | Device Status | | Device Control | |
| 0xAC | Link Capability | | | |
| 0xB0 | Link Status | | Link Control | |
| 0xB4 | Reserved | | | |
| 0xB8 | Reserved | | Reserved | |
| 0xBC | Reserved | | | |
| 0xC0 | Reserved | | Reserved | |
| 0xC4 | Device Capability 2 | | | |
| 0xC8 | Reserved | | Device Control 2 | |
| 0xCC | Reserved | | | |
| 0xD0 | Link Status 2 | | Link Control 2 | |
| 0xD4 | Reserved | | | |
| 0xD8 | Reserved | | Reserved | |

### 9.3.10.1 Capability ID Register (0xA0; RO)

This field equals 0x10 indicating that the linked list item as being the PCIe Capabilities registers.

### 9.3.10.2 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, a value of 0x00 value indicates that it is the last item in the capability-linked list.

### 9.3.10.3 PCIe Capabilities Register (0xA2; RO)

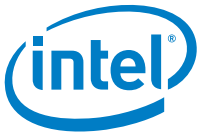
The PCIe Capabilities register identifies PCIe device type and associated capabilities. This is a read-only register identical to all functions.

| Bits | RW | Default | Description |
|---|---|---|---|
| 3:0 | RO | 0010b | Capability Version.<br>Indicates the PCIe capability structure version. The 82599 supports PCIe version 2 (also loaded from the PCIe *Capability Version* bit in the EEPROM). |
| 7:4 | RO | 0000b | Device/Port Type.<br>Indicates the type of PCIe functions. All functions are native PCI functions with a value of 0000b. |
| 8 | RO | 0b | Slot Implemented.<br>The 82599 does not implement slot options. Therefore, this field is hard wired to 0b. |
| 13:9 | RO | 00000b | Interrupt Message Number.<br>The 82599 does not implement multiple MSI per function. As a result, this field is hard wired to 0x0. |
| 15:14 | RO | 00b | Reserved. |

### 9.3.10.4 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 | RW | Default | Description |
|---|---|---|---|
| 2:0 | RO | 010b | Max Payload Size Supported.<br>This field indicates the maximum payload that The 82599 can support for TLPs. It is loaded from the EEPROM with a default value of 512 bytes. |
| 4:3 | RO | 00b | Phantom Function Supported.<br>Not supported by the 82599. |
| 5 | RO | 0b | Extended Tag Field Supported.<br>Maximum supported size of the *Tag* field. The 82599 supports a 5-bit *Tag* field for all functions. |



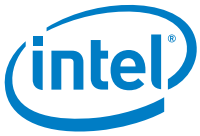
| Bits | RW | Default | Description |
|---|---|---|---|
| 8:6 | RO | 011b | Endpoint L0s Acceptable Latency.<br>This field indicates the acceptable latency that the 82599 can withstand due to the transition from L0s state to the L0 state. All functions share the same value loaded from the EEPROM PCIe Init Configuration 1 bits [8:6].<br>A value of 011b equals 512 ns. |
| 11:9 | RO | 110b | Endpoint L1 Acceptable Latency.<br>This field indicates the acceptable latency that the 82599 can withstand due to the transition from L1 state to the L0 state.<br>A value of 110b equals 32 µs-64 µs.<br>All functions share the same value loaded from the EEPROM. |
| 12 | RO | 0b | Attention Button Present.<br>Hard wired in the 82599 to 0b for all functions. |
| 13 | RO | 0b | Attention Indicator Present.<br>Hard wired in the 82599 to 0b for all functions. |
| 14 | RO | 0b | Power Indicator Present.<br>Hard wired in the 82599 to 0b for all functions. |
| 15 | RO | 1b | Role Based Error Reporting.<br>Hard wired in the 82599 to 1b for all functions. |
| 17:16 | RO | 000b | Reserved 0b. |
| 25:18 | RO | 0x00 | Slot Power Limit Value.<br>Used in upstream ports only. Hard wired in the 82599 to 0x00 for all functions. |
| 27:26 | RO | 00b | Slot Power Limit Scale.<br>Used in upstream ports only. Hard wired in the 82599 to 0b for all functions. |
| 28 | RO | 1b | Function Level Reset Capability.<br>A value of 1b indicates the<br>Function supports the optional Function Level Reset (FLR) mechanism. |
| 31:29 | RO | 0000b | Reserved. |

## 9.3.10.5 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.<br>Enable error report. |
| 1 | RW | 0b | Non-Fatal Error Reporting Enable.<br>Enable error report. |
| 2 | RW | 0b | Fatal Error Reporting Enable.<br>Enable error report. |

| Bits | RW | Default | Description |
|---|---|---|---|
| 3 | RW | 0b | Unsupported Request Reporting Enable.<br>Enable error report. |
| 4 | RW | 1b | Enable Relaxed Ordering.<br>If this bit is set, the 82599 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<br>(128 bytes) | Max Payload Size.<br>This field sets the maximum TLP payload size for the 82599 functions. As a receiver, the 82599 must handle TLPs as large as the set value. As a transmitter, the 82599 must not generate TLPs exceeding the set value.<br>The *Max Payload Size* field supported in the Device Capabilities register indicates permissible values that can be programmed.<br>In ARI mode, *Max Payload Size* is determined solely by the field in function 0 while it is meaningless in the other function(s). |
| 8 | RW | 0b | Extended Tag field Enable.<br>Not implemented in the 82599. |
| 9 | RW | 0b | Phantom Functions Enable.<br>Not implemented in the 82599. |
| 10 | RWS | 0b | Auxiliary Power PM Enable.<br>When set, enables the 82599 to draw AUX power independent of PME AUX power. The 82599 is a multi-function device, therefore allowed to draw AUX power if at least one of the functions has this bit set. |
| 11 | RW | 1b | Reserved |
| 14:12 | RW | 010b | Max Read Request Size.<br>This field sets maximum read request size for the 82599 as a requester.<br>000b = 128 bytes.<br>001b = 256 bytes.<br>010b = 512 bytes.<br>011b = 1024 bytes.<br>100b = 2048 bytes.<br>101b = 4096 bytes (unsupported by 82599).<br>110b = Reserved.<br>111b = Reserved. |
| 15 | RW | 0b | Initiate FLR.<br>A write of 1b initiates FLR to the function. The value read by software from this bit is always 0b. |



### 9.3.10.6 Device Status Register (0xAA; RW1C)

This register provides information about PCIe device specific parameters. Note that there is a dedicated register per each function.

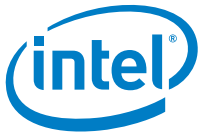
| Bits | RW | Default | Description |
|---|---|---|---|
| 0 | RW1C | 0b | Correctable Detected.<br>Indicates status of correctable error detection. |
| 1 | RW1C | 0b | Non-Fatal Error Detected.<br>Indicates status of non-fatal error detection. |
| 2 | RW1C | 0b | Fatal Error Detected.<br>Indicates status of fatal error detection. |
| 3 | RW1C | 0b | Unsupported Request Detected.<br>Indicates that the 82599 received an unsupported request. This field is identical in all functions. The 82599 can't distinguish which function causes the error. |
| 4 | RO | 0b | Aux Power Detected.<br>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.<br>Indicates whether the 82599 has ANY transactions pending. (transactions include completions for any outstanding non-posted request for all used traffic classes). |
| 15:6 | RO | 0x00 | Reserved. |

### 9.3.10.7 Link Capabilities Register (0xAC; RO)

This register identifies PCIe link-specific capabilities. This is a read-only register identical to all functions.

| Bits | RW | Default | Description |
|---|---|---|---|
| 3:0 | RO | 0010b | Supported Link Speeds.<br>This field indicates the supported Link speed(s) of the associated link port.<br>Defined encodings are:<br>0001b = 2.5 GbE link speed supported.<br>0010b = 5 GbE and 2.5 GbE link speeds supported. |
| 9:4 | RO | 0x08 | Max Link Width.<br>Indicates the maximum link width. The 82599 supports a x1, x2, x4 and x8-link width with a default value of eight lanes.<br>Defined encoding:<br>000000b = Reserved<br>000001b = x1<br>000010b = x2<br>000100b = x4<br>001000b = x8 |

| Bits | RW | Default | Description |
|---|---|---|---|
| 11:10 | RO | 11b | Active State Link PM Support.<br>Indicates the level of the active state of power management supported in the 82599. Defined encodings are:<br>00b = Reserved<br>01b = L0s Entry Supported.<br>10b = Reserved<br>11b = L0s and L1 Supported.<br>All functions share the same value loaded from the EEPROM Act_Stat_PM_Sup field in the EEPROM PCIe Init Configuration 3 word at offset 0x3. |
| 14:12 | RO | 001b<br>(64-128 ns) | L0s Exit Latency.<br>Indicates the exit latency from L0s to L0 state. 000b = Less than 64 ns.<br>001b = 64 ns – 128 ns.<br>010b = 128ns – 256 ns.<br>011b = 256 ns – 512 ns.<br>100b = 512 ns — 1 µs.<br>101b = 1 µs – 2 µs.<br>110b = 2 µs – 4 µs.<br>111b = More than 64 µs.<br>All functions share the same value loaded from the EEPROM. |
| 17:15 | RO | 111b | L1 Exit Latency.<br>Indicates the exit latency from L1 to L0 state.<br>000b = Less than 1 µs.<br>001b = 1 µs — 2 µs.<br>010b = 2 µs — 4 µs.<br>011b = 4 µs — 8 µs.<br>100b = 8 µs — 16 µs.<br>101b = 16 µs — 32 µs.<br>110b = 32 µs — 64 µs.<br>111b = L1 transition not supported.<br>All functions share the same value loaded from the EEPROM. |
| 18 | RO | 0 | Clock Power Management |
| 19 | RO | 0 | Surprise Down Error Reporting Capable.<br>Hard wired to 0b. |
| 20 | RO | 0 | Data Link Layer Link Active Reporting Capable. |
| 21 | RO | 0 | Link Bandwidth Notification Capability.<br>Hard wired to 0b. |
| 23:22 | RO | 00b | Reserved. |
| 31:24 | HwInit | 0x0 | Port Number.<br>The PCIe port number for the given PCIe link. This field is set in the link training phase. |



### 9.3.10.8 Link Control Register (0xB0; RO)

This register controls PCIe link specific parameters. There is a dedicated register per each function.

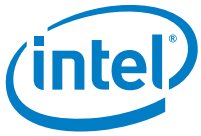
| Bits | RW | Default | Description |
|---|---|---|---|
| 1:0 | RW | 00b | Active State Link PM Control.<br>This field controls the active state PM supported on the link. Link PM functionality is determined by the lowest common denominator of all functions. Defined encodings are:<br>00b = PM Disabled.<br>01b = L0s Entry Supported.<br>10b = Reserved.<br>11b = L0s and L1 Supported.<br>In ARI mode, the ASPM is determined solely by the field in function 0 while it is meaningless in the other function(s). |
| 2 | RO | 0b | Reserved. |
| 3 | RW | 0b | Read Completion Boundary. |
| 4 | RO | 0b | Link Disable.<br>Reserved for endpoint devices. Hard wired to 0b. |
| 5 | RO | 0b | Retrain Clock.<br>Not applicable for endpoint devices. Hard wire to 0b. |
| 6 | RW | 0b | Common Clock Configuration.<br>When set, indicates that the 82599 and the component at the other end of the link are operating with a common reference clock. A value of 0b indicates that they are operating with an asynchronous clock. This parameter affects the L0s exit latencies.<br>In ARI mode, the common clock configuration is determined solely by the field in function 0 while it is meaningless in the other function(s). |
| 7 | RW | 0b | Extended Sync.<br>When set, this bit forces an extended Tx of the FTS ordered set in FTS and an extra TS1 at the exit from L0s prior to entering L0. |
| 8 | RO | 0b | Reserved. |
| 9 | WR | 0b | Hardware Autonomous Width Disable.<br>When set to 1b, this bit disables hardware from changing the link width for reasons other than attempting to correct an unreliable link operation by reducing link width.<br>This bit can be written only by function 0. |
| 10 | RO | 0b | Link Bandwidth Management Interrupt Enable.<br>Not supported in the 82599. Hard wired to 0b. |
| 11 | RO | 0b | Link Autonomous Bandwidth Interrupt Enable<br>.Not supported in the 82599. Hard wired to 0b. |
| 15:12 | RO | 0x0 | Reserved. |

### 9.3.10.9 Link Status Register (0xB2; RO)

This register provides information about PCIe Link specific parameters. This is a read only register identical to all functions.

| Bits | RW | Default | Description |
|---|---|---|---|
| 3:0 | RO | 0001b | Current Link Speed.<br>This field indicates the negotiated link speed of the given PCIe link.<br>Defined encodings are:<br>0001b = 2.5 GbE PCIe link.<br>0010b = 5 GbE PCIe link.<br>All other encodings are reserved. |
| 9:4 | RO | 000001b | Negotiated Link Width.<br>Indicates the negotiated width of the link.<br>Relevant encodings for the 82599 are:<br>000001b = x1<br>000010b = X2<br>000100b = x4<br>001000b = x8 |
| 10 | RO | 0b | Undefined. |
| 11 | RO | 0b | Link Training.<br>Indicates that link training is in progress.<br>This field is not applicable and is reserved for endpoint devices, and is hard wired to 0b. |
| 12 | HwInit | 1b | Slot Clock Configuration.<br>When set, indicates that the 82599 uses the physical reference clock that the platform provides at the connector. This bit must be cleared if the 82599 uses an independent clock. The *Slot Clock Configuration* bit is loaded from the *Slot_Clock_Cfg* EEPROM bit. |
| 13 | RO | 0b | Data Link Layer Link Active.<br>Not supported in the 82599. Hard wire to 0b. |
| 14 | RO | 0b | Link Bandwidth Management Status.<br>Not supported in the 82599. Hard wire to 0b. |
| 15 | RO | 0b | Link Autonomous Bandwidth Status.<br>This bit is not applicable and is reserved for endpoints. |

The following registers are supported only if the capability version is two and above.



## 9.3.10.10 Device Capability 2 Register (0xC4; RO)

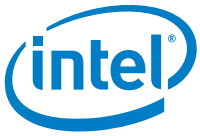
This register identifies the PCIe device-specific capabilities. It is a read-only register with the same value for both LAN functions.

| Bits | RW | Default | Description |
|---|---|---|---|
| 3:0 | RO | 1111b | Completion Timeout Ranges Supported.<br>This field indicates the 82599's support for the optional completion timeout programmability mechanism.<br>Four time value ranges are defined:<br>Range A = 50 µs to 10 ms.<br>Range B = 10 ms to 250 ms.<br>Range C = 250 ms to 4 s.<br>Range D = 4 s to 64 s.<br>Bits are set according to the following values to show the timeout value ranges that the 82599 supports.<br>0000b = Completion timeout programming not supported. The 82599 must implement a timeout value in the range of 50 µs to 50 ms.<br>0001b = Range A.<br>0010b = Range B.<br>0011b = Ranges A and B.<br>0110b = Ranges B and C.<br>0111b = Ranges A, B and C.<br>1110b = Ranges B, C and D.<br>1111b = Ranges A, B, C and D.<br>All other values are reserved. |
| 4 | RO | 1b | Completion Timeout Disable Supported |
| 5 | RO | 0b | ARI Forwarding Supported.<br>Applicable only to Switch Downstream. Ports and Root Ports; must be 0b for other function types. |
| 31:6 | RO | 0x0000 | Reserved. |

## 9.3.10.11 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.<br>For devices that support completion timeout programmability, this field enables system software to modify the completion timeout value.<br>Defined encodings:<br>0000b = Default range: 50 µs to 50 ms<br>*Note:* It is strongly recommended that the completion timeout mechanism not expire in less than 10 ms.<br>Values available if Range A (50 µs to 10 ms) programmability range is supported:<br>0001b = 50 µs to 100 µs<br>0010b = 1 ms to 10 ms<br>Values available if Range B (10 ms to 250 ms) programmability range is supported:<br>0101b = 16 ms to 55 ms<br>0110b = 65 ms to 210 ms<br>Values available if Range C (250 ms to 4 s) programmability range is supported:<br>1001b = 260 ms to 900 ms<br>1010b = 1 s to 3.5 s<br>Values available if the Range D (4 s to 64 s) programmability range is supported:<br>1101b = 4 s to 13 s<br>1110b = 17 s to 64 s<br>Values not defined are reserved.<br>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. |
| 4 | RW | 0b | Completion Timeout Disable.<br>When set to 1b, this bit disables the completion timeout mechanism.<br>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. |
| 5 | RO | 0b | ARI Forwarding Enable.<br>Applicable only to switch devices. |
| 15:6 | RO | 0b | Reserved. |



## 9.3.10.12 Link Control 2 Register (0xD0; RWS)

All RW fields in this register affect the device behavior only through function 0. In function 1 these fields are reserved read as zeros.

| Bits | RW | Default | Description |
|---|---|---|---|
| 3:0 | RWS | 0010b | Target Link Speed.<br>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.<br>Defined encodings are:<br>0001b = 2.5 GbE target link speed.<br>0010b = 5 GbE target link speed.<br>All other encodings are reserved.<br>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.<br>The default value of this field is the highest link speed supported by the 82599 (as reported in the *Supported Link Speeds* field of the Link Capabilities register). |
| 4 | RWS | 0b | Enter Compliance.<br>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.<br>The default value of this field following a fundamental reset is 0b. |
| 5 | RWS | 0b | Hardware Autonomous Speed Disable.<br>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.<br>This bit is not applicable and reserved for endpoints. |
| 9:7 | RWS | 000b | Transmit Margin.<br>This field controls the value of the non de emphasized voltage level at the Transmitter pins.<br>Encodings:<br>000b = Normal operating range.<br>001b = 800-1200 mV for full swing and 400-700 mV for half-swing.<br>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.<br>111b = (n) reserved. |
| 10 | RWS | 0b | Enter Modified Compliance.<br>When this bit is set to 1b, the device transmits modified compliance pattern if the LTSSM enters Polling.Compliance state. |
| 11 | RWS | 0b | Compliance SOS.<br>When set to 1b, the LTSSM is required to send SOS periodically in between the (modified) compliance patterns. |

### 9.3.10.13 Link Status 2 Register (0xD2; RO)

| Bits | RW | Default | Description |
|---|---|---|---|
| 0 | RO | 0b | Current De-emphasis Level.<br>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<br>Encodings:<br>0b = 6 dB.<br>1b = 3.5 dB. |

## 9.4 PCIe Extended Configuration Space

PCIe configuration space is located in a flat memory-mapped address space. PCIe extends the configuration space beyond the 256 bytes available for PCI to 4096 bytes. The 82599 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 |

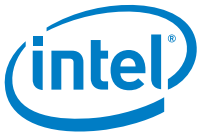
PCIe extended configuration space is allocated using a linked list of optional or required PCIe extended capabilities following a format resembling PCI capability structures. The first PCIe extended capability is located at offset 0x100 in the device configuration space. The first Dword of the capability structure identifies the capability/version and points to the next capability.

The 82599 supports the following PCIe extended capabilities:

**Table 9-6 Extended Capabilities list**

| Capability | Offset | Next Header |
|---|---|---|
| Advanced Error Reporting Capability | 0x100 | 0x140/0x150/0x000[1] |
| Serial Number | 0x140 | 0x150/0x000[1] |
| Alternative RID Interpretation (ARI) | 0x150 | 0x160 |
| IOV support | 0x160 | 0x000 |

1. Depends on EEPROM settings enabling the serial numbers and ARI/IOV structures.



## 9.4.1 Advanced Error Reporting Capability (AER)

The PCIe advanced error reporting capability is an optional extended capability to support advanced error reporting. The tables that follow list the PCIe advanced error reporting extended capability structure for PCIe devices.

| Byte Offset | Byte 3 | Byte 2 | Byte 1 | Byte 0 |
|---|---|---|---|---|
| 0x100 | Next Capability Ptr. (0x140) | 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 | | | |

### 9.4.1.1 Advanced Error Reporting Enhanced Capability Header Register (0x100; RO)

| Bits | RW | Default | Description |
|---|---|---|---|
| 15:0 | RO | 0x0001 | Extended Capability ID.<br>PCIe extended capability ID indicating advanced error reporting capability. |
| 19:16 | RO | 0x1 | Version Number.<br>PCIe advanced error reporting extended capability version number. |
| 31:20 | RO | 0x0140/0x0150/0x0000 | Next Capability Pointer.<br>Next PCIe extended capability pointer.<br>See Table 9-6 for possible values of the next capability pointer. |

### 9.4.1.2 Uncorrectable Error Status Register (0x104; RW1CS)

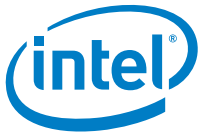
The Uncorrectable Error Status register reports error status of individual uncorrectable error sources on a PCIe device. An individual error status bit that is set to 1b indicates that a particular error occurred; software can clear an error status by writing a 1b to the respective bit. Register is cleared by LAN_PWR_GOOD.

| Bits | RW | Default | Description |
|---|---|---|---|
| 3:0 | RO | 0b | Reserved. |
| 4 | RW1CS | 0b | Data Link Protocol Error Status. |
| 11:5 | RO | 0b | Reserved. |
| 12 | RW1CS | 0b | Poisoned TLP Status. |
| 13 | RW1CS | 0b | Flow Control Protocol Error Status. |
| 14 | RW1CS | 0b | Completion Timeout Status. |
| 15 | RW1CS | 0b | Completer Abort Status. |
| 16 | RW1CS | 0b | Unexpected Completion Status. |
| 17 | RW1CS | 0b | Receiver Overflow Status. |
| 18 | RW1CS | 0b | Malformed TLP Status. |
| 19 | RW1CS | 0b | ECRC Error Status. |
| 20 | RW1CS | 0b | Unsupported Request Error Status. |
| 21 | RO | 0b | ACS Violation Status. |
| 31:22 | RO | 0b | Reserved. |

### 9.4.1.3 Uncorrectable Error Mask Register (0x108; RWS)

The Uncorrectable Error Mask register controls reporting of individual uncorrectable errors by device to the host bridge via a PCIe error message. A masked error (respective bit set in mask register) is not reported to the host bridge by an individual device. Note that there is a mask bit per bit of the Uncorrectable Error Status register.

| Bits | RW | Default | Description |
|---|---|---|---|
| 3:0 | RO | 0b | Reserved. |
| 4 | RWS | 0b | Data Link Protocol Error Mask. |
| 11:5 | RO | 0b | Reserved. |
| 12 | RWS | 0b | Poisoned TLP Mask. |
| 13 | RWS | 0b | Flow Control Protocol Error Mask. |
| 14 | RWS | 0b | Completion Timeout Mask. |
| 15 | RWS | 0b | Completer Abort Mask. |
| 16 | RWS | 0b | Unexpected Completion Mask. |



| Bits | RW | Default | Description |
|---|---|---|---|
| 17 | RWS | 0b | Receiver Overflow Mask. |
| 18 | RWS | 0b | Malformed TLP Mask. |
| 19 | RWS | 0b | ECRC Error Mask. |
| 20 | RWS | 0b | Unsupported Request Error Mask. |
| 21 | RO | 0b | ACS Violation Mask. |
| 31:22 | RO | 0b | Reserved. |

### 9.4.1.4 Uncorrectable Error Severity Register (0x10C; RWS)

The Uncorrectable Error Severity register controls whether an individual uncorrectable error is reported as a fatal error. An uncorrectable error is reported as fatal when the corresponding error bit in the severity register is set. If the bit is cleared, the corresponding error is considered non-fatal.

| Bits | RW | Default | Description |
|---|---|---|---|
| 3:0 | RO | 0b | Reserved. |
| 4 | RWS | 1b | Data Link Protocol Error Severity. |
| 11:5 | RO | 0b | Reserved. |
| 12 | RWS | 0b | Poisoned TLP Severity. |
| 13 | RWS | 1b | Flow Control Protocol Error Severity. |
| 14 | RWS | 0b | Completion Timeout Severity. |
| 15 | RWS | 0b | Completer Abort Severity. |
| 16 | RWS | 0b | Unexpected Completion Severity. |
| 17 | RWS | 1b | Receiver Overflow Severity. |
| 18 | RWS | 1b | Malformed TLP Severity. |
| 19 | RWS | 0b | ECRC Error Severity. |
| 20 | RWS | 1b | Unsupported Request Error Severity. |
| 21 | RO | 0b | ACS Violation Severity. |
| 31:22 | RO | 0b | Reserved. |

### 9.4.1.5 Correctable Error Status Register (0x110; RW1CS)

The Correctable Error Status register reports error status of individual correctable error sources on a PCIe device. When an individual error status bit is set to 1b it indicates that a particular error occurred; software can clear an error status by writing a 1b to the respective bit. Register is cleared by LAN_PWR_GOOD.

| Bits | RW | Default | Description |
|---|---|---|---|
| 0 | RW1CS | 0b | Receiver Error Status. |
| 5:1 | RO | 0b | Reserved. |
| 6 | RW1CS | 0b | Bad TLP Status. |
| 7 | RW1CS | 0b | Bad DLLP Status. |
| 8 | RW1CS | 0b | REPLAY_NUM Rollover Status. |
| 11:9 | RO | 0b | Reserved. |
| 12 | RW1CS | 0b | Replay Timer Timeout Status. |
| 13 | RW1CS | 0b | Advisory Non-Fatal Error Status. |
| 15:14 | RO | 0b | Reserved. |

### 9.4.1.6 Correctable Error Mask Register (0x114; RWS)

The Correctable Error Mask register controls reporting of individual correctable errors by device to the host bridge via a PCIe error message. A masked error (respective bit set in mask register) is not reported to the host bridge by an individual device. There is a mask bit per bit in the Correctable Error Status register.

| Bits | RW | Default | Description |
|---|---|---|---|
| 0 | RWS | 0b | Receiver Error Mask. |
| 5:1 | RO | 0b | Reserved |
| 6 | RWS | 0b | Bad TLP Mask. |
| 7 | RWS | 0b | Bad DLLP Mask. |
| 8 | RWS | 0b | REPLAY_NUM Rollover Mask. |
| 11:9 | RO | 0b | Reserved. |
| 12 | RWS | 0b | Replay Timer Timeout Mask. |
| 13 | RWS | 1b | Advisory Non-Fatal Error Mask. |
| 15:14 | RO | 0b | Reserved. |

### 9.4.1.7 Advanced Error Capabilities and Control Register (0x118; RO)

| Bits | RW | Default | Description |
|---|---|---|---|
| 4:0 | ROS | 0b | Vector pointing to the first recorded error in the Uncorrectable Error Status register.<br>This is a read-only field that identifies the bit position of the first uncorrectable error reported in the Uncorrectable Error Status register. |
| 5 | RO | 0b | ECRC Generation Capable.<br>If set, this bit indicates that the function is capable of generating ECRC.<br>This bit is loaded from EEPROM. |
| 6 | RWS | 0b | ECRC Generation Enable.<br>When set, ECRC generation is enabled. |
| 7 | RO | 0b | ECRC Check Capable.<br>If set, this bit indicates that the function is capable of checking ECRC.<br>This bit is loaded from EEPROM. |
| 8 | RWS | 0b | ECRC Check Enable.<br>When set Set, ECRC checking is enabled. |

### 9.4.1.8 Header Log Register (0x11C:128; RO)

The header log register captures the header for the transaction that generated an error. This register is 16 bytes.

| Bits | RW | Default | Description |
|---|---|---|---|
| 127:0 | ROS | 0b | Header of the packet in error (TLP or DLLP). |

## 9.4.2 Serial Number

The PCIe device serial number capability is an optional extended capability that can be implemented by any PCIe device. The device serial number is a read-only 64-bit value that is unique for a given PCIe device.

**Note:** All multi-function devices that implement this capability must implement it for function 0; other functions that implement this capability must return the same device serial number value as that reported by function 0.

| Byte Offset | Byte 3 | Byte 2 | Byte 1 | Byte 0 |
|---|---|---|---|---|
| 0x140 | Next Capability Ptr. (0x150) | Version (0x1) | Serial ID Capability ID (0x0003) | |
| 0x144 | Serial Number Register (Lower Dword) | | | |
| 0x148 | Serial Number Register (Upper Dword) | | | |

### 9.4.2.1 Device Serial Number Enhanced Capability Header Register (0x140; RO)

| Bit(s) | RW | Description |
|---|---|---|
| 15:0 | RO | PCIe Extended Capability ID.<br>This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability.<br>The extended capability ID for the device serial number capability is 0x0003. |
| 19:16 | RO | Capability Version.<br>This field is a PCI-SIG defined version number that indicates the version of the capability structure present.<br>*Note:* Must be set to 0x1 for this version of the specification. |
| 31:20 | RO | Next Capability Offset.<br>This field contains the offset to the next PCIe capability structure or 0x000 if no other items exist in the linked list of capabilities.<br>The value of this field is 0x150 to point to the ARI capability structure.<br>If ARI/IOV and Serial ID are disabled in EEPROM this field is zero. See Table 9-6. |

### 9.4.2.2 Serial Number Registers (0x144:0x148; RO)

The Serial Number register is a 64-bit field that contains the IEEE defined 64-bit Extended Unique Identifier (EUI-64*). The register at offset 0x144 holds the lower 32 bits and the register at offset 0x148 holds the higher 32 bits. The following figure details the allocation of register fields in the Serial Number register. The table that follows provides the respective bit definitions.

| Bit(s) | RW | Description |
|---|---|---|
| 63:0 | RO | PCIe Device Serial Number.<br>This field contains the IEEE defined 64-bit EUI-64*. This identifier includes a 24-bit company ID value assigned by IEEE registration authority and a 40-bit extension identifier assigned by the manufacturer. |

The serial number uses the Ethernet MAC address according to the following definition:

| Field | Company ID | | | Extension Identifier | | | | |
|---|---|---|---|---|---|---|---|---|
| Order | Addr+0 | Addr+1 | Addr+2 | Addr+3 | Addr+4 | Addr+5 | Addr+6 | Addr+7 |
| Most Significant Byte | | | | | | Least Significant Byte | | |
| Most Significant Bit | | | | | | Least Significant Bit | | |

The serial number can be constructed from the 48-bit Ethernet MAC address in the following form:

| Field | Company ID | | | MAC Label | | Extension identifier | | |
|---|---|---|---|---|---|---|---|---|
| Order | Addr+0 | Addr+1 | Addr+2 | Addr+3 | Addr+4 | Addr+5 | Addr+6 | Addr+7 |
| Most Significant Bytes | | | | | | Least Significant Byte | | |
| Most Significant Bit | | | | | | Least Significant Bit | | |

In this case, the MAC label is 0xFFFF.

For example, assume that the company ID is (Intel) 00-A0-C9 and the extension identifier is 23-45-67. In this case, the 64-bit serial number is:

| Field | Company ID | | | MAC Label | | Extension Identifier | | |
|---|---|---|---|---|---|---|---|---|
| Order | Addr+0 | Addr+1 | Addr+2 | Addr+3 | Addr+4 | Addr+5 | Addr+6 | Addr+7 |
| | 00 | A0 | C9 | FF | FF | 23 | 45 | 67 |
| Most Significant Byte | | | | | | Least Significant Byte | | |
| Most Significant Bit | | | | | | Least Significant Bit | | |

The Ethernet MAC address for the serial number capability is loaded from the Serial Number Ethernet MAC Address EEPROM field (not the same field that is loaded from EEPROM into the RAL and RAH registers).

**Note:** The official document that defines EUI-64* is: http://standards.ieee.org/regauth/oui/tutorials/EUI64.html

## 9.4.3 Alternate Routing ID Interpretation (ARI) Capability Structure

In order to allow more than eight functions per endpoint without requesting an internal switch, as is usually needed in virtualization scenarios, the PCI-SIG defines a new capability that allows a different interpretation of the *Bus*, *Device*, and *Function* fields. The ARI capability structure is as follows:

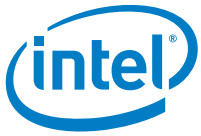
| Byte Offset | Byte 3 | Byte 2 | Byte 1 | Byte 0 |
|---|---|---|---|---|
| 0x150 | Next Capability Ptr. (0x160) | Version (0x1) | ARI Capability ID (0x000E) | |
| 0x154 | ARI Control Register | | ARI Capabilities | |

### 9.4.3.1 PCIe ARI Header Register (0x150; RO)

| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| ID | 15:0 | RO | 0x000E | PCIe Extended Capability ID.<br>PCIe extended capability ID for the alternative RID interpretation. |
| Version | 19:16 | RO | 1b | Capability Version.<br>This field is a PCI-SIG defined version number that indicates the version of the capability structure present.<br>Must be 0x1 for this version of the specification. |
| Next Capability Ptr. | 31:20 | RO | 0x160 | Next Capability Offset.<br>This field contains the offset to the next PCIe extended capability structure. The value of the 0x160 points to the IOV structure. |

### 9.4.3.2 PCIe ARI Capabilities and Control Register (0x154; RO)

| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| Reserved | 0 | RO | 0b | Not supported in the 82599. |
| Reserved | 1 | RO | 0b | Not supported in the 82599. |
| Reserved | 7:2 | RO | 0b | Reserved. |
| NFP | 15:8 | RO | 0x1 (func 0)<br>0x0 (func 1)[1] | Next Function Pointer.<br>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 | RO | 0b | Not supported in the 82599. |
| Reserved | 17 | RO | 0b | Not supported in the 82599. |
| Reserved | 19:18 | RO | 00b | Reserved. |



| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| Reserved | 22:20 | RO | 0b | Not supported in the 82599. |
| Reserved | 31:23 | RO | 0b | Reserved |

1. Even if port 0 and port 1 are switched or function zero is a dummy function, this register should keep its attributes according to the function number. If LAN1 is disabled, the value of this field in function zero should be zero.

## 9.4.4 IOV Capability Structure

This is the new structure used to support the IOV capabilities reporting and control. The following tables shows the possible implementations of this structure in the 82599.

| Byte Offset | Byte 3 | Byte 2 | Byte 1 | Byte 0 |
|---|---|---|---|---|
| 0x150 | Next Capability Ptr. (0x160) | Version (0x1) | Capability ID (0x000E) | |
| 0x154 | Control Register | | Capabilities | |
| 0x160 | Next Capability Offset (0x0) | Version (0x1) | IOV Capability ID (0x0010) | |
| 0x164 | SR IOV Capabilities | | | |
| 0x168 | SR IOV Status | | SR IOV Control | |
| 0x16C | Total VFs (RO) | | Initial VF (RO) | |
| 0x170 | Reserved | Function Dependency Link (RO) | Num VF (RW) | |
| 0x174 | VF Stride (RO) | | First VF Offset (RO) | |
| 0x178 | VF Device ID | | Reserved | |
| 0x17C | Supported Page Size (0x553) | | | |
| 0x180 | system page Size (RW) | | | |
| 0x184 | VF BAR0 — Low (RW) | | | |
| 0x188 | VF BAR0 — High (RW) | | | |
| 0x18C | VF BAR2 (RO) | | | |
| 0x190 | VF BAR3 — Low (RW) | | | |
| 0x194 | VF BAR3- High (RW) | | | |
| 0x198 | VF BAR5 (RO) | | | |
| 0x19C | VF Migration State Array Offset (RO) | | | |

#### 9.4.4.1 PCIe SR-IOV Header Register (0x160; RO)

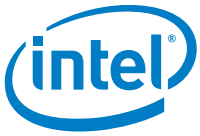
| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| ID | 15:0 | RO | 0x0010 | PCIe Extended Capability ID.<br>PCIe extended capability ID for the SR-IOV capability. |
| Version | 19:16 | RO | 0x1 | Capability Version.<br>This field is a PCI-SIG defined version number that indicates the version of the capability structure present.<br>Must be 0x1 for this version of the specification. |
| Next pointer | 31:20 | RO | 0x0 | Next Capability Offset.<br>This field contains the offset to the next PCIe extended capability structure or 0x000 if no other items exist in the linked list of capabilities. |

#### 9.4.4.2 PCIe SR-IOV Capabilities Register (0x164; RO)

| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| Reserved | 0 | RO | 0b | Not supported in the 82599. |
| Reserved | 20:1 | RO | 0x0 | Reserved. |
| Reserved | 31:21 | RO | 0x0 | Not supported in the 82599. |

#### 9.4.4.3 PCIe SR-IOV Control/Status Register (0x168; RW)

| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| VFE | 0 | RW | 0b | VF Enable/Disable.<br>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.<br>If set to 0b, VFs must be disabled and not visible in the PCIe fabric; VFs cannot respond to or issue PCIe transactions.<br>In addition, if *VF Enable* is cleared after having been set, all of the VFs must no longer:<br>• Issue PCIe transactions<br>• Respond to configuration space or memory space accesses.<br>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 | RO | 0b | Not supported in the 82599. |
| Reserved | 2 | RO | 0b | Not supported in the 82599. |



| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| VF MSE | 3 | RW | 0b | Memory Space Enable for Virtual Functions.<br>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.<br>When VF Enable is 1, virtual function memory space access is permitted only when VF MSE is Set. VFs shall follow the same error reporting rules as defined in the base specification if an attempt is made to access a virtual functions memory space when VF Enable is 1 and VF MSE is zero.<br>*Implementation Note:* Virtual functions memory space cannot be accessed when VF Enable is zero. Thus, VF MSE is "don't care" when VF Enable is zero, however, software may choose to set VF MSE after programming the VF BARn registers, prior to setting VF Enable to 1. |
| VF ARI | 4 | RW (func 0)<br>RO (func 1)[1] | 0b | VF ARI Enable. Device can locate VFs in function numbers 8 to 255 of the captured bus number. |
| Reserved | 15:5 | RO | 0x0 | Reserved. |
| Reserved | 16 | RO | 0b | Not implemented in the 82599. |
| Reserved | 31:17 | RO | 0b | Reserved. |

1. Even if port 0 and port 1 are switched or function zero is a dummy function, this field should keep it's attributes according to the function number.

### 9.4.4.4 PCIe SR-IOV Max/Total VFs Register (0x16C; RO)

| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| InitialVFs | 15:0 | RO | 64 | InitialVFs.<br>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.<br>In the 82599 this parameter is equal to the TotalVFs in this register. |
| TotalVFs | 31:16 | RO | 64 | TotalVFs.<br>Defines the maximum number of VFs that can be associated with the PF. This field is loaded from the *Max VFs* field in the IOV Control Word 1 in the EEPROM. |

### 9.4.4.5 PCIe SR-IOV Num VFs Register (0x170; RW)

| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| NumVFs | 15:0 | RW | 0x0 | Num VFs.<br>Defines the number of VFs software has assigned to the PF. Software sets NumVFs to any value between one and the TotalVFs as part of the process of creating VFs. NumVFs VFs must be visible in the PCIe fabric after both NumVFs is set to a valid value and VF Enable is set to 1b. |
| FDL | 23:16 | RO | 0x0 (func 0)<br>0x1 (func 1)[1] | Function Dependency Link.<br>Defines dependencies between physical functions allocation. In the 82599 there are no constraints. |

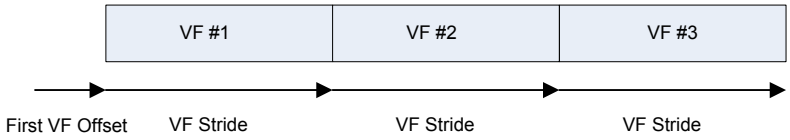
| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| Reserved | 31:24 | RO | 0 | Reserved. |

1. Even if port 0 and port 1 are switched or function zero is a dummy function, this register should keep it's attributes according to the function number.

### 9.4.4.6 PCIe SR-IOV VF RID Mapping Register (0x174; RO)

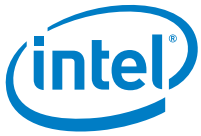
| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| FVO | 15:0 | RO | 0x180 | First VF offset.<br>Defines the requestor ID (RID) offset of the first VF that is associated with the PF that contains this capability structure. The first VFs 16-bit RID is calculated by adding the contents of this field to the RID of the PF containing this field.<br>The content of this field is valid only when *VF Enable* is set. If *VF Enable* is 0b, the contents are undefined.<br>If the ARI *Enable* bit is set, this field changes to 0x80. |
| VFS | 31:16 | RO | 0x2[1] | VF stride.<br>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.<br>The contents of this field is valid only when *VF Enable* is set and *NumVFs* is non-zero. If *VF Enable* is 0b or if NumVFs is zero, the contents are undefined. |

1. See Section 7.10.2.6.1.




### 9.4.4.7 PCIe SR-IOV VF Device ID Register (0x178; RO)

| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| DEVID | 31:16 | RO | 0x10ED | VF Device ID.<br>This field contains the device ID that should be presented for every VF to the Virtual Machine (VM).<br>The value of this field can be read from the IOV Control Word 2 in the EEPROM. |
| Reserved | 15:0 | RO | 0x0 | Reserved. |



### 9.4.4.8 PCIe SR-IOV Supported Page Size Register (0x17C; RO)

| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| Supported page Size | 31:0 | RO | 0x553 | 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.<br>Endpoints are required to support 4 KB, 8 KB, 64 KB, 256 KB, 1 MB and 4 MB page sizes. All other page sizes are optional. |

### 9.4.4.9 PCIe SR-IOV System Page Size Register (0x180; RW)

| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| Page size | 31:0 | RW | 0x1 | 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*.<br>When *System Page Size* is set, the PF and associated VFs are required to align all BAR resources on a *System Page Size* boundary. Each BAR size, including *VF BARn Size* (described later) must be aligned on a *System Page Size* boundary. Each BAR size, including VF *BARn Size* must be sized to consume a multiple of *System Page Size* bytes. All fields requiring page size alignment within a function must be aligned on a *System Page Size* boundary. *VF Enable* must be zero when *System Page Size* is set. The results are undefined if *System Page Size* is set when *VF Enable* is set. |

### 9.4.4.10 PCIe SR-IOV BAR 0 — Low Register (0x184; RW)

| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| Mem | 0 | RO | 0b | 0b indicates memory space. |
| Mem Type | 2:1 | RO | 10b | Indicates the address space size.<br>10b = 64-bit.<br>This bit is loaded from the IOV Control word in the EEPROM. |
| Prefetch Mem | 3 | RO | 0b* | 0b = Non-prefetchable space.<br>1b = Prefetchable space.<br>This bit is loaded from the IOV Control word in the EEPROM. |
| Memory Address Space | 31:4 | RW | 0x0 | Which bits are RW bits and which are RO to 0x0 depend on the memory mapping window size.<br>The size is a maximum between 16 KB and the page size. |

### 9.4.4.11 PCIe SR-IOV BAR 0 — High Register (0x188; RW)

| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| BAR0 — MSB | 31:0 | RW | 0x0 | MSB part of BAR0. |

### 9.4.4.12 PCIe SR-IOV BAR 2 Register (0x18C; RO)

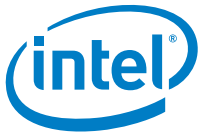
| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| BAR2 | 31:0 | RO | 0x0 | This BAR is not used. |

### 9.4.4.13 PCIe SR-IOV BAR 3 — Low Register (0x190; RW)

| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| Mem | 0 | RO | 0b | 0b indicates memory space. |
| Mem Type | 2:1 | RO | 10b | Indicates the address space size.<br>10b = 64-bit.<br>This bit is loaded from the IOV Control word in the EEPROM. |
| Prefetch Mem | 3 | RO | 0b* | 0b = Non-prefetchable space<br>1b = Prefetchable space<br>This bit is loaded from the IOV Control word in the EEPROM. |
| Memory Address Space | 31:4 | RW | 0x0 | Which bits are RW bits and which are RO to 0x0 depend on the memory mapping window size.<br>The size is a maximum between 16 KB and page size. |

### 9.4.4.14 PCIe SR-IOV BAR 3 — High Register (0x194; RW)

| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| BAR3 — MSB | 31:0 | RW | 0x0 | MSB part of BAR3. |



### 9.4.4.15 PCIe SR-IOV BAR 5 Register (0x198; RO)

| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| BAR5 | 31:0 | RO | 0x0 | This BAR is not used. |

### 9.4.4.16 PCIe SR-IOV VF Migration State Array Offset Register (0x19C; RO)

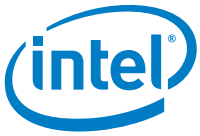
| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| Reserved | 2:0 | RO | 0x0 | Not implemented in the 82599. |
| Reserved | 31:0 | RO | 0x0 | Not implemented in the 82599. |

# 9.5 Virtual Functions Configuration Space

The configuration space reflected to each of the VF is a sparse version of the physical function configuration space. The following table describes the behavior of each register in the VF configuration space.

**Table 9-7 VF PCIe Configuration Space**

| Section | Offset | Name | VF behavior | Notes |
|---|---|---|---|---|
| PCI Mandatory Registers | 0 | Vendor ID | RO — 0xFFFF | |
| | 2 | Device ID | RO — 0xFFFF | |
| | 4 | Command | RW | See Section 9.5.1.1. |
| | 6 | Status | Per VF | See Section 9.5.1.2. |
| | 8 | RevisionID | RO as PF | |
| | 9 | Class Code | RO as PF | |
| | C | Cache Line Size | RO — 0x0 | |
| | D | Latency Timer | RO — 0x0 | |
| | E | Header Type | RO — 0x0 | |
| | F | Reserved | RO — 0x0 | |
| | 10 — 27 | BARs | RO — 0x0 | Emulated by VMM. |
| | 28 | CardBus CIS | RO — 0x0 | Not used. |
| | 2C | Sub Vendor ID | RO as PF | |
| | 2E | Sub System | RO as PF | |
| | 30 | Expansion ROM | RO — 0x0 | Emulated by VMM. |
| | 34 | Cap Pointer | RO — 0x70 | Next = MSI-X capability. |
| | 3C | Int Line | RO — 0x0 | |
| | 3D | Int Pin | RO — 0x0 | |
| | 3E | Max Lat/Min Gnt | RO — 0x0 | |



## Table 9-7 VF PCIe Configuration Space (Continued)

| Section | Offset | Name | VF behavior | Notes |
|---|---|---|---|---|
| MSI-X Capability | 70 | MSI-X Header | RO — 0xA011 | Next = PCIe capability. |
| | 72 | MSI-x Message Control | per VF | See Section 9.5.2.1. |
| | 74 | MSI-X table Address | RO — as PF | |
| | 78 | MSI-X PBA Address | RO | |
| PCIe Capability | A0 | PCIe Header | RO — 0x0010 | Next = Last capability. |
| | A2 | PCIe Capabilities | RO — 0x0 | |
| | A4 | PCIe Dev Cap | RO — 0x0 | |
| | A8 | PCIe Dev Ctrl | RW | As PF apart from FLR.<br>See Table 9.5.2.2.1. |
| | AA | PCIe Dev Status | per VF | See Table 9.5.2.2.2. |
| | AC | PCIe Link Cap | RO — 0x0 | |
| | B0 | PCIe Link Ctrl | RO — 0x0 | |
| | B2 | PCIe Link Status | RO — 0x0 | |
| | C4 | PCIe Dev Cap 2 | RO — 0x0 | |
| | C8 | PCIe Dev Ctrl 2 | RO — 0x0 | |
| | D0 | PCIe Link Ctrl 2 | RO — 0x0 | |
| | D2 | PCIe Link Status 2 | RO — 0x0 | |
| AER Capability | 100 | AER — Header | RO — 0x15010001 | Next = ARI structure. |
| | 104 | AER — Uncorr Status | per VF | See Section 9.5.2.3. |
| | 108 | AER — Uncorr Mask | RO — 0x0 | |
| | 10C | AER — Uncorr Severity | RO — 0x0 | |
| | 110 | AER — Corr Status | Per PF | |
| | 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. |

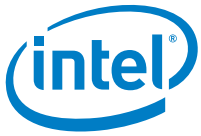
**Table 9-7 VF PCIe Configuration Space (Continued)**

| Section | Offset | Name | VF behavior | Notes |
|---|---|---|---|---|
| ARI Capability | 150 | ARI — Header | 0x0001000E | Next = Last extended Capability. |
| | 154 | ARI — Cap/Ctrl | RO — 0X0 | |

# 9.5.1 Mandatory Configuration Space

## 9.5.1.1 VF Command Register (0x4; RW)

| Bit(s) | RW | Init Val | Description |
|---|---|---|---|
| 0 | RO | 0b | I/O Access Enable (IOAE).<br>RO as zero field. |
| 1 | RO | 0b | Memory Access Enable (MAE).<br>RO as zero field. |
| 2 | RW | 0b | Bus Master Enable (BME).<br>Disabling this bit prevents the associated VF from issuing any memory or I/O requests.<br>*Note:* 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.<br>Requests other than memory or I/O requests are not controlled by this bit.<br>*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.<br>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 | RO | 0b | Special Cycle Enable (SCM).<br>Hard wired to 0b |
| 4 | RO | 0b | MWI Enable (MWIE).<br>Hard wired to 0b. |
| 5 | RO | 0b | Palette Snoop Enable (PSE).<br>Hard wired to 0b. |
| 6 | RO | 0b | Parity Error Response (PER).<br>Zero for VFs. |
| 7 | RO | 0b | Wait Cycle Enable (WCE).<br>Hard wired to 0b. |
| 8 | RO | 0b | SERR# Enable (SERRE).<br>Zero for VFs. |
| 9 | RO | 0b | Fast Back-to-Back Enable (FB2BE).<br>Hard wired to 0b. |



| Bit(s) | RW | Init Val | Description |
|---|---|---|---|
| 10 | RO | 0b | Interrupt Disable (INTD).<br>Hard wired to 0b. |
| 15:11 | RO | 0b | Reserved (RSV). |

### 9.5.1.2 VF Status Register (0x6; RW)

| Bits | RW | Init Val | Description |
|---|---|---|---|
| 2:0 | RO | 0x0 | Reserved (RSV). |
| 3 | RO | 0b | Interrupt Status (IS).<br>Hard wired to 0b. |
| 4 | RO | 1b | New Capabilities (NC).<br>Indicates that the 82599 VFs implement extended capabilities. The 82599 VFs implement a capabilities list, to indicate that it supports MSI-X and PCIe extensions. |
| 5 | RO | 0b | 66 MHz Capable (66E).<br>Hard wired to 0b. |
| 6 | RO | 0b | Reserved (RSV). |
| 7 | RO | 0b | Fast Back-to-Back Capable (FB2BC).<br>Hard wired to 0b. |
| 8 | RW1C | 0b | Data Parity Reported (MPERR). |
| 10:9 | RO | 00b | DEVSEL Timing (DEVSEL).<br>Hard wired to 0b. |
| 11 | RW1C | 0b | Signaled Target Abort (STA). |
| 12 | RW1C | 0b | Received Target Abort (RTA). |
| 13 | RW1C | 0b | Received Master Abort (RMA). |
| 14 | RW1C | 0b | Signaled System Error (SSERR). |
| 15 | RW1C | 0b | Detected Parity Error (DSERR). |

## 9.5.2 PCI Capabilities

### 9.5.2.1 MSI-X Capability

The only registers with a different layout than the PF for MSI-X, is the control register.

**Note:** The message address and data registers in enhanced mode use the first MSI-X entry of each VF in the regular MSI-X table.

#### 9.5.2.1.1 VF MSI-X Control Register (0x72; RW).

| Bits | RW | Init Val | Description |
|---|---|---|---|
| 10:0 | RO | 0x002[1] | Table Size (TS). |
| 13:11 | RO | 0x0 | Reserved (RSV). |
| 14 | RW | 0b | Function Mask (Mask). |
| 15 | RW | 0b | MSI-X Enable (En). |

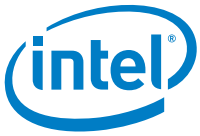
1. Default value is read from the EEPROM.

#### 9.5.2.1.2 MSI-X PBA Register (0x78; RO)

| Bits | RW | Default | Description |
|---|---|---|---|
| 31:3 | RO | 0x400 | PBA Offset.<br>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.<br>This value is changed by hardware to be half of the value programmed to the IOV System Page Size register. |
| 2:0 | RO | 0x3 | PBA BIR.<br>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.<br>A BIR value of three indicates that the PBA is mapped in BAR 3. |

### 9.5.2.2 PCIe Capability Registers

The device control and device status registers have some fields which are specific per VF.



#### 9.5.2.2.1 VF Device Control Register (0xA8; RW)

| Bits | RW | Default | Description |
|---|---|---|---|
| 0 | RO | 0b | Correctable Error Reporting Enable.<br>Zero for VFs. |
| 1 | RO | 0b | Non-Fatal Error Reporting Enable.<br>Zero for VFs. |
| 2 | RO | 0b | Fatal Error Reporting Enable.<br>Zero for VFs. |
| 3 | RO | 0b | Unsupported Request Reporting Enable.<br>Zero for VFs. |
| 4 | RO | 0b | Enable Relaxed Ordering.<br>Zero for VFs. |
| 7:5 | RO | 0b | Max Payload Size.<br>Zero for VFs. |
| 8 | RO | 0b | Not implemented in the 82599. |
| 9 | RO | 0b | Not implemented in the 82599. |
| 10 | RO | 0b | Auxiliary Power PM Enable.<br>Zero for VFs. |
| 11 | RO | 0b | Reserved |
| 14:12 | RO | 000b | Max Read Request Size.<br>Zero for VFs. |
| 15 | RW | 0b | Initiate Function Level Reset.<br>Specific to each VF. |

#### 9.5.2.2.2 VF Device Status Register (0xAA; RO)

| Bits | RW | Default | Description |
|---|---|---|---|
| 0 | RO | 0b | Correctable Detected.<br>Indicates status of correctable error detection. Zero for VF. |
| 1 | RO | 0b | Non-Fatal Error Detected.<br>Indicates status of non-fatal error detection. Zero for VF. |
| 2 | RO | 0b | Fatal Error Detected.<br>Indicates status of fatal error detection. Zero for VF. |
| 3 | RO | 0b | Unsupported Request Detected.<br>Indicates that the 82599 received an unsupported request. This field is identical in all functions. The 82599 can't distinguish which function caused an error. Zero for VF. |

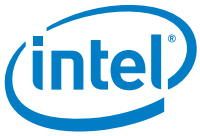
| Bits | RW | Default | Description |
|---|---|---|---|
| 4 | RO | 0b | Aux Power Detected.<br>Zero for VFs. |
| 5 | RO | 0b | Transaction Pending.<br>Specific per VF. When set, indicates that a particular function (PF or VF) has issued non-posted requests that have not been completed. A function reports this bit cleared only when all completions for any outstanding non-posted requests have been received. |
| 15:6 | RO | 0x00 | Reserved. |

### 9.5.2.3 AER Registers

The following registers in the AER capability have a different behavior in a VF function.

#### Uncorrectable Error Status Register (0x104; RW1C)

| Bits | RW | Default | Description |
|---|---|---|---|
| 3:0 | RO | 0x0 | Reserved. |
| 4 | RO | 0b | Data Link Protocol Error Status. |
| 5 | RO | 0b | Surprise Down Error Status (Optional). |
| 11:6 | RO | 0x0 | Reserved. |
| 12 | RW1C | 0b | Poisoned TLP Status. |
| 13 | RO | 0b | Flow Control Protocol Error Status. |
| 14 | RW1C | 0b | Completion Timeout Status. |
| 15 | RW1C | 0b | Completer Abort Status. |
| 16 | RW1C | 0b | Unexpected Completion Status. |
| 17 | RO | 0b | Receiver Overflow Status. |
| 18 | RO | 0b | Malformed TLP Status. |
| 19 | RO | 0b | ECRC Error Status. |
| 20 | RW1C | 0b | Unsupported Request Error Status.<br>When caused by a function that claims a TLP. |
| 21 | RO | 0b | ACS Violation Status. |
| 31:21 | RO | 0x0 | Reserved. |



# 10.0 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® System Management Bus (SMBus) Interface and the Network Controller Sideband Interface (NC-SI) 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 Baseboard Management Controller (BMC), or simply MC.

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 82599 in each platform configuration.

## 10.1 Platform Configurations

This section describes the hardware configurations for platform management. It describes the partitioning of platform manageability among system components and the functionality provided by the 82599 in each of the platform configurations.

The 82599 supports pass-through manageability to an on-board BMC. The link between the 82599 and the BMC is either SMBus or the DMTF NC-SI.

### 10.1.1 On-Board BMC Configurations

Figure 10-1 (left option) depicts an SMBus-only connection between the 82599 and the BMC. The SMBus is used for all communication between the 82599 and the BMC (pass-through traffic, configuration, and status). The protocol details for this configuration follow the SMBus commands described in Section 10.2.2. Figure 10-1 (right option) depicts an NC-SI-only connection between the 82599 and the BMC. The NC-SI is used for all communication between the 82599 and the BMC (pass-through traffic, configuration, and status). The protocol details for this configuration follow the DMTF NC-SI protocol.

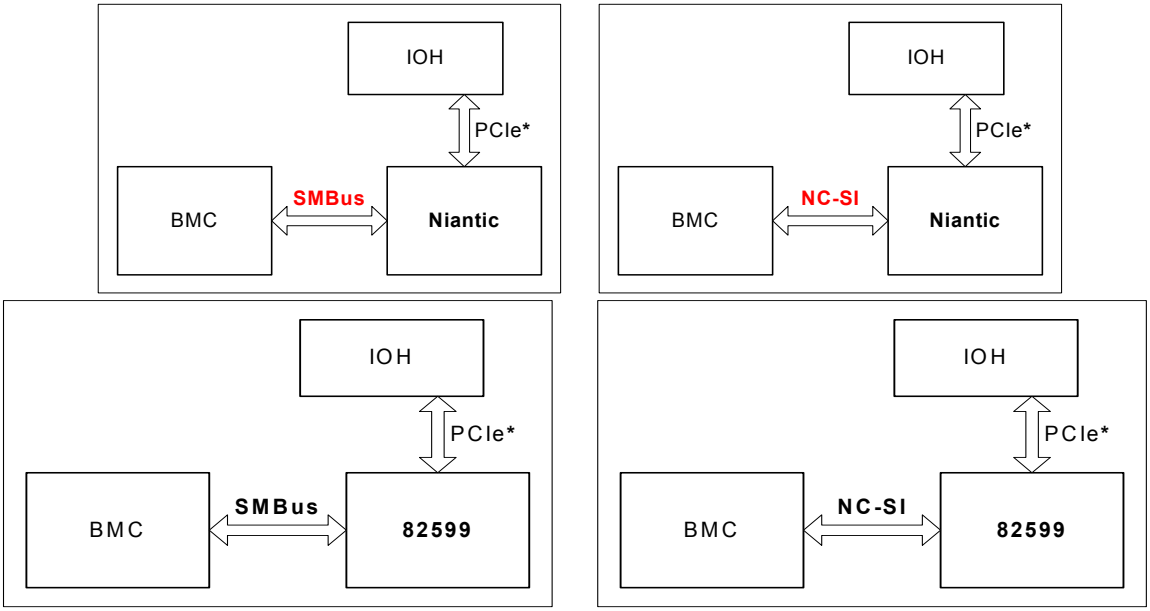



**Figure 10-1 the 82599 to BMC Connectivity Through SMBus Link or NC-SI Link**

Refer to the sections that follow for a description of the traffic types that use the NC-SI and/or SMBus interfaces.

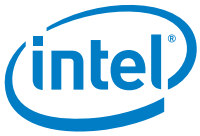
### 10.1.2 82599 NIC

BMC connection to a NIC is not expected.

## 10.2 Pass Through (PT) Functionality

The 82599 supports traffic pass through to an external BMC. The pass-through traffic is carried through an NC-SI interface or SMBus (legacy devices) based on the Redirection Sideband Interface setting in the EEPROM (loaded on power up). The usable bandwidth for either direction is up to 100 Mb/s in NC-SI mode and up to 400 Kb/s in SMBus mode. Supplemental descriptions on SMBus and NC-SI interfaces can be found in Section 3.2 and in Section 3.3. The following list describes usage models for the pass through traffic:

- BMC management traffic
- Keyboard or mouse traffic for KVM (low data rate)
- Video traffic for KVM (low average rate of 150 Kb/s to 200 Kb/s) — transmit only
- USB 2.0 redirect (up to 50 Mb/s)
- IDE redirect for remote CD/floppy (rate — priority 1 — CDx7 = 1.05 Mb/s. Priority 2 — CDx24 = 64 Mb/s



- Serial Over LAN (SoL) — 300 Kb/s

## 10.2.1 DMTF NC-SI Mode

The 82599 supports all the mandatory features of the DMTF NC-SI spec rev1.0.0a.

### 10.2.1.1 Supported Features

Table 10-1 lists the commands supported by the 82599.

**Table 10-1 Supported NC-SI Commands**

| Command | Supported |
|---|---|
| Clear Initial State | Yes |
| Get Version ID | Yes |
| Get Parameters | Yes |
| Get Controller Packet Statistics | No |
| Get Link Status | Yes |
| Enable Channel | Yes |
| Disable Channel | Yes |
| Reset Channel | Yes |
| Enable VLAN | Yes (filtering only by the VLAN ID. No filtering by the User priority) |
| Disable VLAN | Yes |
| Enable BCast | Yes |
| Disable BCast | Yes |
| Set MAC Address | Yes |
| Get NC-SI Statistics | Yes, partially |
| Set NC-SI Flow Control | No |
| Set Link Command | Yes (support for 10 GbE is not fully defined in the specification) |
| Enable Global MCast Filter | Yes |
| Disable Global MCast Filter | Yes |
| Get Capabilities | Yes |

**Table 10-1 Supported NC-SI Commands (Continued)**

| Command | Supported |
|---|---|
| Set VLAN Filters | Yes |
| AEN Enable | Yes |
| Get Pass-Through Statistics | Yes, partially |
| Select Package | Yes |
| Deselect Package | Yes |
| Enable Channel Network Tx | Yes |
| Disable Channel Network Tx | Yes |
| OEM Command | Yes |

Table 10-2 lists the NC-SI features supported by the 82599:

**Table 10-2 Optional NC-SI Features Support**

| Feature | Supported | Details |
|---|---|---|
| AENs | Yes. | *Note:* The driver state AEN might be emitted up to 15 seconds after actual driver change. |
| Get NC-SI statistics command | Yes, partially | Supports the following counters: 1-4, 7 |
| Get NC-SI pass-through statistics command | Yes, partially | Supports the following counters: 2<br>Supports the following counters only when the operating system is down:<br>1, 6, 7 |
| VLAN modes | Yes, partially | Supports only modes 1,3 |
| Buffering capabilities | Yes | 8 K |
| Ethernet MAC address filters | Yes | Supports 2 Ethernet MAC addresses as mixed per port |
| Channel count | Yes | Supports 2 channels |
| VLAN filters | Yes | Supports 8 VLAN filters per port |
| Broadcast filters | Yes | Supports the following filters:<br>• ARP<br>• DHCP<br>• NetBIOS |
| Multicast filters | Yes | Supports the following filters (supported only when all three are enabled):<br>• IPv6 neighbor advertisement<br>• IPv6 router advertisement<br>• DHCPv6 relay and server multicast |

**Table 10-2 Optional NC-SI Features Support (Continued)**

| Feature | Supported | Details |
|---|---|---|
| NC-SI flow control command | No | |
| Hardware arbitration | No | |

## 10.2.2 SMBus Pass Through (PT) Functionality

When operating in SMBus mode, the 82599 provides the following manageability services to the BMC on top of the pass through traffic functionality:

- ARP handling — The 82599 can be programmed to auto-ARP replying for ARP request packets and sending gratuitous ARP to reduce the traffic over the SMBus.
- Teaming and fail-over — The 82599 can be configured to either teaming or non-teaming modes. When operated in teaming mode the 82599 can also provide auto fail-over configurations as detailed in the following sub-sections.
- Default configuration of filters by EEPROM — When working in SMBus mode, the default values of the manageability receive filters can be set according to the PT LAN (Section 6.4.3) and flex TCO EEPROM structure (Section 6.4.5).

### 10.2.2.1 Pass Through (PT) Modes

PT configuration depends on how the LAN ports are configured. If the LAN ports are configured as two different channels (non-teaming mode) then the 82599 is presented on the manageability link as two different devices (via two different SMBus addresses) on which each device is connected to a different LAN port. In this mode (the same as in the LAN channels), there is no logical connection between the two devices. In this mode, the fail-over between the two LAN ports are done by the external BMC (by sending/receiving packets through different devices). The status reports to the BMC, ARP handling, DHCP and other pass through functionality are unique for each port.

When the 82599 operates in teaming mode, it presents itself on the SMBus as a single device. In this mode, the external BMC is not aware that there are two LAN ports. The 82599 determines how to route the packets that it receives on the manageability channel according to the fail-over algorithm. The status reports to the BMC and other pass through configurations are common to both ports.

In pass through mode most of the manageability traffic is handled by the BMC. However, portion of the network traffic can be offloaded and by the 82599 as described in the following sub-sections. This configuration can be done by issuing configuration commands over the SMBus channel or the 82599 can load it from its EEPROM at power up (or both).

#### 10.2.2.2 LAN Fail-Over in LAN Teaming Mode

Manageability fail-over is the ability to detect that the LAN connection on one port is lost, and enable the other port for manageability traffic. When the 82599 operates in teaming mode, the operating system and the external BMC consider it as one logical network device. The decision on which of the 82599 ports are used is done internally by the 82599 (or by the ANS driver in case of the regular receive/transmit traffic). This section deals with fail-over in teaming mode only. In non-teaming mode, the external BMC should consider the 82599's network ports as two different network devices, and the BMC is solely responsible for the fail-over mechanism.

In teaming mode, the 82599 maps both network ports into a single SMBus slave device. The 82599 automatically handles the configurations of both network ports. Thus, for configurations, receiving and transmitting the BMC should consider both ports as a single entity.

When the currently active transmission port becomes unavailable (such as the link is down), the 82599 automatically switches transmission to the other port. Thus, as long as one of the ports is valid, the BMC will have a valid link indication for the SMBus slave.

**Note:** As both ports might be active (such as with a valid link) packets might be received on the currently non-active port. To avoid packet duplication, failover should not be enabled when connected to a hub.

**Note:** Fail over and teaming are not supported in NC-SI mode.

##### 10.2.2.2.1 Port Switching (Fail-Over)

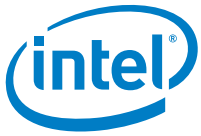
While in teaming mode, transmit traffic is always transmitted by the 82599 through only one of the ports at any given time. The 82599 might switch the traffic transmission between ports under any of the following conditions:

1. The current transmitting port link is not available
2. The preferred primary port is enabled and becomes available for transmission.

##### 10.2.2.2.2 Driver Interactions

When the LAN driver is present, the decision to switch between the two ports is done by the driver. When the driver is absent, this decision is done internally by the 82599.

**Note:** When the driver releases teaming mode (such as, when the system state changes), the 82599 reconfigures the LAN ports to teaming mode. The 82599 accomplishes this by re-setting the Ethernet MAC address of the two ports to be the teaming address in order to re-start teaming. This is followed by transmission of gratuitous ARP packets to notify the network of teaming mode re-setting.



#### 10.2.2.2.3 Fail-Over Configuration

Fail-over operation is configured through the fail-over configuration structure (see Section 10.2.2.2.4).

The BMC should configure this register after a the 82599 initialization indication (following a firmware reset). The different configurations available to the BMC are detailed in this section.

**Note:** In teaming mode both ports should be configured with the same receive manageability filters parameters (EEPROM sections for port 0 and port 1 should be identical).

**Preferred Primary Port** — The BMC might choose one of the network ports (LAN0 or LAN1) as a preferred primary port for packet transmission. The 82599 always switches to the preferred primary port when it is available.

**Gratuitous ARPs** — In order to notify the link partner that a port switching has occurred, the 82599 can be configured to automatically send gratuitous ARPs. These gratuitous ARPs cause the link partner to update its ARP tables to reflect the change. The BMC might enable/disable gratuitous ARPs, configure the number of gratuitous ARPs or the interval between them by modifying the Fail Over configuration register.

**Link Down Timeout** — The BMC can control the timeout for a link to be considered invalid. The 82599 waits this timeout before attempting to switch from an inactive port.

#### 10.2.2.2.4 Fail-Over Structure

The fail-over structure (listed in the following table) is loaded on power up from the EEPROM (see Section 6.4.4.5), or through the Set Fail-Over Configuration host command by the LAN driver (see Section 10.5.3.8). The bits in this register can also be modified by the 82599 hardware reflecting its current state.

| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| RMP0EN | 0 | RO | 0x1 | RCV MNG port 0 Enable.<br>When this bit is set, it reports that MNG traffic is received from port 0. |
| RMP1EN | 1 | RO | 0x1 | RCV MNG port 1 Enable.<br>When this bit is set, it reports that MNG traffic is received from port 1. |
| MXP | 2 | RO | 0x0 | MNG XMT Port.<br>0b = MNG traffic should be transmitted through port 0.<br>1b = MNG traffic should be transmitted through port 1. |
| PRPP | 3 | RW | 0x0 | Preferred Primary Port.<br>0b = Port 0 is the preferred primary port.<br>1b = Port 1 is the preferred primary port. |
| PRPPE | 4 | RW | 0x0 | Preferred Primary Port enables. |
| Reserved | 5 | RO | 0x0 | Reserved. |
| RGAEN | 6 | RW | 0x0 | Repeated Gratuitous ARP Enable.<br>If this bit is set, the 82599 sends a configurable number of gratuitous ARP packets (GAC bits of this register) using configurable interval (GATI bits of this register) after the following events: System move to Dx, or fail-over event initiated the 82599. |

| Field | Bit(s) | RW | Init Val | Description |
|---|---|---|---|---|
| Reserved | 7 | RO | 0x0 | Reserved. |
| Reserved | 8 | RO | 0x0 | Reserved. |
| TFOENODX | 9 | RW | 0x0 | Teaming Fail Over Enable on Dx.<br>Enable fail-over mechanism. Bits 3-8 are valid only if this bit is set. |
| Reserved | 10-11 | RO | 0x0 | Reserved. |
| GAC | 12-15 | RW | 0x0 | Gratuitous ARP counter.<br>Counts the number of gratuitous ARP that should be done after a fail-over event and after a move to Dx. When it is set to zero, there is no limit on the gratuitous ARP packets. |
| LDFOT | 16-23 | RW | 0x0 | Link Down Fail-Over Time.<br>Defines the time in seconds the link should be down before doing a fail over to the other port.<br>This is also the time that the primary link should be up (after it was down) before the 82599 switches back to the primary port. |
| GATI | 24-31 | RW | 0x0 | Gratuitous ARP Transmission Interval.<br>Defines the GAP in seconds before retransmission of gratuitous ARP packets. |

### 10.2.2.3 ARP Handling

Independent of the management interface, the 82599 can be programmed by the BMC to provide ARP services. The 82599 supports auto-ARP replying for ARP request packets and sending Gratuitous ARP. Auto-ARP is done in both ports in either modes: dual-channel and one-channel. In dual-channel mode, each channel uses its own IP and Ethernet MAC address (either the operating system Ethernet MAC address or independent addresses). In one-channel mode, both ports use the same IP and Ethernet MAC address and the ARP is responded to through the port it was received.

The following ARP parameters are loaded from the EEPROM on power up or configured through the management interface:

- ARP auto-reply enabled
- ARP IP address (to filter ARP packets)
- ARP Ethernet MAC Addresses (for ARP response)

When an ARP request packet is received on the wire and ARP auto-reply is enabled, the 82599 checks the targeted IP address (after the packet has passed L2 checks and ARP checks). If the targeted IP matches the 82599 IP configuration, then it replies with an ARP response. The 82599 responds to the ARP request targeted to the ARP IP address with its ARP Ethernet MAC address. In a case where there is no match, the 82599 silently discards the packets. If the 82599 is not configured to do auto-ARP response, it forwards the ARP packets to the BMC.

When the external BMC uses the same IP and MAC of the operating system, the ARP operation should be coordinated with the operating system operation. In this mode, the external BMC has the responsibility and ARP auto-reply should be disabled.

**Note:** When configured in NC-SI mode, the 82599 does not provide ARP services. All ARP handling is done by the BMC.

# 10.3 Manageability Receive Filtering

## 10.3.1 Overview and General Structure

For completeness, this section summarizes the MAC and VLAN filters described in Section 7.1.1.1 and Section 7.1.1.2. In addition, this section describes the manageability receive packet filtering flow. The description applies to any of the 82599 LAN ports. Receive packet filtering can have one of the following routing results:

- Discard packets (packets that do not pass the host nor manageability filtering)
- Send packets to host memory (default hardware setting)
- Send packets to the external BMC (two modes):
  - Receive All — All received packets are routed to the BMC in this mode. It is enabled by setting the RCV_TCO_EN bit (which enables packets to be routed to the BMC) and RCV_ALL bit (which routes all packets to the BMC) in the MANC register.
  - Receive Filtering — In this mode only some of the packet types are directed to the manageability block. The BMC should set the RCV_TCO_EN bit together with the required packet types bits in the manageability filtering registers. Note that the RCV_ALL bit must be cleared).
- Send packets to both the external BMC and host memory:
  - The BMC can enable this mode by setting the EN_MNG2HOST bit in the MANC register and enable specific packet types in the MANC2H register.

The BMC controls its packet filtering by programming the receive manageability filters listed in the following table. These registers are not write-accessible by the host (protecting the BMC from erroneous/malicious host software).

| Filters | Functionality | When Reset? |
|---|---|---|
| Filters enable | General configuration of the manageability filters. | Internal Power On Reset |
| Manageability to host | Enables routing of manageability packets to host. | Internal Power On Reset |
| Manageability decision filters [7:0] | Configuration of manageability decision filters. | Internal Power On Reset |
| MAC address [3:0] | Four unicast MAC manageability addresses. | Internal Power On Reset |
| VLAN filters [7:0] | Eight VLAN tag values. | Internal Power On Reset |
| UDP/TCP port filters [15:0] | 16 destination port values. | Internal Power On Reset |
| Flexible 128-byte TCO filters | Length values for four flex TCO filters. | Internal Power On Reset |
| IPv4 and IPv6 address filters[3:0] | IP address for manageability filtering. | Internal Power On Reset |
| L2 EtherType filters [3:0] | Four L2 EtherType values. | Internal Power On Reset |

Manageability filtering follows these steps and are detailed in the following sections:

1. L2 Ethernet MAC address and VLAN filtering
2. L3/L4 manageability filters — Port, IP, flex filters (packets must also match the above L2 filtering).

Filtering exceptions:

- Fragmented packets can be routed to manageability but not parsed beyond the IP header.
- Packets with L2 errors (CRC, alignment, etc.) are never forwarded to manageability.

**Note:** Jumbo packets above 2 KB are not expected to be received by the manageability data path. If the manageability unit uses a dedicated Ethernet MAC address/VLAN tag, it should not use further L3/L4 filters on top of it. Otherwise, packets that match the L2 filters but fail the L3/L4 filters are routed to the host.

The complete filtering flow is described in the following flow diagram:

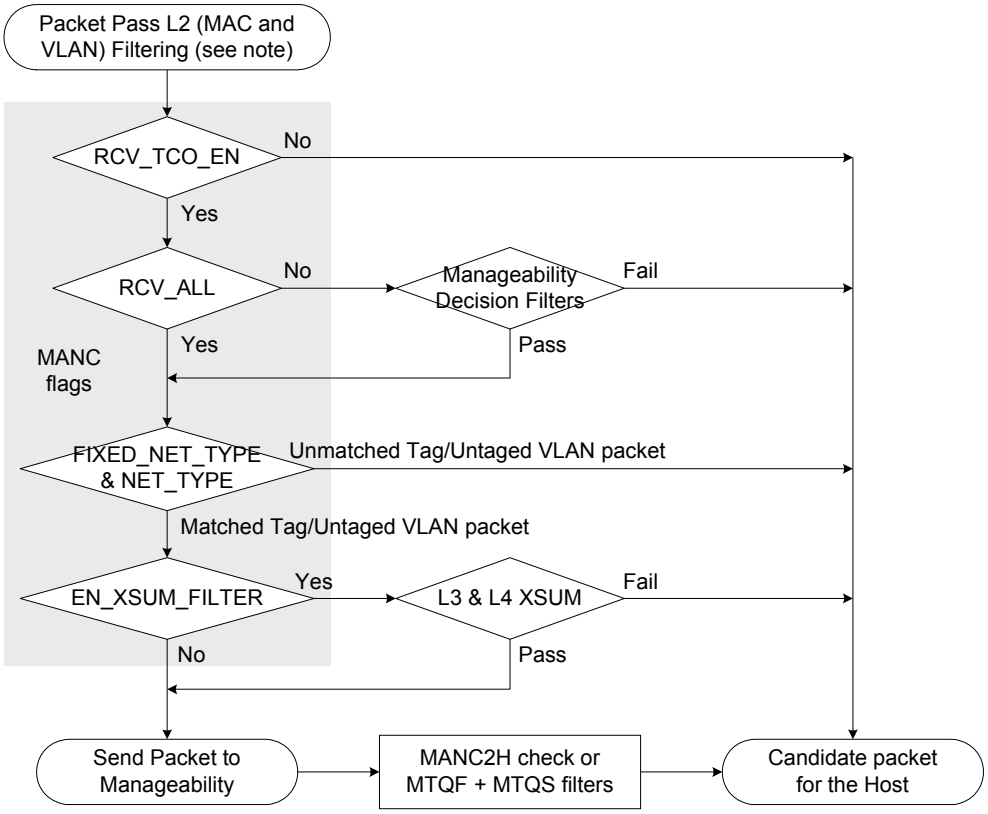
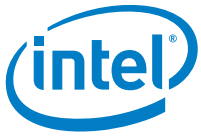



**Figure 10-2 Flow Diagram**

**Note:** L2 MAC address and VLAN filtering are described in Section 7.1.1.1 and Section 7.1.1.2.



## 10.3.2 L2 EtherType Filters

Packets are compared against the EtherType filters programmed in the METF.EType (up to 4 filters) and the result is incorporated to the decision filters.

Each of the manageability EtherType filters can be configured as pass (positive) or reject (negative) polarity. When negative polarity filters are used, all negative filters should be included in all enabled decision filters.

Examples for usages of the L2 EtherType filters are:

- Block routing of packets with the NC-SI EtherType from being routed to the BMC. The NC-SI EtherType is used for communications between the BMC on the NC-SI link and the 82599. Packets coming from the network are not expected to carry this EtherType and such packets are blocked to prevent attacks on the BMC.
- Determine the destination of 802.1X control packets. The 802.1X protocol is executed at different times in either the BMC or by the host. The L2 EtherType filters are used to route these packets to the proper agent.

## 10.3.3 VLAN Filters - Single and Double VLAN Cases

The 82599 supports eight VLAN filters per port defined by the MAVTV[n] and controlled by the MANC register as described in the text that follows.

- When MANC.NET_TYPE = 1b and MANC.FIXED_NET_TYPE = 1b (pass only VLAN tagged packets)
  - A packet without any VLAN or a single VLAN header is not routed to manageability
  - A packet with 2 VLANs is a candidate for manageability
- When MANC.NET_TYPE = 0b and MANC.FIXED_NET_TYPE = 1b (pass only un-tagged packets)
  - A packet without any VLAN or a single VLAN header is a candidate for manageability
  - A packet with 2 VLANs is not routed to manageability
- When MANC.FIXED_NET_TYPE = 0b (both tagged and untagged packets are candidates for manageability)
  - A packet with no VLAN header skips successfully to the next filtering level
  - A packet with a single VLAN or 2 VLANs are filtered by its VLAN header as described in Section 7.1.1.2

## 10.3.4 L3 and L4 Filters

ARP Filtering: The 82599 supports filtering of both ARP request packets (initiated externally) and ARP responses (to requests initiated by the BMC or the 82599).

Neighbor Discovery Filtering: The 82599 supports filtering of neighbor discovery packets. Neighbor discovery filters use the IPV6 destination address filters defined in the MIPAF registers (such as match to any of the enabled IPv6 addresses).

Port 0x298/0x26F Filtering: The 82599 supports filtering by fixed destination ports numbers: 0x26F and 0x298.

Flex Port Filtering: The 82599 implements 16 flex destination port filters. The 82599 directs packets whose L4 destination port matches the value of the respective word in the MFUTP registers. The BMC must ensure that only valid entries are enabled in the decision filters that follow.

Flex TCO Filters: See Section 10.3.4.1.

IP Address Filtering: The 82599 supports filtering by IP address through dedicated IPv4 and IPv6 address filters to manageability. Two modes are possible, depending on the value of the MANC.EN_IPv4_FILTER bit:

- EN_IPv4_FILTER = 0b: The 82599 provides four IPv6 address filters.
- EN_IPv4_FILTER = 1b: The 82599 provides three IPv6 address filters and four IPv4 address filters.
- The MFVAL register indicates which of the IP address filters are valid (contains a valid entry and should be used for comparison).

Checksum Filter: If bit MANC.EN_XSUM_FILTER is set, the 82599 directs packets to the BMC only if they match all other filters previously described as well as pass L3/L4 checksum (if it exists).

### 10.3.4.1 Flexible 128 Bytes Filter (TCO Filters)

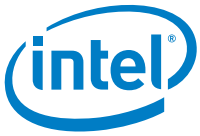
#### 10.3.4.1.1 Overview

The flexible 128 filters are a set of filters designed to enable dynamic filtering of received packets. These filters are part of the manageability receive filters. The filters do not make a decision on the packet's destination. They participate in the decision mechanism for each received packet (Section 10.3.5).

Each filter enables a flexible testing of the first 128 bytes of the packet against a given value. The filter also enables testing of specific bytes by defining a byte-wise mask on the filter.

The 82599 provides four flex TCO filters. Each filter looks for a pattern match within the 1st 128 bytes of the packet. The BMC must ensure that only valid entries are enabled in the decision filters.

**Note:** The flex filters are temporarily disabled when read or written by the host. Any packet received during a read or write operation is dropped. Filter operation resumes once the read or write access completes.



#### 10.3.4.1.2 Structure

Each filter is composed of the following fields:

1. Flexible Filter Length: This field indicates the number of bytes in the packet header that should be inspected. This field also indicates the minimal length of packets in order to be inspected by the filter. A packet below that length is not inspected by the filter. Valid values for this field are: 8*n, where n=1…8.
2. Data: This is a set of up to 128 bytes comprising the values that the header bytes of each packet are tested against.
3. Mask: This is a set of 128 bits corresponding to the 128 data bytes that indicate for each corresponding byte if is tested against its corresponding byte.

Overall, each filter tests the first 128 bytes (or less) of a packet, where not necessarily all bytes must be tested.

#### 10.3.4.1.3 Programming

Programming each filter is done using the following two commands (NC-SI or SMBus) in a sequential manner:

1. Filter Mask and Length. This command configures the following fields.
   a. Mask: A set of 16 bytes containing the 128 bits of the mask. Bit 0 of the first byte corresponds to the first byte on the wire.
   b. Length: A 1-byte field indicating the length.
2. Filter Data.

   The filter data is divided into groups of bytes. as follows:

| Group | Test Bytes |
|---|---|
| 0x0 | 0-29 |
| 0x1 | 30-59 |
| 0x2 | 60-89 |
| 0x3 | 90-119 |
| 0x4 | 120-127 |

Each group of bytes needs to be configured using a separate command, where the group number is given as a parameter.

The command has the following parameters:

a. Group number. A 1-byte field indicating the current group addressed.

b. Data bytes. Up to 30 bytes of test-bytes for the current group.

## 10.3.5 Manageability Decision Filters

The manageability decision filters are a set of eight filters with the same structure (MDEF[7:0] and MDEF_EXT[7:0]). The filtering rule for each decision filter is programmed by the BMC and defines which of the L2, VLAN, and manageability filters participate in the decision (host software can't modify their setting). A packet that passes at least one set of decision filters is directed to manageability and possibly to the host as well. The inputs to each decision filter are:

- Packet passed a valid management L2 unicast address filter.
- Packet is a broadcast packet.
- Packet has a VLAN header and it passed a valid manageability VLAN filter.
- Packet matched one of the valid IPv4 or IPv6 manageability address filters.
- Packet is a multicast packet.
- Packet passed ARP filtering (request or response).
- Packet passed neighbor discovery filtering.
- Packet passed 0x298/0x26F port filter.
- Packet passed a valid flex port filter.
- Packet passed a valid flex TCO filter.
- Packet passed or failed an L2 EtherType filter.

The structure of each of the decision filters is shown in Figure 10-3 . A boxed "x.y" number indicates that the input is conditioned on a mask bit "y" defined in register index "x", while x=0 denotes MDEF and x=1 denotes MDEF_EXT. The decision filter rules are as follows:

- Any bit set in the MDEF and MDEF_EXT registers enables its corresponding filter. Any filter that is not enabled in the MDEF and MDEF_EXT registers is ignored. If all bits in the MDEF and MDEF_EXT registers of a specific decision filter are cleared, it is disabled and ignored.
- All enabled AND filters must pass for the decision filter to match.
- If at least one OR filter is enabled, then at least one of the enabled OR filters must pass for the decision filter to match.

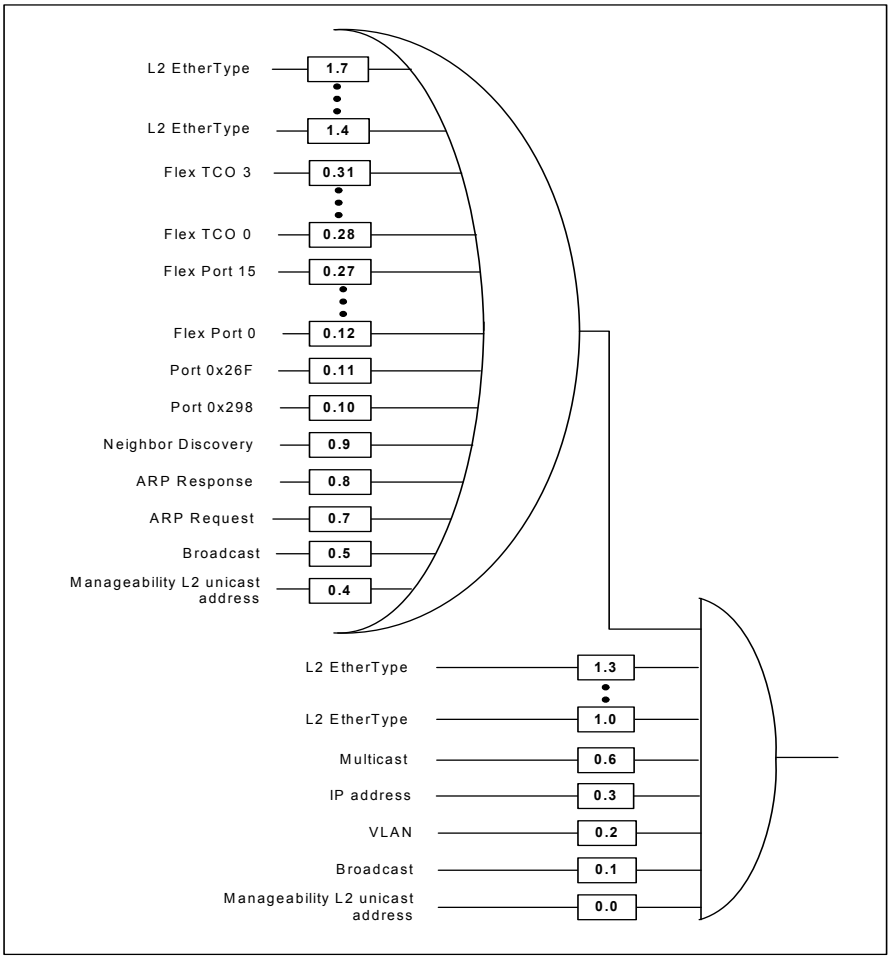



**Figure 10-3 Manageability Decision Filters**

## 10.3.6 Possible Configurations

This section describes possible ways of using the management filters. Actual usage might vary.

### Dedicated MAC packet filtering

- Select one of the eight rules for broadcast filtering
- Set bit 0 of the decision rule to enforce Ethernet MAC address filtering
- Set other bits to qualify which packets are allowed to pass through. For example:
  - Set bit 2 to qualify with manageability VLAN
  - Set bit 3 to qualify with a match to an IP address
  - Set any L3/L4 bits (30:7) to qualify with any of a set of L3/L4 filters

### Broadcast packet filtering

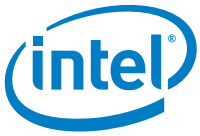
- Select one of the eight rules for broadcast filtering
- Set bit 1 of the decision rule to enforce broadcast filtering
- Set other bits to qualify which broadcast packets are allowed to pass through. For example:
  - Set bit 2 to qualify with manageability VLAN
  - Set bit 3 to qualify with a match to an IP address
  - Set any L3/L4 bits (30:7) to qualify with any of a set of L3/L4 filters

### VLAN packet filtering

- Select one of the eight rules for VLAN filtering
- Set bit 2 of the decision rule to enforce VLAN filtering
- Set other bits to qualify which VLAN packets are allowed to pass through. For example:
  - Set any L3/L4 bits (30:7) to qualify with any of a set of L3/L4 filters

IPv6 filtering is done via the following IPv6-specific filters:

- IP unicast filtering — requires filtering for link local address and a global address. Filtering setup might depend on whether an Ethernet MAC address is shared with the host or dedicated to manageability:
  - Dedicated Ethernet MAC address (such as dynamic address allocation with DHCP does not support multiple IP addresses for one Ethernet MAC address). In this case, filtering can be done at L2 using two dedicated unicast MAC filters.
  - Shared Ethernet MAC address (such as static address allocation sharing addresses with the host). In this case, filtering needs to be done at L3, requiring two IPv6 address filters, one per address.



- A Neighbor discovery filter — The 82599 supports IPv6 neighbor discovery protocol. Since the protocol relies on multicast packets, the 82599 supports filtering of these packets. IPv6 multicast addresses are translated into corresponding Ethernet multicast addresses in the form of 33-33-xx-xx-xx-xx, where the last 32 bits of the address are taken from the last 32 bits of the IPv6 multicast address. Therefore, two direct MAC filters can be used to filter IPv6 solicited-node multicast packets as well as IPv6 all node multicast packets.

### Receive filtering with shared IP — CPMP

When the BMC shares the MAC and IP address with the host, receive filtering is based mainly on identifying specific flows through port allocation. The following setting can be used:

Select one of the eight rules:

- Set a manageability dedicated MAC filter to the host Ethernet MAC address and set bit 0 in the MNG_ FILTER_RULE register.
- If VLAN is used for management, load one of more management VLAN filters and set bit 2 in the MNG_ FILTER_RULE register
- ARP filter / neighbor discovery filter is enabled when the BMC is responsible to handle the ARP protocol. Set bit 7 or bit 8 in the MNG_ FILTER_RULE register for this functionality.
- Program flex port filters with the port values for management flows such as DHCP, HTTP, HTTPS, SMWG, SoL/IDER/KVM, WS-MAN, Telnet, USB redirection, SSH, DNS, and more. Set the respective bits 26:11 in the MNG_ FILTER_RULE register.
- An IP address filter can be loaded as well by setting bit 3 in the MNG_ FILTER_RULE register.
- Management flex filters are programmed to correspond to remaining flows such as DNS update response packets. Set appropriate bits 30:27 in the MNG_ FILTER_RULE register.

# 10.4 LinkSec and Manageability

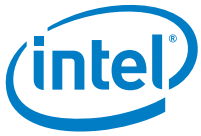
For details on LinkSec and the role of manageability in it, see Section 7.8.

Pass-through mode is supported in a LinkSec environment in one of the following modes of operations:

- Management traffic not protected by LinkSec — The management traffic from and to the BMC is carried over a separate Ethernet MAC address and/or a separate VLAN and the network switch is configured to enable such traffic to pass unprotected.
- Management traffic is protected by LinkSec — The 82599 supports a single secure channel for both host and BMC. At a given time, the host and BMC can be active or inactive. When only BMC is active, it acts as the KaY controlling the secured channel. The host can act as the KaY when it is functional and after it acquires control over LinkSec. In this case, the BMC uses the secured channel set by the host. Even when operating in this mode, the BMC can transmit packets on the clear (as required for 802.1x control packets). The BMC must disable MACsec operation before sending such packets and re-enable MACsec operation afterwards. The messages that control MACsec operation are described in Section 10.5.1.15.

The 82599 provides the following functionality that enables management traffic over the same secure channel with the host:

- Handover of LinkSec ownership between the BMC and the host. Several transitions in ownership are possible:
  - Power-on — The 82599 powers up with LinkSec **not** being owned by the BMC. If the BMC is configured for LinkSec, it takes ownership over LinkSec as follows. If the BMC is not configured for LinkSec, the host takes ownership when it boots. If LinkSec is not owned by the BMC, the host is not required for any handshake with the BMC as there are cases where the BMC is not connected to the 82599. If there is a race between the BMC and the host, the BMC wins over LinkSec, and the host is then interrupted so that the LinkSec resources are not accessible.
  - Handover of LinkSec responsibility from BMC to host — The host can initiate a transfer of ownership from the BMC (such as on operating system boot).
  - Handover of LinkSec responsibility from host to BMC — The host can initiate a transfer of ownership to the BMC (such as on entry to low power state). This is done through the host slave command interface.
  - Forced handover of LinkSec responsibility from host to BMC — The BMC can acquire ownership of LinkSec on its own, for example when the host fails to acquire a secure channel. See Section 10.4.1 for the different transition sequences.
- Configuration of LinkSec resources by the BMC — When the BMC owns the secure channel, it configures LinkSec operation through the SMBus or NC-SI vendor-specific commands (see Section 10.5.1.15).
- Alerts — The 82599 initiates an SMBus or NC-SI alert to the BMC on several LinkSec events as follows (see alerts message format in Section 10.5.1.16, Section 10.5.2.2.3, and Section 10.5.2.2.8).
  - Packet arrived with a LinkSec error (no SA match, replay detection, or a bad LinkSec signature).



  - — Key-exchange event — relevant on Tx when the packet number counter reaches the exhaustion threshold as described in Section 7.8.5.1.
  - — Host request for LinkSec ownership.
  - — Host request to relinquish LinkSec ownership.
- Interrupt causes — The 82599 issues a management interrupt to the host on the following LinkSec events:
  - — Acknowledge of handover of LinkSec responsibility from BMC to host.
  - — Forced handover of LinkSec responsibility from host to BMC.

  The host might identify the ownership status by reading the *Operating System Status* field in the LSWFW register.

## 10.4.1 Handover of LinkSec Responsibility Between BMC and Host

### 10.4.1.1 KaY Ownership Release by the Host

The following procedure is used by the host in order to release ownership of the LinkSec capability. This procedure is usually done before an ordered shutdown of the host.

- The host should stop accessing the LinkSec registers and set the *Release LinkSec* bit in the LSWFW register.
- Setting the *Release LinkSec* bit causes an interrupt to the firmware that is forwarded to the BMC.
- The BMC then takes ownership as described in Section 10.4.1.2.
- The host can then wait for an interrupt from the firmware indicating that the BMC took the KaY ownership.

### 10.4.1.2 KaY Ownership Takeover by BMC

As previously mentioned, the BMC can acquire ownership over LinkSec either by ownership relinquish by the host or without any negotiation (such as on power-up and on a forced transition when the host failed to bring up a LinkSec connection). The BMC acquires ownership of LinkSec by taking the following actions:

- Locking access to LinkSec resources to the host by setting the *Lock LinkSec Logic* bit in the LSWFW register.
- Blocking host packets' transmission from the wire by setting the *Block Host Traffic* bit in the LSWFW register.
- Set the *OS Status* field in the LSWFW register to 1b indicating a BMC takeover of the LinkSec logic.
- Issue a manageability event interrupt to the host.

### 10.4.1.3 KaY Ownership Request by the Host

The following procedure is used by the host in order to request ownership of the LinkSec capability:

- The host should read the LSWFW.OS status field to check if the KaY is currently owned by the BMC.
- if KaY is owned by the BMC, then the host should set the *Request LinkSec* bit in the LSWFW register prior to assuming responsibility over LinkSec connection.
- Setting the *Request LinkSec* bit causes an interrupt to the firmware that is forwarded to the BMC.
- The host should then wait for an interrupt from the firmware indicating that the BMC released the KaY ownership.
- Following the manageability interrupt, the host should check the *OS Status* and *Lock LinkSec Logic* fields in the LSWFW register to make sure the BMC released the KaY ownership.

### 10.4.1.4 KaY Ownership Release by BMC

In order to release ownership of LinkSec, the BMC should take the following actions:

- Disconnect the LinkSec connection with the switch (such as EAP logoff).
- Clear the *Lock LinkSec Logic* bit in the LSWFS register enabling the host setting of the LinkSec registers.
- Clear the *OS Status* bit to 0b in the LSWFS register indicate a LinkSec release.
- Issue a manageability event interrupt to the host.
- Poll the connection state to check if the LinkSec channel was set by the host.

If the BMC decides to deny the release request, it silently ignores the request.

### 10.4.1.5 Control Registers

The complete set of manageability registers are described in Section 8.2.3.26 and Section 8.2.3.26. The following configuration fields are dedicated for manageability control over LinkSec:

| LSWFW Field | LSWFW Field Functionality |
|---|---|
| Block Host Traffic | Enables or disables host transmit traffic for this PCI function from going to the wire.<br>Default is to enable. |
| OS Status | Set by firmware to indicate the status of the LinkSec ownership:<br>0b = LinkSec owned by host (default).<br>1b = LinkSec owned by BMC. |
| LinkSec Request | Bit used by host to request KaY ownership. |

| LSWFW Field | LSWFW Field Functionality |
|---|---|
| LinkSec Release | Bit used by host to release KaY ownership. |
| Lock LinkSec Logic | Serves two purposes. It indicates who owns LinkSec (default value is host ownership). Second, it enables or disables host accesses to the LinkSec registers. Default is to enable. The following registers are blocked:<br>LSECTXCAP; LSECRXCAP; LSECTXCTRL; LSECRXCTRL; LSECTXSCL; LSECTXSCH; LSECTXSA; LSECTXPN0; LSECTXPN1; LSECTXKEY0 (4 registers); LSECTXKEY1 (4 registers); LSECRXSCL; LSECRXSCH; LSECRXSA (0 and 1); LSECRXSAPN (0 and 1); LSECRXKEY (4 registers / SA); LSECTXUT; LSECTXPKTE; LSECTXPKTP; LSECTXOCTE; LSECTXOCTP; LSECRXUTnS; LSECRXUTyS; LSECRXOCTE; LSECRXOCTP; LSECRXBAD; LSECRXNOSCInS; LSECRXNOSCIyS; LSECRXNOSCI; LSECRXDELAY; LSECRXLATE; LSECRXOK; LSECRXINVCK; LSECRXINVST; LSECRXNSAST; LSECRXNSA |

# 10.5 Manageability Programming Interfaces

## 10.5.1 NC-SI Programming

The 82599 supports the mandatory NC-SI commands as listed in Table 10-1. On top of these commands, the 82599 also supports Intel vendor specific commands. The vendor specific commands are based on the NC-SI — OEM Command. These commands are listed in the following sub-sections and are used to enable the BMC to control the 82599 specific features:

- Rx filters:
  - — Packet addition decision filters 0x0…0x4
  - — Packet reduction decision filters 0x5…0x7
  - — MNG2HOST register (controls the forwarding of manageability packets to the host)
  - — Flex 128 filters 0x0…0x3
  - — Flex TCP/UDP port filters 0..0xA
  - — IPv4/IPv6 filters
  - — Ether type filters
- Get System Ethernet MAC Address — This command enables the BMC to retrieve the system Ethernet MAC address used by the NC. This Ethernet MAC address can be used for shared Ethernet 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 MC to reset the network adapter.
- Checksum Offloading — Offloads IP/UDP/TCP checksum checking from the MC.
- LinkSec logic programming

These commands are designed to be compliant with their corresponding SMBus commands (if existing). All of the commands are based on a single DMTF defined NC-SI command, known as OEM Command described in Section 10.5.1.1.

### 10.5.1.1 OEM Command (0x50)

The OEM command can be used by the MC to request the sideband interface to provide vendor-specific information. The Vendor Enterprise Number (VEN) is the unique MIB/ SNMP private enterprise number assigned by IANA per organization. Vendors are free to define their own internal data structures in the vendor data fields.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20.. | Intel Command Number | Optional Data | | |

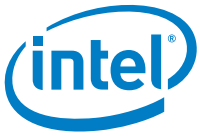
### 10.5.1.2 OEM Response (0xD0)

The sideband interface must return an Unknown Command Type reason code for any un-recognized enterprise number using the following frame format. If the command is valid, the response, if any, is allowed to be vendor-specific. It is recommended to use the 0x8000 range for vendor-specific code.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..27 | Intel Command Number | Optional Return Data | | |

**Table 10-3 OEM Specific Command Response and Reason Codes**

| Response Code | | Reason Code | |
|---|---|---|---|
| **Value** | **Description** | **Value** | **Description** |
| 0x1 | Command Failed | 0x5081 | Invalid Intel Command Number |
| | | 0x5082 | Invalid Intel Command Parameter Number |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |



### 10.5.1.3 Intel Commands

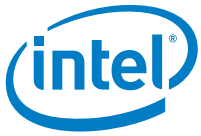
Table 10-4 lists the Intel commands and their associated Intel Command Number values. For detailed description of the commands and their parameters refer to the following sections.

**Table 10-4 Intel Command Summary**

| Intel Command | Parameter | Command Name |
|---|---|---|
| 0x00 | 0x00 | Set IP Filters Control |
| 0x01 | 0x00 | Get IP Filters Control |
| 0x02 | 0x0A | Set Manageability to Host |
| | 0x10 | Set Flexible 128 Filter 0 Mask and Length |
| | 0x11 | Set Flexible 128 Filter 0 Data |
| | 0x20 | Set Flexible 128 Filter 1 Mask and Length |
| | 0x21 | Set Flexible 128 Filter 1 Data |
| | 0x30 | Set Flexible 128 Filter 2 Mask and Length |
| | 0x31 | Set Flexible 128 Filter 2 Data |
| | 0x40 | Set Flexible 128 Filter 3 Mask and Length |
| | 0x41 | Set Flexible 128 Filter 3 Data |
| | 0x61 | Set Packet Addition Filters |
| | 0x63 | Set Flex TCP/UDP Port Filters |
| | 0x64 | Set Flex IPv4 Address Filters |
| | 0x65 | Set Flex IPv6 Address Filters |
| | 0x67 | Set EtherType Filter |
| | 0x68 | Set Packet Addition Extended Decision Filter |

## Table 10-4 Intel Command Summary (Continued)

| Intel Command | Parameter | Command Name |
|---|---|---|
| 0x3 | 0x0A | Get Manageability to Host |
| | 0x10 | Get Flexible 128 Filter 0 Mask and Length |
| | 0x11 | Get Flexible 128 Filter 0 Data |
| | 0x20 | Get Flexible 128 Filter 1 Mask and Length |
| | 0x21 | Get Flexible 128 Filter 1 Data |
| | 0x30 | Get Flexible 128 Filter 2 Mask and Length |
| | 0x31 | Get Flexible 128 Filter 2 Data |
| | 0x40 | Get Flexible 128 Filter 3 Mask and Length |
| | 0x41 | Get Flexible 128 Filter 3 Data |
| | 0x61 | Get Packet Addition Filters |
| | 0x63 | Get Flex TCP/UDP Port Filters |
| | 0x64 | Get Flex IPv4 Address Filters |
| | 0x65 | Get Flex IPv6 Address Filters |
| | 0x67 | Get EtherType Filter |
| | 0x68 | Get Packet Addition Extended Decision Filter |
| 0x04 | 0x00 | Set Unicast Packet Reduction |
| | 0x01 | Set Multicast Packet Reduction |
| | 0x02 | Set Broadcast Packet Reduction |
| | 0x10 | Set Unicast Extended Packet Reduction |
| | 0x11 | Set Multicast Extended Packet Reduction |
| | 0x12 | Set Broadcast Extended Packet Reduction |
| 0x05 | 0x00 | Get Unicast Packet Reduction |
| | 0x01 | Get Multicast Packet Reduction |
| | 0x02 | Get Broadcast Packet Reduction |
| | 0x10 | Get Unicast Extended Packet Reduction |
| | 0x11 | Get Multicast Extended Packet Reduction |
| | 0x12 | Get Broadcast Extended Packet Reduction |



## Table 10-4 Intel Command Summary (Continued)

| Intel Command | Parameter | Command Name |
|---|---|---|
| 0x06 | N/A | Get System Ethernet MAC Address |
| 0x20 | N/A | Set Intel Management Control |
| 0x21 | N/A | Get Intel Management Control |
| 0x22 | N/A | Perform TCO Reset |
| 0x23 | N/A | Enable IP/UDP/TCP Checksum Offloading |
| 0x24 | N/A | Disable IP/UDP/TCP Checksum Offloading |
| 0x30 | 0x10 | Transfer LinkSec Ownership to BMC |
| | 0x11 | Transfer LinkSec Ownership to Host |
| | 0x12 | Initialize LinkSec Rx |
| | 0x13 | Initialize LinkSec Tx |
| | 0x14 | Set LinkSec Rx Key |
| | 0x15 | Set LinkSec Tx Key |
| | 0x16 | Enable Network Tx Encryption |
| | 0x17 | Disable Network Tx Encryption |
| | 0x18 | Enable Network Rx Decryption |
| | 0x19 | Disable Network Rx Decryption |
| 0x31 | 0x01 | Get LinkSec Rx Parameters |
| | 0x02 | Get LinkSec Tx Parameters |

# 10.5.1.4 Set Intel Filters Control Command (Intel Command 0x00)

## 10.5.1.4.1 Set Intel Filters Control – IP Filters Control Command (Intel Command 0x00)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..23 | 0x00 | 0x00 | IP Filters control (3-2) | |
| 24..27 | IP Filters Control (1-0) | | | |

While IP Filters Control has the following format:

**Table 10-5 IP Filter Formats**

| Bit # | Name | Description | Default |
|---|---|---|---|
| 0 | IPv4/IPv6 Mode | IPv6 (0b): There are 0 IPv4 filters and 4 IPv6 filters<br>IPv4 (1b): There are 4 IPv4 filters and 3 IPv6 filters<br>See Section 8.2.3.25.2 or Section 10.3.4 for details. | 1b |
| 1..15 | Reserved | | |
| 16 | IPv4 Filter 0 Valid | Indicates if the IPv4 address configured in IPv4 address 0 is valid.<br>*Note:* The network controller must automatically set this bit to 1b if the Set Intel Filter – IPv4 Filter Command is used for filter 0. | 0b |
| 17 | IPv4 Filter 1 Valid | Indicates if the IPv4 address configured in IPv4 address 1 is valid.<br>*Note:* The network controller must automatically set this bit to 1b if the Set Intel Filter – IPv4 Filter Command is used for filter 1. | 0b |
| 18 | IPv4 Filter 2 Valid | Indicates if the IPv4 address configured in IPv4 address 2 is valid.<br>*Note:* The network controller must automatically set this bit to 1b if the Set Intel Filter – IPv4 Filter Command is used for filter 2. | 0b |
| 19 | IPv4 Filter 3 Valid | Indicates if the IPv4 address configured in IPv4 address 3 is valid.<br>*Note:* The network controller must automatically set this bit to 1b if the Set Intel Filter – IPv4 Filter Command is used for filter 3. | 0b |
| 20..23 | Reserved | | |
| 24 | IPv6 Filter 0 Valid | Indicates if the IPv6 address configured in IPv6 address 0 is valid.<br>*Note:* The network controller must automatically set this bit to 1b if the Set Intel Filter – IPv6 Filter Command is used for filter 0. | 0b |
| 25 | IPv6 Filter 1 Valid | Indicates if the IPv6 address configured in IPv6 address 1 is valid.<br>*Note:* The network controller must automatically set this bit to 1b if the Set Intel Filter – IPv6 Filter Command is used for filter 1. | 0b |

**Table 10-5 IP Filter Formats (Continued)**

| Bit # | Name | Description | Default |
|---|---|---|---|
| 26 | IPv6 Filter 2 Valid | Indicates if the IPv6 address configured in IPv6 address 2 is valid.<br>*Note:* The network controller must automatically set this bit to 1b if the Set Intel Filter – IPv6 Filter Command is used for filter 2. | 0b |
| 27 | IPv6 Filter 3 Valid | Indicates if the IPv6 address configured in IPv6 address 3 is valid.<br>*Note:* The network controller must automatically set this bit to 1b if the Set Intel Filter – IPv6 Filter Command is used for filter 3. | 0b |
| 28..31 | Reserved | Reserved | |

#### 10.5.1.4.2 Set Intel Filters Control – IP Filters Control Response (Intel Command 0x00, Filter Control Index 0x00)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..27 | 0x00 | 0x00 | | |

### 10.5.1.5 Get Intel Filters Control Command (Intel Command 0x01)

#### 10.5.1.5.1 Get Intel Filters Control – IP Filters Control Command (Intel Command 0x01, Filter Control Index 0x00)

This command controls different aspects of the Intel filters.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..21 | 0x01 | 0x00 | | |

#### 10.5.1.5.2 Get Intel Filters Control – IP Filters Control Response (Intel Command 0x01, Filter Control Index 0x00)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..27 | 0x01 | 0x00 | IP Filters Control (3-2) | |
| 28..29 | IP Filters Control (1-0) | | | |

IP Filter Control: See Table 10-5.

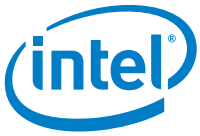
### 10.5.1.6 Set Intel Filters Formats

#### 10.5.1.6.1 Set Intel Filters Command (Intel Command 0x02)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..21 | 0x02 | Filter Parameter | Filters Data (optional) | |

#### 10.5.1.6.2 Set Intel Filters Response (Intel Command 0x02)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24.. | 0x02 | Filter Parameter | Return Data (Optional) | |



#### 10.5.1.6.3 Set Intel Filters – Manageability to Host Command (Intel Command 0x02, Filter Parameter 0x0A)

This command sets the Mng2Host register. The Mng2Host register controls whether pass-through packets destined to the BMC are also forwarded to the host operating system.

The Mng2Host register has the following structure:

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..23 | 0x02 | 0x0A | Manageability to Host (3-2) | |
| 24..25 | Manageability to Host (1-0) | | | |

**Table 10-6 Manageability to Host Field**

| Bits | Name | Description | Default |
|---|---|---|---|
| 0 | Decision Filter 0 | Determines if packets that have passed Decision Filter 0 are also forwarded to the host operating system. | 0b |
| 1 | Decision Filter 1 | Determines if packets that have passed Decision Filter 1 are also forwarded to the host operating system. | 0b |
| 2 | Decision Filter 2 | Determines if packets that have passed Decision Filter 2 are also forwarded to the host operating system. | 0b |
| 3 | Decision Filter 3 | Determines if packets that have passed Decision Filter 3 are also forwarded to the host operating system. | 0b |
| 4 | Decision Filter 4 | Determines if packets that have passed Decision Filter 4 are also forwarded to the host operating system. | 0b |
| 5 | Unicast & Mixed | Determines if unicast and mixed packets are also forwarded to the host operating system. | 0b |
| 6 | Global Multicast | Determines if global multicast packets are also forwarded to the host operating system. | 1b |
| 7 | Broadcast | Determines if broadcast packets are also forwarded to the host operating system. | 1b |
| 31:8 | Reserved | Reserved. | N/A |

#### 10.5.1.6.4 Set Intel Filters – Manageability to Host Response (Intel Command 0x02, Filter Parameter 0x0A)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24.. | 0x02 | 0x0A | | |

#### 10.5.1.6.5 Set Intel Filters – Flex Filter 0/1/2/3 Enable Mask and Length Command (Intel Command 0x02, Filter Parameter 0x10/0x20/0x30/0x40)

The following command sets the Intel flex filters mask and length. Use filter parameters 0x10/0x20/0x30/0x40 for flexible filters 0/1/2/3 accordingly. See Section 10.3.4.1 for details of the programming.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..23 | 0x02 | 0x10/<br>0x20/<br>0x30/<br>0x40 | Mask Byte 1 | Mask Byte 2 |
| 24..27 | .. | .. | .. | .. |
| 28...31 | .. | .. | .. | .. |
| 32...35 | .. | .. | .. | .. |
| 35...37 | .. | Mask Byte 16 | Reserved | Reserved |
| 38 | Flexible Filter Length (8-128 bytes) | | | |

### 10.5.1.6.6 Set Intel Filters – Flex Filter 0/1/2/3 Data Command (Intel Command 0x02, Filter Parameter 0x11/0x21/ 0x31/0x41)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x02 | 0x10/<br>0x20/<br>0x30/<br>0x40 | | |

### 10.5.1.6.7 Set Intel Filters – Flex Filter 0/1/2/3 Data Command (Intel Command 0x02, Filter Parameter 0x11/0x21/ 0x31/0x41)

The following command sets the Intel flex filters data. Use filter parameters 0x11/0x21/ 0x31/0x41 for flexible filters 0/1/2/3 accordingly.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20.. | 0x02 | 0x11/<br>0x21/<br>0x31/<br>0x41 | Filter Data Group | Filter Data 1 |
| | .. | Filter Data N | | |

The filter data group parameter defines which bytes of the flex filter are set by this command:

**Table 10-7 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 |

### 10.5.1.6.8 Set Intel Filters – Flex Filter 0/1/2/3 Data Response (Intel Command 0x02, Filter Parameter 0x11/0x21/ 0x31/0x41)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x02 | 0x11/<br>0x21/<br>0x31/<br>0x41 | | |

**Note:** If filter data length is larger than specified in Table 10-7 an out of range reason code is returned.

### 10.5.1.6.9 Set Intel Filters – Packet Addition Decision Filter Command (Intel Command 0x02, Filter Parameter 0x61)

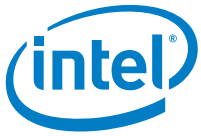
| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..23 | 0x02 | 0x61 | Filter index | Decision Filter (MSB) |
| 24..26 | ..... | | Decision Filter (LSB) | |

Filter index range: 0x0..0x4

If the filter index is bigger than four, a command failed response code is returned with no reason.

**Table 10-8 Filter Values**

| Bit # | Name | Description |
|---|---|---|
| 0 | Unicast (AND) | If set, packets must match a unicast filter. |
| 1 | Broadcast (AND) | If set, packets must match the broadcast filter. |
| 2 | VLAN (AND) | If set, packets must match a VLAN filter. |
| 3 | IP Address (AND) | If set, packets must match an IP filter. |



## Table 10-8 Filter Values (Continued)

| Bit # | Name | Description |
|---|---|---|
| 4 | Unicast (OR) | If set, packets must match a unicast filter or a different OR filter. |
| 5 | Broadcast | If set, packets must match the broadcast filter or a different OR filter. |
| 6 | Multicast (AND) | If set, packets must match the multicast filter. |
| 7 | ARP Request (OR) | If set, packets must match the ARP request filter or a different OR filter. |
| 8 | ARP Response (OR) | If set, packets can pass if match the ARP response filter. |
| 9 | Neighbor Discovery (OR) | If set, packets can pass if match the neighbor discovery filter. |
| 10 | Port 0x298 (OR) | If set, packets can pass if match a fixed TCP/UDP port 0x298 filter. |
| 11 | Port 0x26F (OR) | If set, packets can pass if match a fixed TCP/UDP port 0x26F filter. |
| 12 | Flex port 0 (OR) | If set, packets can pass if match the TCP/UDP port filter 0. |
| 13 | Flex port 1 (OR) | If set, packets can pass if match the TCP/UDP port filter 1. |
| 14 | Flex port 2 (OR) | If set, packets can pass if match the TCP/UDP port filter 2. |
| 15 | Flex port 3 (OR) | If set, packets can pass if match the TCP/UDP port filter 3. |
| 16 | Flex port 4 (OR) | If set, packets can pass if match the TCP/UDP port filter 4. |
| 17 | Flex port 5 (OR) | If set, packets can pass if match the TCP/UDP port filter 5. |
| 18 | Flex port 6 (OR) | If set, packets can pass if match the TCP/UDP port filter 6. |
| 19 | Flex port 7 (OR) | If set, packets can pass if match the TCP/UDP port filter 7. |
| 20 | Flex port 8 (OR) | If set, packets can pass if match the TCP/UDP port filter 8. |
| 21 | Flex port 9 (OR) | If set, packets can pass if match the TCP/UDP port filter 9. |
| 22 | Flex port 10 (OR) | If set, packets can pass if match the TCP/UDP port filter 10. |
| 23 | DHCPv6 (OR) | If set, packets can pass if match the DHCPv6 port (0x0223). |
| 24 | DHCP Client (OR) | If set, packets can pass if match the DHCP server port (0x0043). |
| 25 | DHCP Server (OR) | If set, packets can pass if match the DHCP client port (0x0044). |
| 26 | NetBIOS Name Service (OR) | If set, packets can pass if match the NetBIOS name service port (0x0089). |
| 27 | NetBIOS Datagram Service (OR) | If set, packets can pass if match the NetBIOS datagram service port (0x008A). |
| 28 | Flex TCO 0 (OR) | If set, packets can pass if match the flex 128 TCO filter 0. |
| 29 | Flex TCO 1 (OR) | If set, packets can pass if match the flex 128 TCO filter 1. |

**Table 10-8 Filter Values (Continued)**

| Bit # | Name | Description |
|---|---|---|
| 30 | Flex TCO 2 (OR) | If set, packets can pass if match the flex 128 TCO filter 2. |
| 31 | Flex TCO 3 (OR) | If set, packets can pass if match the flex 128 TCO filter 3. |

The filtering is divided into 2 decisions:

Bits 0,1,2,3,6 works in an AND manner.As a result, they all must be true in order for a packet to pass (if any were set).

Bits 5,7-31 operate in an OR manner. Thus, at least one of them must be true for a packet to pass (if any were set).

See Section 10.3.5 for description of the decision filters.

**Note:** These filter settings operate according to the VLAN mode, as configured according to the DMTF NC-SI specification. After disabling packet reduction filters, the BMC must re-set the VLAN mode using the Set VLAN command.

### 10.5.1.6.10 Set Intel Filters – Packet Addition Decision Filter Response (Intel Command 0x02, Filter Parameter 0x61)

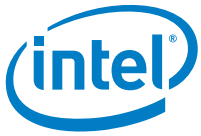
| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x02 | 0x61 | | |

### 10.5.1.6.11 Set Intel Filters – Flex TCP/UDP Port Filter Command (Intel Command 0x02, Filter Parameter 0x63)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..23 | 0x02 | 0x63 | Port filter index | TCP/UDP Port MSB |
| 24 | TCP/UDP Port LSB | | | |

Filter index range: 0x0..0xA.

If the filter index is bigger than 10, a command failed response code is returned with no reason.



#### 10.5.1.6.12 Set Intel Filters – Flex TCP/UDP Port Filter Response (Intel Command 0x02, Filter Parameter 0x63)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x02 | 0x63 | | |

#### 10.5.1.6.13 Set Intel Filters – IPv4 Filter Command (Intel Command 0x02, Filter Parameter 0x64)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..23 | 0x02 | 0x64 | IP filter index | IPv4 Address (MSB) |
| 24..26 | ... | | IPv4 Address (LSB) | |

**Note:** The filters index range can vary according to the IPv4/IPv6 mode setting in the Filters Control command

IPv4 Mode: Filter index range: 0x0..0x3.

IPv6 Mode: This command should not be used in IPv6 mode.

#### 10.5.1.6.14 Set Intel Filters – IPv4 Filter Response (Intel Command 0x02, Filter Parameter 0x64)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x02 | 0x64 | | |

If the IP filter index is bigger than three, a command failed response code is returned with no reason.

### 10.5.1.6.15 Set Intel Filters – IPv6 Filter Command (Intel Command 0x02, Filter Parameter 0x65)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..23 | 0x02 | 0x65 | IP filter index | IPv6 Address (MSB, byte 15) |
| 24..27 | .. | .. | .. | .. |
| 28..31 | .. | .. | .. | .. |
| 32..35 | .. | .. | .. | .. |
| 36..38 | .. | .. | IPv6 Address (LSB, byte 0) | |

**Note:** The filters index range can vary according to the IPv4/IPv6 mode setting in the Filters Control command

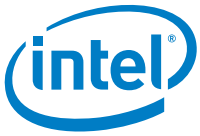
IPv4 Mode: Filter index range: 0x1..0x3.

IPv6 Mode: Filter index range: 0x0..0x3.

### 10.5.1.6.16 Set Intel Filters – IPv6 Filter Response (Intel Command 0x02, Filter Parameter 0x65)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x02 | 0x65 | | |

If the IP filter index is bigger than three, a command failed response code is returned with no reason.



### 10.5.1.6.17 Set Intel Filters – EtherType Filter Command (Intel Command 0x02, Filter Parameter 0x67)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..23 | 0x02 | 0x67 | EtherType Filter Index | EtherType Filter MSB |
| 24..27 | .. | .. | EtherType Filter LSB | |

Where the EtherType filter has the format as described in Section 8.2.3.25.3.

**Table 10-9 Ethertype Usage**

| Filter # | Usage | Note |
|---|---|---|
| 0-1 | Reserved | Not available for generic use. |
| 2 | User defined | |
| 3 | User defined | |

### 10.5.1.6.18 Set Intel Filters - EtherType Filter Response (Intel Command 0x02, Filter Parameter 0x67)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x02 | 0x67 | | |

If the Ethertype filter index is different than two or three, a command failed response code is returned with no reason.

### 10.5.1.6.19 Set Intel Filters – Packet Addition Extended Decision Filter Command (Intel Command 0x02, Filter Parameter 0x68)

DecisionFilter0 Bits 5,7-31 and DecisionFilter1 bits 8..10 work in an OR manner. Thus, at least one of them must be true for a packet to pass (if any were set).

See Figure 10-3 for description of the decision filters structure.

**Note:** The command must overwrite any previously stored value.

Previous Set Intel Filters – Packet Addition Decision Filter command (0x61) should be kept and supported. For legacy reasons, if previous Decision Filter command is called, it should set the Decision Filter 0 as provided and set the extended Decision Filter to 0x0.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..23 | 0x02 | 0x68 | Extended Decision filter Index | Extended Decision filter 1 MSB |
| 24..27 | .. | .. | Extended Decision filter 1 LSB | Extended Decision filter 0 MSB |
| 28..30 | .. | .. | Extended Decision filter 0 LSB | |

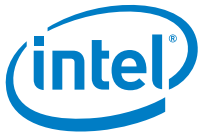
Extended decision filter index range: 0..4.

Filter 0: See Table 10-8.

Filter 1: See the following table:

**Table 10-10 Extended Filter 1 Values**

| Bit # | Name | Description |
|---|---|---|
| 0 | Ethertype 0x88F8 | AND filter |
| 1 | Ethertype 0x8808 | AND filter |
| 3:2 | Ethertype 2 -3 | AND filters |
| 7:4 | Reserved | Reserved |
| 8 | Ethertype 0x88F8 | OR filter |
| 9 | Ethertype 0x8808 | OR filter |
| 11:10 | Ethertype 2 -3 | OR filters |
| 31:12 | Reserved | Reserved |



#### 10.5.1.6.20 Set Intel Filters – Packet Addition Extended Decision Filter Response (Intel Command 0x02, Filter Parameter 0x68)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x02 | 0x68 | | |

If the extended decision filter index is bigger than five, a command failed response code is returned with no reason.

## 10.5.1.7 Get Intel Filters Formats

### 10.5.1.7.1 Get Intel Filters Command (Intel Command 0x03)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..21 | 0x03 | Filter Parameter | | |

### 10.5.1.7.2 Get Intel Filters Response (Intel Command 0x03)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x03 | Filter Parameter | Optional Return Data | |

### 10.5.1.7.3 Get Intel Filters – Manageability to Host Command (Intel Command 0x03, Filter Parameter 0x0A)

This command retrieves the Mng2Host register. The Mng2Host register controls whether pass-through packets destined to the BMC are also 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 | 0x0A | | |

### 10.5.1.7.4 Get Intel Filters – Manageability to Host Response (Intel Command 0x03, Filter Parameter 0x0A)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..27 | 0x03 | 0x0A | Manageability to Host (MSB) | |
| 28..29 | Manageability to Host (LSB) | | | |

The Mng2Host register has the following structure:

**Table 10-11 Mng2Host Structure**

| Bits | Description | Default |
|---|---|---|
| 0 | Decision Filter 0 | Determines if packets that have passed Decision Filter 0 are also forwarded to the host operating system. |
| 1 | Decision Filter 1 | Determines if packets that have passed Decision Filter 1 are also forwarded to the host operating system. |
| 2 | Decision Filter 2 | Determines if packets that have passed Decision Filter 2 are also forwarded to the host operating system. |
| 3 | Decision Filter 3 | Determines if packets that have passed Decision Filter 3 are also forwarded to the host operating system. |
| 4 | Decision Filter 4 | Determines if packets that have passed Decision Filter 4 are also forwarded to the host operating system. |
| 5 | Unicast & Mixed | Determines if unicast and mixed packets are also forwarded to the host operating system. |

**Table 10-11 Mng2Host Structure (Continued)**

| Bits | Description | Default |
|---|---|---|
| 6 | Global Multicast | Determines if global multicast packets are also forwarded to the host operating system. |
| 7 | Broadcast | Determines if broadcast packets are also forwarded to the host operating system. |
| 31:8 | Reserved | Reserved |

### 10.5.1.7.5 Get Intel Filters – Flex Filter 0/1/2/3 Enable Mask and Length Command (Intel Command 0x03, Filter Parameter 0x10/0x20/0x30/0x40)

The following command retrieves the Intel flex filters mask and length. Use filter parameters 0x10/0x20/0x30/0x40 for flexible filters 0/1/2/3 accordingly. See Section 10.3.4.1 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/<br>0x20/<br>0x30/<br>0x40 | | |

### 10.5.1.7.6 Get Intel Filters – Flex Filter 0/1/2/3 Enable Mask and Length Response (Intel Command 0x03, Filter Parameter 0x10/0x20/0x30/0x40)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..27 | 0x03 | 0x10/<br>0x20/<br>0x30/<br>0x40 | Mask Byte 1 | Mask Byte 2 |
| 28..31 | .. | .. | .. | .. |
| 32..35 | .. | .. | .. | .. |
| 36..39 | .. | .. | .. | .. |
| 40..43 | .. | Mask Byte 16 | Reserved | Reserved |
| 44 | Flexible Filter Length | | | |

### 10.5.1.7.7 Get Intel Filters – Flex Filter 0/1/2/3/4 Data Command (Intel Command 0x03, Filter Parameter 0x11/0x21/0x31/0x41)

The following command retrieves the Intel flex filters data. Use filter parameters 0x11/0x21/0x31/0x41 for flexible filters 0/1/2/3 accordingly.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..22 | 0x03 | 0x11/<br>0x21/<br>0x31/<br>0x41 | Filter Data Group<br>0..4 | |

The filter data group parameter defines which bytes of the flex filter are returned by this command:

**Table 10-12 Filter Data Group**

| Code | Bytes Returned |
|---|---|
| 0x0 | Bytes 0-29 |
| 0x1 | Bytes 30-59 |
| 0x2 | Bytes 60-89 |
| 0x3 | Bytes 90-119 |
| 0x4 | Bytes 120-127 |

### 10.5.1.7.8 Get Intel Filters – Flex Filter Data Response (Intel Command 0x03, Filter Parameter 0x11)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24.. | 0x03 | 0x11/<br>0x21/<br>0x31/<br>0x41 | Filter Data Group | Filter Data 1 |
| | .. | Filter Data N | | |

If the filter group number is bigger than four, a command failed response code is returned with no reason.

### 10.5.1.7.9 Get Intel Filters – Packet Addition Decision Filter Command (Intel Command 0x03, Filter Parameter 0x61)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..22 | 0x03 | 0x61 | Decision filter index | |

Filter index range: 0x0..0x4.

### 10.5.1.7.10 Get Intel Filters – Packet Addition Decision Filter Response (Intel Command 0x03, Filter Parameter 0x0A)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..27 | 0x03 | 0x61 | Decision Filter (MSB) | |
| 28..29 | Decision Filter (LSB) | | | |

The decision filter structure returned is listed in Table 10-8.

If the decision filter index is bigger than four, a command failed response code is returned with no reason.

### 10.5.1.7.11 Get Intel Filters – Flex TCP/UDP Port Filter Command (Intel Command 0x03, Filter Parameter 0x63)

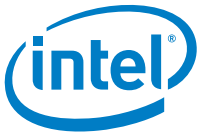
| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..22 | 0x03 | 0x63 | TCP/UDP Filter Index | |

Filter index range: 0x0..0xA.

### 10.5.1.7.12 Get Intel Filters – Flex TCP/UDP Port Filter Response (Intel Command 0x03, Filter Parameter 0x63)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..27 | 0x03 | 0x63 | TCP/UDP Filter Index | TCP/UDP Port (1) |
| 28 | TCP/UDP Port (0) | | | |

Filter index range: 0x0..0xA.



If the TCP/UDP filter index is bigger than 10, a command failed response code is returned with no reason.

### 10.5.1.7.13 Get Intel Filters – IPv4 Filter Command (Intel Command 0x03, Filter Parameter 0x64)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..22 | 0x03 | 0x64 | IPv4 Filter Index | |

**Note:** The filters index range can vary according to the IPv4/IPv6 mode setting in the Filters Control command

IPv4 Mode: Filter index range: 0x0..0x3.

IPv6 Mode: This command should not be used in IPv6 mode.

### 10.5.1.7.14 Get Intel Filters – IPv4 Filter Response (Intel Command 0x03, Filter Parameter 0x64)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..27 | 0x03 | 0x64 | IPv4 Filter Index | IPv4 Address (3) |
| 28..29 | IPv4 Address (2-0) | | | |

If the IPv4 filter index is bigger than three, a command failed response code is returned with no reason.

### 10.5.1.7.15 Get Intel Filters – IPv6 Filter Command (Intel Command 0x03, Filter Parameter 0x65)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..22 | 0x03 | 0x65 | IPv6 Filter Index | |

**Note:** The filters index range can vary according to the IPv4/IPv6 mode setting in the Filters Control command.

IPv4 Mode: Filter index range: 0x0..0x2.

IPv6 Mode: Filter index range: 0x0..0x3.

### 10.5.1.7.16 Get Intel Filters – IPv6 Filter Response (Intel Command 0x03, Filter Parameter 0x65)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..27 | 0x03 | 0x65 | IPv6 Filter Index | IPv6 Address (MSB, Byte 16) |
| 28..31 | .. | .. | .. | .. |
| 32..35 | .. | .. | .. | .. |
| 36..39 | .. | .. | .. | .. |
| 40..42 | .. | .. | IPv6 Address (LSB, Byte 0) | |

If the IPv6 filter index is bigger than three, a command failed response code is returned with no reason.

### 10.5.1.7.17 Get Intel Filters – EtherType Filter Command (Intel Command 0x03, Filter Parameter 0x67)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..22 | 0x03 | 0x67 | EtherType Filter Index | |

Valid indices: 2..3.

See Table 10-9 for a list of the various Ethertype filters usage.

### 10.5.1.7.18 Get Intel Filters - EtherType Filter Response (Intel Command 0x03, Filter Parameter 0x67)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..27 | 0x03 | 0x67 | EtherType Filter Index | EtherType Filter MSB |
| 28..30 | .. | .. | EtherType Filter LSB | |

If the Ethertype filter index is different than two or three, a command failed response code is returned with no reason.

### 10.5.1.7.19 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 | |

### 10.5.1.7.20 Get Intel Filters – Packet Addition Extended Decision Filter Response (Intel Command 0x03, Filter Parameter 0x68)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..27 | 0x03 | 0x68 | Decision Filter Index | Decision Filter 1 MSB |
| 28..31 | .. | .. | Decision Filter 1 LSB | Decision Filter 0 MSB |
| 32..34 | .. | .. | Decision Filter 0 LSB | |

Where Decision Filter 0 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 no reason.

## 10.5.1.8 Set Intel Packet Reduction Filters Formats

### 10.5.1.8.1 Set Intel Packet Reduction Filters Command (Intel Command 0x04)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..23 | 0x04 | Filter Parameter | Optional Data | |

**Note:** It is recommended that the BMC only uses the Extended Packet Reduction commands.

The *Packet Reduction Data* field has the following structure:

**Table 10-13 Packet Reduction Data**

| Bit # | Name | Description |
|---|---|---|
| 2:0 | Reserved | |
| 3 | IP Address | If set, all packets must also match an IP filter. |
| 9:4 | Reserved | |
| 10 | Port 0x298 | If set, all packets can pass if match a fixed TCP/UDP port 0x298 filter. |
| 11 | Port 0x26F | If set, all packets can pass if match a fixed TCP/UDP port 0x26F filter. |
| 12 | Flex port 0 | If set, all packets can pass if match the TCP/UDP port filter 0. |
| 13 | Flex port 1 | If set, all packets can pass if match the TCP/UDP port filter 1. |
| 14 | Flex port 2 | If set, all packets can pass if match the TCP/UDP port filter 2. |
| 15 | Flex port 3 | If set, all packets can pass if match the TCP/UDP port filter 3. |
| 16 | Flex port 4 | If set, all packets can pass if match the TCP/UDP port filter 4. |
| 17 | Flex port 5 | If set, all packets can pass if match the TCP/UDP port filter 5. |
| 18 | Flex port 6 | If set, all packets can pass if match the TCP/UDP port filter 6. |

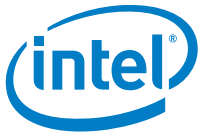


**Table 10-13 Packet Reduction Data (Continued)**

| Bit # | Name | Description |
|---|---|---|
| 19 | Flex port 7 | If set, all packets can pass if match the TCP/UDP port filter 7. |
| 20 | Flex port 8 | If set, all packets can pass if match the TCP/UDP port filter 8. |
| 21 | Flex port 9 | If set, all packets can pass if match the TCP/UDP port filter 9. |
| 22 | Flex port 10 | If set, all packets can pass if match the TCP/UDP port filter 10. |
| 27:23 | Reserved | |
| 28 | Flex TCO 0 | If set, all packets can pass if match the Flex 128 TCO filter 0. |
| 29 | Flex TCO 1 | If set, all packets can pass if match the Flex 128 TCO filter 1. |
| 30 | Flex TCO 2 | If set, all packets can pass if match the Flex 128 TCO filter 2. |
| 31 | Flex TCO 3 | If set, all packets can pass if match the Flex 128 TCO filter 3. |

For the Extended Packet Reduction command, the following fields should also be programmed.

**Table 10-14 Extended Packet Reduction Format**

| Bit # | Name | Description |
|---|---|---|
| 0..1 | Reserved | Used by the regular NC-SI commands. |
| 2 | EtherType2 (AND) | If set, packets must also match the EtherType filter 2. |
| 3 | EtherType3 (AND) | If set, packets must also match the EtherType filter 3. |
| 4..7 | Reserved | |
| 8..9 | Reserved | Used by the regular NC-SI commands |
| 10 | EtherType2 (OR) | If set, packets can pass if it match the EtherType filter 2. |
| 11 | EtherType3 (OR) | If set, packets can pass if it match the EtherType filter 2. |
| 12..31 | Reserved | |

The filtering is divided into two decisions:

Unicast Reduction Filter — Bit 3 and *Extended Unicast Reduction Filter* bits 0..2 work in an AND manner. Thus, they all must be true in order for a packet to pass (if any were set).

*Unicast Reduction Filter* bits 5, 7-31 and *Extended Unicast Reduction Filter* bits 8..10 work in an OR manner, Thus, at least one of them must be true for a packet to pass (if any were set).

See Section 10.3.5 for description of the decision filters.

#### 10.5.1.8.2 Set Intel Packet Reduction Filters Response (Intel Command 0x04)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24.. | 0x04 | Filter Parameter | Optional Return Data | |

#### 10.5.1.8.3 Set Unicast/Multicast/Broadcast packet Reduction Command (Intel Command 0x04, Filter Parameter 0x00/0x01/0x02)

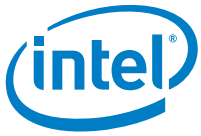
| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..23 | 0x04 | 0x00 / 0x01 / 0x02 | Packet Reduction Table (MSB) | |
| 24..25 | Packet Reduction Table (LSB) | | | |

This command must cause the network controller to filter packets that have passed due to the unicast/multicast/broadcast filter. Note that unicast filtering might be affected by other filters, as specified in the DMTF NC-SI.

The filtering of these packets are done such that the BMC might add a logical condition that a packet must match, or it must be discarded.

**Note:** Packets that might have been blocked can still pass due to other decision filters.

In order to disable unicast/multicast/broadcast packet reduction, the BMC should set all reductions filters to 0b. Following such a setting, the network controller forwards to the BMC all packets that have passed the unicast Ethernet MAC address/global multicast/broadcast filters as specified in the DMTF NC-SI.



#### 10.5.1.8.4 Set Unicast/Multicast/Broadcast Packet Reduction Response (Intel Command 0x04, Reduction Filter Parameter 0x00/0x01/0x02)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x04 | 0x00 / 0x01 / 0x02 | | |

#### 10.5.1.8.5 Set Unicast/Multicast/Broadcast Extended Packet Reduction Command (Intel Command 0x04, Filter Parameter 0x10/0x11/0x12)

In Set Intel Reduction Filters, add another parameter Unicast Extended Packet Reduction (Intel Command 0x04, Filter parameter 0x10) such that the byte count is 0xE. The command 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 /0x11 / 0x12 | Extended Reduction Filter MSB | .. |
| 24..27 | .. | Extended Reduction Filter LSB | Reduction Filter Table (MSB) | .. |
| 28..29 | .. | Reduction Filter Table (LSB) | | |

The command must overwrite any previously stored value.

**Note:** See Table 10-13 and Table 10-14 for a list of the unicast extended packet reduction format.

#### 10.5.1.8.6 Set Unicast/Multicast/Broadcast Extended Packet Reduction Response (Intel Command 0x04, Reduction Filter Index 0x10 / 0x11 / 0x12)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x04 | 0x10 / 0x11 / 0x12 | | |

### 10.5.1.9 Get Intel Packet Reduction Filters Formats

#### 10.5.1.9.1 Get Intel Packet Reduction Filters Command (Intel Command 0x05)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..21 | 0x05 | Filter Parameter | | |

#### 10.5.1.9.2 Get Intel Packet Reduction Filters Response (Intel Command 0x05)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24.. | 0x05 | Filter Parameter | Optional Return Data | |

#### 10.5.1.9.3 Get Unicast/Multicast/Broadcast Packet Reduction Command & Response (Intel Command 0x05, Filter Parameter 0x00/0x01/0x02)

This command retrieves the requested packet reduction filter. The format of the optional return data follows the structure of the Unicast Packet Reduction command described in Section 10.5.1.8.3.

#### 10.5.1.9.4 Get Unicast/Multicast/Broadcast Extended Packet Reduction Command & Response (Intel Command 0x05, Filter Parameter 0x00/0x01/0x02)

This command retrieves the requested extended packet reduction filter. The format of the optional return data follows the structure of the Unicast Extended Packet Reduction command described in Section 10.5.1.8.5.

### 10.5.1.10 System Ethernet MAC Address

#### 10.5.1.10.1 Get System Ethernet MAC Address Command (Intel Command 0x06)

In order to support a system configuration that requires the network controller to hold the Ethernet MAC address for the BMC (such as shared Ethernet MAC address mode), the following command is provided to enable the BMC to query the network controller for a valid Ethernet MAC address.

The network controller must return the system Ethernet MAC addresses. The BMC should use the returned Ethernet MAC addressing as a shared Ethernet MAC address by setting it using the Set Ethernet MAC Address command as defined in NC-SI 1.0.

It is also recommended that the BMC uses the Packet Reduction 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 | | | |

#### 10.5.1.10.2 Get System Ethernet MAC Address Response (Intel Command 0x06)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..27 | 0x06 | Ethernet MAC Address | | |
| 28..30 | Ethernet MAC Address | | | |

The MAC address is returned in network order.

# 10.5.1.11 Set Intel Management Control Formats

## 10.5.1.11.1 Set Intel Management Control Command (Intel Command 0x20)

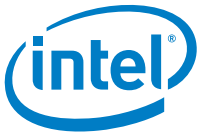
| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..22 | 0x20 | 0x00 | Intel Management Control | |

The Intel management control byte is defined in the following table:

| Bit # | Default | Description |
|---|---|---|
| 0 | 0b | Enable Critical Session Mode (the *Keep PHY Link Up* and *Veto* bits)<br>0b = Disabled<br>1b = Enabled<br>When critical session mode is enabled, the PHY is not reset on PE_RST# nor PCIe resets (in-band and link drop). Other reset events are not affected — LAN Power Good reset, Device Disable, Force TCO, and PHY reset by software.<br>The PHY does not change its power state. As a result, link speed does not change.<br>The device does not initiate configuration of the PHY to avoid losing link. |
| 1…7 | 0x0 | Reserved. |

## 10.5.1.11.2 Set Intel Management Control Response (Intel Command 0x20)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x20 | 0x00 | | |



## 10.5.1.12 Get Intel Management Control Formats

### 10.5.1.12.1 Get Intel Management Control Command (Intel Command 0x21)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..21 | 0x20 | 0x00 | | |

### 10.5.1.12.2 Get Intel Management Control Response (Intel Command 0x21)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..26 | 0x21 | 0x00 | Intel Management Control 1 | |

Intel Management Control 1 byte is described in Section 10.5.1.11.1.

## 10.5.1.13 TCO Reset

This command causes the network controller to perform TCO Reset, if Force TCO reset is enabled in the EEPROM.

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 network controller to enable the BMC to transmit/receive packets through the network controller.

When this command is issued to a channel in a package, it applies only to the specific channel.

After successfully performing the command the network controller considers Force TCO command as an indication that the operating system is hung and clears the DRV_LOAD flag (disable the driver).

#### 10.5.1.13.1 Perform Intel TCO Reset Command (Intel Command 0x22)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20 | 0x22 | TCO Mode[1] | | |

1. See Section 10.5.2.1.4.

#### 10.5.1.13.2 Perform Intel TCO Reset Response (Intel Command 0x22)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..26 | 0x22 | | | |

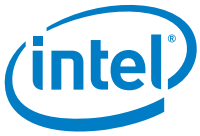
### 10.5.1.14 Checksum Offloading

This command enables the checksum offloading filters in the network controller.

When enabled, these filters block any packets that did not pass IP, UDP and TCP checksums from being forwarded to the BMC. This feature does not support tunneled IPv4/IPv6 packet inspection.

#### 10.5.1.14.1 Enable Checksum Offloading Command (Intel Command 0x23)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20 | 0x23 | | | |



### 10.5.1.14.2 Enable Checksum Offloading Response (Intel Command 0x23)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..26 | 0x23 | | | |

### 10.5.1.14.3 Disable Checksum Offloading Command (Intel Command 0x24)

This command causes the network controller to stop verifying the IP/UDP/TCP checksums.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20 | 0x24 | | | |

### 10.5.1.14.4 Disable Checksum Offloading Response (Intel Command 0x24)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..26 | 0x24 | | | |

## 10.5.1.15 LinkSec Support Commands

The following commands can be used by the BMC to control the different aspects of the LinkSec engine.

### 10.5.1.15.1 Transfer LinkSec Ownership to BMC Command (Intel Command 0x30, Parameter 0x10)

This command causes the 82599 to clear all LinkSec parameters, forcefully release host ownership and grant the ownership to the BMC. The BMC might allow the host to use the BMC's key for traffic by setting the *Host Control – Allow Host Traffic* bit. Activating this command clears all the LinkSec parameters.

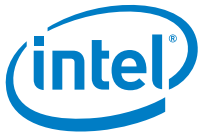
| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..22 | 0x30 | 0x10 | Host Control | |

**Table 10-15 LinkSec Host Control**

| Bit | Description |
|---|---|
| 0 | Reserved. |
| 1 | Allow Host Traffic:<br>0b = Host traffic is blocked.<br>1b = Host traffic is allowed. |
| 2..7 | Reserved. |

### 10.5.1.15.2 Transfer LinkSec Ownership to BMC Response (Intel Command 0x30, Parameter 0x10)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x30 | 0x10 | | |



### 10.5.1.15.3 Transfer LinkSec Ownership to Host Command (Intel Command 0x30, Parameter 0x11)

This command causes the 82599 to clear all LinkSec parameters, release BMC ownership and grant ownership to the host.

In this scenario traffic from/to the MC must be validated by the host's programmed keys. It is recommended that the MC try to establish network communication with a remote station to verify that the host was successful in programming the keys.

Activating this command clears all the LinkSec parameters.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..21 | 0x30 | 0x11 | | |

### 10.5.1.15.4 Transfer LinkSec Ownership to Host Response (Intel Command 0x30, Parameter 0x11)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x30 | 0x11 | | |

### 10.5.1.15.5 Initialize LinkSec Rx Command (Intel Command 0x30, Parameter 0x12)

This command can be used by the MC to initialize the LinkSec Rx engine. This command should be followed by a Set LinkSec Rx Key command to establish a LinkSec environment.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..23 | 0x30 | 0x12 | Rx Port Identifier | |
| 24..27 | Rx SCI [0..3] | | | |
| 28..29 | Rx SCI [4..5] | | | |

Where:

- **Rx Port Identifier** — the port number by which the NC identifies Rx packets. It is recommended that the MC use 0x0 as the port identifier. Note that the MC should use the same port identifier when performing the key-exchange.
- **Rx SCI** — A 6-byte unique identifier for the LinkSec Tx CA. It is recommended that the MC use its Ethernet MAC address value for this field.

### 10.5.1.15.6 Initialize LinkSec Rx Response (Intel Command 0x30, Parameter 0x12)

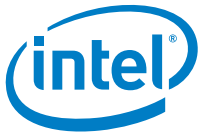
| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x30 | 0x12 | | |

### 10.5.1.15.7 Initialize LinkSec Tx Command (Intel Command 0x30, Parameter 0x13)

This command can be used by the MC to initialize the LinkSec Tx engine. This command should be followed by a Set LinkSec Tx Key command to establish a LinkSec environment.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..23 | 0x30 | 0x13 | Tx Port Identifier | |
| 24..27 | Tx SCI [0..3] | | | |
| 28..31 | Tx SCI [4..5] | | Reserved | |
| 32..35 | Packet Number Threshold | | | |
| 36 | Tx Control | | | |

- **Tx Port Identifier** — For this implementation this field is a don’t care and is automatically set to 0x0.
- **Tx SCI** — A 6-byte unique identifier for the LinkSec Tx CA. It is recommended that the MC use its Ethernet MAC address value for this field.
- **PN Threshold** — When a new key is programmed, the packet number is reset to 0x1. With each Tx packet, The packet number increments by one and is inserted to the packet (to avoid replay attacks). The PN threshold value is the 3 MSBytes of the Tx packet number after which a Key Exchange Required AEN is sent to the MC.



Example: a PN threshold of 0x123456 means that when the PN reaches 0x123456FF a notification is sent. The fourth byte of the PN threshold can be seen as a reserved bit, because it is always treated as 0xFF by the NC.

- **Tx Control**:

| Bit | Description |
|---|---|
| 0..4 | Reserved. |
| 5 | Always Include SCI in Tx:<br>0b = Do not include SCI in Tx packets.<br>1b = Include SCI in Tx packets. |
| 6..7 | Reserved. |

### 10.5.1.15.8 Initialize LinkSec Tx Response (Intel Command 0x30, Parameter 0x13)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x30 | 0x13 | | |

### 10.5.1.15.9 Set LinkSec Rx Key Command (Intel Command 0x30, Parameter 0x14)

This command can be used by the MC to set a new LinkSec Rx key. Upon receiving this command the NC must switch to the new Rx key and send the response.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..23 | 0x30 | 0x14 | Reserved | Rx SA AN |
| 24..27 | Rx LinkSec Key MSB | .. | .. | .. |
| 28..31 | .. | .. | .. | .. |
| 32..35 | .. | .. | .. | .. |
| 36..39 | .. | .. | .. | Rx LinkSec Key LSB |

Where:

- **Rx SA AN** — The association number to be used with this key.

- **Rx LinkSec Key** — the 128 bits (16 bytes) key to be used for Rx

### 10.5.1.15.10 Set LinkSec Rx Key Response (Intel Command 0x30, Parameter 0x14)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x30 | 0x14 | | |

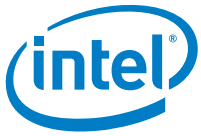
### 10.5.1.15.11 Set LinkSec Tx Key Command (Intel Command 0x30, Parameter 0x15)

This command can be used by the MC to set a new LinkSec Tx key. Upon receiving this command the NC must switch to the new Tx key and send the response.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..23 | 0x30 | 0x15 | Reserved | Tx SA AN |
| 24..27 | Tx LinkSec Key MSB | .. | .. | .. |
| 28..31 | .. | .. | .. | .. |
| 32..35 | .. | .. | .. | .. |
| 36..39 | .. | .. | .. | Tx LinkSec Key LSB |

Where:

- **Tx SA AN** — The association number to be used with this key.
- **Tx LinkSec Key** — the 128 bits (16 bytes) key to be used for Tx



### 10.5.1.15.12 Set LinkSec Tx Key Response (Intel Command 0x30, Parameter 0x15)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x30 | 0x15 | | |

### 10.5.1.15.13 Enable Network Tx Encryption Command (Intel Command 0x30, Parameter 0x16)

This command can be used by the MC to re-enable encryption of outgoing pass-through packets.

After this command is issued and until a response is received, the state of any outgoing packets is undetermined.

By default network Tx encryption is enabled.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..21 | 0x30 | 0x16 | | |

### 10.5.1.15.14 Enable Network Tx Encryption Response (Intel Command 0x30, Parameter 0x16)

Following sending this response the NC must stop encrypting outgoing pass-through packets.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x30 | 0x16 | | |

### 10.5.1.15.15 Disable Network Tx Encryption Command (Intel Command 0x30, Parameter 0x17)

This command can be used by the MC to disable encryption of outgoing pass-through packets.

After this command is issued and until a response is received, the state of any outgoing packets is undetermined.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..21 | 0x30 | 0x17 | | |

### 10.5.1.15.16 Disable Network Tx Encryption Response (Intel Command 0x30, Parameter 0x17)

Following sending this response the NC must start encrypting outgoing pass-through packets.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x30 | 0x17 | | |

### 10.5.1.15.17 Enable Network Rx Decryption Command (Intel Command 0x30, Parameter 0x18)

This command can be used by the MC to re-enable decryption of incoming pass-through packets. This causes the NC to execute LinkSec offload and to post the frames to the MC (or host) only if the LinkSec operation succeeds.

After this command is issued and until a response is received, the state of any incoming packets is undetermined.

By default network Rx decryption is disabled.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..21 | 0x30 | 0x18 | | |

### 10.5.1.15.18 Enable Network Rx Decryption Response (Intel Command 0x30, Parameter 0x18)

Following sending this response the NC must begin decrypting incoming pass-through packets.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x30 | 0x18 | | |

### 10.5.1.15.19 Disable Network Rx Decryption Command (Intel Command 0x30, Parameter 0x19)

This command can be used by the MC to disable decryption of incoming pass-through packets.

After this command is issued and until a response is received, the state of any incoming packets is undetermined.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..21 | 0x30 | 0x19 | | |

### 10.5.1.15.20 Disable Network Rx Decryption Response (Intel Command 0x30, Parameter 0x19)

Following sending this response the NC must stop decrypting incoming pass-through packets.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..25 | 0x30 | 0x19 | | |

### 10.5.1.15.21 Get LinkSec Parameters format (Intel Command 0x31)

The following commands can be used by the MC to retrieve the different LinkSec parameters.

These commands responses are valid only if the BMC owns the LinkSec.

### 10.5.1.15.22 Get LinkSec Rx Parameters Command (Intel Command 0x31, Parameter 0x01)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..22 | 0x31 | 0x01 | | |

### 10.5.1.15.23 Get LinkSec Rx Parameters Response (Intel Command 0x31, Parameter 0x01)

This command enables the MC to retrieve the currently configured set of Rx LinkSec parameters.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..27 | 0x31 | 0x01 | Reserved | |
| 28..31 | LinkSec Owner Status | LinkSec Host Control Status | Rx Port Identifier | |
| 32..35 | SCI [0..3] | | | |
| 36..39 | SCI [4..5] | | Reserved | Rx SA AN |
| 40..43 | Rx SA Packet Number | | | |

Where:

**Table 10-16 LinkSec Owner Status**

| Value | Description |
|---|---|
| 0x0 | Host is LinkSec owner. |
| 0x1 | BMC is LinkSec owner. |

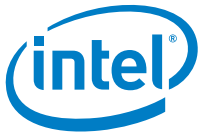


**Table 10-17 LinkSec Host Control Status**

| Bit | Description |
|---|---|
| 0 | Reserved. |
| 1 | Allow Host Traffic:<br>0b = Host traffic is blocked.<br>1b = Host traffic is allowed. |
| 2..7 | Reserved. |

- **Rx Port Identifier** — The Rx Port identifier
- **Rx SCI** — The Rx SCI identifier.
- **Rx SA AN** — The association number associated with the active SA (for which the last valid Rx LinkSec packet was received).
- **Rx SA Packet Number** — Is the last packet number, as read from the last valid Rx LinkSec packet.

### 10.5.1.15.24 Get LinkSec Tx Parameters Command (Intel Command 0x31, Parameter 0x02)

This command enables the MC to retrieve the currently configured set of Tx LinkSec parameter.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Manufacturer ID (Intel 0x157) | | | |
| 20..22 | 0x31 | 0x02 | | |

### 10.5.1.15.25 Get LinkSec Tx Parameters Response (Intel Command 0x31, Parameter 0x02)

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Manufacturer ID (Intel 0x157) | | | |
| 24..27 | 0x31 | 0x2 | Reserved | |
| 28..31 | LinkSec Owner Status | LinkSec Host Control Status | Tx Port Identifier | |
| 32..35 | SCI [0..3] | | | |
| 36..39 | SCI [4..5] | | Reserved | Tx SA AN |
| 40..43 | Tx SA Packet Number | | | |
| 44.47 | Packet Number Threshold | | | |
| 48 | Tx Control Status | | | |

Where:

**Table 10-18 LinkSec Owner Status**

| Value | Description |
|---|---|
| 0x0 | Host is LinkSec owner. |
| 0x1 | BMC is LinkSec owner. |

**Table 10-19 LinkSec Host Control Status**

| Bit | Description |
|---|---|
| 0 | Reserved. |
| 1 | Allow Host Traffic:<br>0b = Host traffic is blocked.<br>1b = Host traffic is allowed. |
| 2..7 | Reserved. |

- **Tx Port Identifier** — Reserved to 0x0 for this implementation.
- **Tx SCI** — The Rx SCI identifier.
- **Tx SA AN** — The association number currently used for the active SA.
- **Tx SA Packet Number** — Is the last packet number, as read from the last valid Rx LinkSec packet.

- **Packet Number Threshold:**

**Table 10-20 Tx Control Status:**

| Bit | Description |
|---|---|
| 0..4 | Reserved. |
| 5 | Include SCI:<br>0b = Do not include SCI in Tx packets.<br>1b = Include SCI in Tx packets. |
| 6..7 | Reserved. |

## 10.5.1.16 LinkSec AEN (Intel AEN 0x80)

The following is the AEN that can be sent by the NC following a LinkSec event.

This AEN must be enabled using the NC-SI AEN Enable command, using bit 16 (0x10000) of the AEN enable mask.

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI AEN Header | | | |
| 20..23 | Reserved | | | 0x80 |
| 24..27 | Reserved | | | LinkSec Event Cause |

Where:

*LinkSec Event Cause* has the following format:

| Bit # | Description |
|---|---|
| 0 | Host requested ownership. |
| 1 | Host released ownership. |
| 2 | Tx Key Packet Number (PN) threshold met. |
| 3..7 | Reserved. |

## 10.5.2 SMBus Programming

This section describes the SMBus transactions supported in Advanced Pass Through (APT) mode.

### 10.5.2.1 Write SMBus Transactions (BMC → the 82599)

The following table lists the different SMBus write transactions supported by the 82599.

| TCO Command | Transaction | Command | | Fragmentation | Section |
|---|---|---|---|---|---|
| Transmit Packet | Block Write | First:<br>Middle:<br>Last: | 0x84<br>0x04<br>0x44 | Multiple | 10.5.2.1.1 |
| Transmit Packet | Block Write | Single: | 0xC4 | Single | 10.5.2.1.1 |
| Receive Enable | Block Write | Single: | 0xCA | Single | 10.5.2.1.3 |
| Management Control | Block Write | Single: | 0xC1 | Single | 10.5.2.1.5 |
| Update MNG RCV filter parameters | Block Write | Single: | 0xCC | Single | 10.5.2.1.6 |
| Force TCO | Block Write | Single: | 0xCF | Single | 10.5.2.1.4 |
| Request Status | Block Write | Single: | 0xDD | Single | 10.5.2.1.2 |
| Update LinkSec parameters | Block Write | Single: | 0xC9 | Single | 10.5.2.1.7 |

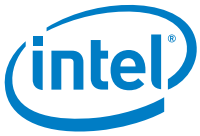
#### 10.5.2.1.1 Transmit Packet Command

The Transmit Packet command behavior is detailed in section 3.2.5. The Transmit Packet fragments have the following format:

| Function | Command | Byte Count | Data 1 | ... | Data N |
|---|---|---|---|---|---|
| Transmit first fragment | 0x84 | N | Packet data MSB | ... | Packet data LSB |
| Transmit middle fragment | 0x04 | | | | |
| Transmit last fragment | 0x44 | | | | |
| Transmit single fragment | 0xC4 | | | | |

The payload length is limited to the maximum payload length set in the EEPROM.

If the overall packet length is bigger than 1536 bytes, the packet is silently discarded by the 82599.



#### 10.5.2.1.2 Request Status Command

The BMC can initiate a request to read the 82599 manageability status by sending this command.

When it receives this command, the 82599 initiates a notification to the BMC (when it is ready with the status), and then the BMC is able to read the status, by issuing a Read Status command (see section 10.5.2.2.3). Request Status Command format:

| Function | Command | Byte Count | Data 1 |
|---|---|---|---|
| Request status | 0xDD | 1 | 0 |

#### 10.5.2.1.3 Receive Enable Command

The Receive Enable command is a single fragment command that is used to configure the 82599.

This command has two formats: short, 1-byte legacy format (providing backward compatibility with previous components) and long, 14-byte advanced format (allowing greater configuration capabilities).

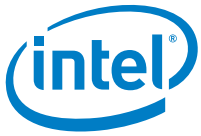
**Note:** If the Receive Enable command is short and thus does not include all the parameters, then the parameters are taken from most recent previous configuration (either the most recent long Receive Enable command in which the particular value was set, or the EEPROM if there was no such previous long Receive Enable command).

| Func. | Cmd | Byte Count | Data 1 | Data 2 | ... | Data 7 | Data 8 | ... | Data 11 | Data 12 | Data 13 | Data 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Legacy receive enable | 0xCA | 1 | Receive control byte | - | ... | - | - | ... | - | - | - | - |
| Advanced receive enable | | 14 0x0E | | MAC addr. MSB | | MAC addr. LSB | IP addr. MSB | | IP addr. LSB | BMC SMBus addr. | Interf. data byte | Alert value byte |

While...

- **Receive control byte** (data byte 1) has the following format:

| Field | Bit(s) | Description |
|---|---|---|
| RCV_EN | 0 | Receive TCO Enable.<br>0b = Disable Receive TCO packets.<br>Rx Packets are not directed to BMC and Auto ARP response is not enabled.<br>1b = Enable Receive TCO packets.<br>Setting this bit enables all manageability receive filtering operation. The enable of the specific filtering is done through loading the Receive Enable 1 word in the EEPROM, or through special configuration command (see Section 10.5.2.1.6). |
| RCV_ALL | 1 | Receive All Enable.<br>When set to 1b, all LAN packets received over the wire that passed L2 filtering are forwarded to the BMC. This flag is meaningful only if the RCV_EN bit is set as well. |
| EN_STA | 2 | Enable Status reporting when set to 1b. |
| EN_ARP_RES | 3 | Enable ARP Response.<br>0b = Disable.<br>The 82599 treats ARP packets as any other packet. These packets are forwarded to BMC if it passes other (non-ARP) filtering.<br>1b = Enable.<br>The 82599 automatically responds to all received ARP requests that match its IP address.<br>*Note:* Setting this bit doesn't change the Rx filtering settings. Appropriate Rx filtering to enable ARP request packets to reach the manageability unit should be set by the BMC or by the EEPROM.<br>The BMC IP address is provided as part of the Receive Enable message (bytes 8-11). If short version of the command is used the 82599 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 82599 uses the IP address configured in the EEPROM as ARP response IPv4 address in pass-through LAN configuration structure. If *CBDM* bit is set the 82599 uses the BMC dedicated Ethernet MAC address in ARP response packets. If the *CBDM* bit is not set, BMC uses the host Ethernet MAC address.<br>Setting this bit requires appropriate assertion of bits RCV_EN and RCV_ALL. Otherwise, the command aborts with no processing. |
| NM | 5:4 | Notification Method.<br>Defines the notification method that the 82599 uses.<br>00b = SMBus alert<br>01b = Asynchronous notify<br>10b = Direct receive<br>11b = Not supported.<br>*Note:* In dual SMBus address mode, both SMBus addresses must be configured to the same notification method. |
| Reserved | 6 | Reserved. |
| CBDM | 7 | Configure BMC dedicated Ethernet MAC address.<br>*Note:* This bit should be 0b when the RCV_EN bit (bit 0) is not set.<br>0b = The 82599 shares the same Ethernet MAC address for manageability and host defined in the EEPROM LAN Core 0/1 Modules in the EEPROM.<br>1b = The 82599 uses a dedicated Ethernet MAC address. The BMC Ethernet MAC address is set in bytes 2-7 in this command.<br>If short version of the command is used, the 82599 uses the Ethernet MAC address configured in the most recent long version of the command in which the *CBDM* bit was set. If no such previous long command exists, then the 82599 uses the Ethernet MAC address configured in the MMAL and MMAH fields in the EEPROM.<br>When the dedicated Ethernet MAC address feature is activated, the 82599 uses the following registers for Rx filtering. The BMC should not modify the following registers:<br>MNG Decision Filter – MDEF7 (and its corresponding bit MANC2H[7])<br>MNG Ethernet MAC Address 3 – MMAL3 and MMAH3 (and its corresponding bit MFVAL[3]). |



- MNG Ethernet MAC address (data bytes 2-7)

Ignored if CBDM bit is not set. This Ethernet MAC address is used for configuration of the dedicated Ethernet MAC address. In addition, it is used in the ARP response packet, when EN_ARP_RES bit is set. This Ethernet MAC address continues to be used when the *CBDM* bit is set in subsequent short versions of this command.

- MNG IP address (data bytes 8-11)

Ignored if EN_ARP_RES bit is not set. This IP address is used to filter ARP request packets. This IP address continues to be used when EN_ARP_RES is set in subsequent short versions of this command.

- Asynchronous notification SMBus address (data byte 12)

This address is used for the asynchronous notification SMBus transaction and for direct receive.

- Interface data (data byte 13)

Interface data byte to be used in asynchronous notification.

- Alert data (data byte 14).

Alert value data byte to be used in the asynchronous notification.

### 10.5.2.1.4 Force TCO Command

This command causes the 82599 to perform a TCO reset, if Force TCO reset is enabled in word Common Firmware Parameters in the EEPROM. The Force TCO reset clears the data path (Rx/Tx) of the 82599 to enable the BMC to transmit/receive packets through the 82599.

**Note:** In single address mode, both ports are reset when the command is issued. In dual address mode, Force TCO reset is asserted only to the port related to the SMB address the command was issued to.

The 82599 considers the Force TCO command as an indication that the operating system is hung and clears the DRV_LOAD flag.

Force TCO Reset command format:

| Function | Command | Byte Count | Data 1 |
|---|---|---|---|
| Force TCO reset | 0xCF | 1 | TCO mode |

TCO mode is listed in the following table:

| Field | Bit(s) | Description |
|---|---|---|
| DO_TCO_RST | 0 | Do TCO reset.<br>0b = Do nothing.<br>1b = Perform TCO reset. |
| Reserved | 1 | Reserved, set to 0b. |

| Field | Bit(s) | Description |
|---|---|---|
| Firmware Reset[1] | 2 | Reset manageability and re-load manageability related EEPROM words<br>0b = Do nothing.<br>1b = Issue firmware reset to manageability.<br>*Note:* Setting this bit generates a one time firmware reset event. Following a firmware reset, management related data from the EEPROM is loaded. |
| Reserved | 7:3 | Reserved, (Set to 0x00). |

1. Before initiating a Firmware Reset command, disable TCO receive via the Receive Enable command, set RCV_EN to 0b, and then wait for 200 milliseconds before initiating the Firmware Reset command. In addition, the BMC should not transmit during this period.

### 10.5.2.1.5 Management Control

This command is used to set generic manageability parameters. The parameters are listed in the following table. The command is 0xC1, which states 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 listed in the table.

**Note:** If in the update configuration, the parameter that the BMC sets is not supported by the 82599, the 82599 does not NACK the transaction. After the transaction ends, the 82599 discards the data and asserts a transaction abort status (see Section 3.2.5.2).

Following is the format of the Management Control command:

| Function | Command | Byte Count | Data 1 | Data 2 | ... | Data N |
|---|---|---|---|---|---|---|
| Management Control | 0xC1 | N | Parameter Number (PN#) | Parameter Dependent | | |

This table lists the different parameters and their content:

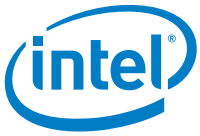
| Parameter | PN# | Parameter Data |
|---|---|---|
| Keep PHY Link Up | 0x00 | A single byte parameter — Data 2:<br>Bit 0 Programming of the MMNGC.MNG_VETO bit.<br>Bit [7:1] Reserved. |

### 10.5.2.1.6 Update MNG RCV Filter Parameters

This command is used to set the manageability receive filters parameters. The parameters are listed in the following table. The command is 0xCC, which states that it is a parameter update. The first data byte is the parameter number and the data afterwards (length and content) are parameter specific as listed in the table.

**Note:** If in the update configuration, the parameter that the BMC sets is not supported by the 82599, the 82599 does not NACK the transaction. After the transaction ends, the 82599 discards the data and asserts a transaction abort status (see Section 3.2.5.2).

Detailed description of receive filtering capabilities and configuration is described in Section 10.3.



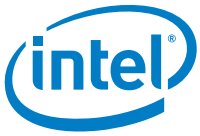
The format of the update MNG RCV filter parameters is listed in the following table:

| Function | Command | Byte Count | Data 1 | Data 2 | ... | Data N |
|---|---|---|---|---|---|---|
| Update MNG RCV Filter Parameters | 0xCC | N | Parameter Number (PN#) | Parameter Dependent | | |

The following table lists the different parameters and their contents**:**

| Parameter | PN# | Parameter Data |
|---|---|---|
| Filters Enable | 0x1 | Defines generic filters configuration.<br>The structure of this parameter is 4 bytes as the MANC Value LSB and MANC Value MSB loaded from the EEPROM.<br>*Note:* General filter enable is in the Receive Enable command, which enable receive filtering. This parameter specifies which filters should be enabled. ARP filtering and dedicated Ethernet MAC address can also be enabled through the Receive Enable command (see Section 10.5.2.1.3). |
| MNG2HOST configuration | 0xA | This parameter defines which manageability packets are directed to the host memory as well.<br>Data 2:5 = MNG2H register setting (Data 2 is the MSB). |
| Fail-Over configuration | 0xB | Fail-Over Structure Configuration (see Section 10.2.2.2.4).<br>The bytes of this parameter are loaded to the fail-over configuration register.<br>Data 2:5 = Fail-over configuration register (Data 2 is the MSB). |
| Flex Filter 0 Enable MASK and Length | 0x10 | Flex Filter 0 Mask.<br>Data 2:17 = MASK. Bit 0 in data 2 is the first bit of the MASK<br>Data 18:19 = Reserved. Should be zero.<br>Data 20 = Flexible Filter length (must be >= 2). |
| Flex Filter 0 Data | 0x11 | Data 2 – Group of flex filter's bytes:<br>0x0 = bytes 0-29.<br>0x1 = bytes 30-59.<br>0x2 = bytes 60-89.<br>0x3 = bytes 90-119.<br>0x4 = bytes 120-127.<br>Data 3:32 = Flex filter data bytes. Data 3 is LSB.<br>Group's length is not mandatory 30 bytes; it can vary according to filter's length and must NOT be padded by zeros. |
| Flex Filter 1 Enable MASK and Length | 0x20 | Same as parameter 0x10 but for filter 1. |
| Flex Filter 1 Data | 0x21 | Same as parameter 0x11 but for filter 1 |
| Flex Filter 2 Enable MASK and Length | 0x30 | Same as parameter 0x10 but for filter 2. |
| Flex Filter 2 Data | 0x31 | Same as parameter 0x11 but for filter 2. |
| Flex Filter 3 Enable MASK and Length | 0x40 | Same as parameter 0x10 but for filter 3. |
| Flex Filter 3 Data | 0x41 | Same as parameter 0x11 but for filter 3. |

| Parameter | PN# | Parameter Data |
|---|---|---|
| Filters Valid | 0x60 | 4 bytes to determine which of the the 82599 filter registers contain valid data.<br>Loaded into the MFVAL0 and MFVAL1 registers. Should be updated after the contents of a filter register are updated.<br>Data 2 = MSB of MFVAL ... Data 5 is the LSB |
| Decision Filters | 0x61 | 5 bytes to load the Manageability Decision Filters (MDEF).<br>Data 2 = Decision filter number.<br>Data 3 = MSB of MDEF register for this decision filter ... Data 6 is the LSB. |
| VLAN Filters | 0x62 | 3 bytes to load the VLAN tag filters (MAVTV).<br>Data 2 = VLAN filter number.<br>Data 3 = MSB of VLAN filter.<br>Data 4 = LSB of VLAN filter. |
| Flex Ports Filters | 0x63 | 3 bytes to load the manageability flex port filters (MFUTP).<br>Data 2 = Flex port filter number.<br>Data 3 = MSB of flex port filter.<br>Data 4 = LSB of flex port filter. |
| IPv4 Filters | 0x64 | 5 bytes to load the IPv4 address filter (MIPAF, DW 15:12).<br>Data 2 = IPv4 address filter number (0-3).<br>Data 3 = MSB of IPv4 address filter … Data 6 is the LSB. |
| IPv6 Filters | 0x65 | 17 bytes to load IPv6 address filter (MIPAF).<br>Data 2 = IPv6 address filter number (0-3).<br>Data 3 = MSB of IPv6 address filter … Data 18 is the LSB. |
| MAC Filters | 0x66 | 7 bytes to load Ethernet MAC address filters (MMAL, MMAH).<br>Data 2 = Ethernet MAC address filters pair number (0-3).<br>Data 3 = MSB of Ethernet MAC address … Data 8 is the LSB. |
| Ethertype Filters | 0x67 | 6 bytes to load Ethertype filters (MTQF).<br>Data 2 = METF filter index (valid values are 0..3).<br>Data 3 = MSB of METF ... Data 6 is the LSB. |
| Extended Decision Filter | 0x68 | 10 bytes to load the extended decision filters (MDEF_EXT & MDEF).<br>Data 2 = MDEF filter index (valid values are 0..6).<br>Data 3 = MSB of MDEF_EXT (DecisionFilter1) ... Data 6 is the LSB.<br>Data 7 = MSB of MDEF (DecisionFilter0) ... Data 10 is the LSB.<br>The command must overwrite any previously stored value.<br>*Note:* Previous Decision Filter command (0x61) is still supported. For legacy reasons — If previous Decision Filter command (0x61) is called — it should set the MDEF as provided and set the extended Decision Filter (MDEF_EXT) to 0x0. |



### 10.5.2.1.7 Update LinkSec Parameters

This command is used to set the manageability LinkSec parameters. The parameters are listed in the following table. The first data byte is the parameter number and the data afterwards (length and content) are parameter specific as listed in the table.

This is the format of the Update LinkSec parameters command:

| Function | Command | Byte Count | Data 1 | Data 2 | ... | Data N |
|---|---|---|---|---|---|---|
| Update LinkSec Filter Parameters | 0xC9 | N | Parameter Number (PN#) | Parameter Dependent | | |

The following table lists the different parameters and their contents**:**

| Parameter | PN# | Parameter Data |
|---|---|---|
| Transfer LinkSec ownership to BMC | 0x10 | Data 2: Host Control:<br>Bit 0 = Reserved.<br>Bit 1 = Allow host traffic (0b – blocked, 1b – allowed).<br>Bit 2...31 = Reserved. |
| Transfer LinkSec ownership to Host | 0x11 | No data needed. |
| Initialize LinkSec Rx | 0x12 | Data 2: Rx Port Identifier (MSB) ... Data 3: (LSB).<br>Rx Port Identifier – the port number by which the 82599 identifies Rx packets. It is recommended that the BMC use 0x0 as the port identifier.<br>*Note:* The BMC should use the same port identifier when performing the key-exchange.<br>Data 4 : Rx MAC SecY (MSB) … Data 9: (LSB). |
| Initialize LinkSec Tx | 0x13 | Data 2: Tx Port Identifier (MSB) ... Data 3: (LSB) — must be set to zero.<br>Data 4: Tx SCI (MSB) … Data 7: Tx SCI (LSB).<br>Tx SCI – A 6-byte unique identifier for the LinkSec Tx CA. It is recommended that the BMC use its Ethernet MAC address value for this field.<br>Data 8: Reserved.<br>Data 9: Reserved.<br>Data 10: Packet Number Threshold (MSB) … Data 12: (LSB).<br>PN Threshold – When a new key is programmed, the packet number is reset to 0x1. With each Tx packet, The packet number is incremented by one and inserted to the packet (to avoid replay attacks). The packet number threshold value is 3 MSBytes of the Tx Packet number after which a Key Exchange Required AEN is sent to the BMC. Example: a PN threshold of 0x123456 means that when the packet number reaches 0x12345600 a notification is sent.<br>Data 22: Tx Control — See Table 10-21. |
| Set LinkSec Rx Key | 0x14 | Data 2: Reserved.<br>Data 3: Rx SA AN (The association number to be used with this key).<br>Data 4: Rx LinkSec Key (MSB) … Data 19: (LSB) — (16 bytes key to be used). |
| Set LinkSec Tx Key | 0x15 | Data 3: Tx SA AN (The association number to be used with this key).<br>Data 4: Tx LinkSec Key (MSB) … Data 19: (LSB) — (16 bytes key to be used). |

| Parameter | PN# | Parameter Data |
|---|---|---|
| Enable LinkSec Network Tx encryption | 0x16 | No data needed. |
| Disable LinkSec Network Tx encryption | 0x17 | No data needed. |

**Table 10-21 Tx Control**

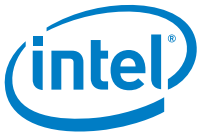
| Bit | Description |
|---|---|
| 0..4 | Reserved. |
| 5 | Always Include SCI in Tx:<br>0b = Do not include SCI in Tx packets.<br>1b = Include SCI in Tx packets. |
| 6..7 | Reserved. |

## 10.5.2.2 Read SMBus Transactions (the 82599 to BMC)

The following table lists the different SMBus read transactions supported by the 82599. All the read transactions are compatible with SMBus Read Block Protocol format.

| TCO Command | Transaction | Command | Op-Code | | Fragmentation | Section |
|---|---|---|---|---|---|---|
| Receive TCO Packet | Block Read | 0xC0 or 0xD0 | First:<br>Middle:<br>Last[1] | 0x90<br>0x10<br>0x50 | Multiple | 10.5.2.2.1 |
| Read Receive Enable configuration | Block Read | 0xDA | Single: | 0xDA | Single | 10.5.2.2.7 |
| Read the 82599 Status | Block Read | 0xC0 or 0xD0 or 0xDE | Single: | 0xDD | Single | 10.5.2.2.3 |
| Read Management parameters | Block Read | 0xD1 | Single: | 0xD1 | Single | 10.5.2.2.5 |
| Read MNG RCV filter parameters | Block Read | 0xCD | Single: | 0xCD | Single | 10.5.2.2.6 |
| Get system Ethernet MAC Address | Block Read | 0xD4 | Single | 0xD4 | Single | 10.5.2.2.4 |
| Read LinkSec parameters | Block Read | 0xD9 | Single | 0xD9 | Single | 10.5.2.2.8 |

1. Last fragment of the receive TCO packet is the packet status.



**Note:** The 82599 responds to one of the commands 0xC0/0xD0 within the time defined in the SMBus notification timeout and flags word in the EEPROM (see Section 6.4.4.3.)

0xC0/0xD0 commands are used for more than one payload. If the BMC issues these read commands, and the 82599 has no pending data to transfer, it always returns as default opcode 0xDD with the 82599 status, and does not NACK the transaction.

If an SMBus Quick Read command is received, it is handled as a Read the 82599 Status command (See Section 10.5.2.2.3 for details).

#### 10.5.2.2.1 Receive TCO LAN Packet Transaction

The BMC uses this command to read the packet received on the LAN and its status. When the 82599 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 LAN receive packet, the BMC should begin issuing a Receive TCO packet command using the block read protocol. The packet can be delivered in more than one SMBus fragment (at least two — one for the packet, and the other one for the status), and the BMC should follow the *F* and *L* bit.

The opcode can have these values:

- 0x90 — First fragment.
- 0x10 — Middle fragment.
- 0x50 — Packet status (last fragment) as described in Section 10.5.2.2.2.

If the external BMC does not finish reading the entire packet within a timeout period since the packet has arrived, the packet is silently discarded. The timeout period is set according to the SMBus notification timeout EEPROM parameter (see Section 6.4.4.3)

| Function | Command |
|---|---|
| Receive TCO packet | 0xC0 or 0xD0 |

Data returned from the 82599:

| Function | Byte Count | Data 1 (Op-Code) | Data 2 | ... | Data N |
|---|---|---|---|---|---|
| Receive TCO First Fragment | N | 90 | Packet Data Byte | ... | Packet Data Byte |
| Receive TCO Middle Fragment | | 10 | | | |
| Receive TCO Last Fragment | | 50 | | | |

#### 10.5.2.2.2 Receive TCO LAN Status Payload Transaction

This transaction is the last transaction that the 82599 issues when a packet that was 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:

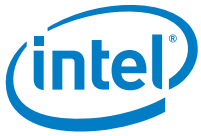
| Function | Byte Count | Data 1 (Op-Code) | Data 2 – Data 17 (Status data) |
|---|---|---|---|
| Receive TCO Long Status | 17 (0x11) | 0x50 | See Table 10-22. For more details on the specific bit fields see Section 7.1.6. |

**Table 10-22 Receive TCO Last Fragment Status Data Content**

| Name | Bit(s) | Description |
|---|---|---|
| Packet Length | 13:0 | Packet length including CRC, only 14 LSB bits. |
| Reserved | 24:14 | Reserved. |
| CRC | 25 | CRC stripped indication. |
| Reserved | 28:26 | Reserved. |
| VEXT | 29 | Additional VLAN present in packet. |
| Reserved | 33:30 | Reserved. |
| Reserved | 34 | Reserved. |
| LAN | 35 | LAN number. |
| Reserved | 63:36 | Reserved. |
| Reserved | 71:64 | Reserved. |
| Status | 79:72 | See Table 10-23. |
| Reserved | 87:80 | Reserved. |
| MNG status | 127:88 | See Table 10-24. This field should be ignored if Receive TCO is not enabled. |

**Table 10-23 Status Info**

| Field | Bit(s) | Description |
|---|---|---|
| Reserved | 7:4 | Reserved. |
| IPCS | 3 | IPv4 Checksum Calculated on Packet. |
| L4CS | 2 | L4 Checksum Calculated on Packet. |



### Table 10-23 Status Info (Continued)

| Field | Bit(s) | Description |
|---|---|---|
| UDPCS | 1 | UDP Checksum Calculated on Packet. |
| Reserved | 0 | Reserved. |

### Table 10-24 MNG Status

| Name | Bits | Description |
|---|---|---|
| Pass RMCP 0x026F | 0 | Set when the UDP/TCP port of the MNG packet is 0x26F. |
| Pass RMCP 0x0298 | 1 | Set when the UDP/TCP port of the MNG packet is 0x298. |
| Pass MNG Broadcast | 2 | Set when the MNG packet is a broadcast packet. |
| Pass MNG Neighbor | 3 | Set when the MNG packet is a neighbor discovery packet. |
| Pass ARP req / ARP Response | 4 | Set when the MNG packet is an ARP response/request packet. |
| Reserved | 7:5 | Reserved. |
| Pass MNG VLAN Filter Index | 10:8 | |
| MNG VLAN Address Match | 11 | Set when the MNG packet matches one of the MNG VLAN filters. |
| Unicast Address Index | 14:12 | Indicates which of the 4 unicast Ethernet MAC addresses match the packet. Valid only if the unicast address match is set. |
| Unicast Address Match | 15 | Set when there is a match to any of the 4 unicast Ethernet MAC addresses. |
| L4 port Filter Index | 22:16 | Indicate the flex filter number. |
| L4 port Match | 23 | Set when there is a match to any of the UDP / TCP port filters. |
| Flex TCO Filter Index | 26:24 | |
| Flex TCO Filter Match | 27 | |
| IP address Index | 29:28 | Set when there is a match to the IP filter number. (IPv4 or IPv6). |
| IP address Match | 30 | Set when there is a match to any of the IP address filters. |
| IPv4 Packet | 31 | Set to 0b when packet is IPv4 (regardless of address match). |
| Decision Filter Match | 39:32 | Set when there is a match to one of the decision filters. |

#### 10.5.2.2.3 Read Status Command

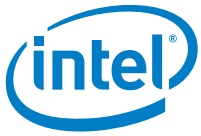
The BMC can read the 82599 status. The 82599 asserts an alert prior to the BMC reading the status bytes. There can be two reasons for the 82599 to send status to the BMC (described in Section 3.2.3):

1. The external BMC asserts a request for reading the 82599 status.
2. The 82599 detects a status change as described in Section 3.2.3.

Note that commands 0xC0/0xD0 are for backward compatibility. 0xD0/0xC0 can be used for other payloads the 82599 defines in the opcode, which payload this transaction is. When 0xDE command is set, the 82599 always returns opcode 0sDD with the 82599 status. The BMC reads the event causing the notification, using the Read Status command as follows:

| Function | Command |
|---|---|
| Read Status | 0xC0 or 0xD0 or 0xDE |

| Function | Byte Count | Data 1 (Op-Code) | Data 2 (Status data 1) | Data 3 (Status data 2) |
|---|---|---|---|---|
| Receive TCO Partial Status | 3 | 0xDD | See the following table | |



The following table lists the status data byte 1:

| Bit | Name | Description |
|---|---|---|
| 7 | LAN Port | 0b = Alert came from LAN port 0.<br>1b = Alert came from LAN port 1 |
| 6 | TCO Command Aborted | 0b = A TCO command abort event has not occurred since the last read status cycle.<br>1b = A TCO command abort event has occurred since the last read status cycle.<br>See Section 3.2.5.2 for command abort flow. |
| 5 | Link Status Indication[1] | 0b = LAN link down.<br>1b = LAN link is up. |
| 4 | PHY Link Forced Up | Contains the value of the MMNGC.MNG_VETO bit. |
| 3 | Initialization Indication[2] | 0b = An EEPROM reload event has not occurred since the last read status cycle.<br>1b = An EEPROM reload event has occurred since the last read status cycle. |
| 2 | Reserved | Reserved as 0b. |
| 1:0 | Power State[3] | 00b = Dr state.<br>01b = D0u state.<br>10b = D0 state.<br>11b = D3 state. |

1. When the 82599 is working in teaming mode, and presented as one SMBus device, the link indication is 0b only when both links (on both ports) are down. If one of the LANs is disabled, its link is considered to be down.
2. This indication is asserted when the 82599 manageability block reloads the EEPROM and its internal database is updated to EEPROM default values. This is an indication that the external BMC should re-configure the 82599, if other values besides the EEPROM default should be configured.
3. In single address mode, the 82599 reports the highest power-state modes in both devices. The D state is marked in this order: D0, D0u, Dr, and D3.

Status data byte 2 is used for the BMC for an indication whether the LAN driver is alive and running.

The driver valid indication is a bit that is set by the driver when it is coming up, and cleared when it goes down to Dx state or cleared by the hardware on PCI reset.

Bits 2 and 1 indicate that the LAN driver is not stuck. Bit 2 indicates whether the interrupt line of the LAN function is asserted, and bit 1 indicates whether the driver between the last read status cycle dealt the interrupt line.

The following table lists status data byte 2:

| Bit | Name | Description |
|---|---|---|
| 7 | Reserved | Reserved. |
| 6 | Reserved | Reserved. |
| 5 | Reserved | Reserved. |
| 4 | LinkSec Indication | If set, indicates that a LinkSec event has occurred.<br>Use the read LinkSec parameters with the LinkSec interrupt cause parameter to read the interrupt cause |

| Bit | Name | Description |
|---|---|---|
| 3 | Driver Valid Indication | 0b = LAN driver is not alive.<br>1b = LAN driver is alive. |
| 2 | Interrupt Pending Indication | 0b = LAN interrupt is not asserted.<br>1b = LAN interrupt is asserted. |
| 1 | ICR Register Read/Write | 0b = ICR register was not read since the last read status cycle.<br>1b = ICR register was read since the last read status cycle.<br>Reading the ICR means that the driver has dealt with the interrupt that was asserted. |
| 0 | Reserved | Reserved. |

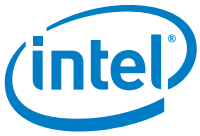
*Note:* When the 82599 is in teaming mode, these bits represent both cores:

- The driver alive indication is set if 1b of the driver is alive.
- The LAN interrupt is considered to be asserted if one of the interrupt lines is asserted.
- The ICR is considered to read if one of the ICRs was read (LAN0 or LAN1).

The following table lists the possible values of bits 2, 1 and what the BMC can assume according to that:

| Previous | Current | |
|---|---|---|
| Don’t care | 00b | Interrupt is not pending – 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 more than one read status cycle – Not OK (possible driver hang state). |
| Don’t Care | 11b | Previous interrupt was read and current interrupt is pending – OK. |
| Don’t Care | 10b | Interrupt is not pending – OK. |

**Note:** The BMC reads should consider the time it takes for the driver to deal with the interrupt (a few microseconds), too frequent reads give false indications.



#### 10.5.2.2.4 Get System Ethernet MAC Address

The Get System Ethernet MAC Address returns the system Ethernet MAC Address (RAL0, RAH0) over the SMBus. This command is a single fragment Read Block transaction, with the following format:

| Function | Command |
|---|---|
| Get system Ethernet MAC Address | 0xD4 |

Data returned from the 82599:

| Function | Byte Count | Data 1 (Op-Code) | Data 2 | ... | Data 7 |
|---|---|---|---|---|---|
| Get system Ethernet MAC Address | 7 | 0xD4 | Ethernet MAC Address MSB | ... | Ethernet MAC Address LSB |

#### 10.5.2.2.5 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.

This is the block write transaction:

| Function | Command | Byte Count | Data 1 |
|---|---|---|---|
| Management Control Request | 0xC1 | 1 | Parameter Number |

Following the block write the BMC should issue a block read that reads the parameter that was set in the Block Write command:

| Function | Command |
|---|---|
| Read Management Parameter | 0xD1 |

Data returned from the 82599:

| Function | Byte Count | Data 1 (Op-Code) | Data 2 | Data 3 | ... | Data N |
|---|---|---|---|---|---|---|
| Read Management Parameter | N | 0xD1 | Parameter Number (PN#) | Parameter Dependent | | |

The returned data is as follows:

| Parameter | PN# | Parameter Data |
|---|---|---|
| Keep PHY Link Up | 0x00 | A single byte parameter — Data 2:<br>Bit 0 = Reflects the setting of the MMNGC.MNG_VETO bit.<br>Bit [7:1] = Reserved. |
| Wrong Parameter Request | 0xFE | the 82599 only: This parameter is returned on a read transaction, if in the previous Read command the BMC sets a parameter that is not supported by the 82599. |
| the 82599 Not Ready | 0xFF | the 82599 only: Returned on Read Parameters command when the data that should have been read is not ready. The BMC should retry the read transaction. |

**Note:** It might be that the parameter that is returned is not the parameter requested by the BMC. The BMC should verify the parameter number (default parameter to be returned is 0x1).

It is BMC's responsibility to follow the procedure Previously defined. If the BMC sends a Block Read command (as previously described) that is not preceded by a Block Write command with bytecount=1b, the 82599 sets the parameter number in the read block transaction to be 0xFE.

### 10.5.2.2.6 Read MNG RCV Filter Parameters

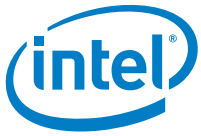
In order to read the MNG RCV filter parameters, the BMC should execute two SMBus transactions. The first transaction is a block write that sets the parameter that the BMC wants to read. The second transaction is block read that reads the parameter.

This is the block write transaction:

| Function | Command | Byte Count | Data 1 | Data 2 |
|---|---|---|---|---|
| Update MNG RCV Filter Parameters | 0xCC | 1 or 2 | Parameter Number (PN#) | Parameter Data |

The following table lists the different parameters and their contents**:**

| Parameter | PN# | Parameter Data |
|---|---|---|
| Filters Enable | 0x1 | None. |
| MNG2HOST Configuration | 0xA | None. |
| Fail-Over Configuration | 0xB | None. |
| Flex Filter 0 Enable Mask and Length | 0x10 | None. |



| Parameter | PN# | Parameter Data |
|---|---|---|
| Flex Filter 0 Data | 0x11 | Data 2 – Group of Flex filter's bytes:<br>0x0 = bytes 0-29.<br>0x1 = bytes 30-59.<br>0x2 = bytes 60-89.<br>0x3 = bytes 90-119.<br>0x4 = bytes 120-127. |
| Flex Filter 1 Enable Mask and Length | 0x20 | None. |
| Flex Filter 1 Data | 0x21 | Same as parameter 0x11 but for filter 1. |
| Flex Filter 2 Enable Mask and Length | 0x30 | None. |
| Flex Filter 2 Data | 0x31 | Same as parameter 0x11 but for filter 2. |
| Flex Filter 3 Enable Mask and Length | 0x40 | None. |
| Flex Filter 3 Data9 | 0x41 | Same as parameter 0x11 but for filter 3. |
| Filters Valid | 0x60 | None. |
| Decision Filters | 0x61 | 1 byte to define the accessed manageability decision filter (MDEF).<br>Data 2 = Decision filter number. |
| VLAN Filters | 0x62 | 1 byte to define the accessed VLAN tag filter (MAVTV).<br>Data 2 = VLAN filter number. |
| Flex Ports Filters | 0x63 | 1 byte to define the accessed manageability flex port filter (MFUTP).<br>Data 2 = Flex port filter number. |
| IPv4 Filter | 0x64 | 1 byte to define the accessed IPv4 address filter (MIPAF).<br>Data 2 = IPv4 address filter number. |
| IPv6 Filters | 0x65 | 1 byte to define the accessed IPv6 address filter (MIPAF).<br>Data 2 = IPv6 address filter number. |
| MAC Filters | 0x66 | 1 byte to define the accessed Ethernet MAC address filters pair (MMAL, MMAH).<br>Data 2 = Ethernet MAC address filters pair number (0-3). |
| Wrong Parameter Request | 0xFE | Returned by the 82599 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 82599. |
| the 82599 Not Ready | 0xFF | Returned by the 82599 only, on Read Parameters command when the data that should have been read is not ready. This parameter has no data. |

Following the block write the BMC should issue a block read that readS the parameter that was set in the Block Write command:

| Function | Command |
|---|---|
| Request MNG RCV Filter Parameters | 0xCD |

Data returned from the 82599:

| Function | Byte Count | Data 1 (Op-Code) | Data 2 | Data 3 | ... | Data N |
|---|---|---|---|---|---|---|
| Read MNG RCV Filter Parameters | N | 0xCD | Parameter Number (PN#) | Parameter Dependent | | |

The returned data is in the same format of the Update command.

**Note:** If the parameter that is returned is not the parameter requested by the BMC, the BMC should verify the parameter number (default parameter to be returned is 0x1).

If the parameter number is 0xFF, it means that the data that the 82599 should supply is not ready yet. The BMC should retry the read transaction.

It is BMC's responsibility to follow the procedure previously defined. If the BMC sends a Block Read command (as previously described) that is not preceded by a Block Write command with bytecount=1b, the 82599 sets the parameter number in the read block transaction to be 0xFE.

### 10.5.2.2.7 Read Receive Enable Configuration

The BMC uses this command to read the receive configuration data. This data can be configured in the Receive Enable command or through EEPROM loading at power up.

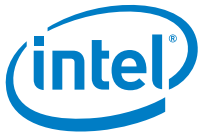
Read Receive Enable Configuration command format (SMBus Read Block Protocol):

| Function | Command |
|---|---|
| Read Receive Enable | 0xDA |

Data returned from the 82599:

| 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 | Ethernet MAC Address MSB | ... | Ethernet MAC Address LSB | IP Address MSB | ... | IP Address LSB | BMC SMBus Address | Interface Data Byte | Alert Value Byte |

The detailed description of each field is specified in the Receive Enable command description in Section 10.5.2.1.3.



### 10.5.2.2.8 Read LinkSec Parameters

In order to read the MNG LinkSec 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.

This is the block write transaction:

| Function | Command | Byte Count | Data 1 | Data 2 |
|---|---|---|---|---|
| Update MNG RCV Filter Parameters | 0xC9 | 1 | Parameter Number (PN#) | Parameter Data |

The following table lists the different parameters and their contents:

| Parameter | PN# | Parameter Data |
|---|---|---|
| LinkSec Interrupt Cause | 0x0 | None. |
| LinkSec Rx Parameters | 0x1 | None. |
| LinkSec Tx Parameters | 0x2 | None. |

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 | Byte Count | Data 1 | Data 2 — n |
|---|---|---|---|---|
| Read LinkSec Parameters | 0xD9 | 2,18 or 22 | Parameter Number (PN#) | Parameter Data |

The following table lists the different parameters and their contents**:**

| Parameter | PN# | Parameter Data |
|---|---|---|
| LinkSec Interrupt Cause | 0x0 | This command must return 1 byte (Data2). This byte contains the LinkSec interrupt cause, according to the following values:<br>Data2:<br>Bit 0 = Tx key packet number threshold met.<br>Bit 1 = Host requested ownership.<br>Bit 2 = Host released ownership.<br>Bit 3...31 = Reserved. |

| Parameter | PN# | Parameter Data |
|---|---|---|
| LinkSec Rx Parameters | 0x1 | Data 2: Reserved.<br>Data 3: LinkSec ownership status. See Table 10-25.<br>Data 4: LinkSec host control status. See Table 10-26.<br>Data 5: Rx host identifier (MSB).<br>Data 6: Rx host identifier (LSB).<br>Data 7: Rx SCI (MSB) … Data 12: (LSB).<br>Data 13: Reserved.<br>Data 14: Rx SA AN — The association number currently used for the active SA.<br>Data 15: Rx SA packet number (MSB) … Data 18: (LSB).<br>Rx SA packet number is the last packet number, as read from the last valid Rx LinkSec packet. |
| LinkSec Tx Parameters | 0x2 | Data 2: Reserved<br>Data 3: LinkSec ownership status. See Table 10-25.<br>Data 4: LinkSec host control status. See Table 10-26.<br>Data 5: Tx port identifier (MSB).<br>Data 6: Tx port identifier (LSB).<br>*Note:* Tx port identifier is reserved to 0x0 for this implementation.<br>Data 7: Tx SCI (MSB) … Data 12: (LSB).<br>Data 13: Reserved.<br>Data 14: Tx SA AN — The association number currently used for the active SA.<br>Data 15: Tx SA packet number (MSB) … Data 18: (LSB).<br>Data 19: packet number threshold (MSB) … Data 21: (LSB).<br>Tx SA packet number is the last packet number, as read from the last valid Tx LinkSec packet.<br>Data 22: Tx Control Status. See Table 10-27. |

### Table 10-25 LinkSec Owner Status

| Value | Description |
|---|---|
| 0x0 | Host is LinkSec owner. |
| 0x1 | BMC is LinkSec owner. |

### Table 10-26 LinkSec Host Control Status

| Bit | Description |
|---|---|
| 0 | Reserved. |
| 1 | Allow host traffic:<br>0b = Host traffic is blocked.<br>1b = Host traffic is allowed. |
| 2..7 | Reserved. |

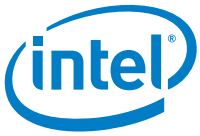


**Table 10-27 Tx Control Status**

| Bit | Description |
|---|---|
| 0..4 | Reserved. |
| 5 | Include SCI:<br>0b = Do not include SCI in Tx packets.<br>1b = Include SCI in Tx packets. |
| 6..7 | Reserved |

## 10.5.2.3 SMBus ARP Transactions

**Note:** All SMBus-ARP transactions include PEC byte.

### 10.5.2.3.1 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. It is used to signal all devices that the ARP master is starting the ARP process:

| 1 | 7 | 1 | 1 | 8 | 1 | 8 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|
| S | Slave Address | Wr | A | Command | A | PEC | A | P |
| | 1100 001 | 0 | 0 | 0000 0001 | 0 | [Data dependent value] | 0 | |

### 10.5.2.3.2 Reset Device (General)

This command clears the Address Resolved flag (set to false). It does not affect the status or validity of the dynamic SMBus address.

| 1 | 7 | 1 | 1 | 8 | 1 | 8 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|
| S | Slave Address | Wr | A | Command | A | PEC | A | P |
| | 1100 001 | 0 | 0 | 0000 0010 | 0 | [Data dependent value] | 0 | |

### 10.5.2.3.3 Reset Device (Directed)

The Command field is NACK-ed if the bits 7 through 1 do not match the current the 82599 SMBus address.

It 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 | Targeted slave address \| 0 | 0 | [Data dependent value] | 0 | |

### 10.5.2.3.4 Assign Address

This command assigns the 82599's SMBus address. The address and command bytes are always acknowledged.

The transaction is aborted immediately (NACK-ed-) if any of the UDID bytes differ from the 82599 UDID bytes. If successful, the manageability interface updates the SMBus address internally. This command also sets the Address Resolved flag to true.

| 1 | 7 | 1 | 1 | 8 | 1 | 8 | 1 | |
|---|---|---|---|---|---|---|---|---|
| | Slave Address | Wr | A | Command | A | Byte Count | A | ••• |
| | 1100 001 | 0 | 0 | 0000 0100 | 0 | 0001 0001 | 0 | |

| 8 | 1 | 8 | 1 | 8 | 1 | 8 | 1 | |
|---|---|---|---|---|---|---|---|---|
| Data-1 | A | Data-2 | A | Data-3 | A | Data-4 | A | ••• |
| UDID byte 15 (MSB) | 0 | UDID byte 14 | 0 | UDID byte 13 | 0 | UDID byte 12 | 0 | |

| 8 | 1 | 8 | 1 | 8 | 1 | 8 | 1 | |
|---|---|---|---|---|---|---|---|---|
| Data-5 | A | Data-6 | A | Data-7 | A | Data-8 | A | ••• |
| UDID byte 11 | 0 | UDID byte 10 | 0 | UDID byte 9 | 0 | UDID byte 8 | 0 | |

| 8 | 1 | 8 | 1 | 8 | 1 | |
|---|---|---|---|---|---|---|
| Data-9 | A | Data-10 | A | Data-11 | A | ••• |
| UDID byte 7 | 0 | UDID byte 6 | 0 | UDID byte 5 | 0 | |

| 8 | 1 | 8 | 1 | 8 | 1 | 8 | 1 | |
|---|---|---|---|---|---|---|---|---|
| Data-12 | A | Data-13 | A | Data-14 | A | Data-15 | A | ••• |
| UDID byte 4 | 0 | UDID byte 3 | 0 | UDID byte 2 | 0 | UDID byte 1 | 0 | |

| 8 | 1 | 8 | 1 | 8 | 1 | 1 |
|---|---|---|---|---|---|---|
| Data-16 | A | Data-17 | A | PEC | A | P |
| UDID byte 0 (LSB) | 0 | Assigned Address | 0 | [Data dependent value] | 0 | |

### 10.5.2.3.5 Get UDID (General and Directed)

The Get UDID command depends on whether this is a directed or general command.

The General Get UDID SMBus transaction 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 will always be 1b.

If the SMBus address has been resolved (Address Resolved flag is true); for a general command the manageability interface does not acknowledge (NACK) this transaction, for a directed command the manageability always acknowledges (ACK) this transaction.

This command does not affect the status or validity of the dynamic SMBus address nor of the Address Resolved flag.

The command returns the UDID bytes as defined in Section 3.2.7.

| S | Slave Address | Wr | A | Command | A | S | ••• |
|---|---|---|---|---|---|---|---|
| | 1100 001 | 0 | 0 | See below | 0 | | |

| 7 | 1 | 1 | 8 | 1 | |
|---|---|---|---|---|---|
| Slave Address | Rd | A | Byte Count | A | ••• |
| 1100 001 | 1 | 0 | 0001 0001 | 0 | |

| 8 | 1 | 8 | 1 | 8 | 1 | 8 | 1 | |
|---|---|---|---|---|---|---|---|---|
| Data-1 | A | Data-2 | A | Data-3 | A | Data-4 | A | ••• |
| UDID byte 15 (MSB) | 0 | UDID byte 14 | 0 | UDID byte 13 | 0 | UDID byte 12 | 0 | |

| 8 | 1 | 8 | 1 | 8 | 1 | 8 | 1 | |
|---|---|---|---|---|---|---|---|---|
| Data-5 | A | Data-6 | A | Data-7 | A | Data-8 | A | ••• |
| UDID byte 11 | 0 | UDID byte 10 | 0 | UDID byte 9 | 0 | UDID byte 8 | 0 | |

| 8 | 1 | 8 | 1 | 8 | 1 | |
|---|---|---|---|---|---|---|
| Data-9 | A | Data-10 | A | Data-11 | A | ••• |
| UDID byte 7 | 0 | UDID byte 6 | 0 | UDID byte 5 | 0 | |

| 8 | 1 | 8 | 1 | 8 | 1 | 8 | 1 | |
|---|---|---|---|---|---|---|---|---|
| Data-12 | A | Data-13 | A | Data-14 | A | Data-15 | A | ••• |
| UDID byte 4 | 0 | UDID byte 3 | 0 | UDID byte 2 | 0 | UDID byte 1 | 0 | |

| 8 | 1 | 8 | 1 | 8 | 1 | 1 |
|---|---|---|---|---|---|---|
| Data-16 | A | Data-17 | A | PEC | ~Ã | P |
| UDID byte 0 (LSB) | 0 | Device Slave Address | 0 | [Data dependent value] | 1 | |

## 10.5.2.4 Example Configuration Steps

This section provides an overview and sample configuration settings for commonly used 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. Here is a sample:

```
Receive Enable[00]
```

Using the simple form of the Receive Enable command, this prevents any packets from reaching the BMC by disabling filtering.

#### 10.5.2.4.1 Example 1 - Shared MAC, RMCP Only Ports

This example is the most basic configuration. The MAC address filtering are shared with the host operating system and only traffic directed the RMCP ports (0x26F and 0x298) are filtered. For this simple example, the BMC must issue gratuitous ARPs because no filter is enabled to pass ARP requests to the BMC.

##### 10.5.2.4.1.1 Example 1 Pseudo Code

**Step 1:** - Disable existing filtering:

```
Receive Enable[00]
```

Using 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:** - Configure MDEF[0]:

```
Update Manageability Filter Parameters [61, 0, 00000C00]
```

Use the Update Manageability Filter Parameters command to update Decision Filters (MDEF) (parameter 0x61). This updates MDEF[0], as indicated by the 2nd parameter (0).

MDEF[0] value of 0x00000C00:

- Bit 10 [1] – port 0x298
- Bit 11 [1] – port 0x26F

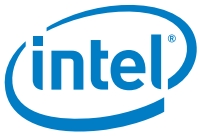
**Step 3:** - Enable filtering:

```
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 = SMBus Alert
- Bit 7 [0] – Use shared MAC



### Table 10-28 Example 1 MDEF Results

| | | Manageability Decision Filter (MDEF) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Filter | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| L2 Unicast Address | AND | | | | | | | | |
| Broadcast | AND | | | | | | | | |
| Manageability VLAN | AND | | | | | | | | |
| IP Address | AND | | | | | | | | |
| L2 Unicast Address | OR | | | | | | | | |
| Broadcast | OR | | | | | | | | |
| Multicast | AND | | | | | | | | |
| ARP Request | OR | | | | | | | | |
| ARP Response | OR | | | | | | | | |
| Neighbor Discovery | OR | | | | | | | | |
| Port 0x298 | OR | x | | | | | | | |
| Port 0x26F | OR | x | | | | | | | |
| Flex Port 15:0 | OR | | | | | | | | |
| Flex TCO 3:0 | OR | | | | | | | | |

#### 10.5.2.4.2 Example 2 - Dedicated MAC, Auto ARP Response and RMCP port filtering

This example shows a common configuration; the BMC has a dedicated MAC and IP address. Automatic ARP responses are enabled as well as RMCP port filtering. By enabling automatic ARP responses the BMC is not required to send the gratuitous ARPs as it did in Example 1. Since ARP requests are now filtered, in order for the host to receive the ARP requests, the manageability-to-host filter is configured to send the ARP requests to the host as well.

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.

Additionally, the XSUM filtering is enabled.

Note that not all Intel Ethernet controllers support automatic ARP responses, please refer to product specific documentation.

##### 10.5.2.4.2.1 Example 2 - Pseudo Code

**Step 1:** - Disable existing filtering:

```
Receive Enable[00]
```

Using 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.

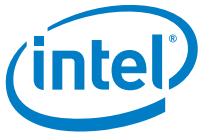
MANC Register 0x00800000:

- Bit 23 [1] – XSUM Filter enable

**Note:** Some of the following configuration steps manipulate the MANC register indirectly, this command sets all bits except XSUM to zero. 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 [61, 0, 00000C00]
```



Use the Update Manageability Filter Parameters command to update Decision Filters (MDEF) (parameter 0x61). This updates MDEF[0], as indicated by the 2nd parameter (0).

MDEF value of 0x00000C00:

- Bit 10 [1] – port 0x298
- Bit 11 [1] – port 0x26F

**Step 5:** - Configure MDEF[1]:

```
Update Manageability Filter Parameters [61, 1, 00000080]
```

Use the Update Manageability Filter Parameters command to update Decision Filters (MDEF) (parameter 61h). This updates MDEF[1], as indicated by the 2nd parameter (1).

MDEF value of 0x00000080:

- Bit 7 [7] – ARP requests

When enabling automatic ARP responses, the ARP requests still go into the manageability filtering system and as such need to be designated as also needing to be sent to the host. For this reason a separate MDEF is created with only ARP request filtering enabled.

Refer to the next step for more details.

**Step 6:** - Configure the Management to Host Filter

```
Update Manageability Filter Parameters [0A, 00000002]
```

Use the Update Manageability Filter Parameters command to update the Management Control-to-Host (MANC2H) register.

MANC2H Register 0x00000002:

- Bit 2 [1] – Enable MDEF[1] traffic to go to the host as well

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-to-host configuration then not only would ARP requests be sent to the BMC and host, RMCP traffic (ports 0x26F and 0x298) would have also been sent to both places.

**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 = SMBus alert
- Bit 7 [1] – Use dedicated MAC

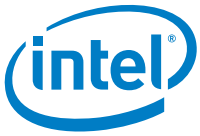
Second parameter is the MAC address (AABBCCDD).

Third parameter is the IP address (01020304).

The last three parameters are zero when the notification method is SMBus Alert.

### Table 10-29 Example 2 MDEF Results

| | | Manageability Decision Filter (MDEF) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Filter | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| L2 Unicast Address | AND | | | | | | | | x |
| Broadcast | AND | | | | | | | | |
| Manageability VLAN | AND | | | | | | | | |
| IP Address | AND | | | | | | | | |
| L2 Unicast Address | 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 15:0 | OR | | | | | | | | |
| Flex TCO 3:0 | OR | | | | | | | | |



#### 10.5.2.4.3 Example 3 - Dedicated MAC and IP Address

This example provides the BMC with a dedicated MAC and IP address and allows it to receive ARP requests. The BMC is then responsible for responding to ARP requests.

For demonstration purposes, the dedicated MAC address is calculated by reading the system MAC address and adding one to it, assume the system MAC is AABBCCDC. The IP address for this example is1.2.3.4. 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 can also receive ARP requests.

##### 10.5.2.4.3.1 Example 3 - Pseudo Code

**Step 1:** - Disable existing filtering:

```
Receive Enable[00]
```

Using 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 manageability filters valid to select the IPv4 [0] and MAC[0]filters:

Step 3 configured one of possibly many IP address filters, this step indicates which of those filters should be used when filtering incoming traffic.

```
Update Manageability Filter Parameters [60, 00010001]
```

Use the Update Manageability Filter Parameters to configure the MVFAL register.

The 1st parameter (0x60) specifies that we are configuring the MFVAL register.

The 2nd parameter (0x00010001) is the new value of the MFVAL register.

MFVAL value of 0x00010000:

- Bit 1  [1] – MAC Address Filter 0
- Bit 16 [1] – IPV4 Filter 0

For more information regarding Manageability Filters Valid, see section Section 8.2.3.25.5.

**Step 6:** - Configure MDEF[0] for IP and MAC filtering:

```
Update Manageability Filter Parameters [61, 0, 00000009]
```

Use the Update Manageability Filter Parameters command to update Decision Filters (MDEF) (parameter 0x61). This will update MDEF[0], as indicated by the 2nd parameter (0).

MDEF value of 0x00000040:

- Bit 1 [1] - MAC Address Filtering
- Bit 3 [1] – IP Address Filtering

**Step 7:** - Configure MDEF[1]:

```
Update Manageability Filter Parameters [61, 1, 00000080]
```

Use the Update Manageability Filter Parameters command to update Decision Filters (MDEF) (parameter 0x61). This will update MDEF[1], as indicated by the 2nd parameter (1).

MDEF value of 00000080:

- Bit 7 [7] – ARP Requests

When filtering ARP requests the requests go into the manageability filtering system and as such need to be designated as also needing to be sent to the host. For this reason a separate MDEF is created with only ARP request filtering enabled.

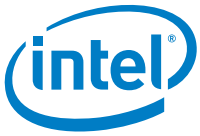
**Step 8:** - Configure the management to host filter:

```
Update Manageability Filter Parameters [0A, 00000002]
```

Use the Update Manageability Filter Parameters command to update the Management Control-to-Host (MANC2H) register.

MANC2H Register 00000002:

- Bit 2 [1] – Enable MDEF[1] traffic to go to the host as well



**Step 9:** - 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 = SMBus Alert

The resulting MDEF filters are as follows:

**Table 10-30 Example 3 MDEF Results**

| | | Manageability Decision Filter (MDEF) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Filter | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| L2 Unicast Address | AND | x | | | | | | | |
| Broadcast | AND | | | | | | | | |
| Manageability VLAN | AND | | | | | | | | |
| IP Address | AND | x | | | | | | | |
| L2 Unicast Address | OR | | | | | | | | |
| Broadcast | OR | | | | | | | | |
| Multicast | AND | | | | | | | | |
| ARP Request | OR | | x | | | | | | |
| ARP Response | OR | | | | | | | | |
| Neighbor Discovery | OR | | | | | | | | |
| Port 0x298 | OR | | | | | | | | |
| Port 0x26F | OR | | | | | | | | |
| Flex Port 15:0 | OR | | | | | | | | |
| Flex TCO 3:0 | OR | | | | | | | | |

#### 10.5.2.4.4 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 0x32 is required for traffic to be sent to the BMC. This means that all traffic with VLAN a matching tag is sent to the BMC.

For demonstration purposes, the dedicated MAC address is calculated by reading the system MAC address and adding one do it, assume the system MAC is AABBCCDC. The IP address for this example is 1.2.3.4 and the VLAN tag will be 0x0032.

It is assumed the host is not using the same VLAN tag as the BMC. If they were to share the same VLAN tag then additional filtering would need to be configured to allow VLAN tagged non-unicast (such as ARP requests) to be sent to the host as well as the BMC using the manageability-to -host filter capability.

Additionally, the XSUM filtering is enabled.

##### 10.5.2.4.4.1 Example 4 - Pseudo Code

**Step 1:** - Disable existing filtering:

```
Receive Enable[00]
```

Using 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:

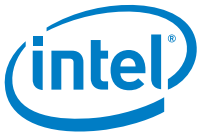
- Bit 23 [1] – XSUM Filter enable

**Note:** Some of the following configuration steps manipulate the MANC register indirectly, this command sets all bits except XSUM to zero. 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 can 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 2$^{nd}$ parameter indicates which VLAN filter (0 in this case), the last parameter is the VLAN ID (0x0032).



**Step 5:** - Enable VLAN 0 filter:

```
Update Manageability Filter Parameters [60, 00000100]
```

The previous step configured a VLAN filter, this step enables it.

Use the Update Manageability Filter Parameters command to enable the VLAN filter (VLAN filter 0) configured in the previous step, this information is written to the Manageability Filters Valid (MFVAL) register. See Section 8.2.3.25.5 for more details about MFVAL.

MFVAL value of 00000100:

- Bit 8 [1] – VLAN Filter 0

**Step 6:** - Configure MDEF[0]:

```
Update Manageability Filter Parameters [61, 0, 00000040]
```

Use the Update Manageability Filter Parameters command to update Decision Filters (MDEF) (parameter 0x61). This updates MDEF[0], as indicated by the 2nd parameter (0).

MDEF value of 00000040:

- Bit 2 [1] – VLAN AND

**Step 7:** - 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] b – Enable receiving of packets
- Bit 2 [1] b – Enable status reporting (such as link lost)
- Bit 5:4 [00]b – Notification method = SMBus Alert
- Bit 7 [1]b – Use dedicated MAC

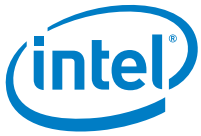
Second parameter is the MAC address: AABBCCDD.

Third parameter is the IP address: 01020304.

The last three parameters are zero when the notification method is SMBus Alert.

### Table 10-31 Example 3 MDEF Results

| | | Manageability Decision Filter (MDEF) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Filter | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| L2 Unicast Address | AND | | | | | | | | x |
| Broadcast | AND | | | | | | | | |
| Manageability VLAN | AND | x | | | | | | | |
| IP Address | AND | | | | | | | | |
| L2 Unicast Address | OR | | | | | | | | |
| Broadcast | OR | | | | | | | | |
| Multicast | AND | | | | | | | | |
| ARP Request | OR | | | | | | | | |
| ARP Response | OR | | | | | | | | |
| Neighbor Discovery | OR | | | | | | | | |
| Port 0x298 | OR | | | | | | | | |
| Port 0x26F | OR | | | | | | | | |
| Flex Port 15:0 | OR | | | | | | | | |
| Flex TCO 3:0 | OR | | | | | | | | |



#### 10.5.2.5 SMBus Troubleshooting and Recommendations

This section outlines the most common issues found while working with pass-through using the SMBus sideband interface.

##### 10.5.2.5.1 SMBus Commands Are Always NACK'd

There are several reasons why all commands sent to the 82599 from a MC could be NACK'd. The following are most common:

- Invalid NVM Image — The image itself might be invalid or it could be a valid image and is not a pass-through image, as such SMBus connectivity is disabled.
- The MC is not using the correct SMBus address — Many MC vendors hard-code the SMBus address(es) into their firmware. If the incorrect values are hard-coded, the 82599 does not respond.
- The SMBus address(es) can be dynamically set using the SMBus ARP mechanism.
- Bus Interference — the bus connecting the MC and the 82599 might be unstable, consult the reference schematics for correct pull-up resisters.

##### 10.5.2.5.2 SMBus Clock Speed Is 16.6666 KHz

This can happen when the SMBus connecting the MC and the 82599 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 82599 and the MC to this device.

##### 10.5.2.5.3 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 MC.

The management control to host configuration, in this situation, was setup not to send RMCP traffic to the operating system (this is typically the correct configuration). This means that no traffic sent to port 623 was being routed.

The solution in this case is to configure the problematic application NOT to use the reserved port 0x26F.

##### 10.5.2.5.4 Unable to Transmit Packets from the MC

If the MC has been transmitting and receiving data without issue for a period of time and then begins to receive NACKs from the 82599 when it attempts to write a packet, the problem is most likely due to the fact that the buffers internal to the 82599 are full of data that has been received from the network but has yet to be read by the MC.

Being an embedded device, the 82599 has limited buffers that are shared for receiving and transmitting data. If a MC does not keep the incoming data read, the 82599 can be filled up This prevents the MC form transmitting more data, resulting in NACKs.

If this situation occurs, the recommended solution is to have the MC issue a Receive Enable command to disable more incoming data, read all the data from the 82599, and then use the Receive Enable command to enable incoming data.

#### 10.5.2.5.5 SMBus Fragment Size

The SMBus specification indicates a maximum SMBus transaction size of 32 bytes. Most of the data passed between the 82599 and the MC over the SMBus is RMCP/RMCP+ traffic, which by its very nature (UDP traffic) is significantly larger than 32 bytes in length. Multiple SMBus transactions may therefore be required to move data from the 82599 to the MC or to send a data from the MC to the 82599.

Recognizing this bottleneck, the 82599 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.

#### 10.5.2.5.6 Losing Link

Normal behavior for the Ethernet controller when the system powers down or performs a reset is for the link to temporarily go down and then back up again to re-negotiate the link speed. This behavior can have adverse affects on manageability.

For example if there is an active FTP or Serial Over LAN (SOL) session to the MC, this connection may be lost. In order to avoid this possible situation, the MC can use the Management Control command detailed in Section 10.5.2.1.5 to ensure the link stays active at all times.

This command is available when using the NC-SI sideband interface as well.

Care should be taken with this command, if the driver negotiates the maximum link speed, the link speed remains the same when the system powers down or resets. This may have undesirable power consumption consequences. Currently, when using NC-SI, the MC can re-negotiate the link speed. That functionality is not available when using the SMBus interface.

#### 10.5.2.5.7 Enable XSum Filtering

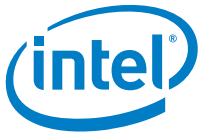
If XSum filtering is enabled, the MC does not need to perform the task of checking this checksum for incoming packets. Only packets that have a valid XSum is passed to the MC. All others are silently discarded.

This is a way to offload some work from the MC.

#### 10.5.2.5.8 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 82599, rather than the NVM image provided by Intel. Parts of the NVM image are changed after writing (such as the physical NVM size).



## 10.5.3 Manageability Host Interface

### 10.5.3.1 HOST CSR Interface (Function 1/0)

The software device driver of function 0/1 communicates with the manageability block through CSR access. The manageability is mapped to address space 0x15800 to 0x15FFF on the slave bus of each function.

**Note:** Writing to address 0x15800 from function 0 or from function 1 is targeted to the same address in the RAM.

### 10.5.3.2 Host Slave Command Interface to Manageability

This interface is used by the software device driver for several of the commands and for delivering various types of data in both directions (manageability-to-host and host-to-manageability).

The address space is separated into two areas:

- Direct access to the internal ARC data RAM: The internal data RAM is mapped to address space 0x15800 to 0x15EFF. Writing/reading to this address space goes directly to the RAM.
- Control registers are located at address 0x15F00.

### 10.5.3.3 Host Slave Command Interface Low Level Flow

This interface is used for the external host software to access the manageability subsystem. Host software writes a command block or read data structure directly from the data RAM. Host software controls these transactions through a slave access to the control register.

The following flow shows the process of initiating a command to the manageability block:

1. The software device driver reads the control register and checks that the *Enable* bit is set.
2. The software device driver writes the relevant command block into the RAM area.
3. The software device driver sets the *Command* bit in the control register. Setting this bit causes an interrupt to the ARC (can be masked).
4. The software device driver polls the control register for the command bit to be cleared by hardware.
5. When manageability finishes with the command, it clears the command bit (if the manageability should reply with data, it should clear the bit only after the data is in the RAM area where the software device driver can read it).

If the software device driver reads the control register and the *SV* bit is set, then there is a valid status of the last command in the RAM. If the *SV* bit is not set, then the command has failed with no status in the RAM.

### 10.5.3.4 Host Slave Command Registers

The Host Slave Command registers (listed below) are described in Section 8.2.3.27.1. These register participates in the Host / Software / Firmware interface:

Host Interface Control Register — CSR Address 0x15F00; AUX 0x0700

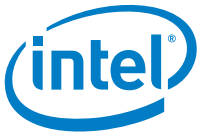
Firmware Status 0 (FWS0R) Register — CSR Address 0x15F0C; AUX 0x0702

Software Status Register — CSR Address 0x15F10; AUX 0x0703

### 10.5.3.5 Host Interface Command Structure

The following table describes the structure used by the host driver to send a command to manageability firmware via the host interface slave command interface:

| #Byte | Description | Bit | Value | Description |
|---|---|---|---|---|
| 0 | Command | 7:0 | Command Dependent | Specifies which host command to process. |
| 1 | Buffer Length | 7:0 | Command Length | Command Data Buffer length: 0 to 252, not including 32 bits of header. |
| 2 | Default/ Implicit Interface | 0 | Command Dependent | Used for commands might refer to one of two interfaces (LAN or SMBus).<br>0b = Use default interface.<br>1b = Use specific interface. |
| | Interface Number | 1 | Command Dependent | Used when bit 0 (Default/Implicit interface) is set:<br>0b = Apply command for interface 0.<br>1b = Apply command for interface 1.<br>When bit 0 is set to 0b, it is ignored. |
| | Reserved | 7:2 | 0x0 | Reserved |
| 3 | Checksum | 7:0 | Defined Below | Checksum signature. |
| 255:4 | Data Buffer | 7:0 | Command Dependent | Command Specific Data<br>Minimum buffer size: 0.<br>Maximum buffer size: 252. |



### 10.5.3.6 Host Interface Status Structure

The following table lists the structure used by manageability firmware to return a status to the host driver via the host interface slave command interface. A status is returned after a command has been executed.

| #Byte | Description | Bit | Value | Description |
|---|---|---|---|---|
| 0 | Command | 7:0 | Command Dependent | Command ID. |
| 1 | Buffer Length | 7:0 | Status Dependent | Status buffer length: 252:0 |
| 2 | Return Status | 7:0 | Depends on Command Executing Results | Defined in commands description. |
| 3 | Checksum | 7:0 | Defined Below | Checksum signature. |
| 255:4 | Data Buffer | | Status Dependent | Status configuration parameters<br>Minimum Buffer Size: 0.<br>Maximal Buffer Size: 252. |

### 10.5.3.7 Checksum Calculation Algorithm

The Host Command/Status structure is summed with this field cleared to 0b. The calculation is done using 8-bit unsigned math with no carry. The inverse of this sum is stored in this field (0b minus the result). Result: The current sum of this buffer (8-bit unsigned math) is 0b.

### 10.5.3.8 Host Slave Interface Commands

In SMBus PT mode the only host interface command that is supported is the fail-over configuration command (besides debug commands that will not be described in this document).

#### 10.5.3.8.1 Fail-Over Configuration Host Command

This command is used to update the Fail-Over Configuration register:

| Byte | Description | Bit | Value | Description |
|---|---|---|---|---|
| 0 | Command | 7:0 | 0x26 | Fail-over configuration command. |
| 1 | Buffer Length | 7:0 | 0x4 | Four bytes of the fail-over configuration register. |
| 2 | | 7:0 | 0x0 | |

| Byte | Description | Bit | Value | Description |
|---|---|---|---|---|
| 3 | Checksum | 7:0 | | Checksum signature of the Host command. |
| 7:4 | Reserved | 7:0 | Reserved | Reserved. |

Following is the status returned on this command:

| Byte | Description | Bit | Value | Description |
|---|---|---|---|---|
| 0 | Command | 7:0 | 0x26 | Four bytes of the fail over register value. |
| 1 | Buffer Length | 7:0 | 0x0 | No data in return status. |
| 2 | Return Status | 7:0 | 0x1 | 0x1 for good status. |
| 3 | Checksum | 7:0 | | Checksum signature. |

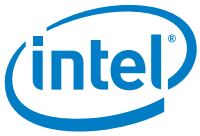
### 10.5.3.8.2 Read Fail-Over Configuration Host Command

This command is used to read the Fail-Over Configuration register:

| Byte | Description | Bit | Value | Description |
|---|---|---|---|---|
| 0 | Command | 7:0 | 0x27 | Read Fail-Over Configuration command. |
| 1 | Buffer Length | 7:0 | 0x0 | No data attached to this command. |
| 2 | | 7:0 | 0x0 | |
| 3 | Checksum | 7:0 | | Checksum signature of the Host command. |
| | | | | |

Following is the status returned on this command:

| Byte | Description | Bit | Value | Description |
|---|---|---|---|---|
| 0 | Command | 7:0 | 0x27 | Fail-over configuration command. |
| 1 | Buffer Length | 7:0 | 0x4 | Indicates four bytes of the fail-over register (7:4 below). |
| 2 | Return Status | 7:0 | 0x1 | Indicates good status. |
| 3 | Checksum | 7:0 | | Checksum signature. |
| 7:4 | Data Buffer | 7:0 | Fail-over configuration Dwords | Fail over register content.<br>Byte 4 is byte 0 of the configuration register. |



## 10.5.4 Software and Firmware Synchronization

Software and firmware synchronize accesses to shared resources in the 82599 through a semaphore mechanism and a shared configuration register between the host interface of the two ports and firmware. This semaphore enables synchronized accesses to the following shared resources:

- EEPROM
- PHY 0 and PHY 1 registers
- MAC (LAN controller) shared registers (reserved option for future use)

The SWSM.SWESMBI bit and the FWSM.FWSMBI bit are used as a semaphore mechanism between software and firmware. Once software or firmware takes control over these semaphore flags, it can access the SW_FW_SYNC register and claim ownership of the specific resources. The SW_FW_SYNC includes pairs of bits (one owned by software and the other by firmware), while each pair of bits control a different resource. A resource is owned by software or firmware when the respective bit is set. It is illegal to have both bits in a pair set at the same time. Following are the required sequences for gaining and releasing control over the shared resources:

Gaining Control of Shared Resource by Software

- Software checks that the software on the other LAN function does not use the software/firmware semaphore
  - Software polls the SWSM.SMBI bit until it is read as 0b or time expires (recommended expiration is ~10 ms+ expiration time used for the SWSM.SWESMBI).
  - If SWSM.SMBI is found at 0b, the semaphore is taken. Note that following this read cycle hardware auto sets the bit to 1b.
  - If time expired, it is assumed that the software of the other function malfunctions. Software proceeds to the next steps checking SWESMBI for firmware use.
- Software checks that the firmware does not use the software/firmware semaphore and then takes its control
  - Software writes a 1b to the SWSM.SWESMBI bit
  - Software polls the SWSM.SWESMBI bit until it is read as 1b or time expires (recommended expiration is ~3 sec). If time has expired 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)
  - Software reads the firmware and software bit(s) of the requested resource(s) in the SW_FW_SYNC register.
  - If time has expired in the previous steps due to a malfunction firmware, the software should clear the firmware bits in the SW_FW_SYNC register. If time has expired in the previous steps due to malfunction software of the other LAN function, software should clear the software bits in the SW_FW_SYNC register that it does not own.

— If the software and firmware bit(s) of the requested resource(s) in the SW_FW_SYNC register are cleared, it means that these resources are accessible. In this case software sets the software bit(s) of the requested resource(s) in the SW_FW_SYNC register. Then the SW clears the SWSM.SWESMBI and SWSM.SMBI bits (releasing the SW/FW 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 SWSM.SWESMBI and SWSM.SMBI bits and then repeats the entire process after some delay (recommended 5-10 ms). If the resources are not released by software of the other LAN function long enough (recommended expiration time is ~1 sec) software can assume that the other software malfunctioned. In that case software should clear all software flags that it does not own and then repeat the entire process once again.

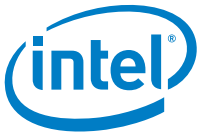
Note that firmware initializes its semaphore flags as part of its initialization flow.

#### Releasing a Shared Resource by Software

- The software takes control over the software/firmware semaphore as previously described for gaining shared resources.
- Software clears the bit(s) of the released resource(s) in the SW_FW_SYNC register.
- Software releases the software/firmware semaphore by clearing the SWSM.SWESMBI and SWSM.SMBI bits
- Software should wait a minimum delay (recommended 5-10 ms) before trying to gain the semaphore again

#### Gaining Control of Shared Resource by Firmware

- Firmware takes control over the software/firmware semaphore (SW_FW_SYNC register)
  - — Firmware writes a 1b to the FWSM.FWSMBI bit
  - — Firmware polls the FWSM.FWSMBI bit until it is read as 1b or time is expired (recommended expiration time is ~10 ms).
  - — If time has expired firmware ignores the FWSM.FWSMBI bit and continues to the next step (assuming software does not function well).
- Firmware takes ownership of the requested resources
  - — Firmware reads the matched software bit(s) to the requested resource(s) in the SW_FW_SYNC register.
  - — If the software bit(s) are cleared (such as software does not own the resource), firmware sets the firmware bit(s) of the requested resource(s). Then firmware clears the FWSM.FWSMBI bit (releasing the software/firmware semaphore) and can use the specific resource(s).
  - — Otherwise (software owns the resource), firmware clears the FWSM.FWSMBI bit and then repeats the previous process after some delay (recommended delay of 5-10 ms). If the resources are not released long enough (~1 sec) firmware accesses by force the requested resources. Firmware also clears the software flags of the requested resources in the SW_FW_SYNC register (assuming software that set those flags malfunctioned).



### 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 FWSM.FWSMBI bit
- Firmware should wait some delay before trying to gain the semaphore once again (recommended 5-10 ms)

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

# 11.0 Electrical / Mechanical Specification

## 11.1 Introduction

This section describes the 82599 DC and AC (timing) electrical characteristics and the 82599 package specification. This includes absolute maximum rating, recommended operating conditions, power sequencing requirements, DC and AC timing specifications. The DC and AC characteristics include generic digital IO specification as well as other specifications of interfaces supported by the 82599.

## 11.2 Operating Conditions

### 11.2.1 Absolute Maximum Ratings

**Table 11-1 Absolute Maximum Ratings**

| Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|
| $T_{case}$ | Case Temperature Under Bias | 0 | 120 | °C |
| $T_{storage}$ | Storage Temperature Range | -65 | 140 | °C |
| Vi | 3.3V I/O input Voltage | Vss-0.5 | 4.0 | V |
| VCC3P3 | 3.3V Periphery Supply Voltage | Vss -0.5 | 4.0 | V |
| VCC1P2 | 1.2V Core/Periphery/Analog Supply Voltage | Vss-0.2 | 1.68V | V |
| ICC3P3 | 3.3V Periphery Supply Current | - | 0.25 | A |
| ICC1P2 | 1.2V Core/Periphery/Analog Supply Current | - | 5.3 | A |

*Note:* Stresses above those listed in the table can cause permanent device damage. These values should not be used as limits for normal device operation. Exposure to absolute maximum rating conditions for an extended period of time can affect device reliability.

### 11.2.2 Recommended Operating Conditions

**Table 11-2 Recommended Operating Conditions**

| Symbol | Parameter | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Ta | Operating Temperature Range Commercial (Ambient; 0 CFS airflow) | 0 | | See Section 13.0 | °C |
| Tj | Junction Temperature | Driven by min Ta | | 123 | °C |
| VCC3P3 | 3.3V Power Supply | 3.14 | 3.3 | 3.46 | V |
| VCC1P2 | 1.2V Power Supply | 1.14 | 1.2 | 1.26 | V |

*Notes:*

1. For normal device operation, adhere to the limits in this table. Sustained operation of a device at conditions exceeding these values, even if they are within the absolute maximum rating limits, can result in permanent device damage or impaired device reliability. Device functionality to stated DC and AC limits is not guaranteed if conditions exceed recommended operating conditions.
2. Recommended operation conditions require accuracy of power supply of ±5% relative to the nominal voltage.
3. External Heat Sink (EHS) is needed.
4. Refer to Section 13.0 for a description of the allowable thermal environment.

## 11.3 Power Delivery

### 11.3.1 Power Supply Specifications

**Table 11-3 VCC3P3 External Power Supply Specifications**

| Title | Description | Min | Max | Units |
|---|---|---|---|---|
| Rise Time | Time from 10% to 90% mark | 0.1 | 100 | ms |
| Monotonicity | Voltage dip allowed in ramp | n/a | 0 | mV |
| Slope | Ramp rate at any given time between 10% and 90%<br>Min 0.8*V(min)/rise time (max)<br>Max 0.8*V(max)/rise time (min) | 24 | 28,800 | V/S |
| Operational Range | Voltage range for normal operating conditions | 3.3 — 5% | 3.3 + 5% | V |
| Ripple | Maximum voltage ripple (peak to peak) | n/a | 70 | mV |
| Overshoot | Maximum overshoot allowed | n/a | 100 | mV |
| Overshoot Settling Time | Maximum overshoot allowed duration.<br>(At that time delta voltage should be lower than 5 mV from steady state voltage) | n/a | 0.05 | ms |

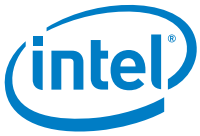


**Table 11-4 VCC1P2 External Power Supply Specification**

| Title | Description | Min | Max | Units |
|---|---|---|---|---|
| Rise Time | Time from 10% to 90% mark | 0.1 | 100 | ms |
| Monotonicity | Voltage dip allowed in ramp | n/a | 0 | mV |
| Slope | Ramp rate at any given time between 10% and 90%<br>Min 0.8*V(min)/rise time (max)<br>Max 0.8*V(max)/rise time (min) | 9.1 | 10000 | V/S |
| Operational Range | Voltage range for normal operating conditions | 1.14 | 1.26 | V |
| Ripple | Maximum voltage ripple (peak to peak) | n/a | 40 | mV |
| Overshoot | Maximum overshoot allowed | n/a | 60 | mV |
| Overshoot Duration | Maximum overshoot allowed duration.<br>(At that time delta voltage should be lower than 5 mV from steady state voltage) | 0.0 | 0.05 | ms |

## 11.3.1.1 Power On/Off Sequence

The following relationships between the rise time of the different power supplies should be maintained at all times when external power supplies are in use to avoid risk of either latch-up or forward-biased internal diodes:

$$T_{3.3} \leq T_{1.2}$$
$$V_{1.2} \leq V_{3.3}$$

At power-on and after 3.3V reaches 90% of its final value, the 1.2V voltage rail is allowed 100 ms to reach it's final operating voltage. Once the 1.2V power supply reaches 80% of it's final value the 3.3V power supply should always be above 80% of it's final value until power down.

For power down, it is recommended to turn off all rails at the same time and allow voltage to decay.

**Table 11-5 Power Sequencing for the 82599**

| Symbol | Parameter | Min | Max | units |
|---|---|---|---|---|
| $T_{3_1}$ | VCC3P3 (3.3V) stable to VCC1P2 (1.2V) stable | 0 | 100 | ms |
| Tm-per, Tm-ppo | 3.3V core to PE_RST_N and MAIN_PWR_OK on | 0 | | ms |
| Tper-m, Tppo-m | PE_RST_Nand MAIN_PWR_OK off before 3.3V core down | 0 | | ms |
| Tlpgw | LAN_PWR_GOOD Minimum Width | 5 | | µs |
| Tlpg-per | LAN_PWR_GOOD High Setup | 100 | | ms |
| Tlpg | LAN_PWR_GOOD High Hold | 40 | 80 | ms |

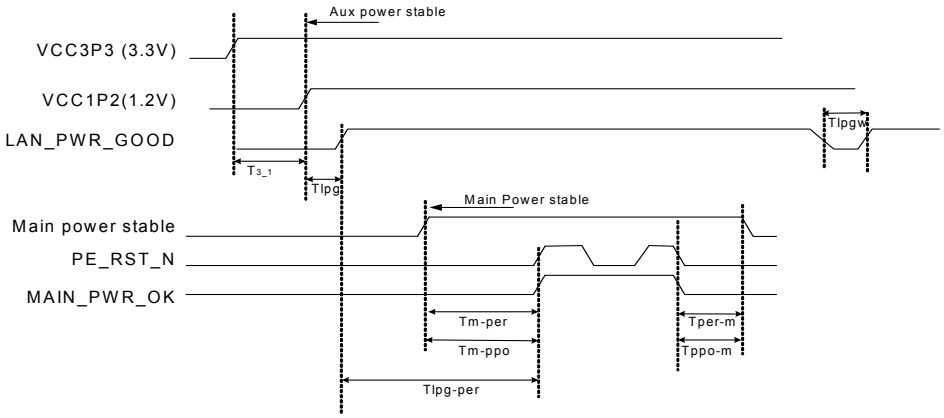



**Figure 11-1 Power and Reset Sequencing**

## 11.3.2 In-Rush Current

**Note:** NICs should limit in-rush current to under 3A from 3.3V power supply and 2.1A from 12V power supply

# 11.4 DC/AC Specification

## 11.4.1 DC Specifications

**Table 11-6 Power Summary for Dual Port Devices (82599ES, 82599EB)**

| Interface | Typical [W] TTT, 60 °C, Vnom, 512 B Security On[1] | | Maximum [W] FFF, 125 °C, Vnom, 512 B Security On[1] | | | |
|---|---|---|---|---|---|---|
| | SP[2] | DP[3] | SP | DP | | |
| KX | 2.3 | 2.7 | 3.1 | 3.5 | | |
| XAUI/KX4 | 3.3 | 4.5 | 4.0 | 5.2 | | |
| SFI Optics | 3.4 | 4.7 | 4.2 | 5.6 | | |
| SFI Twinax | 3.6 | 5.2 | 4.5 | 6.2 | | |
| KR (IEEE) | 3.3 | 4.5 | 4.1 | 5.4 | | |

1. The security engine in 10 GbE mode contributes 200 mW maximum power (180 mW typical power).
2. SP = Single port.
3. DP = Dual port.

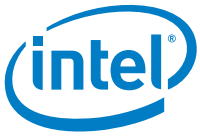


**Table 11-7 Power Summary for Single Port Device (82599EN)**

| Interface | Typical [W] TTT, 60 °C, Vnom, 512 B Security On[1] | | Maximum [W] FFF, 125 °C, Vnom, 512 B Security On[1] | | | |
|---|---|---|---|---|---|---|
| | SP[2] | | SP | | | |
| SFI Optics | 3.4 | | 4.2 | | | |
| SFI Twinax | 3.6 | | 4.5 | | | |

1. The security engine in 10 GbE mode contributes 200 mW maximum power (180 mW typical power).
2. SP = Single port.

## 11.4.1.1 Current Consumption

The 82599 priorities for power reduction are as follows (in descending order):

- TDP: D0 active @ 10 GbE maximum load, fast silicon
- D0 idle: 10 GbE/1 GbE link and no activity
- D0 idle: 10 GbE/1 GbE link with one port disabled
- System sleep: D3cold with wake (link at 10 GbE or 1 GbE)
- System Sleep: D3cold without wake and without manageability
- Other states

The following tables list the targets for device power. The numbers listed apply to device current and power and do not include power losses on external components.

**Table 11-8 D0a — Active Link - Both Ports Active; L0s and L1 Disabled**

| | 1000 Mb/s | | 10 GbE (KX4, CX4, XAUI) | | 10 GbE KR IEEE | | SFI Optics | | 10 GbE Twinax | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Parameter** | **Typ** | **Max** | **Typ** | **Max** | **Typ** | **Max** | **Typ** | **Max** | **Typ** | **Max** |
| **3.3v Idd [mA]** | 42 | 42 | 42 | 42 | 62 | 64 | 126 | 126 | 126 | 126 |
| **1.2v Idd [mA]** | 2170 | 2810 | 3660 | 4380 | 3570 | 4330 | 3590 | 4350 | 4010 | 4790 |
| **Power [mW]** | 2700 | 3500 | 4500 | 5400 | 4500 | 5400 | 4700 | 5600 | 5200 | 6200 |

*Notes:*

1. Typical conditions: typical material TJ = 60 °C, nominal voltages and continuous network traffic at link speed.
2. Maximum conditions: fast material maximum operating temperature (TJ) values, typical voltage values and continuous network traffic at link speed.
3. Maximum power at 110 °C is expected to be ~0.5 W less than the power at 123°C (max TJ).
4. Power numbers are measured with security offload on. Disabling it reduces ~200 mW (max) and 180 mW (Typical).

### Table 11-9 D0a — Active Link - Single Port Active; L0s and L1 Disabled

| | 1000 Mb/s | | 10 GbE (KX4, CX4, XAUI) | | 10 GbE KR IEEE | | SFI Optics | | 10 GbE Twinax | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Parameter** | **Typ** | **Max** | **Typ** | **Max** | **Typ** | **Max** | **Typ** | **Max** | **Typ** | **Max** |
| **3.3v Idd [mA]** | 42 | 42 | 42 | 42 | 53 | 53 | 84 | 84 | 84 | 84 |
| **1.2v Idd [mA]** | 1830 | 2460 | 2600 | 3240 | 2560 | 3260 | 2570 | 3270 | 2780 | 3490 |
| **Power [mW]** | 2300 | 3100 | 3300 | 4000 | 3300 | 4100 | 3400 | 4200 | 3600 | 4500 |

*Notes:*

1. Typical conditions: typical material TJ = 60 °C, nominal voltages and continuous network traffic at link speed.
2. Maximum conditions: fast material maximum operating temperature (TJ) values, typical voltage values and continuous network traffic at link speed.
3. Maximum power at 110 °C is expected to be ~0.5 W less than the power at 123°C (max TJ).
4. Power numbers are measured with security offload on. Disabling it reduces ~200 mW (max) and 180 mW (Typical).

### Table 11-10 D0a — Idle Link - Both Ports Active, L0s and L1 Disabled, No Rx/Tx Traffic

| | 1000 Mb/s | | 10 GbE (KX4, CX4, XAUI) | | 10 GbE KR IEEE | | SFI Optics | | 10 GbE Twinax | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Parameter** | **Typ** | **Max** | **Typ** | **Max** | **Typ** | **Max** | **Typ** | **Max** | **Typ** | **Max** |
| **3.3v Idd [mA]** | 42 | 42 | 42 | 42 | 62 | 64 | 126 | 126 | 126 | 126 |
| **1.2v Idd [mA]** | 2110 | 2740 | 3070 | 3740 | 2930 | 3655 | 2950 | 3675 | 3770 | 4120 |
| **Power [mW]** | 2700 | 3400 | 3800 | 4600 | 3700 | 4600 | 4000 | 4800 | 4500 | 5400 |

*Notes:*

1. Typical conditions: typical material TJ = 60 °C, nominal voltages and no network traffic.
2. Maximum conditions: fast material maximum operating temperature (TJ) values, nominal voltages and no network traffic.

### Table 11-11 Typical D3cold Wake Up Enable - One Port With Wakeup Enabled; Second Port With Wakeup Disabled

| | | | 1000 Mb/s | | 10 GbE (KX4, XAUI, CX4) | |
|---|---|---|---|---|---|---|
| **Parameter** | | | **Typ** | **Max** | **Typ** | **Max** |
| **3.3v Idd [mA]** | | | 31.5 | 31.5 | 26.8 | 31.5 |
| **1.2v Idd [mA]** | | | 1010.1 | 1010.1 | 1291.5 | 1396.5 |
| **Power [mW]** | | | 1316.1 | 1316.1 | 1638.2 | 1779.8 |

*Notes:*

1. In this measurement one port is set to D3 wake-up enabled and one port at D3 no wake up.
2. Typical conditions: typical material TJ = 25, nominal voltages and no network traffic.
3. TJ = 25, nominal voltages and no network traffic.

### Table 11-12 Power Down

| | IDDq (Tj = 123 °C) | | IDDq (Tj = 25 °C) | | D3cold No Wake Up | |
|---|---|---|---|---|---|---|
| **Parameter** | **Typ** | **Max** | **Typ** | **Max** | **Typ** | **Max** |
| **3.3v Idd [mA]** | 2 | 2 | 2 | 2 | 15.8 | 15.8 |
| **1.2v Idd [mA]** | 230 | 670 | 60 | 100 | 720.3 | 805.4 |
| **Power [mW]** | 280 | 810 | 75 | 125 | 916.3 | 1018.4 |

*Notes:*

1. Typical conditions: typical material, nominal voltages and no network traffic.
2. Maximum conditions: fast material, nominal voltages and no network traffic.

## 11.4.1.2 Digital I/O DC Specifications

### Table 11-13 Digital Functional 3.3V I/O DC Electrical Characteristics

| Symbol | Parameter | Conditions | Min | Max | Units | Note |
|---|---|---|---|---|---|---|
| VOH | Output High Voltage | IOH = -8mA; VCC3P3 = Min | 2.4 | | V | |
| VOL | Output Low Voltage | IOL = 8mA; VCC3P3=Min | | 0.4 | V | |
| $VOH_{led}$ | LED Output High Voltage | IOL = 12mA; VCC3P3 = Min | 2.4 | | V | |
| $VOL_{led}$ | LED Output Low Voltage | IOL = 12mA; VCC3P3 = Min | | 0.4 | V | |
| VIH | Input High Voltage | | 2.0 | VCC3P3 + 0.3 | V | |
| VIL | Input Low Voltage | | -0.3 | 0.8 | V | |
| Iil | Input Current | VCC3P3 = Max; VIn = 3.6V/GND | | 15 | µA | |
| PU | Internal pull-up | | 27 | 34 | KΩ | |
| Cin | Pin capacitance | | | 7 | pF | [2] |

*Notes:*

1. Table 11-13 applies to PE_RST_N, LED0[3:0], LED1[3:0], LAN_PWR_GOOD, MAIN_PWR_OK, JTCK, JTDI, JTDO, JTMS, SDP0[7:0], SDP1[7:0], FLSH_SI, FLSH_SO, FLSH_SCK, FLSH_CE_N, EE_DI, EE_DO, EE_SK, EE_CS_N, MDIO0, MDC1 and MDIO1, LAN0_DIS_N, LAN1_DIS_N, AUX_PWR, OSC_SEL.
2. Characterized not tested.

### 11.4.1.3 Open Drain I/O DC Specification

**Table 11-14 Open Drain I/O DC Characteristics**

| Symbol | Parameter | Condition | Min | Max | Units | Note |
|---|---|---|---|---|---|---|
| Vih | Input High Voltage | | VCC3P3 * 0.7 | VCC3P3 + 0.5 | V | |
| Vil | Input Low Voltage | | -0.3 | VCC3P3 * 0.3 | V | |
| Ileakage | Output Leakage Current | 0 ≤ Vin ≤ VCC3P3 max | -10 | 10 | µA | [2] |
| Vol | Output Low Voltage | @ Ipullup = 4 mA | | 0.4 | V | [5] |
| Ipullup | Current Sink | Vol = 0.4V | | 4 | mA | [4] |
| Cin | Input Pin Capacitance | | | 7 | pF | [3] |
| Ioffsmb | Input leakage Current | VCC3P3 off or floating | -10 | 10 | µA | [2] |

*Notes:*

1. Table 11-14 applies to SMBD, SMBCLK, SMBALRT _N, PE_WAKE_N, SCL0, SDA0, SCL1 and SDA1.
2. Device must meet this specification whether powered or unpowered.
3. Characterized, not tested.
4. The IPULLUP max specification is determined primarily by the need to operate at a certain frequency with a certain capacitive load.
5. OD no high output drive. VOL max=0.4V at 8mA, VOL max=0.2V at 0.1mA.

The buffer specification meets the SMBus specification requirements defined at: www.smbus.org.

### 11.4.1.4 NC-SI I/O DC Specification

**Table 11-15 NC-SI I/O DC Characteristics**

| Parameter | Symbol | Conditions | Min. | Typ. | Max | Units |
|---|---|---|---|---|---|---|
| Bus High Reference | Vref[1] | | 3.0 | 3.3 | 3.46 | V |
| Signal Voltage Range | Vabs | | -0.300 | | 3.765 | V |
| Input Low Voltage | Vil | | | | 0.8 | V |
| Input High Voltage | Vih | | 2.0 | | | V |
| Output Low Voltage | Vol | Iol = 4mA, Vref= $Vref_{min}$ | 0 | | 0.4 | V |
| Output High Voltage | Voh | Iol = -4mA, Vref= $Vref_{min}$ | 2.4 | | Vref | V |
| Input High Current | Iih | Vin = 3.6V, Vref = 3.6V | 0 | | 200 | µA |
| Input Low Current | Iil | Vin = 0V, $Vref_{min}$ to $Vref_{max}$ | -20 | | 0 | µA |

### Table 11-15 NC-SI I/O DC Characteristics (Continued)

| Parameter | Symbol | Conditions | Min. | Typ. | Max | Units |
|---|---|---|---|---|---|---|
| Clock Midpoint Reference Level | Vckm | | 1.4 | | | V |
| Leakage Current for Output Signals in High-Impedance State | Iz | $0 \le Vin \le Vih_{max}$ @Vref = $Vref_{max}$ | -20 | | 20 | µA |

*Notes:*

1. Vref = Bus high reference level. This parameter replaces the term 'supply voltage' since actual devices may have internal mechanisms that determine the operating reference for the sideband interface that are different from the devices overall power supply inputs. Vref is a reference point that is used for measuring parameters such as overshoot and undershoot and for determining limits on signal levels that are generated by a device. In order to facilitate system implementations, a device must provide a mechanism (e.g. a power supply pin, internal programmable reference, or reference level pin) to allow Vref to be set to within 20 mV of any point in the specified Vref range. This is to enable a system integrator to establish an interoperable Vref level for devices on the sideband interface. Although the NC-SI spec define the Vrefmax up to 3.6V, the 82599 supports the Vrefmax up to 3.46V (3.3V +5%).
2. Table 11-15 applies to NCSI_CLK_IN, NCSI_CRS_DV, NCSI_RXD[0:0], NCSI_TX_EN and NCSI_TXD[1:0].
3. Please refer also to the *Network Controller Sideband Interface (NC-SI) Specification for more details*.

## 11.4.2 Digital I/F AC Specifications

### 11.4.2.1 Digital I/O AC Specifications

### Table 11-16 Digital Functional 3.3V I/O AC Electrical Characteristics

| Parameters | Description | Min | Max | Condition | Note |
|---|---|---|---|---|---|
| **$F_{max}$** | Maximum Operating Frequency | | 50 MHz | Cload 25 pF | [2] |
| **Tor** | Output Rise Time | 1 ns | 5 ns | Cload 25 pF | |
| **Tof** | Output Fall Time | 1 ns | 5 ns | Cload 25 pF | |
| **Todr** | Core to Output Rise Delay Time | 1 ns | 7 ns | Cload 25 pF | [2] |
| **Todf** | Core to Output Fall Delay Time | 1 ns | 7 ns | Cload 25 pF | [2] |
| **Tidr** | Input to Core Rise Delay Time | 0. 2ns | 1.3 ns | Internal Load 200 pF | [2] |
| **Tidf** | Input to Core Fall Delay time | 0.2 ns | 1.3 ns | Internal Load 200 pF | [2] |
| **Tir** | Internal Core Rise Time | 0.03 ns | 0.1 ns | Internal Load 200 pF | [1],[2] |

## Table 11-16 Digital Functional 3.3V I/O AC Electrical Characteristics (Continued)

| Parameters | Description | Min | Max | Condition | Note |
|---|---|---|---|---|---|
| Tif | Internal Core Fall Time | 0.03 ns | 0.1 ns | Internal Load 200 pF | [1],[2] |

*Notes:*

1. The input delay test conditions: Maximum input level = VIN = 2.7V; Input rise/fall time (0.2VIN to 0.8VIN) = 1 ns (Slew Rate ~ 1.5 ns).
2. Characterized but not tested.
3. Table 11-16 applies to PE_RST_N, LED0[3:0], LED1[3:0], LAN_PWR_GOOD, MAIN_PWR_OK, JTCK, JTDI, JTDO, JTMS, SDP0[7:0], SDP1[7:0], FLSH_SI, FLSH_SO, FLSH_SCK, FLSH_CE_N, EE_DI, EE_DO, EE_SK, EE_CS_N, MDIO0, MDC1 and MDIO1.
4. Table 11-16 applies to PE_RST_N, LED0[3:0], LED1[3:0], LAN_PWR_GOOD, MAIN_PWR_OK, JTCK, JTDI, JTDO, JTMS, SDP0[7:0], SDP1[7:0], FLSH_SI, FLSH_SO, FLSH_SCK, FLSH_CE_N, EE_DI, EE_DO, EE_SK, EE_CS_N, MDIO0, MDC1 and MDIO1, LAN0_DIS_N, LAN1_DIS_N, AUX_PWR, OSC_SEL.
5.

## Table 11-17 Digital Test Port 3.3V I/O AC Electrical Characteristics

| Parameters | Description | Min | Max | Condition | Note |
|---|---|---|---|---|---|
| $F_{max}$ | Maximum Operating Frequency | | 312.5 MHz | Cload 16 pF | [2] |
| Tor | Output Rise Time | 0.2 ns | 1 ns | Cload 16 pF | |
| Tof | Output Fall Time | 0.2 ns | 1 ns | Cload 16 pF | |
| Todr | Core to Output Rise Delay Time | 0.2 ns | 2 ns | Cload 16 pF | [2] |
| Todf | Core to Output Fall Delay Time | 0.2 ns | 2 ns | Cload 16 pF | [2] |
| Tidr | Input to Core Rise Delay Time | 0.2 ns | 1.3 ns | Internal Load 200 pF | [2] |
| Tidf | Input to Core Fall Delay Time | 0.2 ns | 1.3 ns | Internal Load 200 pF | [2] |
| Tir | Internal Core Rise Time | 0.03 ns | 0.1 ns | Internal Load 200 pF | [1],[2] |
| Tif | Internal Core Fall Time | 0.03 ns | 0.1 ns | Internal Load 200 pF | [1],[2] |

*Notes:*

1. The input delay test conditions: Maximum input level = VIN = 2.7V; Input rise/fall time (0.2VIN to 0.8VIN) = 1 ns (Slew Rate ~ 1.5 ns).
2. Characterized but not tested.
3. Table 11-17 applies to Digital Test Pins and Pins used for Scan out during Scan operation.

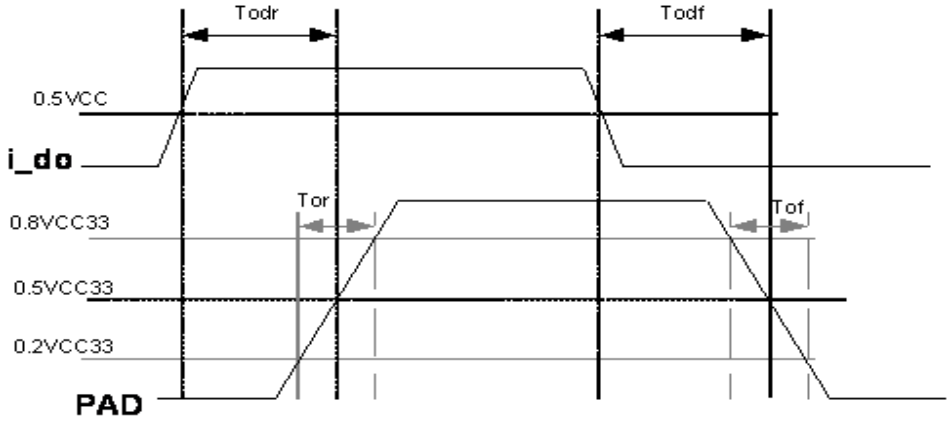



**Figure 11-2 Digital 3.3V I/O Output Timing Diagram**

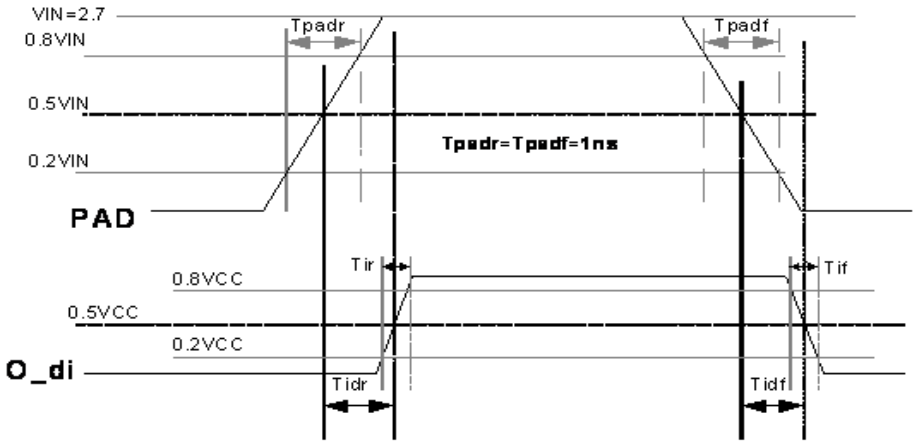



**Figure 11-3 Digital 3.3V I/O Input Timing Diagram**

## 11.4.2.2 SMBus and I²C AC Specifications

The 82599 meets the SMBus AC specification as defined in SMBus specification version 2, section 3.1.1 (http://www.smbus.org/specs/) and the I²C specification.

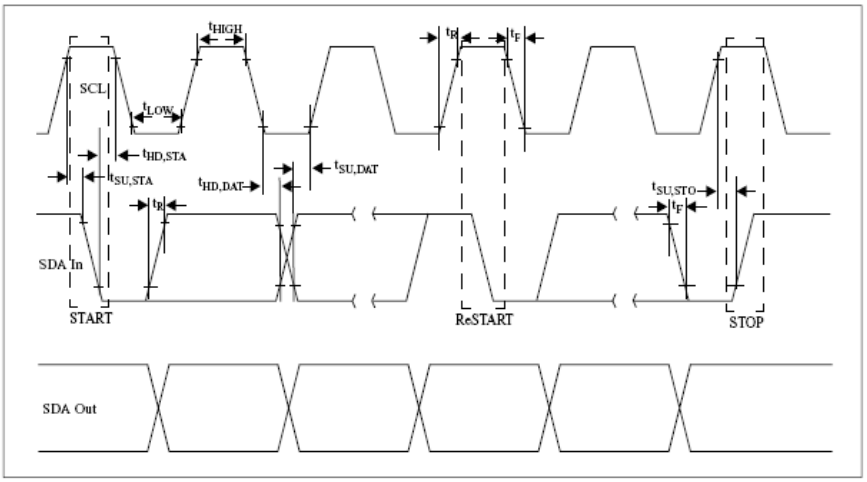
The 82599 also supports a 400 KHz SMBus (as a slave) and meets the specifications listed in the following table:

## Table 11-18 Support for 400 KHz SMBus

| Symbol | Parameter | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| $F_{SMB}$ | SMBus Frequency | 10 | | 400 | KHz |
| $T_{BUF}$ | Time Bus Free Before New Transmission Can Start (Between Stop and Start) | 20 | | | µs |
| $T_{HD,STA}$ | Hold Time After Start Condition. After This Period, the First Clock is Generated. | 0.6 | | | µs |
| $T_{SU,STA}$ | Start Condition Setup Time | 0.6 | | | µs |
| $T_{SU,STO}$ | Stop Condition Setup Time | 0.6 | | | µs |
| $T_{HD,DAT}$ | Data in Hold Time | 0 | | | µs |
| $T_{SU,DAT}$ | Data in Setup Time | 0.1 | | | µs |
| $T_{LOW}$ | SMBClk Low Time | 1.3 | | | µs |
| $T_{HIGH}$ | SMBClk High Time | 0.6 | | | µs |

*Notes:*

1. Table 11-18 applies to SMBD, SMBCLK, SCL0, SDA0, SCL1 and SDA1.




## Figure 11-4 SMBus I/F Timing Diagram

### 11.4.2.3 Flash AC Specification

The 82599 is designed to support a serial Flash. Applicable over recommended operating range from Ta = 0 °C to +70 °C, VCC3P3 = 3.3V, Cload = 16 pF (unless otherwise noted). For Flash I/F timing specifications, see Table 11-19 and Figure 11-5.

**Table 11-19 Flash I/F Timing Parameters**

| Symbol | Parameter | Min | Typ | Max | Units | Note |
|---|---|---|---|---|---|---|
| $t_{SCK}$ | FLSH_SCK Clock Frequency | 0 | 12.5 | 15 | MHz | [2] |
| $t_{RI}$ | FLSH_SO Rise Time | | 2.5 | 20 | ns | |
| $t_{FI}$ | FLSH_SO Fall Time | | 2.5 | 20 | ns | |
| $t_{WH}$ | FLSH_SCK High Time | 20 | 50 | | ns | [1] |
| $t_{WL}$ | FLSH_SCK Low Time | 20 | 50 | | ns | [1] |
| $t_{CS}$ | FLSH_CE_N High Time | 25 | | | ns | |
| $t_{CSS}$ | FLSH_CE_N Setup Time | 25 | | | ns | |
| $t_{CSH}$ | FLSH_CE_N Hold Time | 25 | | | ns | |
| $t_{SU}$ | Data-in Setup Time | 5 | | | ns | |
| $t_{H}$ | Data-in Hold Time | 5 | | | ns | |
| $t_{V}$ | Output Valid | | | 20 | ns | |
| $t_{HO}$ | Output Hold Time | 0 | | | ns | |
| $t_{DIS}$ | Output Disable Time | | | 100 | ns | |
| $t_{EC}$ | Erase Cycle Time per Sector | | | 1.1 | Seconds | |
| $t_{BPC}$ | Byte Program Cycle Time | | 60 | 100 | µs | |

*Notes:*

1. 50% duty cycle.
2. Clock is either 25 MHz or 26.04 MHz divided by 2.
3. Table 11-19 applies to FLSH_SI, FLSH_SO, FLSH_SCK and FLSH_CE_N.

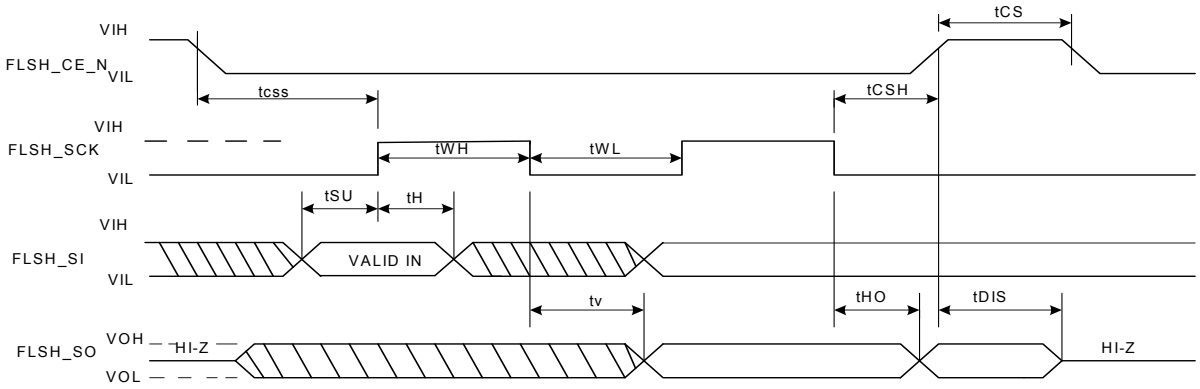



**Figure 11-5 Flash I/F Timing Diagram**

## 11.4.2.4 EEPROM AC Specification

The 82599 is designed to support a standard serial EEPROM. Applicable over recommended operating range from Ta = -0 °C to +70 °C, VCC3P3 = 3.3V, Cload = 16pF (unless otherwise noted). For EEPROM I/F timing specifications, see Table 11-20 and Figure 11-6.

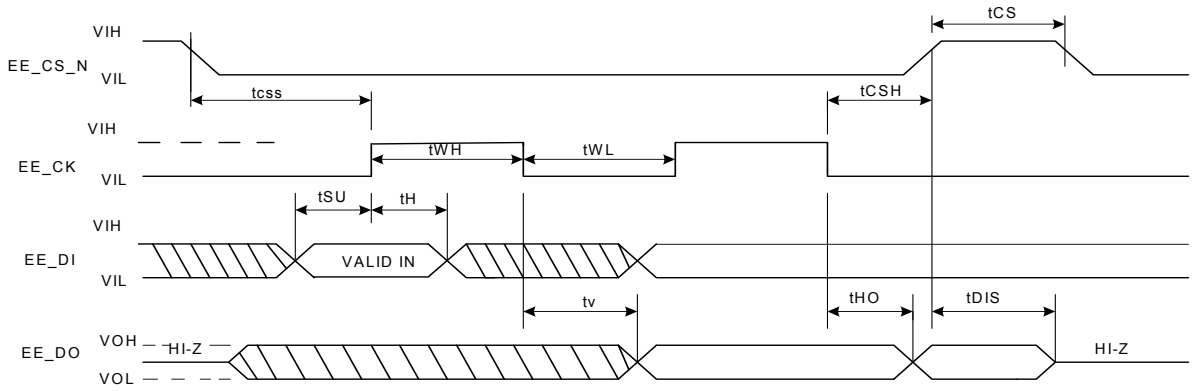
**Table 11-20 EEPROM I/F Timing Parameters**

| Symbol | Parameter | Min | Typ | Max | Units | Note |
|---|---|---|---|---|---|---|
| $t_{SCK}$ | EE_CK Clock Frequency | | 2 | 2.1 | MHz | |
| $t_{RI}$ | EE_DO Rise Time | | 2.5 ns | 2 µs | ns / µs | |
| $t_{FI}$ | EE_DO Fall Time | | 2.5 ns | 2 µs | ns / µs | |
| $t_{WH}$ | EE_CK High Time | 200 | 250 | | ns | |
| $t_{WL}$ | EE_CK Low Time | 200 | 250 | | ns | |
| $t_{CS}$ | EE_CS_N High Time | 250 | | | ns | |
| $t_{CSS}$ | EE_CS_N Setup Time | 250 | | | ns | |
| $t_{CSH}$ | EE_CS_N Hold Time | 250 | | | ns | |
| $t_{SU}$ | Data-in Setup Time | 50 | | | ns | |
| $t_{H}$ | Data-in Hold Time | 50 | | | ns | |
| $t_{V}$ | Output Valid | 0 | | 200 | ns | |
| $t_{HO}$ | Output Hold Time | 0 | | | ns | |

### Table 11-20 EEPROM I/F Timing Parameters (Continued)

| Symbol | Parameter | Min | Typ | Max | Units | Note |
|---|---|---|---|---|---|---|
| $t_{DIS}$ | Output Disable Time | | | 250 | ns | |
| $t_{WC}$ | Write Cycle Time | | | 10 | ms | |

*Notes:*
1. Table 11-20 applies to EE_DI, EE_DO, EE_SK and EE_CS_N.




### Figure 11-6 EEPROM I/F Timing Diagram

## 11.4.2.5 NC-SI AC Specifications

The 82599 supports the NC-SI standard as defined in the DMTF Network Controller Sideband Interface (NC_SI) specification. The NC-SI timing specifications can be found in Table 11-21 and Figure 11-7.

### Table 11-21 NC-SI Interface AC Specifications

| Parameter | Symbol | Conditions | Min. | Typ. | Max. | Units | Notes |
|---|---|---|---|---|---|---|---|
| REF_CLK Frequency | | | | 50 | 50+100 ppm | MHz | |
| REF_CLK Duty Cycle | | | 35 | | 65 | % | 2 |
| Clock-to-Out (10pF<=cload<=50 pF) | Tco | | 2.5 | | 12.5 | ns | 1,3 |
| Skew Between Clocks | Tskew | | | | 1.5 | ns | |
| TXD[1:0], TX_EN, RXD[1:0], CRS_DV, RX_ER Data Setup to REF_CLK Rising Edge | Tsu | | 3 | | | ns | 3 |

## Table 11-21 NC-SI Interface AC Specifications (Continued)

| Parameter | Symbol | Conditions | Min. | Typ. | Max. | Units | Notes |
|---|---|---|---|---|---|---|---|
| TXD[1:0], TX_EN, RXD[1:0], CRS_DV, RX_ER Data Hold From REF_CLK Rising Edge | Thd | | 1 | | | ns | 3 |
| Signal Rise/Fall Time | Tr/Tf | | 1 | | 6 | ns | 4 |
| REF_CLK Rise/Fall Time | Tckr/Tckf | | 0.5 | | 3.5 | ns | |
| Interface Power-Up High Impedance Interval | Tpwrz | | 2 | | | µs | |
| Power Up Transient Interval (recommendation) | Tpwrt | | | | 100 | ns | |
| Power Up Transient Level (recommendation) | Vpwrt | | -200 | | 200 | mV | |
| Interface Power-Up Output Enable Interval | Tpwre | | | | 10 | ms | |
| EXT_CLK Startup Interval | Tclkstrt | | | | 100 | ms | |

*Notes:*

1. This timing relates to the output pins timing while Tsu and Thd relate to timing at the input pins.
2. REF_CLK duty cycle measurements are made from Vckm to Vckm. Clock skew Tskew is measured from Vckm to Vckm of two NC-SI devices and represents maximum clock skew between any two devices in the system.
3. All timing measurements are made between Vckm and Vm. All output timing parameters are measured with a capacitive load between 10 pF and 50 pF.
4. Rise and fall time are measured between points that cross 10% and 90% of Vref (see Table 11-15). The middle points (50% of Vref) are marked as Vckm and Vm for clock and data, respectively.

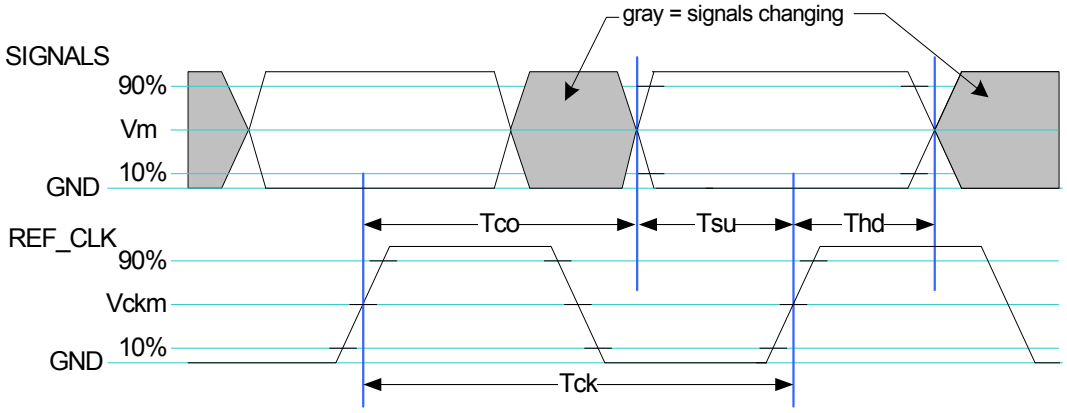



**Figure 11-7 NC-SI AC Timing Diagram**

### 11.4.2.6 JTAG AC Specification

The 82599 is designed to support the IEEE 1149.1 standard. The following timing specifications are applicable over recommended operating range from Ta = 0 °C to +70 °C, VCC3P3 = 3.3V, Cload =
16 pF (unless otherwise noted). For JTAG I/F timing specifications, see Table 11-22 and Figure 11-8.

**Table 11-22 JTAG I/F Timing Parameters**

| Symbol | Parameter | Min | Typ | Max | Units | Note |
|---|---|---|---|---|---|---|
| $t_{JCLK}$ | JTCK Clock Frequency | | | 10 | MHz | |
| $t_{JH}$ | JTMS and JTDI Hold Time | 10 | | | ns | |
| $t_{JSU}$ | JTMS and JTDI Setup Time | 10 | | | ns | |
| $t_{JPR}$ | JTDO Propagation Delay | | | 15 | ns | |

*Notes:*
1. Table 11-22 applies to JTCK, JTMS, JTDI and JTDO.
2. Timing measured relative to JTCK reference voltage of VCC3P3/2.

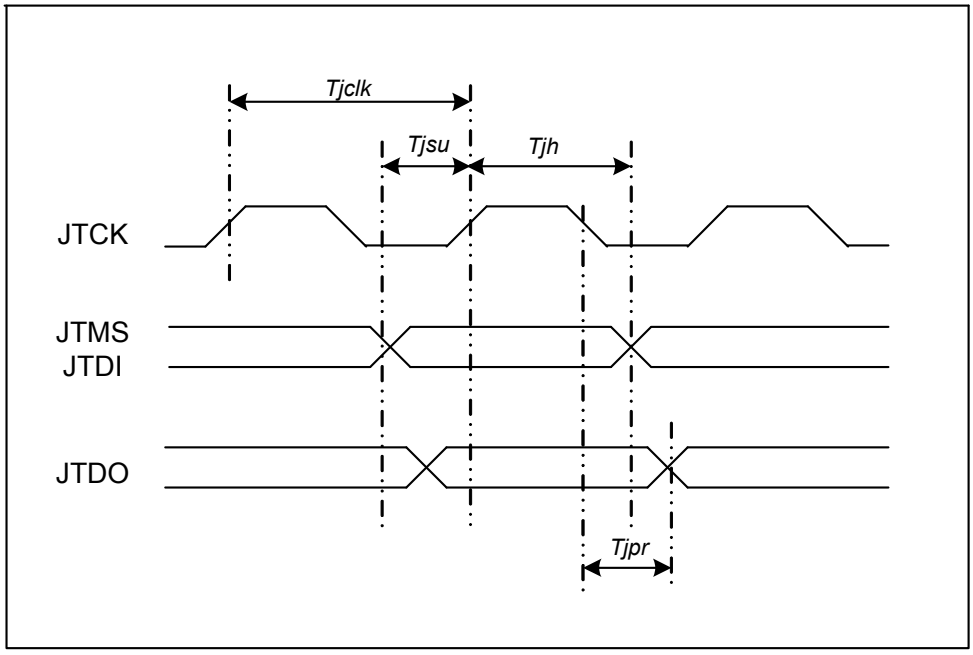



**Figure 11-8 JTAG AC Timing Diagram**

## 11.4.2.7 MDIO AC Specification

The 82599 is designed to support the MDIO specifications defined in IEEE 802.3 clause 22. The following timing specifications are applicable over recommended operating range from Ta = 0 °C to +70 °C, VCC3P3 = 3.3V, Cload = 16 pF (unless otherwise noted). For MDIO I/F timing specifications, see Table 11-23, Figure 11-9 and Figure 11-10.

**Table 11-23 MDIO I/F Timing Parameters**

| Symbol | Parameter | Min | Typ | Max | Units | Note |
|---|---|---|---|---|---|---|
| $t_{MCLK}$ | MDC Clock Frequency | 2.4 | | 24 | MHz | |
| $t_{MH}$ | MDIO Hold Time | 10 | | | ns | |
| $t_{MSU}$ | MDIO Setup Time | 10 | | | ns | |
| $t_{MPR}$ | MDIO Propagation Delay | 10 | | 30 | ns | |

*Notes:*

1. Table 11-23 applies to MDIO0, MDC0, MDIO1 and MDC1.
2. Timing measured relative to MDC reference voltage of 2.0V (Vih).

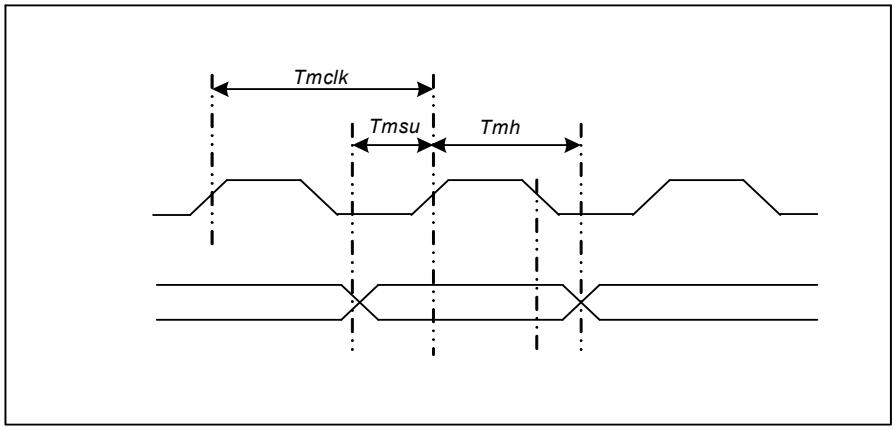



**Figure 11-9 MDIO Input AC Timing Diagram**

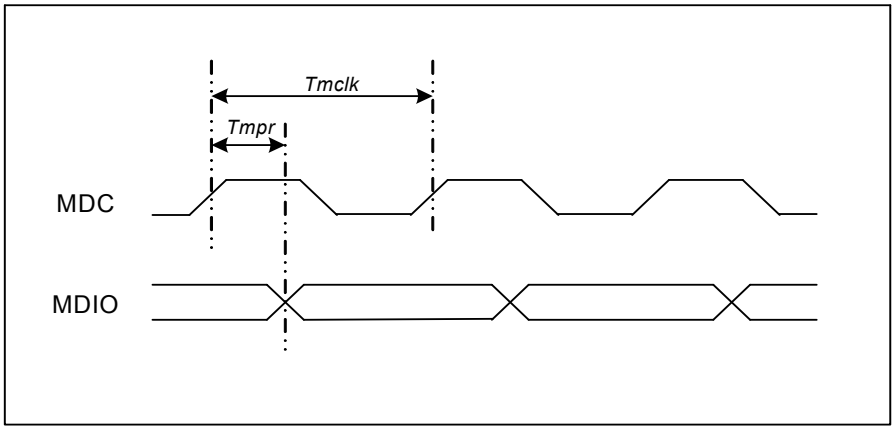



**Figure 11-10 MDIO Output AC Timing Diagram**

## 11.4.2.8 Reset Signals

For power-on indication, the 82599 can either use an internal power-on circuit, which monitors the 1.2V power supply, or external reset using the LAN_PWR_GOOD pin. The POR_BYPASS pin defines the reset source (when high, the device uses the LAN_PWR_GOOD pad as power-on indication).

The timing between the power-up sequence and the different reset signals when using the internal power indication is described in Section 11.3.1.1.

The BYPASS mode is described in Section 11.4.2.8.1.

A schematic of the power-on logic can be found in Figure 11-11:

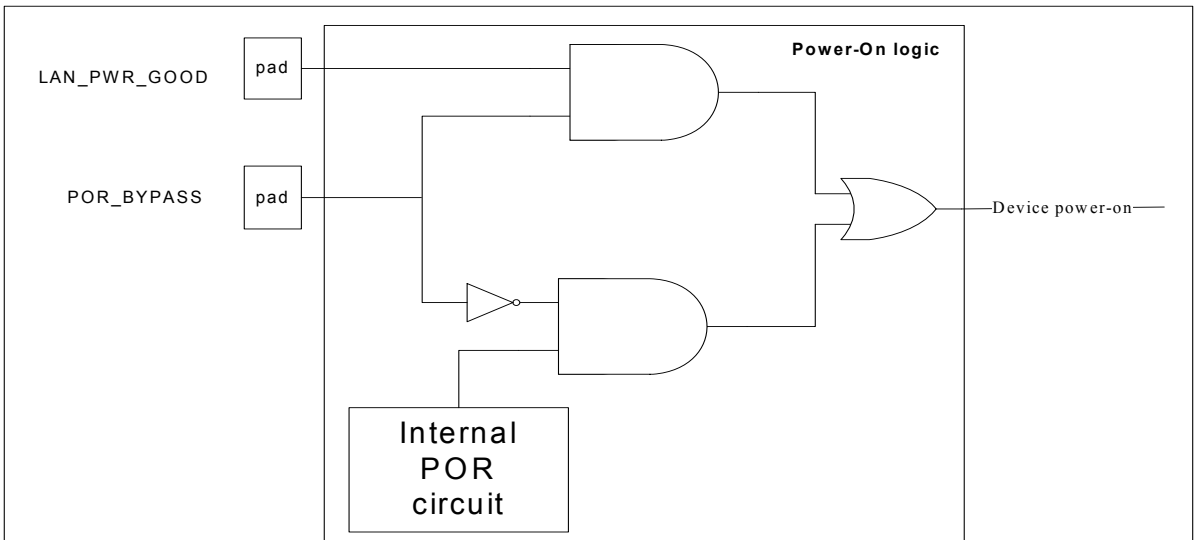



**Figure 11-11 Power-On Reset Logic**

#### 11.4.2.8.1 Power-On Reset BYPASS

When asserting the POR_BYPASS pad, the 82599 uses the LAN_PWR_GOOD pin as power-on indication. Otherwise, the 82599 uses an internal power on detection circuit in order to generate the internal power on reset signal.

Table 11-24 lists the timing for the external power-on signal.

**Table 11-24 External Reset Specification**

| Symbol | Title | Description | Min | Max | Units |
|---|---|---|---|---|---|
| Tlpgw | LAN_PWR_GOOD Minimum Width | Minimum width for LAN_PWR_GOOD. | 5 | N/A | µs |
| Tlpg | LAN_PWR_GOOD High Hold | Hold time following power-up (power supplies in acceptable operating range). | 40 | 80 | ms |

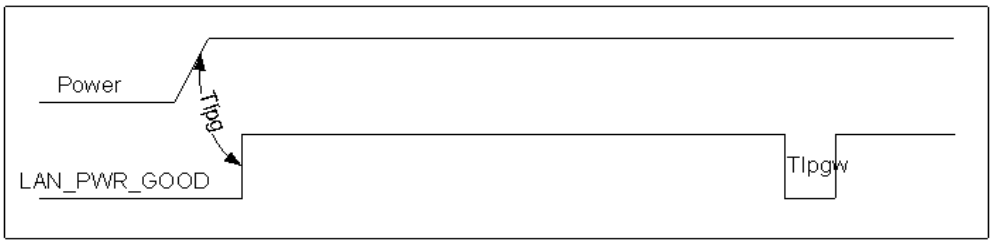



**Figure 11-12 External Reset Timing Diagram**

## 11.4.3 PCIe Interface AC/DC Specification

The 82599 PCIe interface supports the electrical specifications defined in:

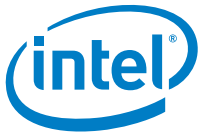
- PCI Express* 2.0 Card Electromechanical Specification.
- PCI Express* 2.0 Base Specification, Chapter 4.

**Note:** Reference clock specifications are detailed in both the base and CEM specification. Please consult both specifications to understand the full set of requirements. Sections 4.3.7 (Base specification) and 2.1.3 (CEM specification) in particular.

## 11.4.4 Network (MAUI) Interface AC/DC Specification

### 11.4.4.1 KR Interface AC/DC Specification

The 82599 MAUI interface supports the 10GBASE-KR electrical specification defined in IEEE802.3ap clause 72.



#### 11.4.4.2 SFP+ Interface AC/DC Specification

The 82599 MAUI interface supports the SFI electrical specification defined in the SFP+ MSA (SFF Committee SFF-8431).

#### 11.4.4.3 KX4 Interface AC/DC Specification

The 82599 MAUI interface supports the 10GBASE-KX4 electrical specification defined in IEEE802.3ap clause 71.

#### 11.4.4.4 BX4 Interface AC/DC Specification

The 82599 MAUI interface supports the 10GBASE-BX4 electrical specification defined in PICMG 3.1.

#### 11.4.4.5 CX4 Interface AC/DC Specification

The 82599 MAUI interface supports the 10GBASE-CX4 electrical specification defined in IEEE802.3ak clause 54.

#### 11.4.4.6 XAUI Interface AC/DC Specification

The 82599 MAUI interface supports the 10G XAUI electrical specification defined in IEEE802.3ae clause 47.

#### 11.4.4.7 KX Interface AC/DC Specification

The 82599 MAUI interface supports the 1000BASE-KX electrical specification defined in IEEE802.3ap clause 70.

#### 11.4.4.8 BX Interface AC/DC Specification

The 82599 MAUI interface supports the 1000BASE-BX electrical specification defined in PICMG® 3.1 specification.

## 11.4.5 SerDes Crystal/Reference Clock Specification

The 82599 SerDes clock can be supplied either by connecting an external differential oscillator of 25 MHz or an external 25 MHz crystal. SerDes clock frequency is set by the OSC_FREQ_SE pin and crystal vs. oscillator are selected by the OSC_SEL pin.

The figures below show connection options to the REFCLKIN_p/_n pins by using a crystal or by using an external oscillator (PECL or CML). These schemes are not part of The 82599 specifications but rather examples that meet the required specification.

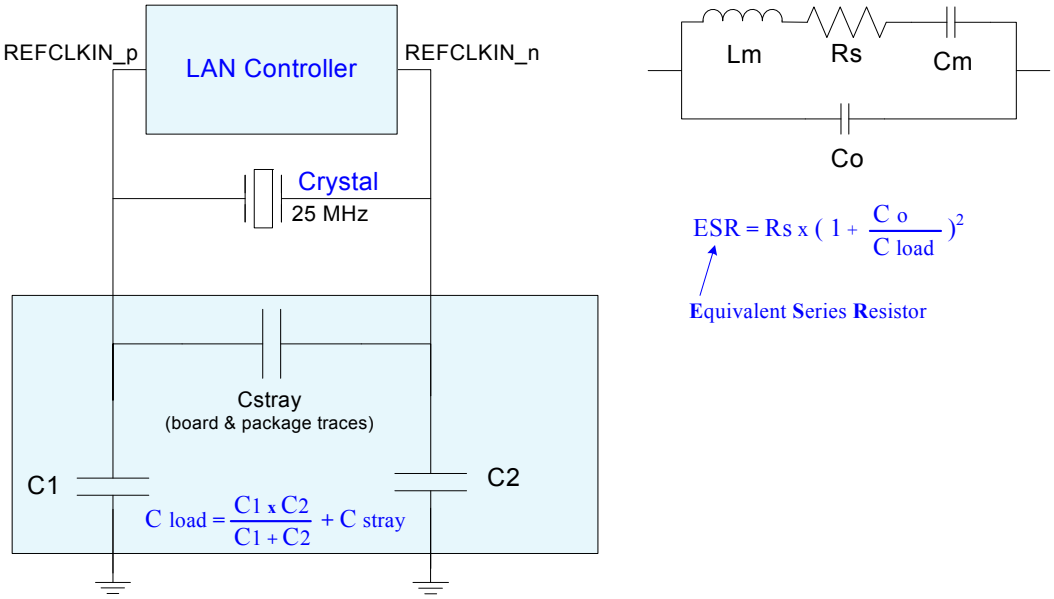
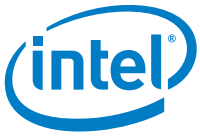



**Figure 11-13 Crystal Connectivity**



### 11.4.5.1 SerDes Crystal Specification

**Table 11-25 SerDes Crystal Specifications**

| Parameter Name | Symbol | Recommended Value | Conditions |
|---|---|---|---|
| Frequency | $f_o$ | 25.000 [MHz] | @25 [°C] |
| Vibration Mode | | Fundamental | |
| Cut | | AT | |
| Operating /Calibration Mode | | Parallel | |
| Frequency Tolerance @25 °C | $Df/f_o$ @25 °C | ±30 [ppm] | @25 [°C] |
| Temperature Tolerance | $Df/f_o$ | ±30 [ppm] | |
| Operating Temperature | $T_{opr}$ | -20 to +70 [°C] | |
| Non Operating Temperature Range | $T_{opr}$ | -40 to +90 [°C] | |
| Equivalent Series Resistance (ESR) | ESR | 50 [Ω] maximum | @25 [MHz] |
| Load Capacitance | $C_{load}$ | 20 [pF] | |
| Shunt Capacitance | $C_o$ | 7 [pF] maximum | |
| Pullability From Nominal Load Capacitance | $Df/C_{load}$ | 15 [ppm/pF] maximum | |
| Max Drive Level | $D_L$ | 750 [µW] | |
| Insulation Resistance | IR | 500 [MW] minimum | @ 100V DC |
| Aging | $Df/f_o$ | ±5 [ppm/year] | |
| External Capacitors | $C_1$, $C_2$ | 20 [pF] | |
| Board Resistance | $R_s$ | 0.1 [W] | |

## 11.4.5.2 SerDes Reference Clock Specification

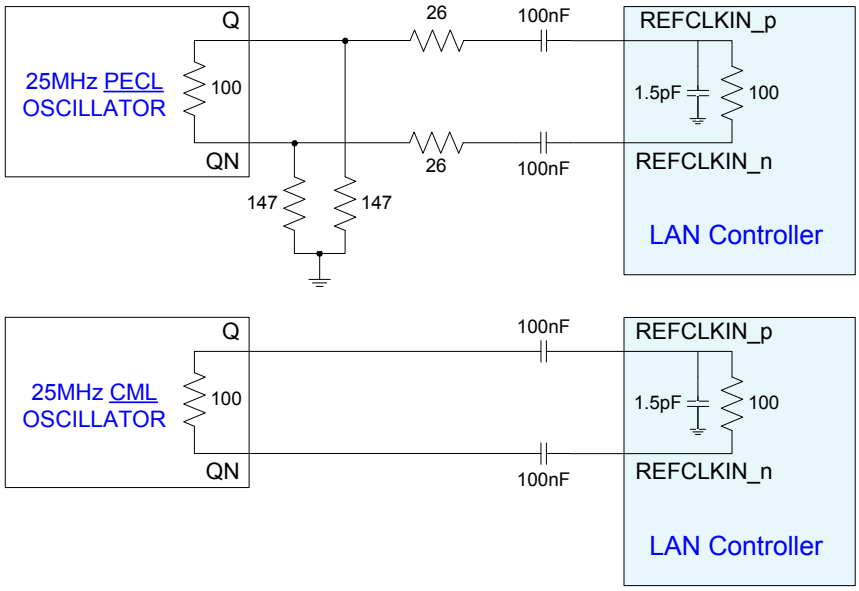



**Figure 11-14 Example Diagram for External PECL or CML Oscillator Connectivity**

### Table 11-26 Input Reference Clock Electrical Characteristics

| Sym | Parameter | Min | Typ | Max | Unit | Comments |
|---|---|---|---|---|---|---|
| f | Frequency | | 25 | | MHz | |
| Δf | Frequency Variation | -100 | | +100 | ppm | |
| DC | Duty Cycle | 40 | | 60 | % | |
| Tr | Rise Time (20% - 80%) | 300 | | 1000 | ps | |
| Tf | Fall Time (20% - 80%) | 300 | | 1000 | ps | |
| AMP | Differential Peak-to-Peak Amplitude | 0.6 | | 1.25 | V | |
| R | Differential Termination Resistance | | 100 | | Ω | |
| C | AC Coupling | | 100 | | nF | |
| Cin | Input Capacitance | | 1.5 | | pF | |

**Table 11-26 Input Reference Clock Electrical Characteristics (Continued)**

| Sym | Parameter | Min | Typ | Max | Unit | Comments |
|---|---|---|---|---|---|---|
| DJ P2P | Deterministic Peak-to-Peak Jitter | | | 10 | ps | |
| p-noise | Phase Noise (high-speed serial - KR and SFI) | | -145 | | dBc/Hz | See Figure 11-16 for phase noise graph |
| p-noise | Phase Noise (non-high speed serial) | | -136 | | dBc/Hz | See Figure 11-17 for phase noise graph |

**Note:** Intel recommends designing to the high-speed serial p-noise specification allowing for maximum flexibility with link configuration options.

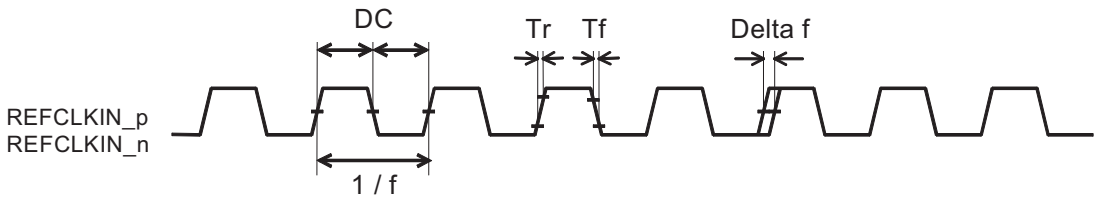



**Figure 11-15 External Clock Reference Timing**

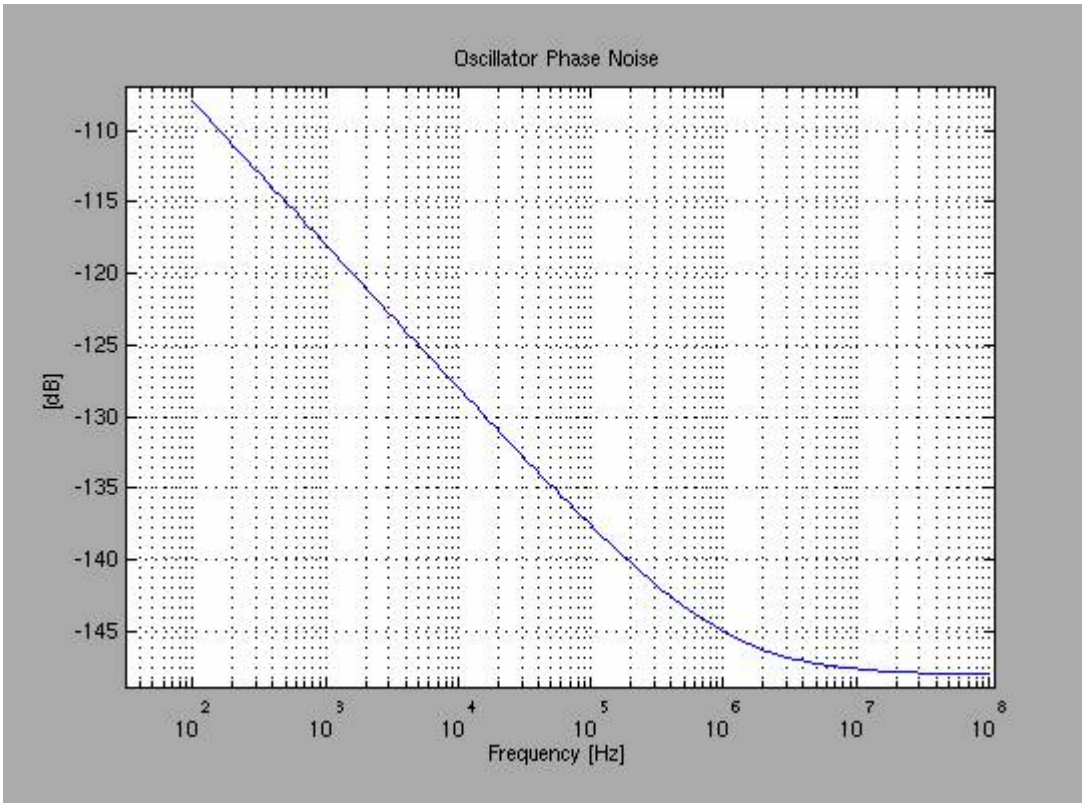



**Figure 11-16 Maximum External Oscillator Phase Noise as a Function of Frequency (High-Speed Serial)**

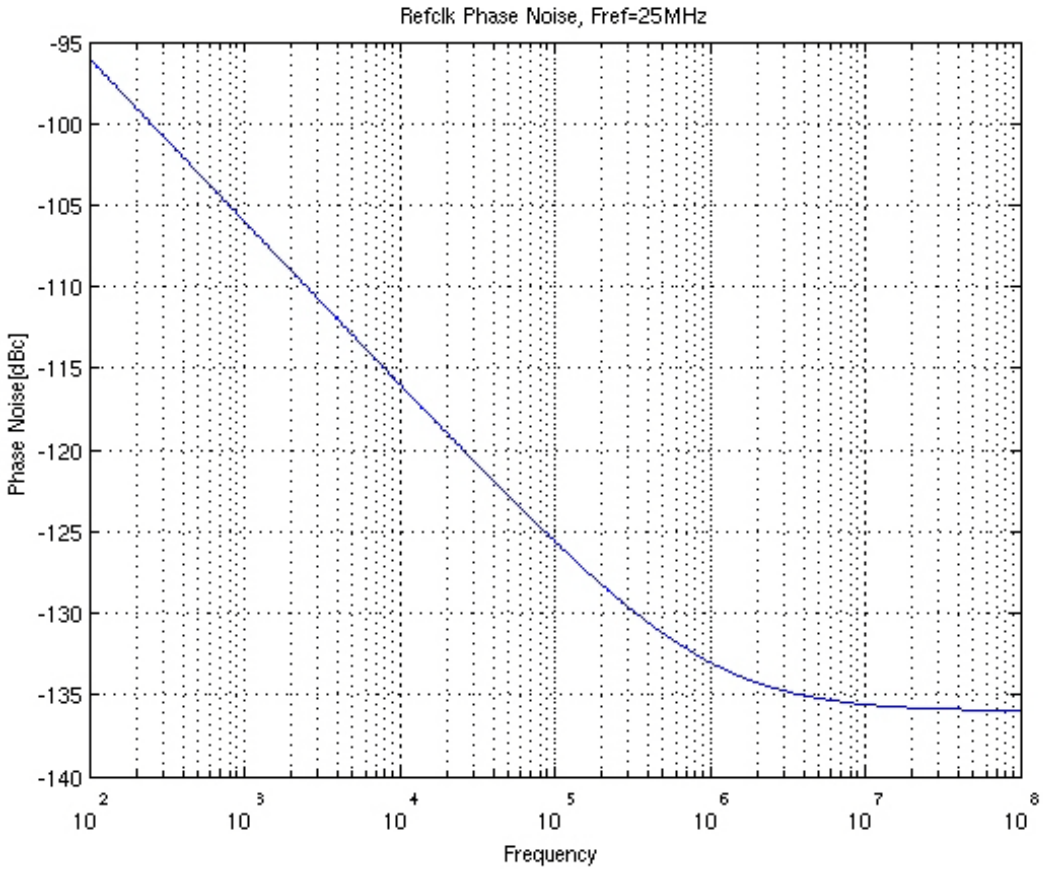



**Figure 11-17 Refclk Phase Noise as a Function of Frequency (Non-High Speed Serial)**

# 11.5 Package

## 11.5.1 Mechanical

The 82599 is assembled in a 25 x 25 FCBGA package with an 8-layer substrate.

**Table 11-27 Package Specifications**

| Body Size | Ball Count | Ball Pitch | Ball Matrix | Substrate |
|---|---|---|---|---|
| 25x25 $mm^2$ | 576 | 1 mm | 24 X 24 | Eight Layers |

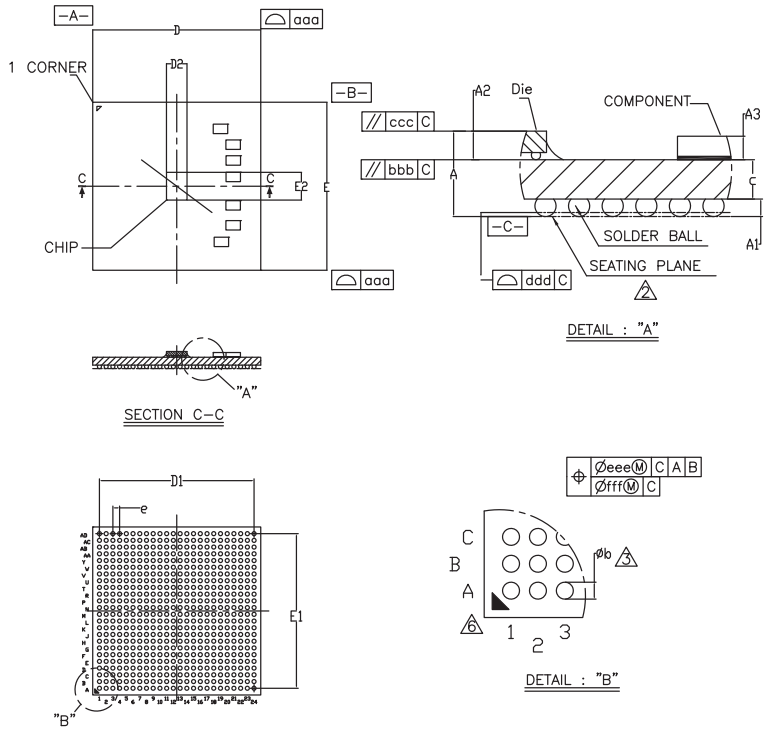
## 11.5.2 Thermal

For the 82599's package thermals please refer to Section 13.0.

## 11.5.3 Electrical

Package electrical models are part of the IBIS files.

## 11.5.4 Mechanical Package




| Symbol | Dimension in mm | | | Dimension in inch | | |
|---|---|---|---|---|---|---|
| | MIN | NOM | MAX | MIN | NOM | MAX |
| A | 2.325 | 2.540 | 2.755 | 0.092 | 0.100 | 0.108 |
| A1 | 0.44 | 0.52 | 0.60 | 0.017 | 0.020 | 0.024 |
| A2 | 0.80 | 0.85 | 0.90 | 0.031 | 0.033 | 0.035 |
| A3 | --- | --- | 0.70 | --- | --- | 0.028 |
| c | 1.085 | 1.170 | 1.255 | 0.043 | 0.046 | 0.049 |
| D/E | 24.95 | 25.00 | 25.05 | 0.982 | 0.984 | 0.986 |
| D1/E1 | --- | 23.00 | --- | --- | 0.906 | --- |
| e | --- | 1.00 | --- | --- | 0.039 | --- |
| b | 0.55 | 0.60 | 0.65 | 0.022 | 0.024 | 0.024 |
| aaa | 0.20 | | | 0.008 | | |
| bbb | 0.25 | | | 0.010 | | |
| ccc | 0.20 | | | 0.008 | | |
| ddd | 0.20 | | | 0.008 | | |
| eee | 0.25 | | | 0.010 | | |
| fff | 0.10 | | | 0.004 | | |

| | Die size type (mm) | Die size type (inch) |
|---|---|---|
| | Niantic | Niantic |
| D2 | 9.44 REF | 0.372 REF |
| E2 | 7.83 REF | 0.308 REF |

NOTE :

1. CONTROLLING DIMENSION : MILLIMETER.
2. PRIMARY DATUM C AND SEATING PLANE ARE DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.
3. DIMENSION b IS MEASURED AT THE MAXIMUM SOLDER BALL DIAMETER, PARALLEL TO PRIMARY DATUM C.
4. THERE SHALL BE A MINIMUM CLEARANCE OF 0.25mm BETWEEN THE EDGE OF THE SOLDER BALL AND THE BODY EDGE.
5. SPECIAL CHARACTERISTICS C CLASS: A , ddd

**Figure 11-18 Mechanical Package**

# 11.6 Devices Supported

## 11.6.1 Flash

The 82599 supports Flash devices with a SPI interface. Section 11-28 lists the specific Flash types supported,

**Table 11-28 Supported Flash Devices**

| Density | Atmel PN | STM PN |
|---|---|---|
| 1 Mb | AT25F1024N-10SI-2.7 or AT25FS010 | M25P10-AVMN6T |
| 2 Mb | AT25F2048N-10SI-2.7 | M25P20-AVMN6T |
| 4 Mb | AT25F4096N-10SI-2.7 | M25P40-AVMN6T |
| 8 Mb | | M25P80-AVMN6T |

**Table 11-28 Supported Flash Devices (Continued)**

| Density | Atmel PN | STM PN |
|---|---|---|
| 16 Mb | | M25P16-AVMN6T |
| 32 Mb | | M25P32-AVMN6T |

*Notes:*

1. Since all SPI Flash memories have similar interface characteristics, there is no need to test the interface with all the proposed types. It's acceptable to test the largest Flash.
2. All supported Flashes have an address size of 24 bits.

## 11.6.2 EEPROM

Section 11-29 lists the specific EEPROM devices supported.

**Table 11-29 Supported EEPROM Devices**

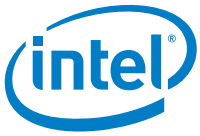
| Density [Kb] | Atmel PN | STM PN | Catalyst PN |
|---|---|---|---|
| 128 | AT25128AN-10SI-2.7 | M95128WMN6T | CAT25CS128-TE13 |
| 256 | AT25256AN-10SI-2.7 | M95256WMN6T | |

### 11.6.2.1 Minimum EEPROM Sizes

- No manageability - 16 KB (128 Kb)
- SMBus/NC-SI - 32 KB (256 Kb)

### 11.6.2.2 Recommended EEPROM Sizes

- No manageability - 32 KB (256 Kb)
- SMBus/NC-SI - 32 KB (256 Kb)



# 12.0 Design Considerations and Guidelines

This section provides recommendations for selecting components, connecting interfaces, dealing with special pins, and layout guidance.

Some unused interfaces should be terminated with pull-up or pull-down resistors. These are indicated in Section 2.0 or reference schematics. There are reserved pins, identified as RSVD_3P3, RSVD_1P2 and RSVD_VSS. The 82599 might enter special test modes unless these strapping resistors are in place.

Some unused interfaces must be left open. Do not attach pull-up or pull-down resistors to any balls identified as No Connect or Reserved No Connect.

## 12.1 Connecting the PCIe Interface

The 82599 connects to the host system using a PCIe interface. The interface can be configured to operate in several link modes. These are detailed in Section 3.0. A link between the ports of two devices is a collection of lanes. Each lane has to be AC-coupled between its corresponding transmitter and receiver; with the AC-coupling capacitor located close to the transmitter side (within 1 inch). Each end of the link is terminated on the die into nominal 100 Ω differential DC impedance. Board termination is not required.

Refer to the *PCI Express* Base Specification, Revision 2.0* and *PCI Express* Card Electromechanical Specification, Revision 2.0.*

### 12.1.1 Link Width Configuration

The 82599 supports link widths of x8, x4, x2, or x1 as determined by the PCIe init configuration. The configuration is loaded using bits 9:4 in the *Max Link Width* field of the Link Capabilities register (0xAC). The 82599 default is the x8 link width.

During link configuration, the platform and the 82599 negotiate a common link width. In order for this to work, the selected maximum number of PCIe lanes must be connected to the host system.

### 12.1.2 Polarity Inversion and Lane Reversal

To ease routing, designers have the flexibility to use the lane reversal modes supported by the the 82599. Polarity inversion can also be used, since the polarity of each differential pair is detected during the link training sequence.

When lane reversal is used, some of the down-shift options are not available. For a description of available combinations, see Section 3.0.

### 12.1.3 PCIe Reference Clock

For LOM designs, the device requires a 100 MHz differential reference clock, denoted PE_CLK_p and PE_CLK_n. This signal is typically generated on the system board and routed to the PCIe port. For add-in cards, the clock is furnished at the PCIe connector.

The frequency tolerance for the PCIe reference clock is +/- 300 ppm.

### 12.1.4 PCIe Analog Bias Resistor

For proper biasing of the PCIe analog interface, a 24.9 Ω 0.5% resistor needs to be connected from the PE_RBIAS to the VCC1P2 supply. The PE_RSENSE pin should be connected directly to PE_RBIAS, as close as possible to the 24.9 Ω resistor pad. To avoid noise coupled onto this reference signal, place the bias resistor close to the 82599 and keep traces as short as possible.

### 12.1.5 Miscellaneous PCIe Signals

The 82599 signals power management events to the system by pulling low the PE_WAKE# signal. This signal operates like the PCI PME# signal. Note that somewhere in the system, this signal has to be pulled high to the auxiliary 3.3 V supply rail.

The PE_RST# signal, which serves as the familiar reset function for the 82599, needs to be connected to the host system's corresponding signal.

### 12.1.6 PCIe Layout Recommendations

For information regarding the PCIe signal routing, refer to the *Intel® 82599 10 GbE Controller Checklists for further layout guidance*.

# 12.2 Connecting the MAUI Interfaces

The 82599 has two high speed network interfaces that can be configured in different 1 Gb/s and 10 Gb/s modes (CX4, KX, KX4, KR, XAUI, KR, and SFI+). Choose the appropriate configuration for your system configuration.

## 12.2.1 MAUI Channels Lane Connections

For BX, KX, KR, and SFI+ connections, only the first lane needs to be connected (TX0_L0_p, TX0_L0_n; RX0_L0_p, RX0_L0_n). For the remainder of the interfaces, all four differential pairs need to be connected for each direction.

These signals are 100 Ω terminated differential signals that are AC coupled near the receiver. Place the AC coupling capacitors less than one inch away from the receiver. For recommended capacitor values, consult the relevant IEEE 802.3 specifications and/or the relevant PICMG specifications. Capacitor size should be small to reduce parasitic inductance. Use X5R or X7R, ±10% capacitors in a 0402 or 0201 package size.

**Notes:**

1. SFI+ board traces generally do not require AC coupling capacitors because they are normally integrated into the SFP+ module.
2. Unused pins can be left as a no connect (NC), including the pins for the port not present on the 82599EN single port SKU.

## 12.2.2 MAUI Bias Resistor

For proper biasing of the MAUI analog interface, a 1 KΩ ±0.5% resistor needs to be connected between the XA_RBIAS and XA_SENSE pins. To avoid noise coupled onto this reference signal, place the bias resistor close to the 82599 and keep traces as short as possible.

## 12.2.3 XAUI, KX/KR, BX4, CX4, BX and SFI+ Layout Recommendations

This section provides recommendations for routing the high-speed interface. The intent is to route this interface optimally using FR4 technology. Intel has tested and characterized these recommendations.

## 12.2.4 Board Stack-Up Example

Printed Circuit Boards (PCBs) for these designs typically have six, eight, or more layers. Although, the the 82599 does not dictate stackup, the following examples are of typical stack-up options.

Microstrip Example:

- Layer 1 is a signal layer.
- Layer 2 is a ground layer.
- Layer 3 is used for power planes.
- Layer 4 is a signal layer. Careful routing is necessary to prevent crosstalk with layer 5.
- Layer 5 is a signal layer. Careful routing is necessary to prevent crosstalk with layer 4.
- Layer 6 is used for power planes.
- Layer 7 is a signal ground layer.
- Layer 8 is a signal layer.

**Note:** Layers 4 and 5 should be used mostly for low-speed signals because they are referenced to potentially noisy power planes that might also be slotted.

Stripline Example:

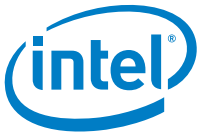
- Layer 1 is a signal layer.
- Layer 2 is a ground layer.
- Layer 3 is a signal layer.
- Layer 4 is used for power planes
- Layer 5 is used for power planes
- Layer 6 is a signal layer.
- Layer 7 is a signal ground layer.
- Layer 8 is a signal layer.

**Note:** To avoid the effect of the potentially noisy power planes on the high-speed signals, use offset stripline topology. The dielectric distance between the power plane and signal layer should be three times the distance between ground and signal layer.

This board stack-up configuration can be adjusted to conform to your company's design rules.

## 12.2.5 Trace Geometries

For 82599 SFI Designs, trace geometries are found in the Intel® 82599 10 Gigabit Ethernet Controller – SFI (SFP + Interface) Design Guide. Contact your Intel representative for details.



## 12.2.6 Other High-Speed Signal Routing Practices

These layout and routing recommendations are applicable for the MAUI interfaces of the the 82599.

In order to keep impedance continuity consistent around signal via anti-pad regions, Intel recommends adding the anti-pad diameter requirement of 9to 12 mils clearance to vias to ground and power. This ensures that the impedance variance is minimized. On plane layers, pairs of signal vias should share the same enlarged elliptical or oval (merged) anti-pads.

Enforce differential symmetry, even for grounds. Along with ensuring that the MAUI interface is routed symmetrically in terms of signal routing and balance, we also recommend that ground paths be routed symmetrically. This helps reduce the imbalance that can occur in the different return current paths.

In cases where there is a via and an AC coupling capacitor on the same trace, the signal trace between the via and the AC coupling capacitors on the MAUI interface, there is an intrinsic impedance mismatch because of the required capacitors. To minimize the overall impact of having vias and AC coupling capacitors, we recommend that both via and capacitor layout pad be placed within 100 mils of each other.

**Note:** For KR interfaces, this is not recommended unless simulations are performed and the results confirm minimal impact to impedance, insertion loss, insertion loss deviation, and crosstalk.

It is best to use a 0402 capacitor or smaller for the AC coupling components on the MAUI interface. The pad geometries for a 0402 or smaller components lend themselves to maintaining a more consistent transmission line environment. For 10 Gb/s KR, the recommended package size for the required AC coupling capacitors is the 0201 package size. Note that SFI+ board traces normally do not require AC coupling capacitors. Contact your Intel sales representative for more details.

**Note:** To reduce shunt capacitance from the AC capacitors' solder-pads to the reference plane beneath the solder-pads, we recommend that you void the reference plane that is directly under the capacitor. The reference plane void should have the same shape as the capacitor and it's solder pads. The size of the reference plane void should be slightly larger than the size of the capacitor and it's solder pad. If you have access to a 3-dimensional field solver, it can and should be used to determine the optimal size and shape for the reference plane void under each capacitor. To prevent noise problems, be careful not to route any traces across the capacitor-shaped voids in the reference plane.

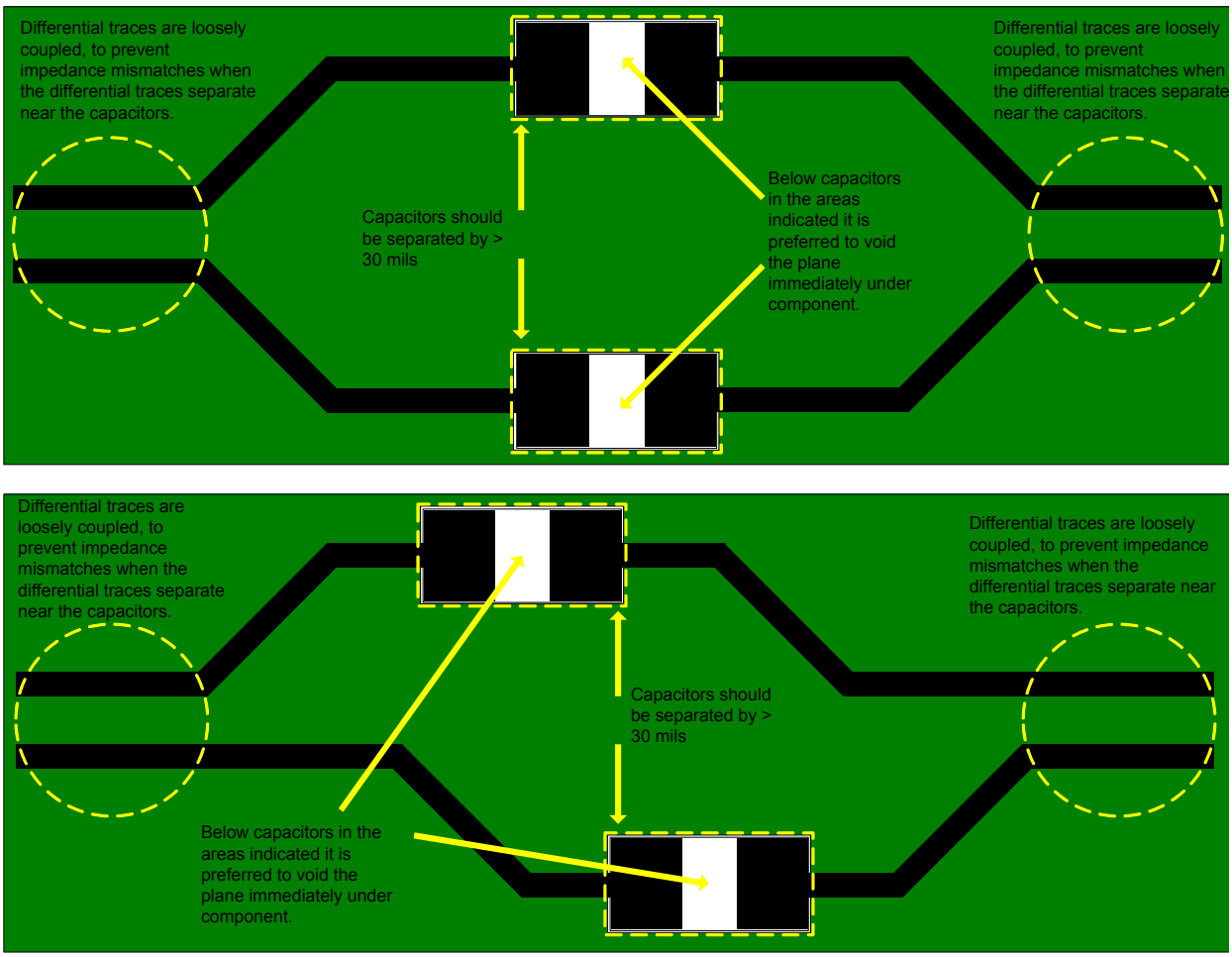



**Figure 12-1 Reference Plane Voids Under AC Caps and AC Caps Separated to Reduce Stray Capacitance Between Caps**

**Note:** The top layout of Figure 12-1 shows acceptable guidelines. The bottom layout of Figure 12-1 shows the preferred guidelines.

Use the smallest possible vias on board to optimize the impedance for the MAUI interface.

### 12.2.6.1 Via Usage

Use vias to optimize signal integrity. Figure 12-2 shows correct via usage. Figure 12-3 shows the type of topology that should be avoided.

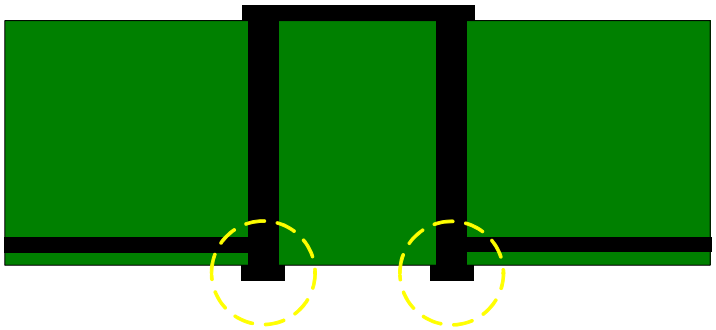


**Figure 12-2 Correct Via Usage**

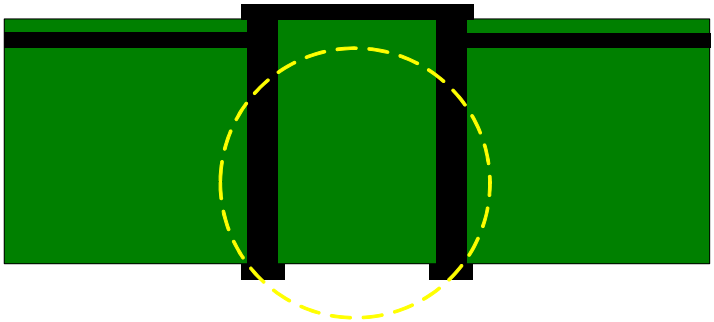


**Figure 12-3 Incorrect Via Usage**

Any via stubs on the KR and/or SFI+ differential signal traces must be less than 35 mils in length. Keeping KR and SFI+ signal via stubs less than or equal to 20 mils is preferable.

**Note:** Vias on SFI+ traces are not recommended. SFI+ Tx signals must not have any vias. Refer to the SFI+ layout section for more information.

Place ground vias adjacent to signal vias used for the MAUI interface. Do NOT embed vias between the high-speed signals, but place them adjacent to the signal vias (see Figure 12-4). This helps to create a better ground path for the return current of the AC signals, which also helps address impedance mismatches and EMC performance.

We recommend that, in the breakout region between the via and the capacitor pad, you target a Z0 for the via to capacitor trace equal to 50 Ω. This minimizes impedance imbalance.

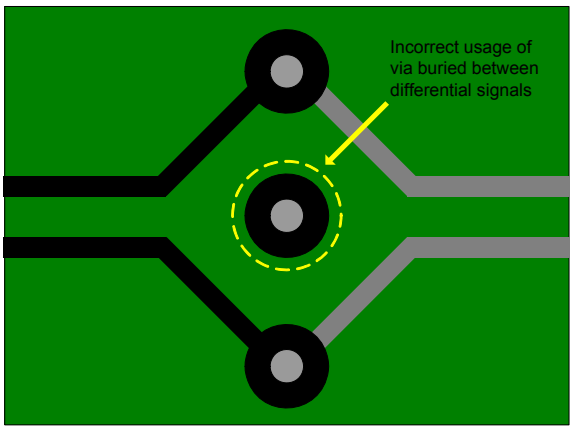



**Figure 12-4 No Vias Between High-Speed Traces in the Same Differential Pair**

## 12.2.7 Reference Planes

Do not cross plane splits with the MAUI high-speed differential signals. This causes impedance mismatches and negatively affects the return current paths for the board design and layout. Refer to Figure 12-5.

Traces should not cross power or ground plane splits if at all possible. Traces should stay seven times the dielectric height away from plane splits or voids. If traces must cross splits, capacitive coupling should be added to stitch the two planes together in order to provide a better AC return path for the high-speed signals. To be effective, the capacitors should be have low ESR and low equivalent series inductance.

**Note:** Even with plane split stitching capacitors, crossing plane splits is extremely high risk for 10 Gb/s KR and 10 Gb/s SFI+ designs.

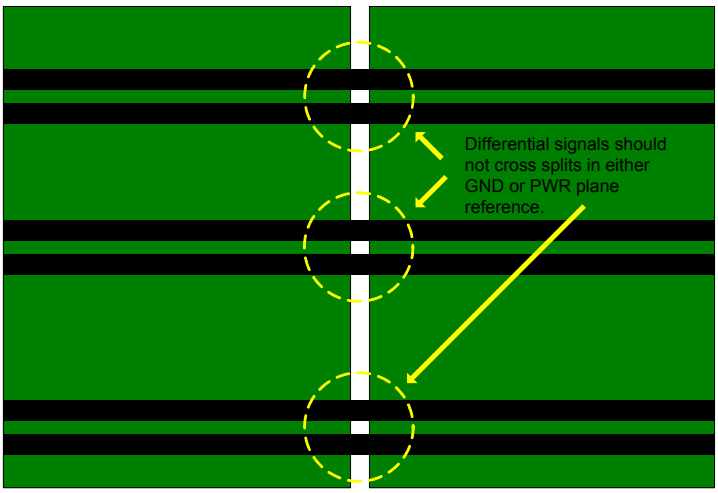



**Figure 12-5 Do Not Cross Plane Splits**

Keep Rx and Tx separate. This helps to minimize crosstalk effects since the TX and RX signals are NOT synchronous. This is the more natural routing method and occurs without much designer interference.

We recommend that the MAUI signals stay at least seven times the dielectric height away from any power or ground plane split (see Figure 12-6). This improves impedance balance and return current paths.

If a high-speed signal needs to reference a power plane, then ensure that the height of the secondary (power) reference plane is at least 3 x the height of the primary (ground) reference plane.

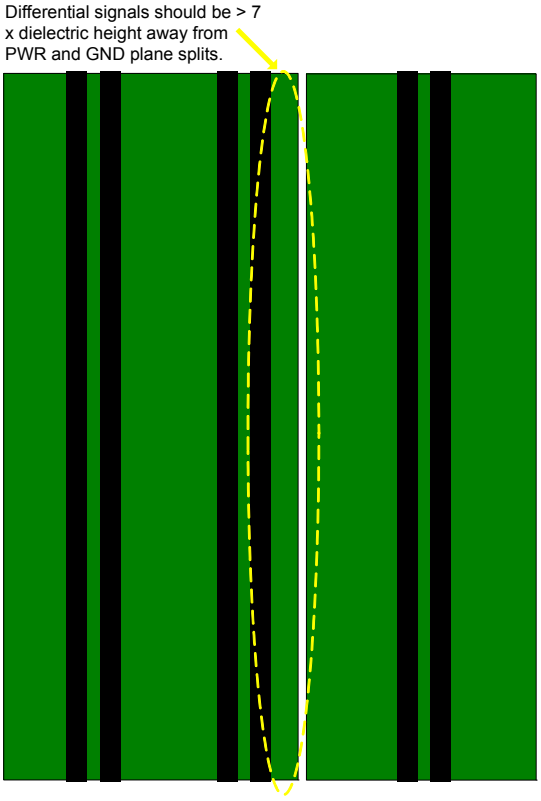
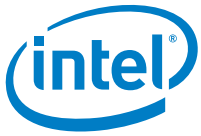



**Figure 12-6 Traces Should Stay Seven Times the Dielectric Height Away From Plane Splits Or Voids**

## 12.2.8 Dielectric Weave Compensation

Because the dielectric weave can cause different propagation velocity on each of the traces within one differential pair, Intel recommends using one or more trace routing techniques that can minimize signal skewing caused by the weave:

- Instead of routing traces parallel to either X or Y axis, traces should be routed at an angle to the weave, and the angle should be between 11 and 45 degrees. Routing both traces within each differential pair at an angle, with respect to the dielectric weave, minimizes the signal skew within each differential pair.
- The center-to-center pitch of the traces within the diff pairs can be matched to the weave pitch of the dielectric material. If you plan to uses a woven glass/epoxy dielectric material, check with the material supplier to find out the glass weave pitch prior to doing final differential trace routing.
- Traces can be routed to include a series of 45 degree bends, with bends separated by several tenths of an inch, to shift the traces in steps by a few millimeters each time. There should be an equal number left turns and right turns along the length of the traces. Trace segments between each pair of bends should be different lengths (if they are all the same length it could create an undesirable resonance in the line).



- If differential traces must be straight and orthogonal to the outline of the circuit board for most of their routed lengths, then rotate CAD artwork by 15°, with respect to the weave of the circuit board's dielectric weave.

## 12.2.9 Impedance Discontinuities

Impedance discontinuities cause unwanted signal reflections. Minimize vias (signal through holes) and other transmission line irregularities. A total of six through holes (a combination of vias and connector through holes) between the two chips connected by the MAUI interface is a reasonable maximum budget for each differential signal path. For example, if a backplane system has a total of three boards (blade server, mid-plane, and switch blade) in the differential signal path, then SFI+ Tx must not have any signal vias and SFI+Rx should not have more than one signal via per SFI+ signal trace. For this purpose, signal pin through-holes for board connectors are also counted as signal vias. Signal via pads on unconnected plane layers can be removed to reduce capacitance between the signal via and the surrounding metal plane. Alternatively, the anti-pad diameter can be increased to provide 9to 12 mils clearance between signal via (pads) and power or gound.

## 12.2.10 Reducing Circuit Inductance

Traces should be routed over a continuous reference plane with no interruptions. If there are vacant areas on a reference or power plane, the signal conductors should not cross the vacant area. Routing over a void in the reference plane causes impedance mismatches and usually increases radiated noise levels. Noisy logic grounds should NOT be located near or under high-speed signals or near sensitive analog pin regions of the LAN silicon. If a noisy ground area must be near these sensitive signals or IC pins, ensure sufficient decoupling and bulk capacitance in these areas. Noisy logic and switching power supply grounds can sometimes affect sensitive DC subsystems such as analog to digital conversion, operational amplifiers, etc.

All ground vias should be connected to every ground plane; and similarly, every power via should be connected to all equally potential power planes. This helps reduce circuit inductance. Another recommendation is to physically locate grounds to minimize the loop area between a signal path and its return path. Rise and fall times should be as slow as possible while still meeting the relevant electrical requirements. Because signals with fast rise and fall times contain many high frequency harmonics, which can radiate significantly. The most sensitive signal returns closest to the chassis ground should be connected together. This results in a smaller loop area and reduces the likelihood of crosstalk. The effect of different configurations on the amount of crosstalk can be studied using electronics modeling and simulation software.

### 12.2.11 Signal Isolation

To maintain best signal integrity, keep digital signals far away from the analog traces. A good rule of thumb is no digital signal should be within 7x to 10x dielectric height of the differential pairs. If digital signals on other board layers cannot be separated by a ground plane, they should be routed at a right angle (90 degrees) to the differential signal traces. If there is another LAN controller on the board, take care to keep the differential pairs from that circuit away. The same thing applies to switching regulator traces.

Rules to follow for signal isolation:

- Separate and group signals by function on separate board layers if possible. Maintain a separation that is at least seven times the thinnest adjacent dielectric height between all differential pairs (Ethernet) and other nets, but group associated differential pairs together. For example, Keep Tx signals with Tx signals and keep Rx signals with Rx signals. Note that if an Rx signal is routed between two Tx signals, the higher levels of Tx crosstalk causes the Rx signal-to-noise ratio to be worse than if the Rx signal is routed between two other Rx signals.)
- Over the length of the trace run, each differential pair should be at least seven times the thinnest adjacent dielectric height away from any parallel signal traces.
- Physically group together all components associated with one clock trace to reduce trace length and radiation.
- Isolate other I/O signals from high-speed signals to minimize crosstalk because crosstalk can increase radiated EMI and can increase susceptibility to EMI from other signals.
- Avoid routing high-speed LAN traces near other high-frequency signals associated with a video controller, cache controller, processor, or other similar devices.

### 12.2.12 Power and Ground Planes

Good grounding requires minimizing inductance levels in the interconnections and keeping ground returns short, signal loop areas small, and locating decoupling capacitors at or near power inputs to bypass to the signal return. This will significantly reduce EMI radiation.

These guidelines reduce circuit inductance in both backplanes and motherboards:

- Route traces over a continuous plane with no interruptions. Do not route over a split power or ground plane. If there are vacant areas on a ground or power plane, avoid routing signals over the vacant area. Routing signals over power or ground voids increases inductance and increases radiated EMI levels.
- Use distance and/or extra decoupling capacitors to separate noisy digital grounds from analog grounds to reduce coupling. Noisy digital grounds may affect sensitive DC subsystems.
- All ground vias should be connected to every ground plane; and every power via should be connected to all power planes at equal potential. This helps reduce circuit inductance.
- Physically locate grounds between a signal path and its return. This will minimize the loop area.

- Avoid fast rise/fall times as much as possible. Signals with fast rise and fall times contain many high frequency harmonics, which can radiate EMI.
- Do not route high-speed signals near switching regulator circuits.
- There should not be any test-point vias or test-point pads on KR and SFI+ traces.
- It's acceptable to put ground fill or thieving on the trace layers, but preferably not closer than 50 mils to the differential traces and the connector pins.
- If differential traces must be routed on another layer, then the signal vias should carry the signal to the opposite side of the circuit board (to be near the top of the circuit board); AND if the high-speed signals are being routed between two connectors on the same board, then before the signal traces reach the second connector, they must return to the original signal layer (before reaching the connector pin). This strategy keeps via stubs short without requiring back drilling.
- Each time differential traces make a layer transition (pass through a pair of signal vias), there must be at least one ground via located near each signal via. Two ground vias near each signal via is better. See Figure 12-7 and Figure 12-8.

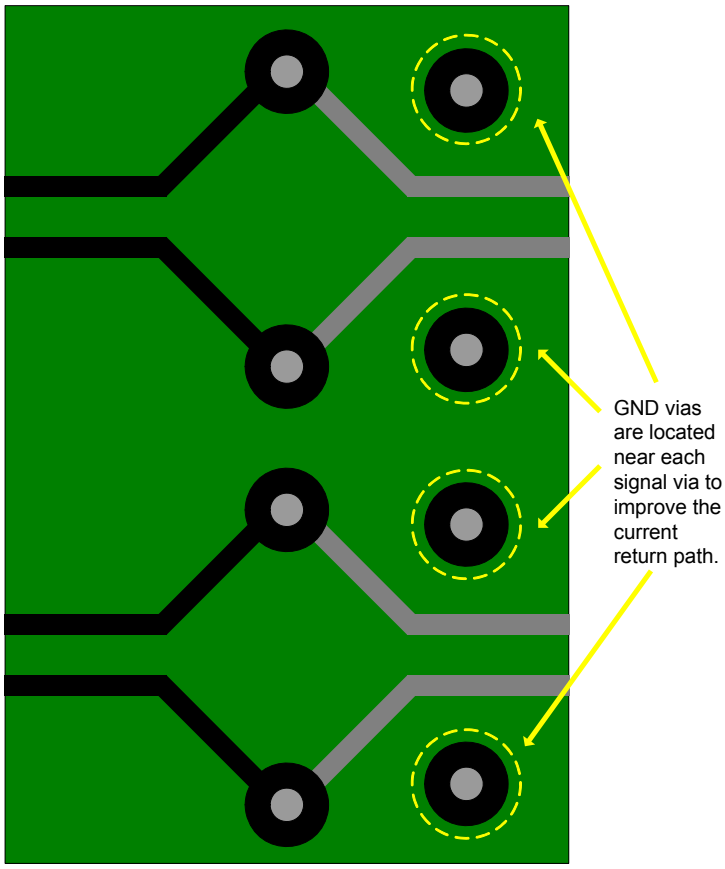



**Figure 12-7 Good Ground Vias for Signal Return Paths – One Return Path Via Per Signal Via**

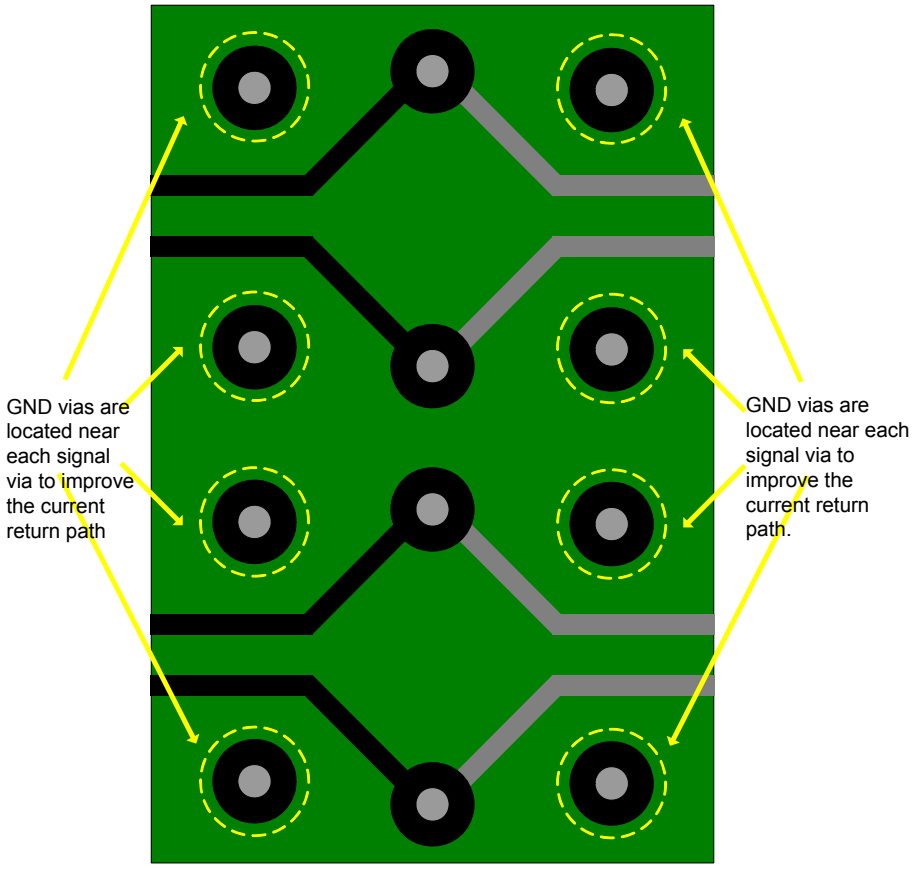



**Figure 12-8 Better Ground Vias for Signal Return Paths – Two Return Path Vias Per Signal Via (Less Reflection)**

If the circuit board fabrication process permits it, it is best to remove signal via pads on unconnected metal layers. See Figure 12-9 and Figure 12-10.

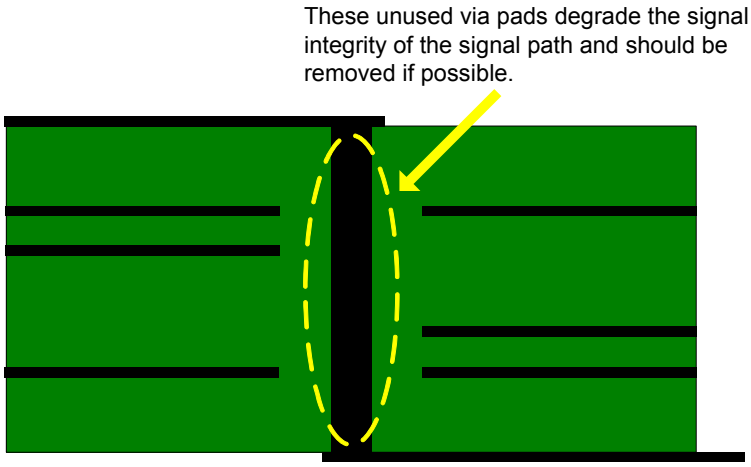



**Figure 12-9 Undesirable: For Signal Vias to Have Pads on the Unused Metal Layers**

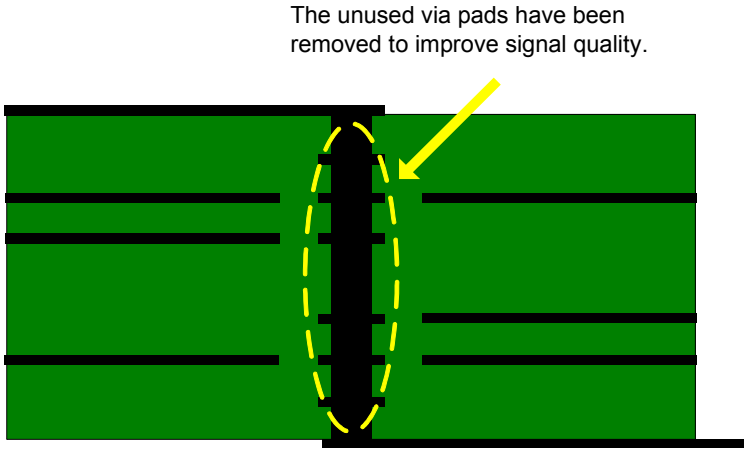



**Figure 12-10 Signal Via Improved by Removing Unused Metal Layer Pads**

On metal layers where signal vias need to have via pads, it is desirable to reduce capacitance between the signal vias and ground plane layers. The anti-pad diameters should be up to 20 mils larger than the via pad diameters. See Figure 12-11. Clearance between the pad and the surrounding metal should be >= 10 mils.

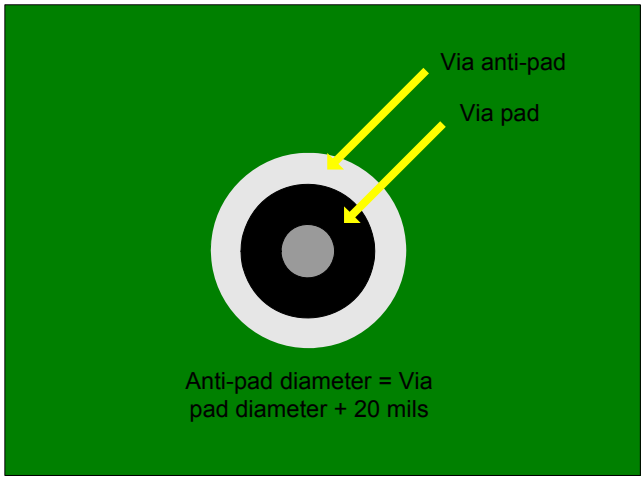



**Figure 12-11 Increase Anti-Pad Diameter To Reduce Shunt Capacitance**

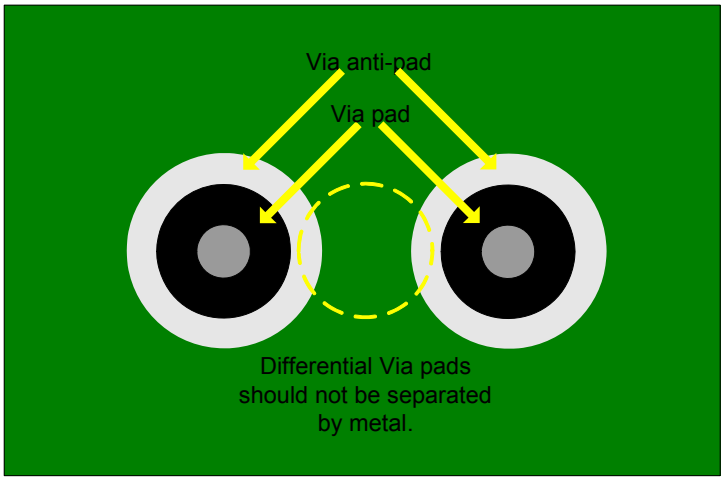



**Figure 12-12 Differential Signal Via Pads Should Not Be Separated By Metal**

Each time differential signal vias pass through a plane layer, within each differential pair, the anti-pads should overlap. See Figure 12-13 and Figure 12-14.

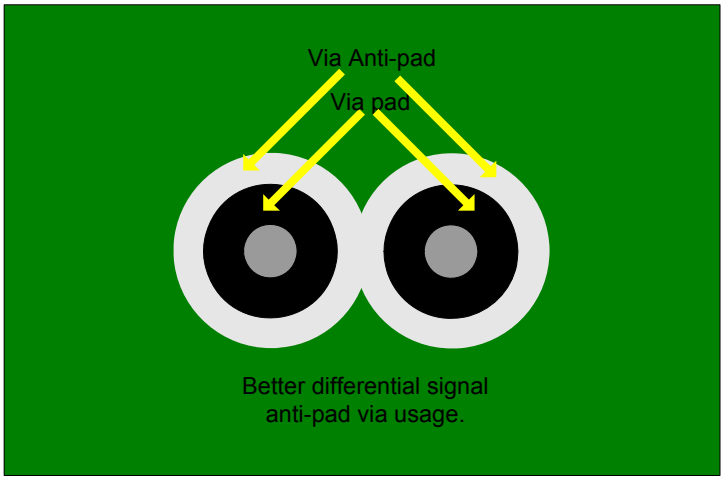



**Figure 12-13 Better Differential Signals Via Anti-Pads**

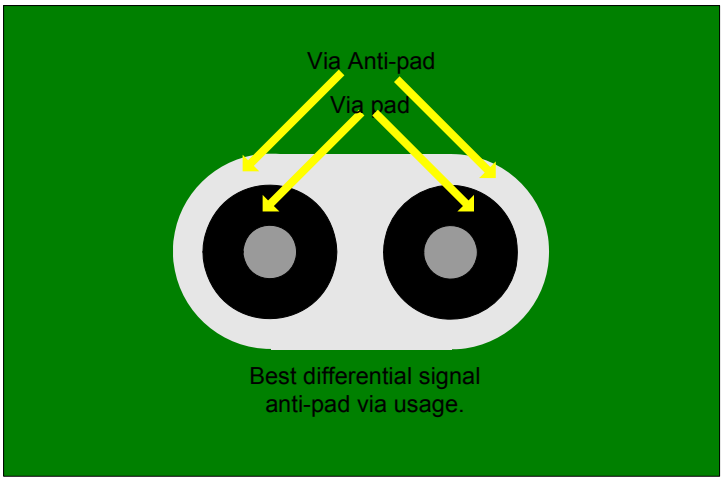



**Figure 12-14 Best Differential Signals Via Anti-Pads**

## 12.2.13 KR and SFI+ Recommended Simulations

KR and SFI+ signaling frequencies extend above 5 GHz: relatively short stubs, small discontinuities, and fairly small in-pair trace length differences can cause an undesirable increase in bit errors. Before ordering circuit boards, verify that:

- Planned KR signal trace routing on the circuit board complies with the interconnect characteristics recommended in IEEE 802.3ap sections 69.3 and 69.4.
- Planned SFI+ signal trace routing on the circuit board complies with the guidance provided in this document and complies with the interconnect characteristics recommended in the SFF-8431 specifications. Contact your Intel sales representative for more details.
- For most KR board traces:
  - If possible, export S-parameters for the planned KR signal channels, and compare them to the IEEE recommended electrical characteristics. Optimize the KR signal path until it complies with the IEEE recommendations.

    IEEE channel characteristics recommendations are for the entire length of the board channels – from the solder-pads for one KR IC device to the solder-pads for another KR device, at the far-end of the entire KR channel path. This end-to-end KR signal path typically includes two or three circuit boards, connectors, and AC coupling caps.
- For unusual routing requirements, which make it difficult to meet the IEEE channel recommendations:
  - KR board trace channels, which have been optimized by following the layout guidelines recommended within this document but which cannot be improved enough to comply with IEEE 802.3ap recommended electrical characteristics, might still work satisfactorily with the 82599 LAN silicon in KR mode (see as follows).
- With sufficient advance notice, Intel engineers can provide assistance:
  - Trace routing should be optimized prior to the next steps – request a layout review (must be willing to provide board stack-up information and the KR traces CAD artwork).
  - After KR traces have been optimized, if the IEEE recommended electrical characteristics are still not being met, then end-to-end KR board channels S-parameter models should be extracted (preferably in Touchstone* S4p format) for additional investigative simulations by Intel signal integrity engineers. Please request the required S-parameter frequency range, step size etc, before extracting Touchstone S-parameter models.
  - KR values are at 2.5 and 5 GHz and pre/post plating information must also be provided.

## 12.2.14 Additional Differential Trace Layout Guidelines for SFI+ Boards

As stated in the SFF-8431 specifications, SFI+ differential traces should have a nominal differential impedance that is 100 Ω with ±10 Ω tolerance and with 7% differential coupling (nominal).

Differential coupling = [(4 x Zcm) - Zdiff] / [(4 x Zcm) + Zdiff], where Zcm is the common mode impedance and Zdiff is the differential impedance. For example, when Zcm is 28.76 Ω and Zdiff is
100 Ω, the coupling is 7%.

**Note:** Zdiff should be 100 Ω nominal with Zdiff tolerance within +/- 10% AND when Zdiff nominal is 100 Ω, then Zcm nominal should be about 28.76 Ω, maximum.

Signal vias should be avoided on SFI+ traces. The SFI+ transmit traces must not have any vias. If there must be one or two signal vias on the SFI+ receive traces, then each via should be accompanied by one or two ground return vias, via stubs should be <= 20 mils long, AND the board's SFI+ channel insertion loss and return loss should still conform to the SFF-828431 Appendix A, SFI+ channel recommendations. To verify conformance, AC channel simulations should be performed.

Because the SFP+ module connector pads on the end of the SFI+ traces are typically wider than the SFI+ traces, in order to avoid excessive capacitance, the reference plane areas directly under each pair of SFP+ module connector pads must be voided. The voided areas under the connector pads should be rectangular-like in shape, and should be a little larger than the area occupied by each pair of SFP+ module connector pads. Simulations should be performed, to verify that 100 Ω differential impedance has been maintained. The size and shape of the reference plane voids should be adjusted to maintain the desired impedance (see Figure 12-6 for an example).

SFI+ trace lengths and trace geometry: To meet the stringent transmitter electrical requirements, some trace geometry guidance is listed in Figure 12-15 and Figure 12-16 show examples.Contact your Intel sales representative for more details.

**Note:** Grey is the ground plane.

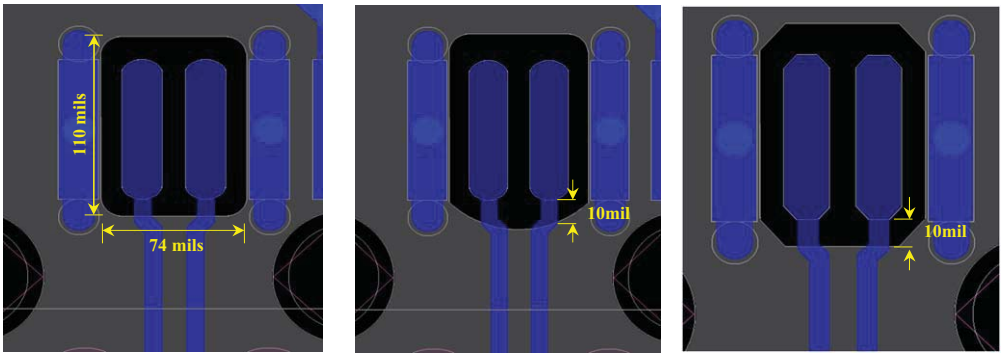



**Figure 12-15 Voiding the Reference Planes Under the SFI+ Connector Pin Solder Pads**

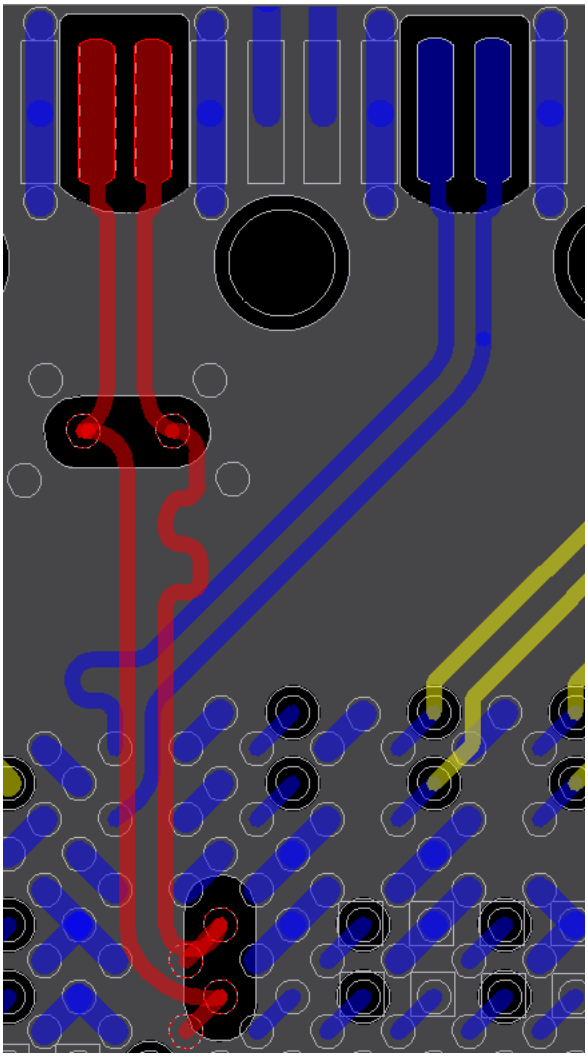
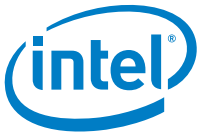


**Figure 12-16 SFI+ Breakout and Routing Example From the 82599 to SFI+ Connector Pad**

**Note:** The SFI+ transmit traces do not have any vias. Contact your Intel sales representative for more details.



# 12.3 Connecting the Serial EEPROM

The 82599 uses a Serial Peripheral Interface (SPI)* EEPROM. Several words of the EEPROM are accessed automatically by the 82599 after reset to provide pre-boot configuration data before it is accessed by host software. The remainder of the EEPROM space is available to software for storing the MAC address, serial numbers, and additional information. For a complete description of the content stored in the EEPROM see Section 6.0.

## 12.3.1 Supported EEPROM Devices

Table 12-1 lists the SPI EEPROMs that operate satisfactorily with the the 82599. SPI EEPROMs used must be rated for a clock rate of at least 2 MHz.

**Table 12-1. Supported SPI EEPROM Devices**

| Density (Kb) | Atmel* PN | STM* PN | Catalyst* PN |
|---|---|---|---|
| 128 | AT25128AN-1DSI-2.7 | M951286DWMN6T | CAT25C9128-TE13 |
| 256 | AT25256AN-1DSI-2.7 | M95256DWMN6T | |

**Note:** Refer to Section 11.6.2.1 and Section 11.6.2.2 for minimum and recommended EEPROM sizes.

For more information on how to properly attach the EEPROM device to the the 82599, follow the example provided in the 82599 reference schematics. Contact your Intel sales representative for access.

# 12.4 Connecting the Flash

The 82599 provides support for an SPI Flash device that is made accessible to the system through the following:

- Flash Base Address register (PCIe Control register at offset 0x14 or 0x18).
- An address range of the IOADDR register, defined by the IO Base Address register (PCIe) Control register at offset 0x18 or 0x20).
- Expansion ROM Base Address register (PCIe Control register at offset 0x30).

### 12.4.1 Supported Flash Devices

The 82599 supports SPI Flash type. All supported Flashes have address size of 24 bits. Table 12-3 lists the Flash types supported.

**Table 12-1 Flash Types Supported**

| Density | Atmel* PN | STM* PN |
|---|---|---|
| 512 Kb | AT25F512N-10SI-2.7 | M25P05-AVMN6T |
| 1 Mb | AT25F1024N-10SI-2.7 | M25P10-AVMN6T |
| 2 Mb | AT25F2048N-10SI-2.7 | M25P20-AVMN6T |
| 4 Mb | AT25F4096N-10SI-2.7 | M25P40-AVMN6T |
| 8 Mb | | M25P80-AVMN6T |
| 16 Mb | | M25P16-AVMN6T |
| 32 Mb | | M25P32-AVMN6T |

For more information on how to properly attach the Flash device to the 82599, follow the example provided in the 82599 reference schematics. Contact your Intel sales representative for access.

**Note:** If no Flash device is used, leave FLSH_CE_N, FLSH_SCK, FLSH_SI, FLSH_SO unconnected.

## 12.5 SMBus and NC-SI

SMBus and NC-SI are interfaces for pass-through and configuration traffic between the Management Controller (MC) and the 82599.

**Note:** Intel recommends that the SMBus be connected to an MC for the EEPROM recovery solution. If the connection is to a MC, it will be able to send the EEPROM release command.

The 82599 can be connected to an external MC. It operates in one of two modes:

- SMBus mode
- NC-SI mode

The clock-out (if enabled) is provided in all power states (unless the device is disabled).

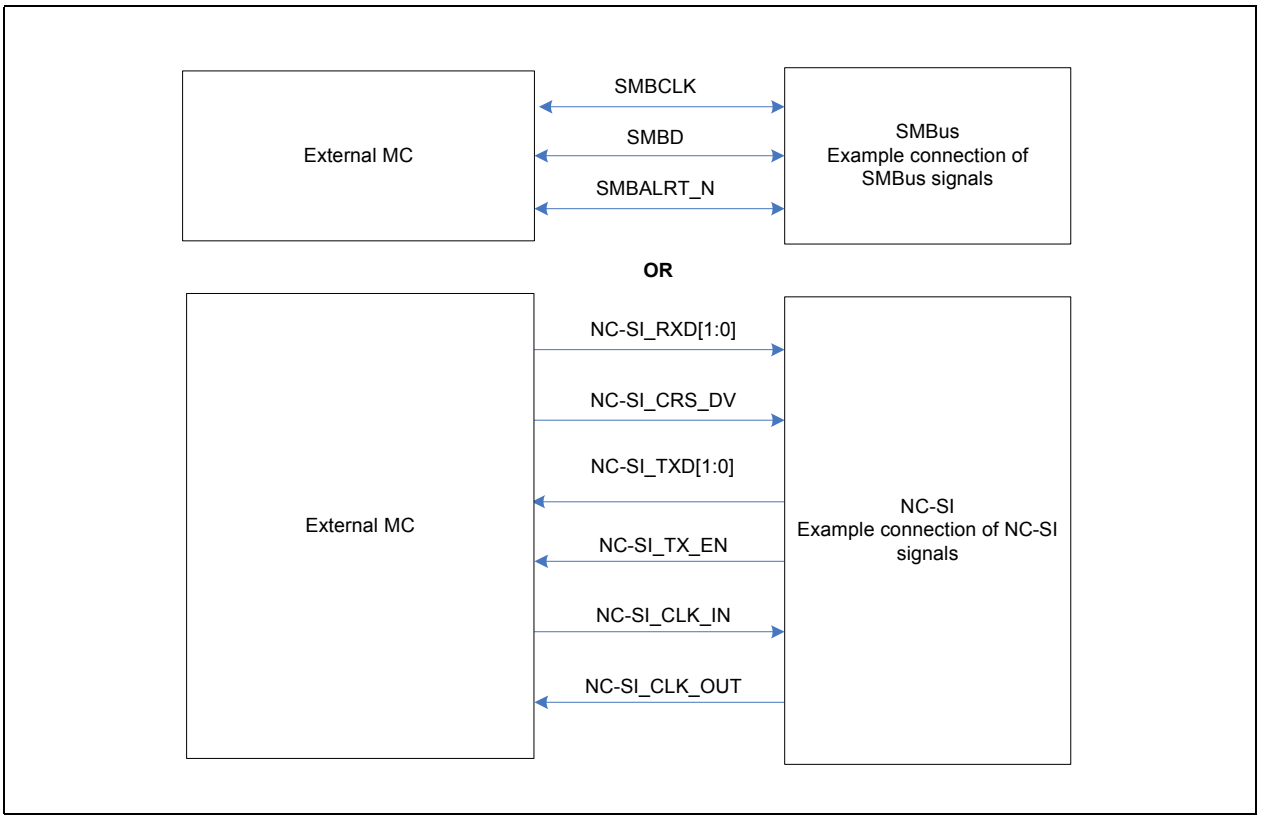



**Figure 12-1 External MC Connections with NC-SI and SMBus**

# 12.6 NC-SI

## 12.6.1 NC-SI Design Requirements

### 12.6.1.1 Network Controller

The NC-SI Interface enables network manageability implementations required by information technology personnel for remote control and alerting via the LAN. Management packets can be routed to or from a management processor.

### 12.6.1.2 External Management Controller (MC)

An external MC is required to meet the requirements called out in the latest NC-SI specification as it relates to this interface.

### 12.6.1.3 Reference Schematic

The following reference schematic (provides connectivity requirements for single and mulit-drop applications. This configuration only has a single connection to the MC. The network device also supports multi-drop NC-SI configuration architecture with software arbitration support from the MC.

Refer to the NC-SI specification for connectivity requirements for multi-drop applications.

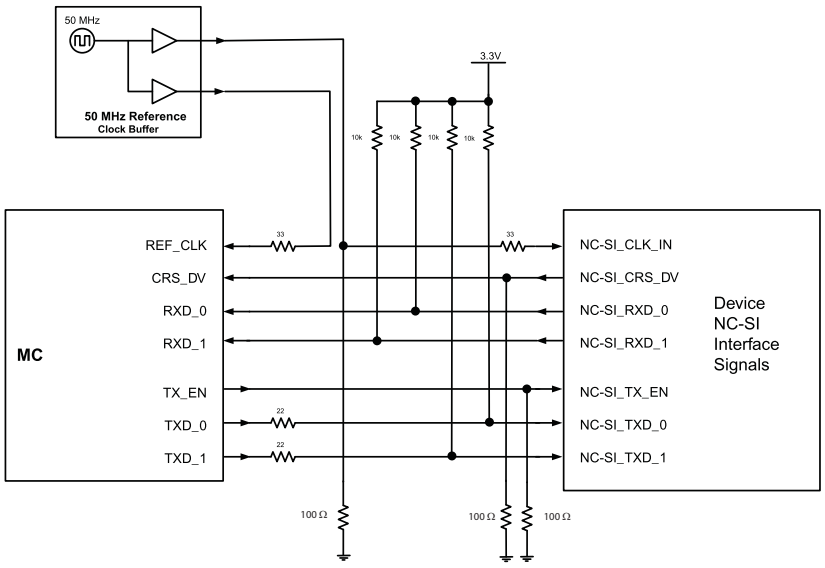



**Table 12-2 NC-SI Connection Schematic: Single-Drop Configuration**

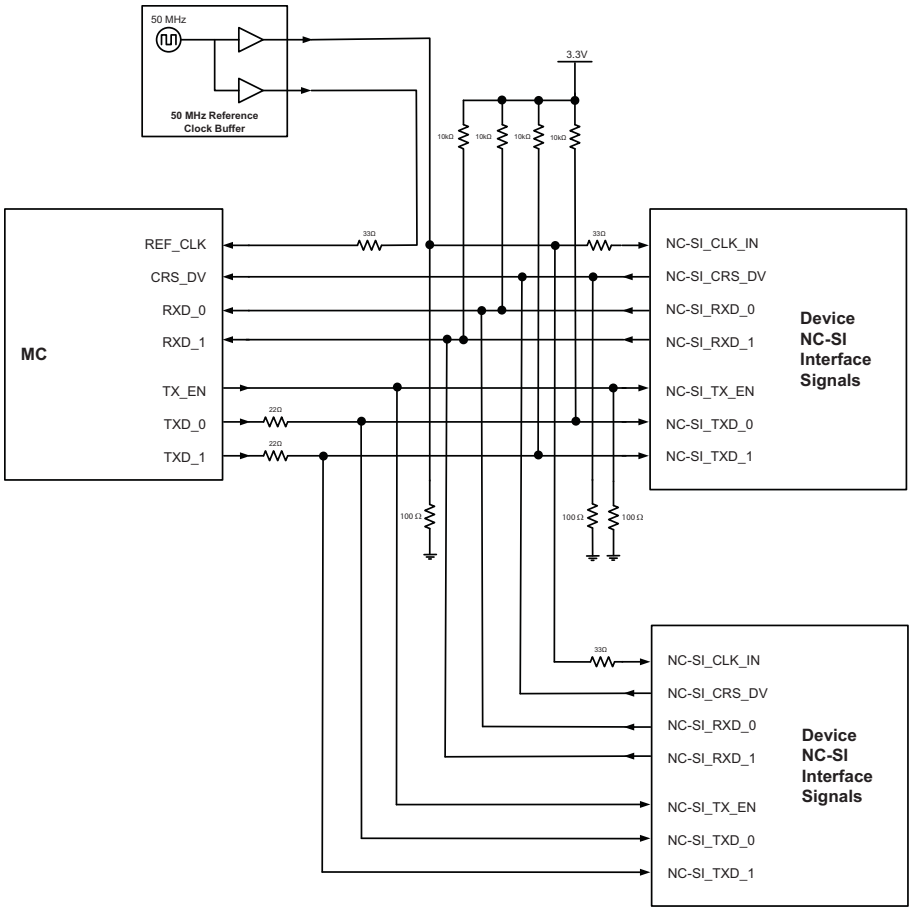



**Figure 12-2 NC-SI Connection Schematic: Multi-Drop Configuration**

## 12.6.2 NC-SI Layout Requirements

### 12.6.2.1 Board Impedance

The NC-SI signaling interface is a single ended signaling environment and as such Intel recommends a target board and trace impedance of 50 Ω plus 20% and minus 10%. This impedance ensures optimal signal integrity and quality.

### 12.6.2.2 Trace Length Restrictions

The recommended maximum trace lengths for each circuit board application is dependent on the number drops and the total capacitive loading from all the trace segments on each NC-SI signal net. The number via's must also be considered. Circuit board material variations and trace etch process variations affect the trace impedance and trace capacitance. For each fixed design, highest trace capacitance occurs when trace impedance is lowest.For the FR4 board stack-up provided in direct connect applications, the maximum length for a 50 Ω NC-SI trace would be approximately 9 inches on a minus 10% board impedance skew. This ensures that signal integrity and quality are preserved and enables the design to comply with NC-SI electrical requirements.

For special applications which require longer NC-SI traces, the total functional NC-SI trace length can be extended with non-compliant rise time by:

- providing good clock and signal alignment
- testing with the target receiver to verify it meets setup and hold requirements.

For multi-drop applications, the total capacitance and the extra resistive loading affect the rise time. A multi-drop of two devices limits the total length to 8 inches. A multi-drop of four limits the total length to 6.5 inches. Capacitive loading of extra via's have a nominal effect on the total load.

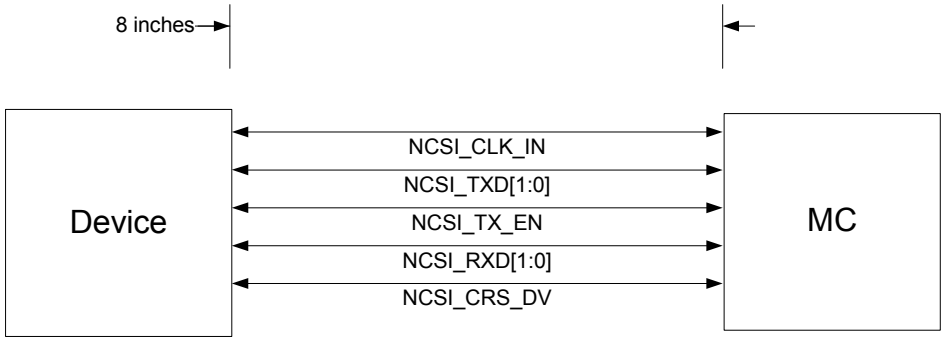



**Figure 12-3 NC-SI Trace Length Requirement for Direct Connect**

Table 12-3 lists how seven more vias increase the rise time by 0.5 ns. Again, longer trace lengths can be achieved.

### Table 12-3 Stack Up, Seven Vias

| Item | Value | Units |
|---|---|---|
| Trace width | 4.5 | mils |
| Trace thickness | 1.9 | mils |
| Dielectric thickness | 3.0 | mils |
| Dielectric constant | 4.1 | -- |
| Loss Tangent | 0.024 | -- |
| Nominal Impedance | 50 | W |
| Trace Capacitance | 1.39 | pf/inch |

Table 12-4 lists the example trace lengths for the multi-drop topology of two and four represented in the figures that follow.

### Table 12-4 Example Trace Lengths for Multi-Drop Topologies, 2 & 4

| Multi-drop length parameter used in Figure 12-4 and Figure 12-5 | Segment length example for multi drop configurations | | | |
|---|---|---|---|---|
| | Two drop configuration | | Four drop configuration | |
| | Length (Inches) | Trace capacitance (Pf) | Length (Inches) | Trace capacitance (Pf) |
| L1 | 2 | 2.8 | 1.5 | 8.8 |
| L2 | 4 | 5.6 | 2 | 16.1 |
| L3 | 2 | 2.8 | 1 | 8.1 |
| Total | 8 | 11.1 | 6.5 | 35.6 |

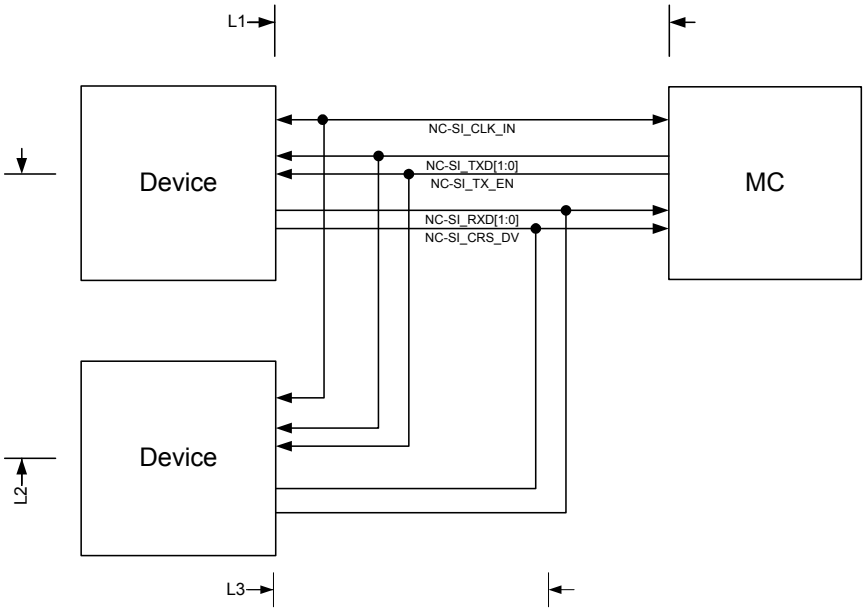



**Figure 12-4 Example 2-Drop Topology**

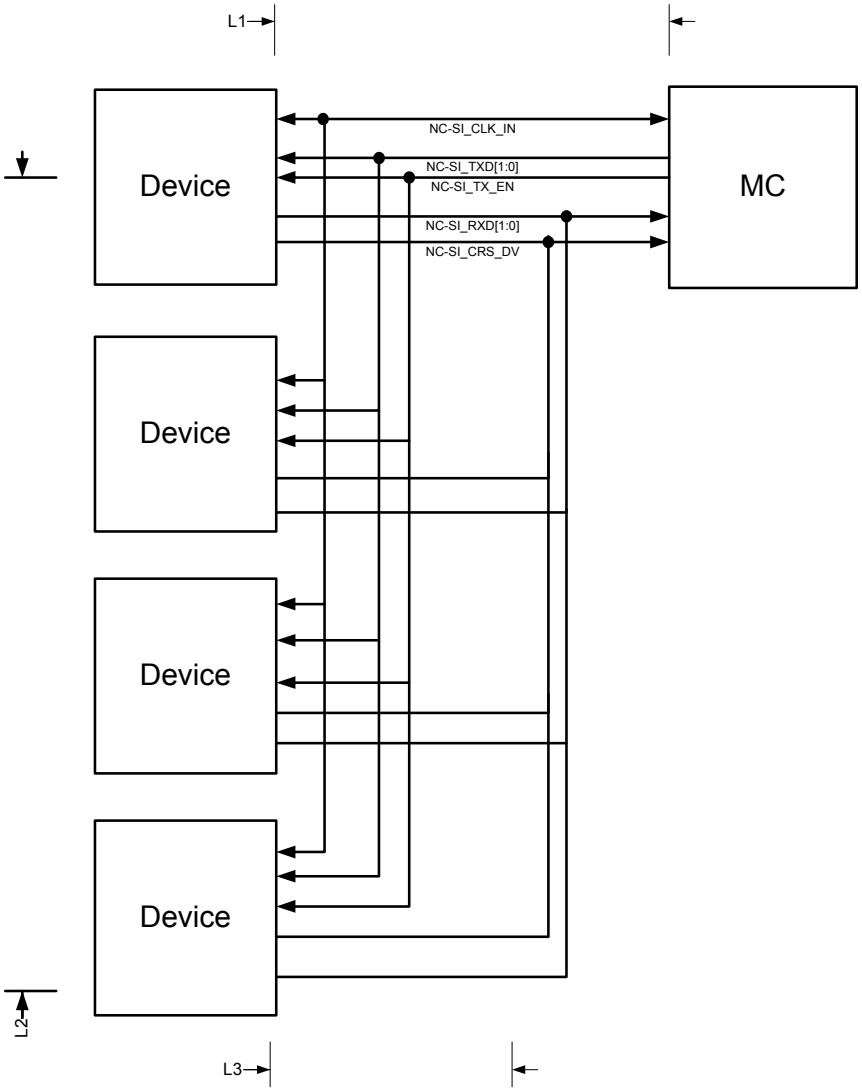



**Figure 12-5 Example 4-Drop Topology**

**Table 12-5 Compliant NC-SI Maximum Length on a 50 Ω -10% Skew-board with Example Stack-up**

| Topology | Total maximum compliant linear bus size (inches) | Number of vias | Approximate Net trace capacitance minus load capacitance (pf) |
|---|---|---|---|
| 4 multi-drop | 6.0 | 1 | 8.3 |
| 4 multi-drop | 5.5 | 8 | 8.3 |
| 2 multi-drop | 8.0 | 1 | 11.1 |
| 2 multi-drop | 7.5 | 8 | 11.1 |

**Table 12-5 Compliant NC-SI Maximum Length on a 50 Ω -10% Skew-board with Example Stack-up (Continued)**

| Topology | Total maximum compliant linear bus size (inches) | Number of vias | Approximate Net trace capacitance minus load capacitance (pf) |
|---|---|---|---|
| Point to point | 9.0 | 1 | 12.5 |
| Point to point | 8.5 | 8 | 12.5 |

Extending NC-SI to a maximum 11ns rise time increases the maximum trace length.

**Table 12-6 Functional NC SI maximum length on a 50 Ω -10% skew board with Example Stack-up (based on actual lab-measured solution)**

| Topology | Total maximum functional linear bus size (inches) | Number of vias | Approximate Net trace capacitance minus load capacitance (pf) |
|---|---|---|---|
| 4 multi-drop | 19 | 1 | 26.4 |
| 4 multi-drop | 18 | 8 | 26.4 |
| 2 multi-drop | 20 | 1 | 27.8 |
| 2 multi-drop | 19 | 8 | 27.8 |
| Point to point | 22 | 1 | 30.6 |
| Point to point | 21 | 8 | 30.6 |

# 12.7 Resets

After power is applied, the the 82599 must be reset. There are two ways to do this:

1. Using the internal power on reset circuit.
2. Using the external LAN_PWR_GOOD signal.

By default, the internal power on reset will reset the 82599.

If the design relies on the internal power on reset, then the power supply sequencing timing requirement between the 3.3V and 1.2V power rails has to be met. If this requirement is impossible to meet, the alternative is to bypass the internal power on reset circuit by pulling POR_BYPASS high and using an external power monitoring solution to provide a LAN_PWR_GOOD signal.

For LAN_PWR_GOOD timing requirements, see Section 4.0 and Section 5.0.

**Table 12-7 Reset Context for POR_BYPASS and LAN_PWR_GOOD**

| POR_BYPASS | Active Reset Circuit | |
|---|---|---|
| If = 0b | Internal POR | |
| If = 1b | External Reset | |

**Table 12-7 Reset Context for POR_BYPASS and LAN_PWR_GOOD**

| | LAN_PWR_GOOD | |
|---|---|---|
| | If = 0b | Held in reset. |
| | If = 1b | Initialized, ready for normal operation. |

It is important to ensure that the resets for the MC and the 82599 are generated within a specific time interval. The important requirement here is ensuring that the NC-SI link is established within two seconds of the MC receiving the power good signal from the platform. Both the 82599 and the external MC need to receive power good signals from the platform within one second of each other.

This causes an internal power on reset within the 82599 and then initialization as well as a triggering and initialization sequence for the MC. Once these power good signals are received by both the 82599 and the external MC, the NC-SI interface can be initialized. The NC-SI specification calls out a requirement of link establishment within two seconds. The MC should poll this interface and establish a link for two seconds to ensure specification compliance.

## 12.8 Connecting the MDIO Interfaces

The the 82599 provides one MDIO interface for each LAN port used as configuration interface for an external PHY attached to the 82599.

Connect the MDIO and MDC signals to the corresponding pins on the PHY chip. Make sure to provide a pull-up resistor to 3.3 V on the MDIO signal.

## 12.9 Connecting the Software-Definable Pins (SDPs)

The 82599 has eight SDPs per port that can be used for miscellaneous hardware or software-controllable purposes. These pins and their function are bound to a specific LAN device. The pins can each be individually configured to act as either input or output pins via EEPROM. The initial value in case of an output can also be configured in the same way. However, the silicon default for any of these pins is to be configured as outputs.

To avoid signal contention, all eight pins are set as input pins until after EEPROM configuration has been loaded.

Choose the right software definable pins for your applications keeping in mind that two of the eight pins (SDPx_6 and SDPx_7) are open drain. The rest are tri-state buffers. Consider that four of these pins (SDPx_0 – SDPx_3) can be used as General Purpose Interrupt (GPI) inputs. To act as GPI pins, the desired pins must be configured as inputs. A separate GPI interrupt-detection enable is then used to enable rising-edge detection of the input pin (rising-edge detection occurs by comparing values sampled at 62.5 MHz, as opposed to an edge-detection circuit). When detected, a corresponding GPI interrupt is indicated in the Interrupt Cause register.

When connecting the SDPs to different digital signals, please keep in mind that these are 3.3 V signals and use level shifting if necessary.

The use, direction, and values of SDPs are controlled and accessed using fields in the Extended SDP Control (ESDP) and Extended OD SDP Control (EODSDP) registers.

# 12.10 Connecting the Light Emitting Diodes (LEDs)

The 82599 provides four programmable high-current push-pull (active high) outputs per port to directly drive LEDs for link activity and speed indication. Each LAN device provides an independent set of LED outputs; these pins and their function are bound to a specific LAN device. Each of the four LED outputs can be individually configured to select the particular event, state, or activity, which is indicated on that output. In addition, each LED can be individually configured for output polarity, as well as for blinking versus non-blinking (steady-state) indication.

The LED ports are fully programmable through the EEPROM interface (LEDCTL register). In addition, the hardware-default configuration for all LED outputs can be specified via an EEPROM field, thus supporting LED displays configurable to a particular OEM preference.

Provide separate current limiting resistors for each LED connected.

Since the LEDs are likely to be placed close to the board edge and to external interconnect, take care to route the LED traces away from potential sources of EMI noise. In some cases, it might be desirable to attach filter capacitors.

# 12.11 Connecting Miscellaneous Signals

## 12.11.1 LAN Disable

The the 82599 has two signals that can be used for disabling Ethernet functions from system BIOS. LAN0_DIS_N and LAN1_DIS_N are the separated port disable signals. Each signal can be driven from a system output port. Choose outputs from devices that retain their values during reset. For example, some ICH GPIO outputs transition high during reset. It is important not to use these signals to drive LAN0_DIS_N or LAN1_DIS_N because these inputs are latched upon the rising edge of PE_RST_N or an in-band reset end.

A LAN port can also be disabled through EEPROM settings. See Section 4.2, Reset Operation for details.

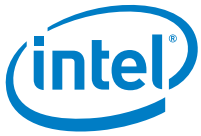


**Table 12-8 PCI Functions Mapping (Legacy Mode)**

| PCI Function # | LAN Function Select | Function 0 | Function 1 |
|---|---|---|---|
| Both LAN functions are enabled | 0 | LAN 0 | LAN 1 |
| | 1 | LAN 1 | LAN 0 |
| LAN 0 is disabled | x | LAN1 | Disable |
| LAN 1 is disabled | x | LAN 0 | Disable |
| Both LAN functions are disabled | Both PCI functions are disabled. 82599 is in low power mode. | | |

**Table 12-9 PCI Functions Mapping (Dummy Function Mode)**

| PCI Function # | LAN Function Select | Function 0 | Function 1 |
|---|---|---|---|
| Both LAN functions are enabled | 0 | LAN 0 | LAN 1 |
| | 1 | LAN 1 | LAN 0 |
| LAN 0 is disabled | 0 | Dummy | LAN1 |
| | 1 | LAN 1 | Disable |
| LAN 1 is disabled | 0 | LAN 0 | Disable |
| | 1 | Dummy | LAN 0 |
| Both LAN functions are disabled | Both PCI functions are disabled. 82599 is in low power mode. | | |

When both LAN ports are disabled following a POR / LAN_PWR_Good/ PE_RST_N/ in-band reset, the LAN_DIS_N signals should be tied statically to low. At this state the 82599 is disabled, LAN ports are powered down, all internal clocks are off and the PCIe connection is powered down (similar to L2 state).

## 12.11.2 BIOS Handling of Device Disable

Assume that in the following power up sequence the LANx_DIS_N signals are driven high (or is already disabled):

1. PCIe link is established following the PE_RST_N.
2. BIOS recognizes that the 82599 should be disabled.
3. BIOS drives the LANx_DIS_N signals to the low level.
4. BIOS issues PE_RST_N or an in-band PCIe reset.
5. As a result, the 82599 samples the LANx_DIS_N signals and enters the desired device-disable mode.
6. Re-enable could be done by driving high one of the LANx_DIS_N signals and then issuing a PE_RST_N to restart the 82599.

# 12.12 Oscillator Design Considerations

This section provides information regarding oscillators for use with the the 82599.

All designs require an external clock. There are two options for this clock source: a 25 MHz differential clock or a 25 MHz crystal. The the 82599 uses the clock source to generate clocks with frequency up to 3.125 GHz for the high speed interfaces.

The chosen oscillator or crystal vendor should be consulted early in the design cycle. Oscillator and crystal manufacturers familiar with networking equipment clock requirements can provide assistance in selecting an optimum, low-cost solution.

## 12.12.1 Oscillator Types

### 12.12.1.1 Fixed Crystal Oscillator

A packaged fixed crystal oscillator comprises an inverter, a quartz crystal, and passive components. The device renders a consistent square wave output. Oscillators used with microprocessors are supplied in many configurations and tolerances.

Crystal oscillators can be used in special situations, such as shared clocking among devices. As clock routing can be difficult to accomplish, it is preferable to provide a separate crystal for each device.

Recommended crystals are:

**Table 12-10 Part Numbers for Recommended Crystals**

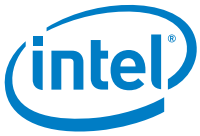
| | |
|---|---|
| Raltron | AS-25.000-20-SMD-TR-NS7 |
| TXC | 9C25000551 |

### 12.12.1.2 Programmable Crystal Oscillators

A programmable oscillator can be configured to operate at many frequencies. The device contains a crystal frequency reference and a Phase Lock Loop (PLL) clock generator. The frequency multipliers and divisors are controlled by programmable fuses.

PLLs are prone to exhibit frequency jitter. The transmitted signal can also have considerable jitter even with the programmable oscillator working within its specified frequency tolerance. PLLs must be designed carefully to lock onto signals over a reasonable frequency range. If the transmitted signal has high jitter and the receiver's PLL loses its lock, then bit errors or link loss can occur.

PHY devices are deployed for many different communication applications. Some PHYs contain PLLs with marginal lock range and cannot tolerate the jitter inherent in data transmission clocked with a programmable oscillator. The American National Standards Institute (ANSI) X3.263-1995 standard test method for transmit jitter is not stringent enough to predict PLL-to-PLL lock failures. Therefore, use of programmable oscillators is generally not recommended.



### 12.12.2 Oscillator Solution

Choose a clock oscillator with a PECL or CML output. When connecting the output of the oscillator to an the 82599, use the layout information shown in Figure 11-14. Also, make sure the oscillator meets the electrical characteristics listed in Table 11-25. Note that the EuroQuartz 3HPW5761-A-25 25MHz PECL Output Crystal-Controlled Oscillator has been used successfully in 82599-based designs.

### 12.12.3 Oscillator Layout Recommendations

Oscillators should not be placed near I/O ports or board edges. Noise from these devices can be coupled onto the I/O ports or out of the system chassis. Oscillators should also be kept away from network interface differential pairs to prevent interference.

The reference clock should be routed differentially; use the shortest, most direct traces possible. Keep potentially noisy traces away from the clock trace. It is critical to place the termination resistors and AC coupling capacitors as close to the the 82599 as possible (less than 250 mils).

### 12.12.4 Reference Clock Measurement Recommendations

A low capacitance, high impedance probe (C < 1 pF, R > 500 K) should be used for testing. Probing parameters can affect the measurement of the clock amplitude and cause errors in the adjustment. A test should be done after the probe has been removed to ensure circuit operation.

## 12.13 Power Supplies

The 82599 requires two power rails: 3.3 V and 1.2 V. A central power supply can provide all the required voltage sources; or power can be derived from the 3.3 V supply and regulated locally using an external regulator. If the LAN wake capability is used, voltages must remain present during system power down. Local regulation of the LAN voltages from system 3.3 Vmain and 3.3 Vaux voltages is recommended.

Make sure that all the external voltage regulators generate the proper voltage, meet the output current requirements (with adequate margin), and provide the proper power sequencing. See Section 11.0.

## 12.13.1 Power Supply Sequencing

Due to the current demand, a Switching Voltage Regulator (SVR) is highly recommended for the 1.2 V power rail. Regardless of the type of regulator used, all regulators need to adhere to the sequencing shown in Section 11.0 of this document to avoid latch-up and forward-biased internal diodes (1.2 V must not exceed 3.3 V).

The power supplies are all expected to ramp during a short power-up interval (recommended interval 20 ms or faster). Do not leave the 82599 in a prolonged state where some, but not all, voltages are applied.

### 12.13.1.1 Using Regulators With Enable Pins

The use of regulators with enable pins is very helpful in controlling sequencing. Connecting the enable of the 1.2 V regulator to 3.3 V ensures that the 1.2 V rail ramps after the 3.3V rail. This provides a quick solution to power sequencing. Make sure to check design parameters for inputs with this configuration. Alternatively, power monitoring chips can be used to provide the proper sequencing by keeping the voltage regulators with lower output in shutdown until the one immediately above doesn't reach a certain output voltage level.

## 12.13.2 Power Supply Filtering

Provide several high-frequency bypass capacitors for each power rail (Table 12-11), selecting values in the range of 0.001 µF to 0.1 µF. If possible, orient the capacitors close to the 82599 and adjacent to power pads.

Traces between decoupling and I/O filter capacitors should be as short and wide as practical. Long and thin traces are more inductive and would reduce the intended effect of decoupling capacitors. Also for similar reasons, traces to I/O signals and signal terminations should be as short as possible. Vias to the decoupling capacitors should be sufficiently large in diameter to decrease series inductance.

**Table 12-11 Minimum Number of Bypass Capacitors per Power Rail**

| Power Rail | Total Bulk Capacitance | 1.0 µF | 0.1 µF | 0.001 µF |
|---|---|---|---|---|
| 3.3 V | 44 µF | 0 | 8 | 0 |
| 1.2 V | 132 µF | 6 | 36 | 12 |

### 12.13.3 Support for Power Management and Wake Up

A designer must connect the MAIN_PWR_OK and the AUX_PWR signals on the board. These are digital inputs to the the 82599 and serve the following purpose:

MAIN_PWR_OK signals the the 82599 controller that the main power from the system is up and stable. For example, it could be pulled up to the 3.3V main rail or connected to a power well signal available in the system.

When sampled high, AUX_PWR indicates that auxiliary power is available to the 82599, and therefore it advertises D3cold wake up support. The amount of power required for the function, which includes the entire network interface card, is advertised in the Power Management Data register, which is loaded from the EEPROM.

If wake-up support is desired, AUX_PWR needs to be pulled high and the appropriate wake-up LAN address filters must also be set. The initial power management settings are specified by EEPROM bits. When a wake-up event occurs, the 82599 asserts the PE_WAKEn signal to wake the system up. PE_WAKEn remains asserted until PME status is cleared in the the 82599 Power Management Control/Status Register.

## 12.14 Connecting the JTAG Port

The the 82599 contains a test access port (3.3 V only) conforming to the IEEE 1149.1-2001 Edition (JTAG) specification. To use the test access port, connect these balls to pads accessible by your test equipment.

For proper operation, a pull-down resistor should be connected to the JTCK and JRST_N signals and pull-up resistors to the JTDO, JTMS and JTDI signals.

A Boundary Scan Definition Language (BSDL) file describing the the 82599 10 Gigabit Ethernet Controller device is available for use in your test environment.

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

# 13.0 Thermal Design Recommendations

This section provides a method for determining the operating temperature of the 82599 in a specific system based on case temperature. Case temperature is a function of the local ambient and internal temperatures of the component. This document specifies a maximum allowable Tcase for the 82599.

## 13.1 Thermal Considerations

In a system environment, the temperature of a component is a function of both the system and component thermal characteristics. System-level thermal constraints consist of the local ambient temperature at the component, the airflow over the component and surrounding board, and the physical constraints at, above, and surrounding the component that may limit the size of a thermal enhancement (heat sink).

The component's case/die temperature depends on:

- Component power dissipation
- Size
- Packaging materials (effective thermal conductivity)
- Type of interconnection to the substrate and motherboard
- Presence of a thermal cooling solution
- Power density of the substrate, nearby components, and motherboard

These parameters are pushed by increased performance levels (higher operating speeds, MHz) and power density (more transistors). As operating frequencies increase and packaging size decreases, the power density increases and the thermal cooling solution space and airflow become more constrained. The result is an increased emphasis on system design to ensure that thermal design requirements are met for each component in the system.

## 13.2 Importance of Thermal Management

The thermal management objective is to ensure that all system component temperatures are maintained within functional limits. The functional temperature limit is the range in which the electrical circuits are expected to meet specified performance. Operation outside the functional limit can degrade system performance, cause logic errors, or cause device and/or system damage. Temperatures exceeding the maximum operating limits may result in irreversible changes in the device operating characteristics. Note that sustained operation at component maximum temperature limit may affect long-term device reliability.

## 13.3 Packaging Terminology

The following terminology is used in this chapter:

- FCBGA Flip Chip Ball Grid Array: A surface-mount package using a combination of flip chip and BGA structure whose PCB-interconnect method consists of solder ball array on the interconnect side of the package. The die is flipped and connected to an organic build-up substrate with C4 bumps. An integrated heat spreader (IHS) may be present for larger FCBGA packages for enhanced thermal performance (but IHS is not present for the 82599).
- Junction: Refers to a P-N junction on the silicon. In this document, it is used as a temperature reference point (for example, JA refers to the “junction” to ambient thermal resistance).
- Ambient: Refers to local ambient temperature of the bulk air approaching the component. It can be measured by placing a thermocouple approximately 1”inch upstream from the component edge.
- Lands: The pads on the PCB to which BGA balls are soldered.
- PCB: Printed circuit board.
- Printed Circuit Assembly (PCA): An assembled PCB.
- Thermal Design Power (TDP): The estimated maximum possible/expected power generated in a component by a realistic application. Use Maximum power requirements listed in Table 13-2.
- LFM: Linear feet per minute (airflow).
- JA (Theta JA): Thermal resistance junction-to-ambient, °C/W.
- $\Psi_{JT}$ (Psi JT): Junction-to-top (of package) thermal characterization parameter, °C/W. ΨJT does not represent thermal resistance, but instead is a characteristic parameter that can be used to convert between Tj and Tcase when knowing the total TDP. $\Psi_{JT}$ is easy to characterize in simulations or measurements, and is equal to Tj minus Tcase divided by the total TDP. This parameter can vary by environment conditions like heat sink and airflow.

# 13.4 Thermal Specifications

To ensure proper operation of the 82599, the thermal solution must maintain a case temperature at or below the values specified in Table 13-1. System-level or component-level thermal enhancements are required to dissipate the generated heat to ensure the case temperature never exceeds the maximum temperatures, listed in Table 13-2. Table 13-1 lists the thermal performance parameters per JEDEC JESD51-2 standard. In Table 13-1 the $\theta_{JA}$ values should be used as reference only and can vary by system environment. $\Psi_{JT}$ values also can vary by system environment, and are given in Table 13-1 as the maximum value for the 82599 simulations.

Analysis indicates that real applications are unlikely to cause the 82599 to be at Tcase-max for sustained periods of time. Given that Tcase should reasonably be expected to be a distribution of temperatures, sustained operation at Tcase-max may affect long-term reliability of the 82599 and the system, and sustained operation performance at Tcase-max should be evaluated during the thermal design process and steps taken to further reduce the Tcase temperature.

Good system airflow is critical to dissipate the highest possible thermal power. The size and number of fans, vents, and/or ducts, and, their placement in relation to components and airflow channels within the system determine airflow. Acoustic noise constraints may limit the size and types of fans, vents and ducts that can be used in a particular design.

To develop a reliable, cost-effective thermal solution, all of the system variables must be considered. Use system-level thermal characteristics and simulations to account for individual component thermal requirements.

**Table 13-1 Package Thermal Characteristics in Standard JEDEC Environment**

| Package | $\theta_{JA}$ (°C/W) | $\Psi_{JT}$ (°C/W) |
|---|---|---|
| 25 mm FCBGA without IHS[1] | 22.6[4] | 0.54[6] |
| 25 mm FCBGA without IHS –HS (7.11mm height)[2] | 14.5[5] | 0.54 |
| 25 mm FCBGA without IHS –HS (11.43mm height)[3] | 11.3[5] | 0.54 |

*Notes:*

1. Integrated Heat Spreader (the 82599 is Bare die).
2. Heat sink with low profile 7.11 mm
3. Heat sink with high profile 11.43 mm
4. Integrated Circuit Thermal Measurement Method-Electrical Test Method EIA/JESD51-1, Integrated Circuits Thermal Test Method Environmental Conditions — Natural Convection (Still Air), No Heat sink attached EIAJESD51-2.
5. Natural Convection (Still Air), Heat sink attached.
6. $\Psi_{JT}$ is given as maximum value for a worst-case the 82599 scenario, and might vary to a lesser values in some scenarios

**Table 13-2 Absolute Thermal Maximum Rating (°C)**

| Application | TDP Power (W)[1] | Tcase Max-hs[2] (°C) |
|---|---|---|
| Intel® 82599 10 GbE Controller | 7.1 @ 123 °C Tj_max | 119 |

*Notes:*

1. Maximum power, also known as Thermal Design Power (TDP), is a system design target associated with the maximum component operating temperature specifications. Maximum power values are determined based on typical DC electrical specification and maximum ambient temperature for a worst-case realistic application running at maximum utilization.
2. Tcase Max-hs is defined as the maximum case temperature at TDP conditions. Tcase Max-hs is dictated by Tj_max equals to 123 °C.

Thermal parameters defined above are based on simulated results of packages assembled on standard multi layer 2s2p 1.0-oz Cu layer boards in a natural convection environment. The maximum case temperature is based on the maximum junction temperature and defined by the relationship, maximum Tcase = Tjmax — ($_{JT}$ x Power) where $_{JT}$ is the junction-to-top (of package) thermal characterization parameter. If the case temperature exceeds the specified Tcase max, thermal enhancements such as heat sinks or forced air will be required. $_{JA}$ is the thermal resistance junction-to-ambient of the package.

# 13.5 Case Temperature

The 82599 is designed to operate properly as long as Tcase rating is not exceeded. This chapter discusses proper guidelines for measuring the case temperature.

# 13.6 Thermal Attributes

## 13.6.1 Designing for Thermal Performance

Section 13.14 and Section 13.15 provide system design recommendations required to optimize product line thermal performance.

## 13.6.2 Model System Definition

A system with the following attributes was used to generate thermal characteristics data:

- Heatsink case described in Section 13.9.
- Six-layer, 4.5 x 4 inch PCB.

**Note:** All data is preliminary and is not validated against physical samples. Your system design may be significantly different.A larger board with more than six copper layers may improve thermal performance.

## 13.6.3 Package Thermal Characteristics

See Table 13-3 to determine the optimum airflow and heatsink combination for the 82599. Figure 13-1 shows the required ambient temperature versus airflow for a typical 82599 system.

Table 13-3 shows Tcase as a function of airflow and ambient temperature at the Thermal Design Power (TDP) for a typical 82599 system. Your system design may vary from the typical system board environment used to generate the values.

**Note:** Thermal models are available upon request (Flotherm*: 2-Resistor, Delphi, or Detailed). Contact your local Intel sales representative for product line thermal models.

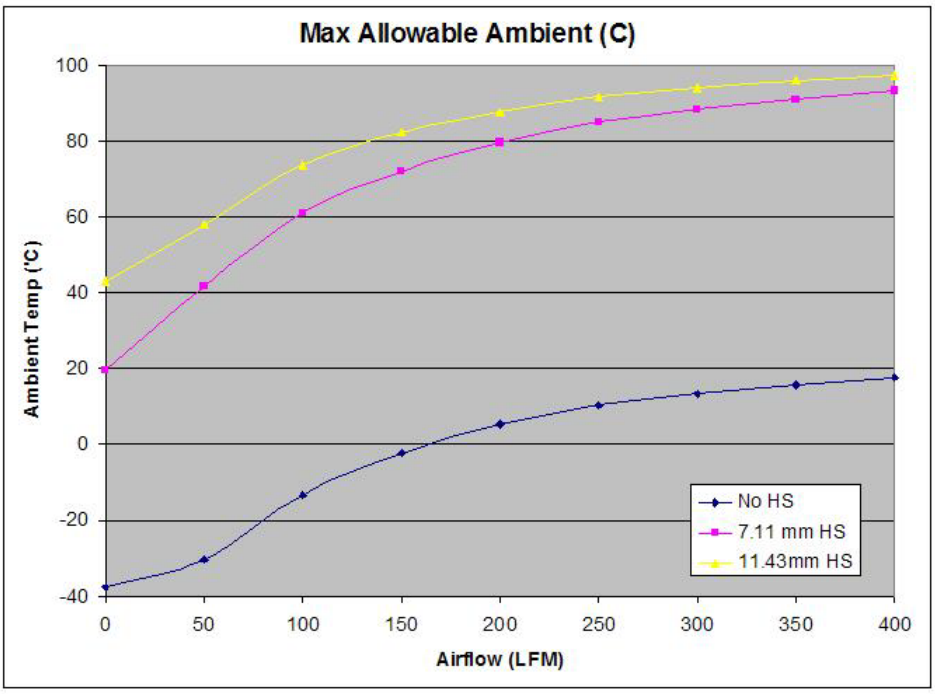



**Figure 13-1 82599 Max Allowable Ambient @ 8.82W (JEDEC Card)**

**Table 13-3 82599 Expected Tcase (°C) for Two Heat Sinks at 8.82 W (JEDEC Card)**

| | Case Temperature (Max = 119C) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **11.43 mm High Heat Sink** | | | | | | | | |
| | | Airflow (LFM) | | | | | | | |
| | | 0 | 50 | 100 | 150 | 200 | 250 | 300 | 350 | 400 |
| Ambient Temp (°C) | 45 | 122.8 | 104.2 | 91.84 | 83.02 | 77.09 | 73.2 | 70.68 | 68.84 | 67.41 |
| | 50 | 129.2 | 107.35 | 96.42 | 87.855 | 82.07 | 78.2 | 75.68 | 73.835 | 72.405 |
| | 55 | 135.6 | 110.5 | 101 | 92.69 | 87.05 | 83.2 | 80.68 | 78.83 | 77.4 |
| | 60 | 141.7 | 115.35 | 105.4 | 97.545 | 92.015 | 88.2 | 85.675 | 83.82 | 82.39 |
| | 65 | 147.8 | 120.2 | 109.8 | 102.4 | 96.98 | 93.2 | 90.67 | 88.81 | 87.38 |
| | 70 | 153.1 | 125.05 | 112.9 | 107.2 | 101.94 | 98.2 | 95.685 | 93.81 | 92.38 |
| | 75 | 158.4 | 129.9 | 116 | 112 | 106.9 | 103.2 | 100.7 | 98.81 | 97.38 |
| | 80 | 162.15 | 134.65 | 120.75 | 115.35 | 111.85 | 108.2 | 105.7 | 103.805 | 102.39 |
| | 85 | 165.9 | 139.4 | 125.5 | 118.7 | 116.8 | 113.2 | 110.7 | 108.8 | 107.4 |
| | | | | | | | | | | |
| | **7.11 mm High Heat Sink** | | | | | | | | |
| | | Airflow (LFM) | | | | | | | |
| | | 0 | 50 | 100 | 150 | 200 | 250 | 300 | 350 | 400 |
| Ambient Temp (°C) | 45 | 149.7 | 124.4 | 104.7 | 93.65 | 85.72 | 80.19 | 76.51 | 73.73 | 71.59 |
| | 50 | 154.7 | 129.1 | 109.2 | 98.47 | 90.63 | 85.15 | 81.49 | 78.73 | 76.59 |
| | 55 | 158.7 | 133.8 | 113.7 | 103.2 | 95.7 | 90.09 | 86.56 | 83.73 | 81.59 |
| | 60 | 162 | 138.6 | 120 | 107.9 | 100.6 | 95.25 | 91.56 | 88.73 | 86.58 |
| | 65 | 166 | 143.6 | 124.7 | 112.7 | 105.6 | 100.1 | 96.51 | 93.72 | 91.58 |
| | 70 | 169.8 | 148.6 | 129.5 | 117.5 | 110.5 | 105.1 | 101.5 | 98.72 | 96.58 |
| | 75 | 173.9 | 153.7 | 134.2 | 122.9 | 115.4 | 110.1 | 106.5 | 103.7 | 101.6 |
| | 80 | 177.3 | 158.7 | 138.9 | 127.9 | 120.2 | 115 | 111.5 | 108.7 | 106.6 |
| | 85 | 179.6 | 163.7 | 143.6 | 132.8 | 125.5 | 120 | 116.4 | 113.7 | 111.6 |

**Note:** The Orange blocked value(s) indicate airflow/ambient combinations that exceed the allowable case temperature for the 82599 at 8.82 W.

# 13.7 Thermal Enhancements

One method frequently used to improve thermal performance is to increase the device surface area by attaching a metallic heatsink to the component top. Increasing the surface area of the heatsink reduces the thermal resistance from the heatsink to the air, increasing heat transfer.

# 13.8 Clearances

A heatsink should have a pocket of air around it that is free of obstructions. Though each design may have unique mechanical restrictions, the recommended clearances for a heatsink used with the 82599 are shown in Figure 13-2 assuming one of the 40 x 40mm reference heat sinks is selected. Retention clip selection is open, and example keep-outs and board through holes are given in Figure 13-2 and Figure 13-3 for a torsion retention clip.

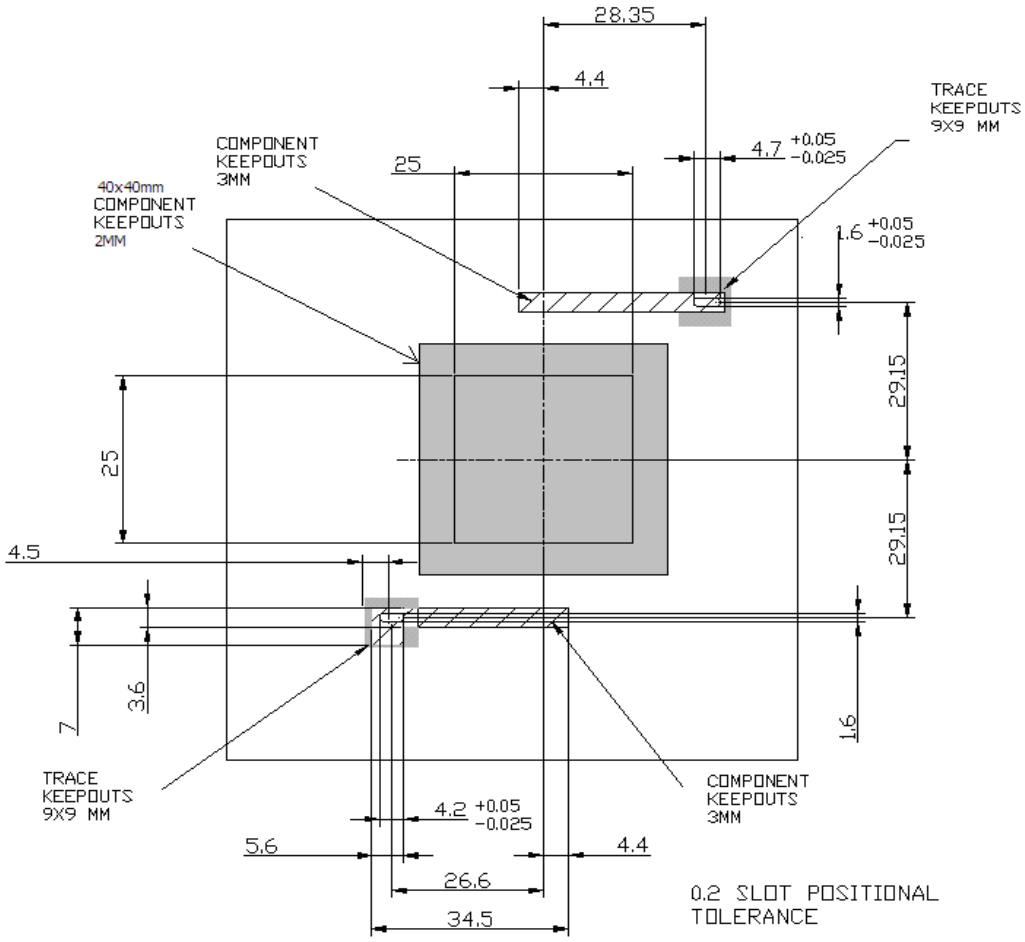



**Figure 13-2 Heatsink Keep-out Restrictions**

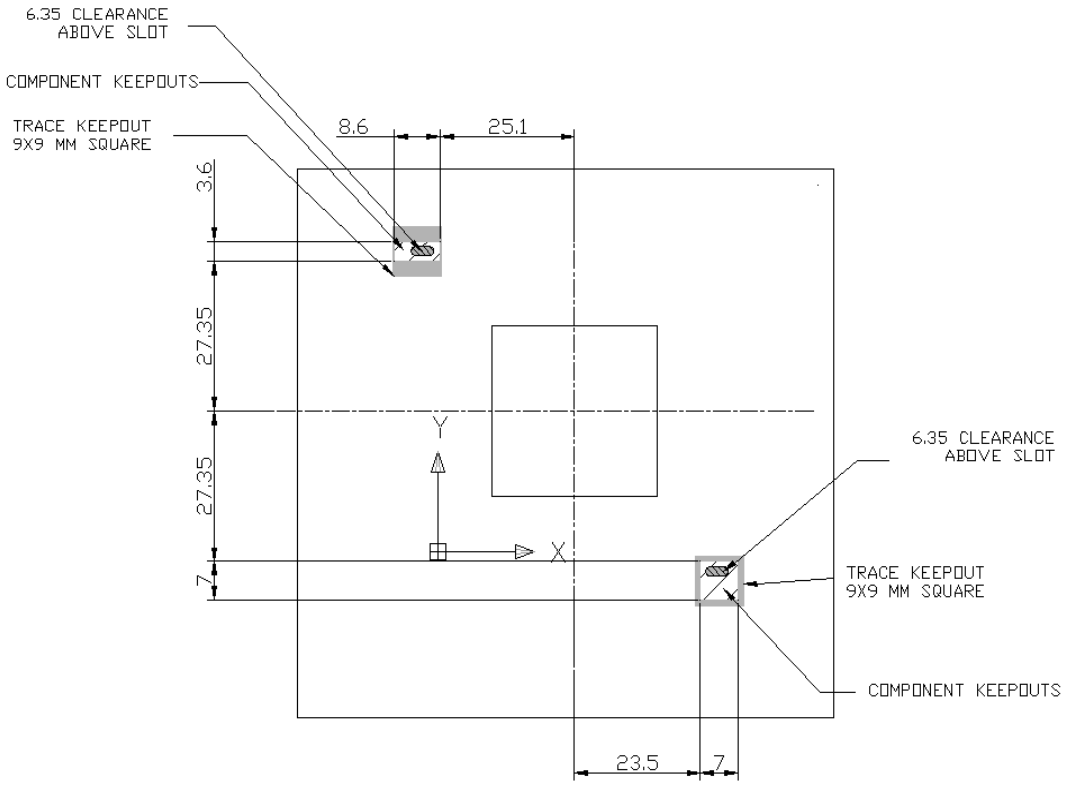



**Figure 13-3 Bottom Side Keep-out**

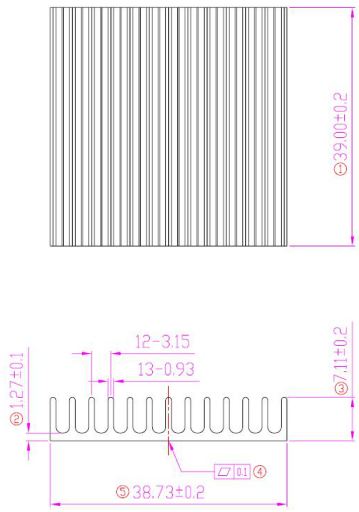
# 13.9 Default Enhanced Thermal Solution

If you have no control over the end-user's thermal environment or you wish to bypass the thermal modeling and evaluation process, use the Default Enhanced Thermal Solution (see Figure 13-4).

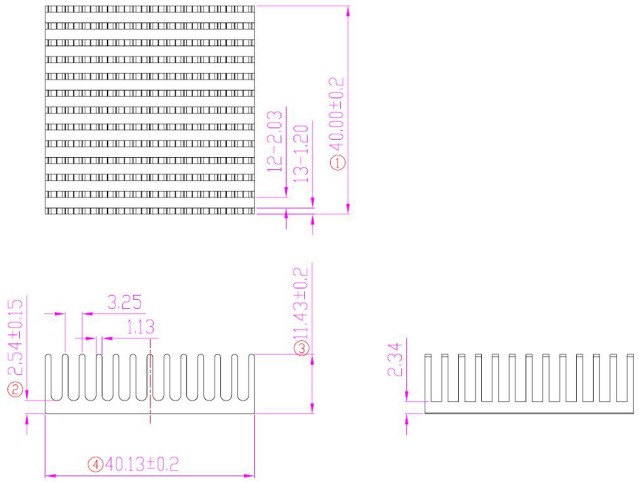
If, after implementing the recommended enhanced thermal solution, the case temperature continues to exceed allowable values, then additional cooling is needed. This additional cooling may be achieved by improving airflow to the component and/or adding additional thermal enhancements.

# 13.10 Extruded Heatsinks

If required, the following extruded heatsinks are the suggested for the 82599 thermal solutions. Other equivalent heatsinks and their sources are provided in Section 13.14.




Low Profile 7.11 mm Height Passive Heat Sink




High Profile 11.43 mm Height Passive Heat Sink

**Figure 13-4 Extruded Heatsinks**

# 13.11 Attaching the Extruded Heatsink

The extruded heatsink may be attached using clips with a phase change thermal interface material.

## 13.11.1 Clips

A well-designed clip, in conjunction with a thermal interface material (tape, grease, etc.) often offers the best combination of mechanical stability and reworkability. Use of a clip requires significant advance planning as mounting holes are required in the PCB.

Use non-plated mounting with a grounded annular ring on the solder side of the board surrounding the hole. For a typical low-cost clip, set the annular ring inner diameter to 150 mils and an outer diameter to 300 mils. Define the ring to have at least eight ground connections. Set the solder mask opening for these holes with a radius of 300 mils.

## 13.11.2 Thermal Interface (PCM45 Series)

The recommended thermal interface is PCM45 Series from Honeywell. The PCM45 Series thermal interface pads are phase change materials formulated for use in high performance devices requiring minimum thermal resistance for maximum heat sink performance and component reliability. These pads consist of an electrically non-conductive, dry film that softens at device operating temperatures resulting in "greasy-like" performance. However, Intel has not fully validated the PCM45 Series TIM.

Each PCA, system and heatsink combination varies in attach strength. Carefully evaluate the reliability of tape attaches prior to high-volume use.

## 13.11.3 Avoid Damaging Die-Side Capacitors with Heat Sink Attached

Capacitors on the die side are not protected and can be damaged during heat sink attachment. If the heat sink is tilted from the die it is possible that the heat sink will make contact with the capacitors prior to making contact with the package substrate. Figure 13-5 shows how the capacitors can be exposed to heat sink contact by drawing a plane from the die edge to the substrate edge. Figure 13-6 shows an example of the damage caused by heat sink contact. It is recommended that heat sinks be attached vertically, with the heat sink bottom surface parallel to the die surface to avoid contact with the capacitors.

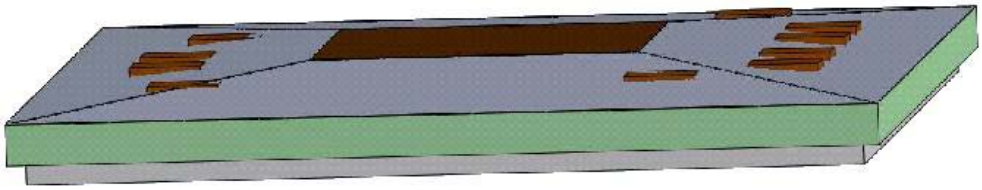


**Figure 13-5 Die-Side Capacitors Exposed to Heat Sink Contact**

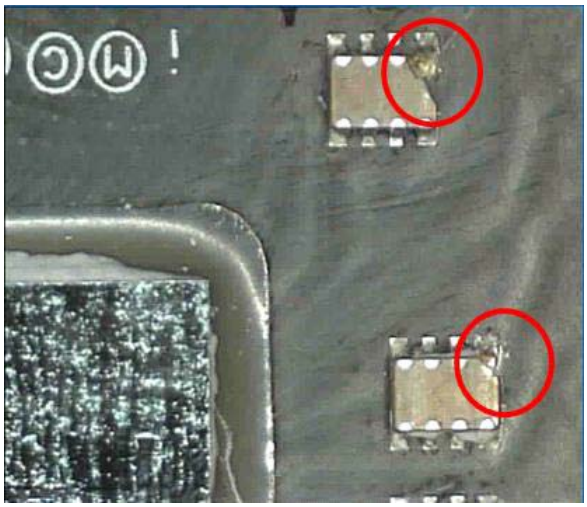


**Figure 13-6 Example for Damage Caused by Heat Sink Contact**

## 13.11.4 Maximum Static Normal Load

the 82599 package has a bare die that is capable of sustaining a maximum static normal load of 15 lbf (66.7 N). This load is a uniform compressive load in a direction perpendicular to the die top surface. This mechanical load limit must not be exceeded during heatsink installation, mechanical stress testing, standard shipping conditions, and/or any other use condition. Note that the heat sink attach solution must not include continuous stress to the package, with the exception of a uniform load to maintain the heatsink-to-package thermal interface. This load specification is based on limited testing for design characterization, and is for the package only.

# 13.12 Reliability

Each PCA, system and heatsink combination varies in attach strength and long-term adhesive performance. Carefully evaluate the reliability of the completed assembly prior to high-volume use. Some reliability recommendations are shown in Table 13-4.

**Table 13-4 Reliability Validation**

| Test[1] | Requirement | Pass/Fail Criteria[2] |
|---|---|---|
| Mechanical Shock | 50G trapezoidal, board level<br>11 ms, 3 shocks/axis | Visual and Electrical Check |
| Random Vibration | 7.3G, board level<br>45 minutes/axis, 50 to 2000 Hz | Visual and Electrical Check |
| High-Temperature Life | 85 °C<br>2000 hours total<br>Checkpoints occur at 168, 500, 1000, and 2000 hours | Visual and Mechanical Check |
| Thermal Cycling | Per-Target Environment<br>(for example: -40 °C to +85 °C)<br>500 Cycles | Visual and Mechanical Check |
| Humidity | 85% relative humidity<br>85 °C, 1000 hours | Visual and Mechanical Check |

1. Performed the above tests on a sample size of at least 12 assemblies from 3 lots of material (total = 36 assemblies)
2. Additional pass/fail criteria can be added at your discretion.

## 13.12.1 Thermal Interface Management for Heat-Sink Solutions

To optimize the 82599 heatsink design, it is important to understand the interface between the heat spreader and the heatsink base. Thermal conductivity effectiveness depends on the following:

- Bond line thickness
- Interface material area
- Interface material thermal conductivity

### 13.12.1.1 Bond Line Management

The gap between the heat spreader and the heatsink base impacts heat-sink solution performance. The larger the gap between the two surfaces, the greater the thermal resistance. The thickness of the gap is determined by the flatness of both the heatsink base and the heat spreader, plus the thickness of the thermal interface material (for example, PSA, thermal grease, epoxy) used to join the two surfaces.

#### 13.12.1.2 Interface Material Performance

The following factors impact the performance of the interface material between the heat spreader and the heatsink base:

- Thermal resistance of the material
- Wetting/filling characteristics of the material

#### 13.12.1.3 Thermal Resistance of the Material

Thermal resistance describes the ability of the thermal interface material to transfer heat from one surface to another. The higher the thermal resistance, the less efficient the heat transfer. The thermal resistance of the interface material has a significant impact on the thermal performance of the overall thermal solution. The higher the thermal resistance, the larger the temperature drop required across the interface.

#### 13.12.1.4 Wetting/Filling Characteristics of the Material

The wetting/filling characteristic of the thermal interface material is its ability to fill the gap between the heat spreader top surface and the heatsink. Since air is an extremely poor thermal conductor, the more completely the interface material fills the gaps, the lower the temperature-drop across the interface, increasing the efficiency of the thermal solution.

## 13.13 Measurements for Thermal Specifications

Determining the thermal properties of the system requires careful case temperature measurements. Guidelines for measuring the 82599 case temperature are provided in Section 13.13.1.

### 13.13.1 Case Temperature Measurements

Maintain the 82599 Tcase at or below the maximum case temperatures listed in Table 13-2 to ensure functionality and reliability. Special care is required when measuring the Tcase temperature to ensure an accurate temperature measurement. Use the following guidelines when making Tcase measurements:

- Measure the surface temperature of the case in the geometric center of the case top.
- Calibrate the thermocouples used to measure Tcase before making temperature measurements.
- Use 36-gauge (maximum) K-type thermocouples.

Care must be taken to avoid introducing errors into the measurements when measuring a surface temperature that is a different temperature from the surrounding local ambient air. Measurement errors may be due to a poor thermal contact between the thermocouple junction and the surface of the package, heat loss by radiation, convection, conduction through thermocouple leads, and/or contact between the thermocouple cement and the heat-sink base (if used).

## 13.13.2 Attaching the Thermocouple (No Heatsink)

The following approach is recommended to minimize measurement errors for attaching the thermocouple with no heatsink:

- Use 36-gauge or smaller-diameter K-type thermocouples.
- Ensure that the thermocouple has been properly calibrated.
- Attach the thermocouple bead or junction to the top surface of the package (case) in the center of the heat spreader using high thermal conductivity cements.

**Note:** It is critical that the entire thermocouple lead be butted tightly to the heat spreader.

Attach the thermocouple at a 0° angle if there is no interference with the thermocouple attach location or leads (see Figure 13-7). This is the preferred method and is recommended for use with packages not having a heat sink.

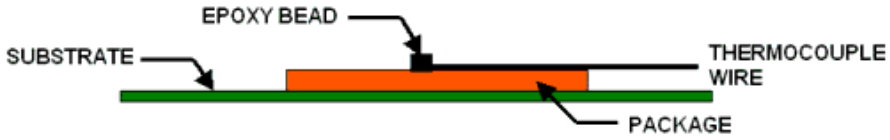



**Figure 13-7 Technique for Measuring Tcase with 0° Angle Attachment, No Heatsinks**

## 13.13.3 Attaching the Thermocouple (Heatsink)

The following approach is recommended to minimize measurement errors for attaching the thermocouple with heatsink:

- Use 36-gauge or smaller diameter K-type thermocouples.
- Ensure that the thermocouple is properly calibrated.
- Attach the thermocouple bead or junction to the case's top surface in the geometric center using a high thermal conductivity cement.

**Note:** It is critical that the entire thermocouple lead be butted tightly against the case.

- Attach the thermocouple at a 90° angle if there is no interference with the thermocouple attach location or leads (Figure 13-8). This is the preferred method and is recommended for use with packages with heatsinks.

- For testing purposes, a hole (no larger than 0.150 inches in diameter) must be drilled vertically through the center of the heatsink to route the thermocouple wires out.
- Ensure there is no contact between the thermocouple cement and heatsink base. Any contact affects the thermocouple reading.

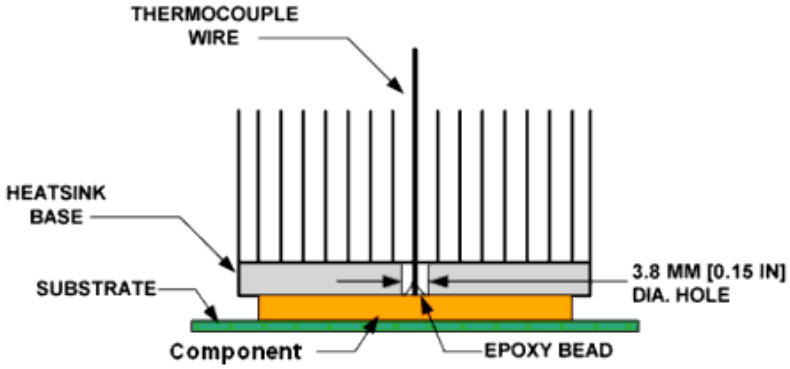



**Figure 13-8 Technique for Measuring Tcase with 90° Angle Attachment**

# 13.14 Heatsink and Attach Suppliers

**Table 13-5 Heatsink and Attach Suppliers**

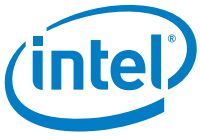
| Part | Part Number | Supplier | Contact |
|---|---|---|---|
| Extruded Al Heat sink<br>+ Clip<br>+ PCM45 (TIM)<br>Assembly | Generated specific to customer numbering scheme | Cooler Master | Eugene Lai<br>Cooler Master USA, INC. (Fremont)<br>Office: 510-770-8566 # 222<br>yuchin_lai@coolermaster.com |
| PCM45 Series | PCM45F | Honeywell | North America Technical Contact: Paula Knoll<br>1349 Moffett Park Dr.<br>Sunnyvale, CA 94089<br>Cell: 1-858-705-1274 Business: 858-279-2956<br>paula.knoll@honeywell.com |

# 13.15 PCB Guidelines

The following general PCB design guidelines are recommended to maximize thermal performance of FCBGA packages:

- When connecting ground (thermal) vias to the ground planes, do not use thermal-relief patterns.
- Thermal-relief patterns are designed to limit heat transfer between vias and the copper planes, thus constricting the heat flow path from the component to the ground planes in the PCB.
- As board temperature also has an effect on the thermal performance of the package, avoid placing the 82599 adjacent to high-power dissipation devices.
- If airflow exists, locate the components in the mainstream of the airflow path for maximum thermal performance. Avoid placing the components downstream, behind larger devices or devices with heat sinks that obstruct the air flow or supply excessively heated air.

**Note:** The previous information is provided as a general guideline to help maximize the thermal performance of the components.



# 14.0 Diagnostics

## 14.1 Link Loopback Operations

Loopback operations are supported by the 82599 to assist with system and device debug. Loopback operation can be used to test transmit and receive aspects of software drivers, as well as to verify electrical integrity of the connections between the 82599 and the system (such as PCIe bus connections, etc.). Loopback operation is supported as follows:

Internal Loopback:

- Tx->Rx MAC Loopback — This loopback is closed on the internal XGMII interface of the MAC core (Does not apply to PCS or analog cores).

To configure the 82599 for Tx->Rx loopback operation:

1. Configure the desire speed (10 GbE/1 GbE).
2. In AUTOC register (see Section 8.2.3.22.19) set the LMS field value to the desired speed (0x0 —
   1 GbE, 0x1 — 10 GbE) and set the *FLU* bit to 1b in order to force linkup. All other bits are a don't care. In the HLREG0 register (see Section 8.2.3.22.8), set the *LPBK* bit to 1b.

Link indication in register LINKS (see Section 8.2.3.22.20) should be ignored:

LINK configuration should be done as in regular functional mode (see Section 4.6.4). All LINK modes can be configured, but auto-negotiation should be disabled.

- Rx->Tx Loopback — This loopback is closed in the internal XGMII interface (covers analog and the enabled PCS blocks). Note that 10b/8b encoding is done through this loopback, so IDLE patterns might be different between the received and transmitted data.

For the loopback to be functional, a functional link (with a link partner) should be achieved (sync and alignment).

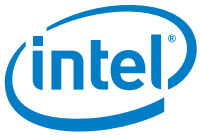
LINK configuration should be done as in regular functional mode (see Section 4.6.3), and all LINK modes can be configured.

Loopback limitations:

- Short preamble with minimal IPG is not supported with loopback operation.

**Note:** Transmitted data might violate the minimum IPG specification requirements.For more details on the 82599 loopback modes, refer to the *Intel® Ethernet Controllers Loopback Modes* Guide.

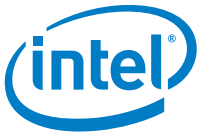
***NOTE:*** ***This page intentionally left blank.***



# 15.0 Glossary and Acronyms

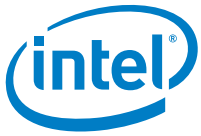
| Term | Definition |
|---|---|
| 1 KB | A value of 1 KB equals 1024 bytes. |
| 1’s complement | A system known as ones' complement can be used to represent negative numbers in a binary system. The ones' complement form of a negative binary number is the bitwise NOT applied to it. |
| 2’s complement | A system of two's-complement arithmetic represents negative integers by counting backwards and wrapping around. Any number whose left-most bit is 1 is considered negative. |
| 1000BASE-BX | 1000BASE-BX is the PICMG 3.1 electrical specification for transmission of 1 Gb/s Ethernet or 1 Gb/s fibre channel encoded data over the backplane. |
| 1000BASE-T | 1000BASE-T is the specification for 1 Gb/s Ethernet over category 5e twisted pair cables as defined in IEEE 802.3 clause 40. |
| 1000BASE-BX | 1000BASE-BX is the PICMG 3.1 electrical specification for transmission of 1 Gb/s Ethernet or 1 Gb/s Fibre Channel encoded data over the backplane. |
| 1000BASE-CX | 1000BASE-CX over specially shielded 150 Ω balanced copper jumper cable assemblies as specified in IEEE 802.3 Clause 39. |
| 10GBASE-BX4 | 10GBASE-BX4 is the PICMG 3.1 electrical specification for transmission of 10 Gb/s Ethernet or 10 Gb/s Fibre Channel encoded data over the backplane. |
| 10GBASE-CX4 | 10GBASE-CX4 over shielded 100 Ω balanced copper jumper cable assemblies as specified in IEEE 802.3 Clause 54. |
| AAD | Additional Authentication Data input, which is authenticated data that must be left un-encrypted. |
| ACK | Acknowledgement |
| ACPI | Advanced Configuration and Power Interface — ACPI reset is also known as D3hot-D0 transition. |
| AEN | Address Enable |
| AER | Advanced Error Reporting |
| AFE | Analog Front End |
| AH | IP Authentication Header — An IPsec header providing authentication capabilities defined in RFC 2402 For an example of an AH packet diagram see below:<br>• Next Header: Identifies the protocol of the transferred data.<br>• Payload Length: Size of AH packet.<br>• RESERVED: Reserved for future use (all zero until then).<br>• Security Parameters Index (SPI): Identifies the security parameters, which, in combination with the IP address, then identify the Security Association implemented with this packet.<br>• Sequence Number: Monotonically increasing number, used to prevent replay attack<br>• Authentication Data: Contains the integrity check value (ICV) necessary to authenticate the packet; it may contain padding. |
| AN | Auto negotiation |
| AN | Association Number |
| APIC | Advanced Programming Interrupt Controller |
| APM | Advanced Power Management |

| Term | Definition |
|---|---|
| APT | Advanced Pass Through mode |
| ARI | Alternative Routing ID capability structure– This is a new capability that allows an interpretation of the Device and Function fields as a single identification of a function within the bus. |
| ARP | Address Resolution Protocol |
| b/w or BW | Bandwidth |
| backbone | A bus shared by many clients for example a management backbone or a host backbone |
| BAR | Base Address Register |
| BCN | Backward Congestion Notification. |
| BCNA | BCN Address |
| BDF | Bus/Device/Function |
| BER | Bit Error Rate |
| BIOS | Basic Input/Output System. |
| BIST | Built-In Self Test |
| BKM | Best Known Method |
| BMC | Baseboard Management Controller |
| BME | Bus Master Enable |
| BT | Byte Time. |
| BYTE alignment | Implies that the physical addresses can be odd or even. Examples: 0FECBD9A1h, 02345ADC6h. |
| BWG | Bandwidth Group. |
| CA | Secure Connectivity Association (CA): A security relationship, established and maintained by key agreement protocols, that comprises a fully connected subset of the service access points in stations attached to a single LAN that are to be supported by LinkSec. |
| CAM | Content Addressable Memory |
| Ciphertext | Encrypted data, whose length is exactly that of the plaintext. |
| CFI | Canonical Form Indicator |
| CM-Tag | Congestion Management tag |
| concurrency | The concurrent (simultaneous) execution of multiple interacting computational tasks. These tasks may be implemented as separate programs, or as a set of processes or threads created by a single program. |
| corner case | A problem or situation that occurs only outside of normal operating parameters — specifically one that manifests itself when multiple environmental variables or conditions are simultaneously at extreme levels.<br>For example, a computer server may be unreliable, but only with the maximum complement of 64 processors, 512 GB of memory, and over 10,000 signed-on users. From Wiki. |
| CPID | Congestion Point Identifier –which should include the congestion point Ethernet MAC Address, as well as a local identifier for the local congestion entity, usually a queue in the switch. |
| CRC | Cyclic Redundancy Check<br>A cyclic redundancy check (CRC) is a type of function that takes as input a data stream of unlimited length and produces as output a value of a certain fixed size. The term CRC is often used to denote either the function or the function's output. A CRC can be used in the same way as a checksum to detect accidental alteration of data during transmission or storage. CRCs are popular because they are simple to implement in binary hardware, are easy to analyze mathematically, and are particularly good at detecting common errors caused by noise in transmission channels. From Wiki |
| CRS | Carrier Sense Indication. |
| CSMA/CD | 802.3 Carrier Sense Multiple Access / Collision Domain Ethernet LCI-2 Interface to an external LAN Connected Device to provide wired LAN connectivity. |



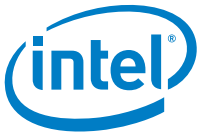
| Term | Definition |
|---|---|
| CSR | Control / Status Register |
| CTS | Cisco Trusted Security |
| D0a<br>D0 Active | Active fully operational state. Once memory space is enabled all internal clocks are activated and the LAN Controller enters an active state. |
| D0u<br>D0 uninitialized | The D0u state is a low-power state used after PCI Reset (SPXB Reset) is de-asserted following power-up (cold or warm), or on D3 exit. |
| D3Hot | In D3 the LAN Controller only responds to PCI configuration accesses and does not generate master cycles. |
| D3Cold | Power Off if Vcc is removed from the device and all of its PCI functions transition immediately to D3 cold. When power is restored a PCI Reset must be asserted. |
| Dr | Internal Power management state when minimal function is provided (WoL, Manageability) |
| DA | Destination Address |
| DAC | Digital to Analog Converter |
| DAC | Dual Address Cycle messages |
| Data Frame | FC Frames that carry read or write data. |
| DBU | Data Buffer Unit |
| DCA | Direct Cache Access |
| DCB | Data Center Bridging. |
| DCX | DCB Configuration Exchange protocol |
| DDP | Direct Data placement |
| DFT | Testability. |
| DFX | Design for * |
| DHCP | Dynamic Host Configuration Protocol (protocol for automating the configuration of computers that use TCP/IP) |
| DLLP | Data Link Layer Packet /PCIe |
| DMTF NC-SI | Distributed Management Task Force<br>BMC-NIC interconnect for management |
| DQ | Descriptor Queue. |
| DSP | Digital Signal Processor |
| DUT | Device Under Test |
| DWORD (Double-Word) alignment | Implies that the physical addresses may only be aligned on 4-byte boundaries; i.e., the last nibble of the address may only end in 0, 4, 8, or Ch. For example, 0FECBD9A8h. |
| EAPOL | Extensible Authentication Protocol over LAN |
| EAS | External Architecture Specification. |
| ECC | Error Correction Coding |
| ECRC | End to End CRC |
| EDB | End Data Bit |
| ECC | Error Correction Coding |
| EEPROM | Electrically Erasable Programmable Memory. A non-volatile memory located on the LAN controller that is directly accessible from the host. |
| EHS | External Heat Sink |
| EOP | End-Of-Packet; when set indicates the last descriptor making up the packet. |

| Term | Definition |
|---|---|
| EP | End point |
| ESN | Extended Sequence Number |
| E-SOF | FCoE Start of Frame |
| ESP | IP Encapsulating Security Payload — An IPsec header providing encryption and authentication capabilities defined in RFC 4303. The Encapsulating Security Payload (ESP) extension header provides origin authenticity, integrity, and confidentiality protection of a packet. ESP also supports encryption-only and authentication-only configurations, but using encryption without authentication is strongly discouraged. Unlike the AH header, the IP packet header is not accounted for. ESP operates directly on top of IP, using IP protocol number 50. ESP fields:<br>• Security Parameters Index (SPI): See AH<br>• Sequence Number: See AH<br>• Payload Data: See AH<br>• Padding: Used with some block ciphers to pad the data to the full length of a block.<br>• Pad Length: Size of padding in bytes.<br>• Next Header: Identifies the protocol of the transferred data.<br>• Authentication Data: Contains the data used to authenticate the packet. |
| EUI | IEEE defined 64-bit Extended Unique Identifier |
| Extension Header | IPv6 protocol. |
| Fail-over | fail-over is the ability to detect that the LAN connection on one port is lost, and enable the other port for traffic. |
| FC | Fiber Channel |
| FC | Flow Control. |
| FCoE | Fiber Channel over Ethernet |
| FC Exchange | Complete Fiber Channel Read or Fiber Channel Write flow. It starts with the read or write requests by the initiator (the host system) till the completion indication from the target (the remote disk). |
| FCS | Frame Check Sequence of Ethernet frames |
| FC Sequence | A Fiber Channel Exchange is composed of multiple Fiber Channel sequences. Fiber Channel Sequence can be a single or multiple frames that are sent by the initiator or the target. Each FC Sequence has a unique “Sequence ID”. |
| FC Frame | Fiber Channel Frames are the smallest units sent between the initiator and the target. The FC-FS-2 spec define the maximum frame size as 2112 bytes. Each Fiber Channel frame includes an FC header and optional FC payload. It can also may include Extended headers and FC optional headers. Extended headers are not expected in FCoE network and FC optional headers may not be used as well. |
| FCP_RSP Frame | Fiber Channel control Frames that are sent from the target to the initiator which defines the completion of an FC read or write exchange. |
| FEC | Forward Error Correction |
| FEXT | Far End Crosstalk |
| Firmware (FW) | Embedded code on the LAN controller that is responsible for the implementation of the NC-SI protocol and pass through functionality. |
| FLR | Function level reset An OS in a VM must have complete control over a device, including its initialization, without interfering with the rest of the functions. |
| FML | Fast Management Link |
| Fragment Header | An IPv6 extension Header |
| Frame | A unit composed of headers, data and footers that are sent or received by a device. Same as a Packet |
| FSM | Finite State Machine |
| FTS | Fast Training Sequence |
| GbE | Gigabit Ethernet (IEEE 802.3z-1998) |



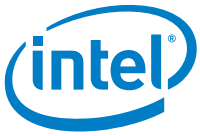
| Term | Definition |
|---|---|
| GMRP | GARP Multicast Registration Protocol (Cisco) |
| GPIO | General Purpose I/O |
| GSP | Group Strict Priority |
| HBA | Host Bus Adapters |
| Host Interface | RAM on the LAN controller that is shared between the firmware and the host. RAM is used to pass commands from the host to firmware and responses from the firmware to the host. |
| HPC | High — Performance Computing. |
| HT core option | Hyper Thread Intel's trademark for implementation of the simultaneous multithreading technology on the Pentium 4 microarchitecture. It is a more advanced form of Super-threading that debuted on the Intel Xeon processors and was later added to Pentium 4 processors. The technology improves processor performance under certain workloads by providing useful work for execution units that would otherwise be idle, for example during a cache miss. A Pentium 4 with Hyper-Threading enabled is treated by the operating system as two processors instead of one. From Wiki |
| I2C | Two Serial Management Interfaces |
| IANA | Internet Assigned Number Authority |
| ICV | 128-bits Integrity Check Value (referred also as authentication tag). used for LinkSec header and signature |
| IDS | Intrusion detection systems |
| IFCS | Insert Frame Check Sequence of Ethernet frames |
| IFS | Inter Frame Spacing |
| IKE | Internet Key Exchange |
| IOAT | I/O Acceleration Technology |
| IOH | I/O Hub |
| IOV | Input Output Virtualization |
| IOV mode | Operating through an IOVM or IOVI |
| IOVI | I/O Virtual Intermediary: A special virtual machine that owns the physical device and is responsible for the configuration of the physical device.<br>Also Known As IOVM |
| IOVM | I/O Virtual Machine: A special virtual machine that owns the physical device and is responsible for the configuration of the physical device.<br>Also Known As IOVI |
| IP tunneling | IP tunneling is the process of embedding one IP packet inside of another, for the purpose of simulating a physical connection between two remote networks across an intermediate network.<br>IP tunnels are often used in conjunction with IPSec protocol to create a VPN between two or more remote networks across a "hostile" network such as the Internet. |
| IPC | Inter Processor Communication. |
| IP — CPMP | Carrier Performance Measurement Plan |
| IPG | Inter Packet Gap. |

| Term | Definition |
|---|---|
| IP Sec | IP security) is a suite of protocols for securing Internet Protocol (IP) communications by authenticating and/or encrypting each IP packet in a data stream. IPsec also includes protocols for cryptographic key establishment.<br>IPsec is implemented by a set of cryptographic protocols for (1) securing packet flows and (2) internet key exchange. There are two families of key exchange protocols.<br>The IP security architecture uses the concept of a security association as the basis for building security functions into IP. A security association is simply the bundle of algorithms and parameters (such as keys) that is being used to encrypt a particular flow. The actual choice of algorithm is left up to the users. A security parameter index (SPI) is provided along with the destination address to allow the security association for a packet to be looked up.<br>For multicast, therefore, a security association is provided for the group, and is duplicated across all authorized receivers of the group. There may be more than one security association for a group, using different SPIs, thereby allowing multiple levels and sets of security within a group. Indeed, each sender can have multiple security associations, allowing authentication, since a receiver can only know that someone knowing the keys sent the data. Note that the standard doesn't describe how the association is chosen and duplicated across the group; it is assumed that a responsible party will make the choice. From Wiki |
| iSCSI | Internet SCSI (iSCSI) is a network protocol standard, officially ratified on 2003-02-11 by the Internet Engineering Task Force, that allows the use of the SCSI protocol over TCP/IP networks. iSCSI is a transport layer protocol in the SCSI-3 specifications framework. Other protocols in the transport layer include SCSI Parallel Interface (SPI), Serial Attached SCSI (SAS) and Fibre Channel. From Wiki. |
| ISR | Interrupt Service Routine |
| ITR | Interrupt Throttling |
| IV | Integrity Value |
| IV | Initialization Vector |
| IV | Initial Value |
| KaY | Key agreement entity (KaY – in 802.1AE spec terminology) i.e. control and access the off loading engine (SecY in 802.1AE spec terminology) |
| KVM | Keyboard — Video — Mouse |
| LAN Auxiliary Power-Up | The event of connecting the LAN controller to a power source (occurs even before system power-up). |
| landing Zone requirements | General targets for the product. |
| LF | Local Fault |
| LinkSec (or MACsec, 802.1AE) | A MAC level encryption/authentication scheme defined in IEEE 802.1AE that uses symmetric cryptography. The 802.1AE defines an AES-GCM 128 bit key as a mandatory cipher suite which can be processed by the LAN controller. |
| LLC header | 802.2 defines a special header that includes a SNAP (subnetwork access protocol) header. Some protocols, particularly those designed for the OSI networking stack, operate directly on top of 802.2 LLC, which provides both datagram and connection-oriented network services. This 802.2 header is currently embedded in modern 802.3 frames (Ethernet II frames, aka. DIX frames).<br>The LLC header includes two additional eight-bit address fields, called service access points or SAPs in OSI terminology; when both source and destination SAP are set to the value 0xAA, the SNAP service is requested. The SNAP header allows EtherType values to be used with all IEEE 802 protocols, as well as supporting private protocol ID spaces. In IEEE 802.3x-1997, the IEEE Ethernet standard was changed to explicitly allow the use of the 16-bit field after the Ethernet MAC Addresses to be used as a length field or a type field. This definition is from Wiki |
| LLDP | Link Layer Discovery Protocol |
| LLINT | Low Latency Interrupt |
| Local Traffic | In a virtual environment traffic between virtual machines. |
| LOM | LAN on Motherboard. |
| LP | Link Partner |
| LSC | Link Status Change |
| LS | Least significant / Lowest order (for example: LS bit = Least significant bit) |



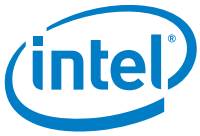
| Term | Definition |
|---|---|
| LSO | Large Send Offload, same as TSO |
| LSP | Link Strict Priority |
| LTSSM | Link Training and Status State Machine Defined in the PCIe specs. |
| MAC | Media Access Control. |
| MAUI | Multi Speed Attachment Unit Interface |
| MCH | Memory Controller Hub |
| MDC | Management Data Clock |
| MDI | Management Data Interface |
| MDIC | MDI Control Register |
| MDIO | Management Data Input/Output Interface over MDC/MDIO lines. |
| MFVC | Multi-Function Virtual Channel Capability structure |
| MIB | Management Interface Bus |
| MIFS/MIPG | Minimum Inter Frame Spacing/Minimum Inter Packet Gap. |
| MMD | MDIO Managed Device |
| MMW | Maximum Memory Window. |
| Mod / Modulo | In computing, the modulo operation finds the remainder of division of one number by another. |
| MPA | Marker PDU Aligned Framing for TCP |
| MPDU | MACSEC Protocol Data Unit including SecTag, User Data and ICV |
| MRQC | Multiple Receive Queues Command register |
| MS | Most significant / Highest order (for example: MS byte = Most significant byte) |
| MSFT RSS | Microsoft RSS specification |
| MSI | Message Signaled Interrupt |
| MSS | Maximum Segment Size |
| MTA | Multicast Table Array |
| MTU | Maximum Transmission Unit |
| NACK | Negative Acknowledgement |
| native mode | Used for GPIO pin that is set to be controlled by the internal logic rather than by software. |
| NC-SI | Network Controller — Sideband Interface |
| NEXT | Near End Crosstalk |
| Next Generation VMDq | SW switch acceleration mode–central management of the networking resources by an IOVM or by the VMM.<br>Virtual Machine Devices queue (VMDq) is a mechanism to share I/O resources among several consumers. For example, in a virtual system, multiple OSs are loaded and each executes as though the whole system's resources were at its disposal. However, for the limited number of I/O devices, this presents a problem because each OS may be in a separate memory domain and all the data movement and device management has to be done by a VMM (Virtual Machine Monitor). VMM access adds latency and delay to I/O accesses and degrades I/O performance. VMDs (Virtual Machine Devices) are designed to reduce the burden of VMM by making certain functions of an I/O device shared and thus can be accessed directly from each guest OS or Virtual Machine (VM). |
| NIC | Network Interface Controller. |
| NFTS | Number of Fast Training Signals |
| NFS | Network File Server |

| Term | Definition |
|---|---|
| non-teaming mode | If the LAN is in non-teaming mode, the SMBus is presented as two SMBus devices on the SMBus (two SMBus addresses). |
| Nonce | 96-bits initialization vector used by the AES-128 engine, which is distinct for each invocation of the encryption operation for a fixed key. It is formed by the AES-128 SALT field stored for that IPsec flow in the Tx SA Table, appended with the Initialization Vector (IV) field included in the IPsec packet: |
| NOS | Network Operating System |
| NPRD | Non-Posted Request Data |
| NRZ | Non-return-to-zero signaling |
| NTL | No Touch Leakage |
| NTP | Network Time Protocol |
| OEM | Original Equipment manufacturer |
| Core | Network Interface Registers |
| Packet | A unit composed of headers, data and footers that are sent or received by a device. Also known as a frame. |
| Pass Filters | Needs Definition Packets that match this type of filter continue on to their destination |
| PB | Packet Buffer |
| PBA | The nine-digit (Printed Board Assembly) number used for Intel manufactured adapter cards. |
| PBA | Pending Bit Array |
| PBA | Printed Board Assembly |
| PCS | Physical Coding Sub layer. |
| PDU | Protocol Data Units |
| PF | Physical Function (in a virtualization context). |
| PFC | Priority Flow Control |
| PHY | Physical Layer Device. |
| Plaintext | Data to be both authenticated and encrypted. |
| PMA | Physical Medium Attachment |
| PMC | Power Management Capabilities |
| PMD | Physical Medium Dependent. |
| PME | Power Management Event |
| PN | Packet Number (PN) in a LinkSec context: Monotonically increasing value used to uniquely identify a LinkSec frame in the sequence. |
| Pool | Virtual ports |
| Power State D0a | Active fully operational state. Once memory space is enabled all internal clocks are activated and the LAN Controller enters an active state. |
| Power State D0u | The D0u state is a low-power state used after SPXB Reset is de-asserted following power-up (cold or warm), or on D3 exit. |
| Power State D3Hot | A Power down state with the PCI continuing to receive a proper power supply. |
| Power State D3Cold | A Power down state with the PCI also in a power down state. |
| Power State Dr | Device state when PCIe reset is asserted. |
| Power State Sx | Lan Connected Device: SMBus Active and PCI Powered down. |
| PPM | Packet Processor Module |
| PRBS | Pseudo-Random Binary Sequence |



| Term | Definition |
|---|---|
| PT | Pass Through |
| PTP | Precision Time Protocol |
| QoS | Quality of Service |
| QWORD (Quad-Word) alignment | Implies that the physical addresses may only be aligned on 8byte boundaries; i.e., the last nibble of the address may only end in 0, or 8. For example, 0FECBD9A8h. |
| Receive latency | Measured from packet reception from the wire and until the descriptor is updated on PCIe. |
| RDMA | Remote Direct Memory Access |
| RDMAP | Remote Direct Memory Access Protocol |
| Relax ordering | When the strict order of packets is not required, the device can send packets in an order that allows for less power consumption and greater CPU efficiency. |
| RID | Requester ID |
| RLT | Rate-limited flag bit |
| RMCP | Remote Management and Control Protocol (Distributed Management Task Force) |
| RMII | Reduced Media Independent Interface (Reduced MII) |
| RMII NC-SI | Reduced Media Independent Interface (Reduced MII). |
| RMON statistics | Remote Network Monitoring or Remote Monitoring |
| RPC header | Remote Procedure Call |
| RS | Rate Scheduler |
| RSC | Receive Side Coalescing coalesces incoming TCP/IP (and potentially UDP/IP) packets into larger receive segments |
| RSS | Receive-Side Scaling is a mechanism to distribute received packets into several descriptor queues. Software then assigns each queue to a different processor, therefore sharing the load of packet processing among several processors |
| RSTD | Reset Sequence Done |
| RSTI | Reset Sequence in Process |
| RTT | Round Trip Time |
| Rx, RX | Receive |
| SA | Security Association or source address |
| SA (in a LinkSec context) | Secure Association (SA): A security relationship that provides security guarantees for frames transmitted from one member of a CA to the others. Each SA is supported by a single secret key, or a single set of keys where the cryptographic operations used to protect one frame require more than one key. |
| SAC | Single Address Cycle (SAC) messages |
| SAK | Security Associations Key |
| salt | In cryptography, a salt consists of random bits used as one of the inputs to a key derivation function. Sometimes the initialization vector, a previously generated (preferably random) value, is used as a salt. The other input is usually a password or passphrase. The output of the key derivation function is often stored as the encrypted version of the password. A salt value can also be used as a key for use in a cipher or other cryptographic algorithm. A salt value is typically used in a hash function. from Wiki |
| SAN | Storage Area Networks |
| SAP | Service Access Point –an identifying label for network endpoints used in OSI networking. |
| SC | Secure Channel – Authentication and key exchange |
| SC | Secure Channel (SC): A security relationship used to provide security guarantees for frames transmitted from one member of a CA to the others. An SC is supported by a sequence of SAs thus allowing the periodic use of fresh keys without terminating the relationship. |

| Term | Definition |
|---|---|
| SCI | Secure Channel Identifier A globally unique identifier for a secure channel, comprising a globally unique Ethernet MAC Address and a Port Identifier, unique within the system allocated that address. |
| SCSI | Small Computer System Interface is a set of standards for physically connecting and transferring data between computers and peripheral devices. The SCSI standards define commands, protocols, and electrical and optical interfaces. SCSI is most commonly used for hard disks and tape drives, but it can connect a wide range of other devices, including scanners, and optical drives (CD, DVD, etc. From Wiki. |
| SCL signal | SM Bus Clock |
| SCTP | Stream Control Transmission Protocol |
| SDA signal | SM Bus Data |
| SDP | Software-Definable Pins |
| SecY | 802.1AE spec terminology Security entity |
| Segment | subsections of a packet |
| SerDes | Serializer and De-Serializer Circuit. |
| SFD | Start Frame Delimiter |
| SGMII | Serialized Gigabit Media Independent Interface. |
| SKU | subsets of features of a chip that can be disabled for marketing purposes. |
| SNMP | Standard Network Management Protocol |
| SMB | Semaphore Bit |
| SMBus | System Management Bus. A bus that carries various manageability components, including the LAN controller, BIOS, sensors and remote-control devices. |
| SN | Sequence Number — contains a counter value that increases by one for each Ethernet frame sent. |
| SNAP | Subnetwork Access Protocol |
| SoL | Serial Over LAN<br>Serial Over LAN is a mechanism that enables the input and output of the serial port of a managed system to be redirected via an IPMI (Internet Protocol Multicast Initiative) session over IP. |
| SPD | Smart Power Down |
| SPI | The Security Parameter Index is an identification tag added to the header while using IPSec for tunneling the IP traffic. This tag helps the kernel discern between two traffic streams where different encryption rules and algorithms may be in use.<br>The SPI (as per RFC 2401) is an essential part of an IPSec SA (Security Association) because it enables the receiving system to select the SA under which a received packet will be processed. An SPI has only local significance, since is defined by the creator of the SA; an SPI is generally viewed as an opaque bit string. However, the creator of an SA may interpret the bits in an SPI to facilitate local processing. from Wikipedia |
| SPXB interface | PCI Express Backbone |
| Spoofing | In computer networking, the term IP address spoofing is the creation of IP packets with a forged (spoofed) source IP address with the purpose to conceal the identity of the sender or impersonating another computing system. IP stands for Internet Protocol. from Wiki |
| SR-IOV | PCI-SIG single-root I/O Virtualization initiative |
| SW Switch acceleration mode | Central management of the networking resources by an IOVM or by the VMM. Also known as VMDq2 mode. |
| SWIZZLE | To convert external names, array indices, or references within a data structure into address pointers when the data structure is brought into main memory from external storage (also called pointer swizzling); |
| Sx | Lan Connected Device: SMBus Active and PCI Powered down. |
| SYN Attack | A SYN attack is a form of denial-of-service attack in which an attacker sends a succession of SYN (synchronize) requests to a target's system. |



| Term | Definition |
|---|---|
| TC | Traffic Class |
| TCI | For 802.1q, Tag Header field Tag Control Information (TCI); 2 octets. |
| TCO | Total Cost of Ownership (Management) |
| TCP/IP | Transmission Control Protocol/Internet Protocol |
| TDESC | Transmit Descriptor |
| TDP | Total Device Power |
| TDR | Time Domain Reflectometry |
| Teaming Mode | When the LAN is in Teaming mode, the 82599 is presented over the SMBus as one device and has one SMBus address. |
| TFCS | Transmit Flow Control Status |
| TLP | Transaction layer Packets |
| ToS | Type of Service |
| TPID | For 802.1q, Tag Header field Tag Protocol Identifier; 2 octets. |
| TPPAC | Transmit Packet Plane Arbitration Control |
| Transmit latency | Measured from Tail update until the packet is transmitted on the wire. It is assumed that a single packet is submitted for this traffic class and its latency is then measured in presence of traffic belonging to other traffic classes. |
| TS | Time Stamp |
| TSO | TCP or Transmit Segmentation offload — A mode in which a large TCP/UDP I/O is handled to the device and the device segments it to L2 packets according to the requested MSS. |
| TSS | Transmit Side Scaling |
| Tx, TX | Transmit |
| UBWG | User Bandwidth Group |
| ULP | Upper Layer Protocol |
| UP | User Priority |
| UR | Error Reporting Unsupported Request Error |
| VF | Virtual Function– A part of a PF assigned to a VI |
| VI | Virtual Image – A virtual machine to which a part of the I/O resources is assigned. Also known as a VM. |
| VM | Virtual Machine |
| VMM | Virtual Machine Monitor |
| VPD | Vital Product Data (PCI protocol). |
| VT | Virtualization |
| WB | Write Back |
| WC | Worst Case |
| WfM | Wired for Management<br>was a primarily hardware-based system allowing a newly built computer without any software to be manipulated by a master computer that could access the hard disk of the new PC to paste the install program. It could also be used to update software and monitor system status remotely. Intel developed the system in the 1990s; it is now considered obsolete. |
| WoL | Wake-on-LAN Now called APM Wake up or Advanced power management Wake up. |
| WORD alignment | Implies that physical addresses must be aligned on even boundaries; i.e., the last nibble of the address may only end in 0, 2, 4, 6, 8, Ah, Ch, or Eh. For example, 0FECBD9A2h. |

| Term | Definition |
|---|---|
| WRR | Weighted Round-Robin |
| WSP | Weighted Strict Priority |
| XAUI | 10 Gigabit Attachment Unit Interface |
| XFP | 10 Gigabit Small Form Factor Pluggable modules |
| XGMII | 10 Gigabit Media Independent Interface |
| XGXS | XGMII Extender Sub layer |
| XMT Frame Transmit | Most Recent Transmit Buffer Tail Register content |

# 15.1 Register Attributes

| Attribute | Description |
|---|---|
| RO | Read Only: Writes to this register setting do not affect the register value. Reads return either a constant or variable device state. |
| RW | Read Write: Writes to this register setting alter the register value. Reads return the value of the register. |
| RW1C | Read Write Clear: Register that is set to 1b by hardware, and cleared to 0b by software writing a 1b to the register |
| RWS | Read Write Set: Register that is set to1b by software by writing a 1b to the register, and cleared to 0b by hardware. Notation used in CSR sections |
| 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. Notation used in "PCI Express* Programming Interface" chapter. |
| WO | Write Only: Writes to this register alters the register value. Reads always return 0b. |
| 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. |

# Appendix A Packets and Frames

## 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 | 4 | Possible VLAN Tag | | Stored |
| 12 | 8 | Possible Len/LLC/SNAP Header | | Stored |
| 12 | 2 | Type | 0806 | Compare |
| 14 | 2 | HW Type | 0001 | Compare |
| 16 | 2 | Protocol Type | 0800 | Compare |
| 18 | 1 | Hardware Size | 06 | Compare |
| 19 | 1 | Protocol Address Length | 04 | Compare |
| 20 | 2 | Operation | 0001 | Compare |
| 22 | 6 | Sender HW Address | - | Stored |
| 28 | 4 | Sender IP Address | - | Stored |
| 32 | 6 | Target HW Address | - | Ignore |
| 38 | 4 | Target IP Address | ARP IP address | Compare |

### 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 EEPROM or BMC |
| 12 | 4 | Possible VLAN Tag | From ARP Request |
| 12 | 8 | Possible Len/LLC/SNAP Header | From ARP Request |
| 12 | 2 | Type | 0x0806 |
| 14 | 2 | HW Type | 0x0001 |
| 16 | 2 | Protocol Type | 0x0800 |
| 18 | 1 | Hardware Size | 0x06 |
| 19 | 1 | Protocol Address Length | 0x04 |
| 20 | 2 | Operation | 0x0002 |
| 22 | 6 | Sender HW Address | Programmed from EEPROM or BMC |
| 28 | 4 | Sender IP Address | Programmed from EEPROM or BMC |
| 32 | 6 | Target HW Address | ARP Request Sender HW Address |
| 38 | 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 | 2 | Type | 0x0806 |
| 14 | 2 | Hardware Type | 0x0001 |
| 16 | 2 | Protocol Type | 0x0800 |
| 18 | 1 | Hardware Size | 0x06 |
| 19 | 1 | Protocol Address Length | 0x04 |
| 20 | 2 | Operation | 0x0001 |
| 22 | 6 | Sender Hardware Address | |

| Offset | # of bytes | Field | Value |
|---|---|---|---|
| 28 | 4 | Sender IP Address | |
| 32 | 6 | Target Hardware Address | |
| 38 | 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 82599 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 82599 are highlighted in the figures below.

**Note:** The IP header is first shown in the traditional (i.e. RFC 791) representation, and because byte and bit ordering is confusing in that representation, the IP header is also shown in Little Endian format. The actual data will be fetched from memory in Little Endian format.

<table>
<tr><th colspan="4">0 1 2 3 4 5 6 7 | 1<br>8 9 0 1 2 3 4 5 | 2<br>6 7 8 9 0 1 2 3 | 3<br>4 5 6 7 8 9 0 1</th></tr>
<tr><td>Version</td><td>IP Hdr Length</td><td>TYPE of service</td><td>Total length (IP header + payload length)</td></tr>
<tr><td colspan="3">Identification</td><td>Flags | Fragment Offset</td></tr>
<tr><td colspan="2">Time to Live</td><td>Layer 4 Protocol ID</td><td>Header Checksum</td></tr>
<tr><td colspan="4">Source Address</td></tr>
<tr><td colspan="4">Destination Address</td></tr>
<tr><td colspan="4">Options</td></tr>
</table>

**Figure A-1 IPv4 Header (Traditional Representation - most left byte first on the wire)**

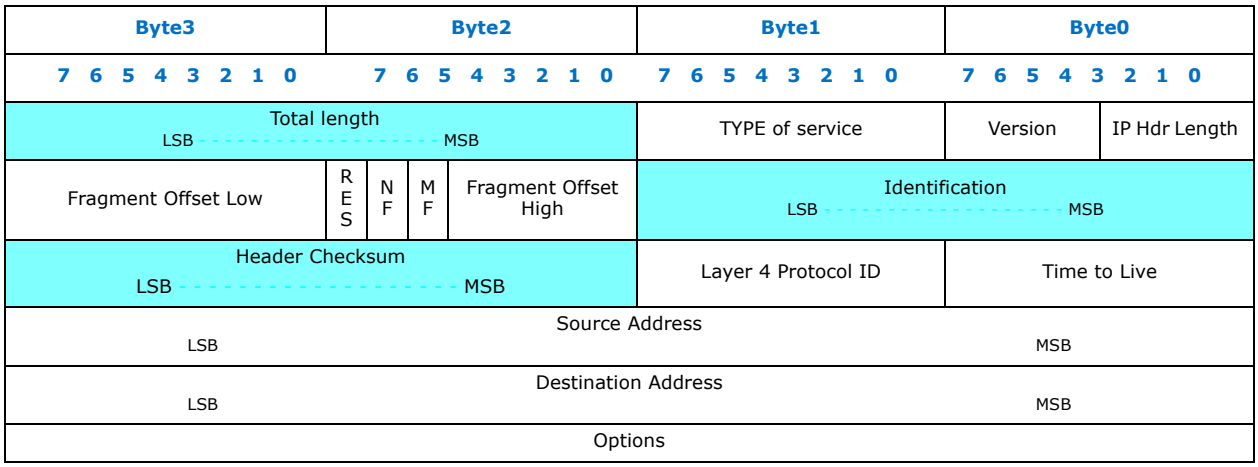



**Figure A-2 IPv4 Header (Little Endian Order - Byte 0 first on the wire)**

Identification is incremented on each packet.

Flags Field Definitions:

The Flags field is defined below. Note that hardware does not evaluate or change these bits.

- MF - More Fragments
- NF - No Fragments
- Reserved

the 82599 does TCP segmentation, not IP Fragmentation. IP Fragmentation may occur in transit through a network's infrastructure.

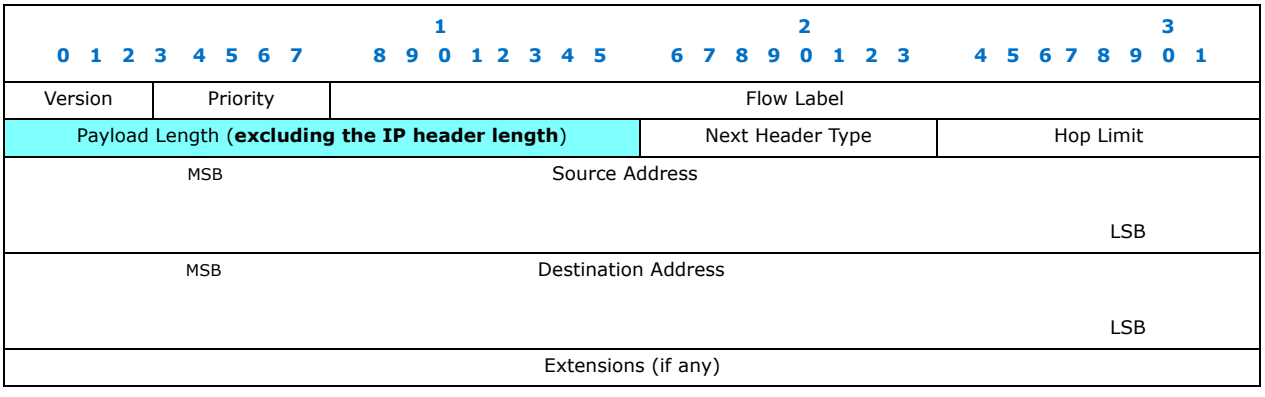



**Figure A-3 IPv6 Header (Traditional Representation - most left byte first on the wire)**

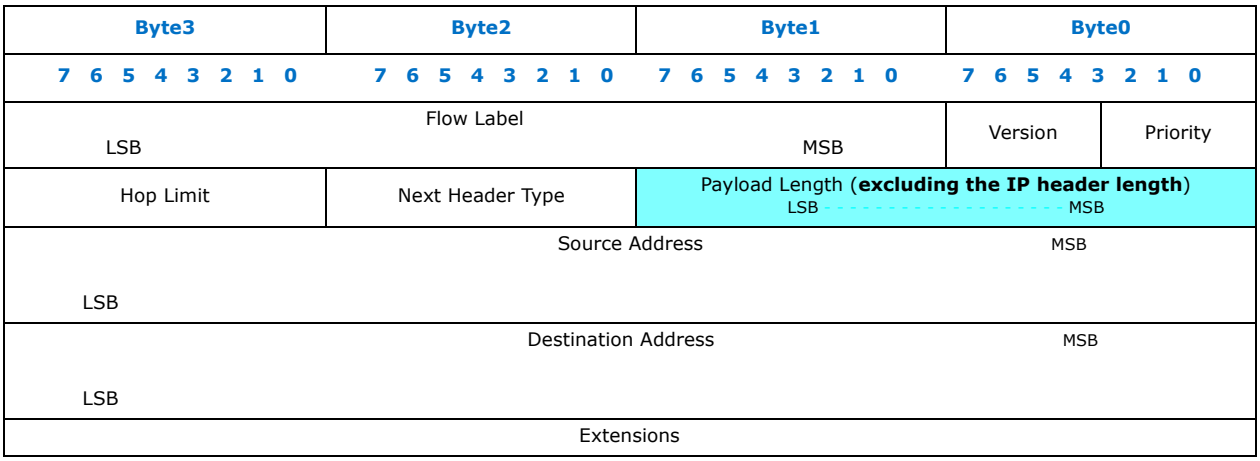



**Figure A-4 IPv6 Header (Little Endian Order - byte 0 first on the wire)**

A TCP or UDP frame uses a 16 bit wide one's complement checksum. The checksum word is computed on the outgoing TCP or UDP header and payload, and on the Pseudo Header. Details on checksum computations are provided in Section 7.2.4.6.

**Note:** TCP and UDP over IPv6 requires the use of checksum, where it is optional for UDP over IPv4.

The TCP header is first shown in the traditional (i.e. RFC 793) representation, and because byte and bit ordering is confusing in that representation, the TCP header is also shown in Little Endian format. The actual data will be fetched from memory in Little Endian format.

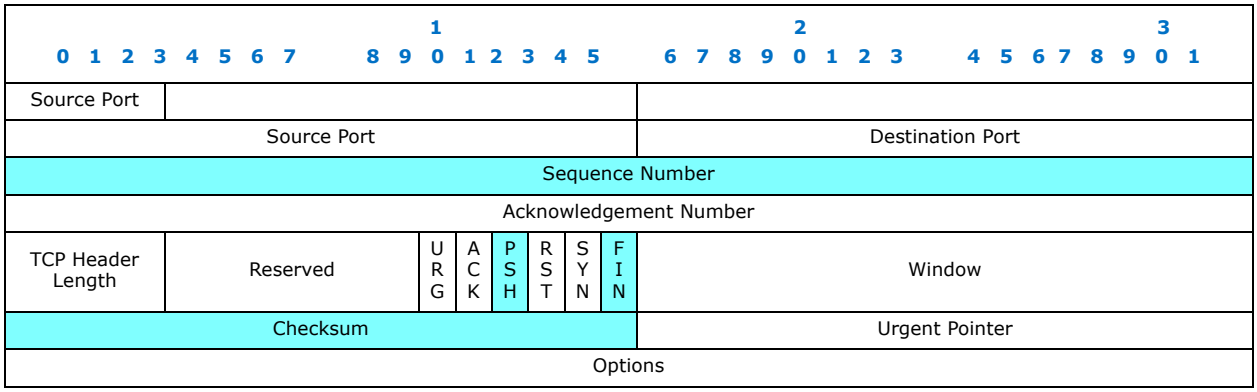



**Figure A-5 TCP Header (Traditional Representation)**

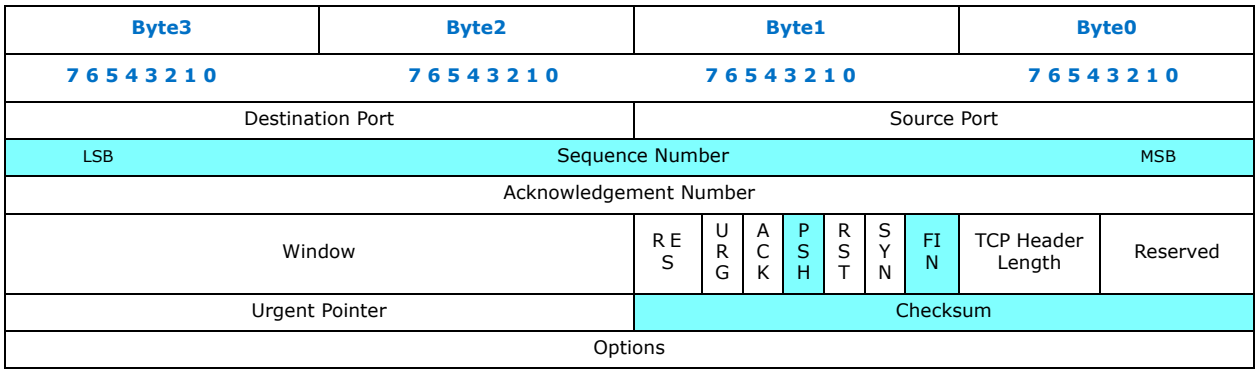



**Figure A-6 TCP Header (Little Endian)**

The TCP header is always a multiple of 32 bit words. TCP options may occupy space at the end of the TCP header and are a multiple of 8 bits in length. All options are included in the checksum.

The checksum also covers a 96-bit pseudo header prefixed to the TCP Header (see Figure A-7 below). For IPv4 packets, this pseudo header contains the IP Source Address, the IP Destination Address, the IP Protocol field, and TCP Length. Software pre-calculates the partial pseudo header sum, that includes IPv4 SA, DA and protocol types, but NOT the TCP length, and stores this value into the TCP checksum field of the packet. For both IPv4 and IPv6, hardware needs to factor in the TCP length to the software supplied pseudo header partial checksum.

**Note:** When calculating the TCP pseudo header, the byte ordering can be tricky. One common question is whether the Protocol ID field is added to the "lower" or "upper" byte of the 16- bit sum. The Protocol ID field should be added to the least significant byte (LSB) of the 16-bit pseudo header sum, where the most significant byte (MSB) of the 16-bit sum is the byte that corresponds to the first checksum byte out on the wire.

The TCP Length field is the TCP Header Length including option fields plus the data length in bytes, which is calculated by hardware on a frame-by-frame basis. The TCP Length does not count the 12 bytes of the pseudo header. The TCP length of the packet is determined by hardware as:

TCP Length = min (MSS,PAYLOADLEN) + L5_LEN

The two flags that may be modified are defined as:

- PSH: receiver should pass this data to the app without delay
- FIN: sender is finished sending data

The handling of these flags is described in Section 7.2.4.7.

"Payload" is normally MSS except for the last packet where it represents the remainder of the payload.

| IPv4 Source Address | | |
|---|---|---|
| IPv4 Destination Address | | |
| Zero | Layer 4 Protocol ID | TCP/UDP Length |

**Figure A-7 TCP/UDP Pseudo Header Content for IPv4 (Traditional Representation)**

| IPv6 Source Address | |
|---|---|
| IPv6 Final Destination Address | |
| TCP/UDP Packet Length | |
| Zero | Next Header |

**Figure A-8 TCP/UDP Pseudo Header Content for IPv6 (Traditional Representation)**

**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 will be in the last element of the Routing header; at the recipient(s), that address will be in the Destination Address field of the IPv6 header.
- The Next Header value in the pseudo-header identifies the upper-layer protocol (e.g., 6 for TCP, or 17 for UDP). It will differ from the Next Header value in the IPv6 header if there are extension headers between the IPv6 header and the upper-layer header.
- The Upper-Layer Packet Length in the pseudo-header is the length of the upper-layer header and data (e.g., TCP header plus TCP data). Some upper-layer protocols carry their own length information (e.g., the Length field in the UDP header); for such protocols, that is the length used in the pseudo- header. Other protocols (such as TCP) do not carry their own length information, in which case the length used in the pseudo-header is the Payload Length from the IPv6 header, minus the length of any extension headers present between the IPv6 header and the upper-layer header.
- Unlike IPv4, when UDP packets are originated by an IPv6 node, the UDP checksum is not optional. That is, whenever originating a UDP packet, an IPv6 node must compute a UDP checksum over the packet and the pseudo-header, and, if that computation yields a result of zero, it must be changed to hex FFFF for placement in the UDP header. IPv6 receivers must discard UDP packets containing a zero checksum, and should log the error.

A type 0 Routing header has the following format:

| Next Header | Hdr Ext Len | Routing Type<br>"0" | Segments Left<br>"n" |
|---|---|---|---|
| Reserved | | | |
| Address[1] | | | |
| Address[2] | | | |
| ... | | | |
| Final Destination Address [n] | | | |

**Figure A-9 IPv6 Routing Header (Traditional Representation)**

- Next Header - 8-bit selector. Identifies the type of header immediately following the Routing header. Uses the same values as the IPv4 Protocol field [RFC-1700 et seq.].
- Hdr Ext Len - 8-bit unsigned integer. Length of the Routing header in 8-octet units, not including the first 8 octets. For the Type 0 Routing header, Hdr Ext Len is equal to two times the number of addresses in the header.
- Routing Type - 0.
- Segments Left - 8-bit unsigned integer. Number of route segments remaining, i.e., number of explicitly listed intermediate nodes still to be visited before reaching the final destination. Equal to "n" at the source node.

Reserved - 32-bit reserved field. Initialized to zero for transmission; ignored on reception.

- Address[1...n] - Vector of 128-bit addresses, numbered 1 to n.

The UDP header is always 8 bytes in size with no options.

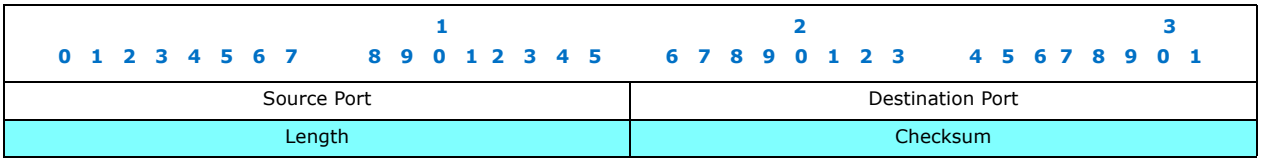



**Figure A-10 UDP Header (Traditional Representation)**

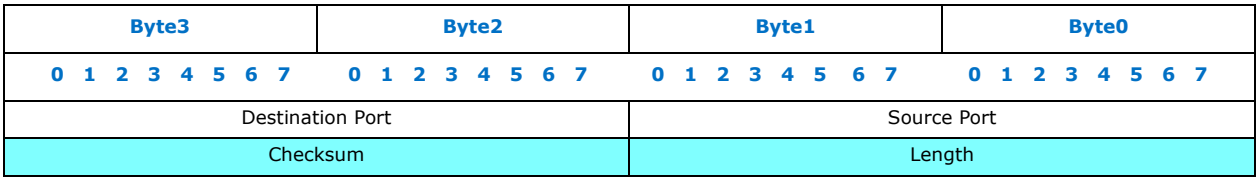



**Figure A-11 UDP Header (Little Endian Order)**

UDP pseudo header has the same format as the TCP pseudo header. The pseudo header prefixed to the UDP header contains the IPv4 source address, the IPv4 destination address, the IPv4 protocol field, and the UDP length (same as the TCP Length discussed above). This checksum procedure is the same as is used in TCP.

Unlike the TCP checksum, the UDP checksum is optional (for IPv4). Software must set the TXSM bit in the TCP/IP Context Transmit Descriptor to indicate that a UDP checksum should be inserted. Hardware will not overwrite the UDP checksum unless the TXSM bit is set.

### A.1.3 Magic Packet

The Magic Packet is a broadcast frame, but could also be a Multicast or Unicast Ethernet MAC Addresses. the 82599 accepts this packet if it matches any of its pre-programmed Ethernet MAC Addresses. Magic packet can be sent over a variety of connectionless protocols (usually UDP or IPX). The Magic packet pattern is composed of the following sequences:

- Synchronization stream composed of 6 bytes equal to 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
- Unique pattern composed of 16 times the end node Ethernet MAC Address. the 82599 expect for the Ethernet MAC Address stored in the RAL[0] and RAH[0] registers.

the 82599 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.1.4 SNAP Packet Format

the 82599 supports SNAP packets for all offload capabilities as described in this document. The device identifies SNAP packet by the following fields highlighted in “light green” color in the diagram below.

<table>
<tr><td rowspan="2">L2 Destination & Source MAC Addresses</td><td rowspan="2">L2 Length must be less equal than 1500</td><td colspan="4">SNAP Header</td><td rowspan="2">Payload</td><td rowspan="2">CRC</td></tr>
<tr><td>DSAP<br>0xAA</td><td>SSAP<br>0xAA</td><td>CTRL<br>0x03</td><td>ORG<br>0x00 0x00 0x00</td><td>EtherType</td></tr>
<tr><td></td><td colspan="5"><----------- Compared by the Hardware ----------></td><td></td><td></td></tr>
</table>

The packet may carry VLAN header(s). In that case, the EtherType in the SNAP header has the EtherType value of the first VLAN header.

The packet may be encapsulated by LinkSec. In that case, the LinkSec header appears before the L2 length field.

## A.2 Packet Types for Packet Split Filtering

The following packet types are supported by the 'Packet Split' feature in the 82599. 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 | S=(0/4) | Possible VLAN Tag | 8100 **** | Compare | |
| 12+S | D=(0/8) | Possible Len/LLC/SNAP Header | | Compare | |
| 12+D+S | 2 | Type | 0800h | Compare | IP |
| **IPv4 header** | | | | | |
| 14+D+S | 1 | Version/ HDR length | 4Xh | Compare | Check IPv4 and header length. |
| 15+D+S | 1 | Type of Service | - | Ignore | |
| 16+D+S | 2 | Packet Length | - | Ignore | |
| 18+D+S | 2 | Identification | - | Ignore | |
| 20+D+S | 2 | Fragment Info | >0 or MF bit is set | Check | Check that the packet is fragmented. |
| 22+D+S | 1 | Time to live | - | Ignore | |
| 23+D+S | 1 | Protocol | | Ignore | Has no meaning if the packet is fragmented. |
| 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.

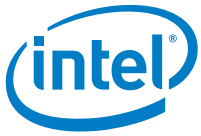
| Offset | # of bytes | Field | Value | Action | Comment |
|---|---|---|---|---|---|
| 0 | 6 | Destination Address | | Ignore | MAC Header – processed by main address filter, or broadcast filter. |
| 6 | 6 | Source Address | | Ignore | |
| 12 | S=(0/4) | Possible VLAN Tag | 8100 **** | Check | |
| 12+S | D=(0/8) | Possible Len/LLC/SNAP Header | | Check | |
| 12+D+S | 2 | Type | 0800h | Compare | IP |
| **IP header** | | | | | |
| 14+D+S | 1 | Version/ HDR length | 4Xh | Compare | Check IPv4 and header length. |
| 15+D+S | 1 | Type of Service | - | Ignore | |
| 16+D+S | 2 | Packet Length | - | Ignore | |
| 18+D+S | 2 | Identification | - | Ignore | |
| 20+D+S | 2 | Fragment Info | (xx00) 000h | Compare | |
| 22+D+S | 1 | Time to live | - | Ignore | |
| 23+D+S | 1 | Protocol | 0x11 | Compare | UDP header |
| 24+D+S | 2 | Header Checksum | - | Ignore | |
| 26+D+S | 4 | Source IP Address | - | Ignore | |
| 30+D+S | 4 | Destination IP Address | - | Ignore | |
| 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 | S=(0/4) | Possible VLAN Tag | 8100 **** | Check | |
| 12+S | D=(0/8) | Possible Len/LLC/SNAP Header | | Check | |
| 12+D+S | 2 | Type | 0800h | Compare | IP |
| **IPv4 header** | | | | | |
| 14+D+S | 1 | Version/ HDR length | 4Xh | Compare | Check IPv4 and header length. |
| 15+D+S | 1 | Type of Service | - | Ignore | |
| 16+D+S | 2 | Packet Length | - | Ignore | |
| 18+D+S | 2 | Identification | - | Ignore | |
| 20+D+S | 2 | Fragment Info | 0x00 | Compare | |
| 22+D+S | 1 | Time to live | - | Ignore | |
| 23+D+S | 1 | Protocol | 0x06 | Compare | TCP header |
| 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 | |



| Offset | # of bytes | Field | Value | Action | Comment |
|---|---|---|---|---|---|
| 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.

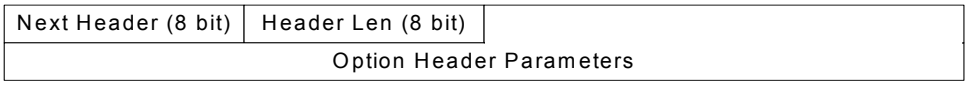
- N = (IP HDR length -5) * 4.
- F = (TCP header length - 5) * 4.

## 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 82599 does not parse all possible extension headers and if there is an extension header that is not supported by the 82599 then the packet is treated as an unknown payload after the IPv6 header.

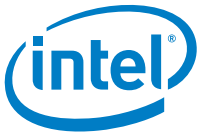
| Value | Header type |
|---|---|
| 0x00 | Hop by Hop |
| 0x2B | Routing |
| 0x2C | Fragment |
| 0x3B | No next header (EOL) |
| 0x3C | Destination option header |

- 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 | S=(0/4) | Possible VLAN Tag | 8100 | Check | |
| 12+S | D=(0/8) | Possible Len/LLC/SNAP Header | | Check | |
| 12+D+S | 2 | Type | 0800h | Compare | IP |
| **Ipv4 header** | | | | | |
| 14+D+S | 1 | Version/ HDR length | 4Xh | Compare | Check IPv4 and header length. |
| 15+D+S | 1 | Type of Service | - | Ignore | |
| 16+D+S | 2 | Packet Length | - | Ignore | |
| 18+D+S | 2 | Identification | - | Ignore | |
| 20+D+S | 2 | Fragment Info | 0x00 | Compare | |
| 22+D+S | 1 | Time to live | - | Ignore | |
| 23+D+S | 1 | Protocol | 0x29 | Compare | Ipv6 |
| 24+D+S | 2 | Header Checksum | - | Ignore | |
| 26+D+S | 4 | Source IP Address | - | Ignore | |
| 30+D+S | 4 | Destination IP Address | - | Ignore | |
| 34+D+S | N | Possible IP Options | | Ignore | |
| **Ipv6 header** | | | | | |
| 34+D+S+N | 1 | Version/ Traffic Class | 6Xh | Compare | Check IPv6 |
| 35+D+S+N | 3 | Traffic Class/Flow Label | - | Ignore | |
| 38+D+S+N | 2 | Payload Length | - | Ignore | |
| 40+D+S+N | 1 | Next Header | IPv6 extension headers | Check | |
| 41+D+S+N | 1 | Hop Limit | - | Ignore | |
| 42+D+S+N | 16 | Source Address | - | Ignore | |



| Offset | # of bytes | Field | Value | Action | Comment |
|---|---|---|---|---|---|
| 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 | S=(0/4) | Possible VLAN Tag | 8100 | Check | |
| 12+S | D=(0/8) | Possible Len/LLC/SNAP Header | | Check | |
| 12+D+S | 2 | Type | 0800h | Compare | IP |
| **Ipv4 header** | | | | | |
| 14+D+S | 1 | Version/ HDR length | 4Xh | Compare | Check IPv4 and header length |
| 15+D+S | 1 | Type of Service | - | Ignore | |
| 16+D+S | 2 | Packet Length | - | Ignore | |
| 18+D+S | 2 | Identification | - | Ignore | |
| 20+D+S | 2 | Fragment Info | 0x00 | Compare | |
| 22+D+S | 1 | Time to live | - | Ignore | |
| 23+D+S | 1 | Protocol | 0x29 | Compare | Ipv6 |
| 24+D+S | 2 | Header Checksum | - | Ignore | |
| 26+D+S | 4 | Source IP Address | - | Ignore | |
| 30+D+S | 4 | Destination IP Address | - | Ignore | |

| Offset | # of bytes | Field | Value | Action | Comment |
|---|---|---|---|---|---|
| 34+D+S | N | Possible IP Options | | Ignore | |
| **Ipv6 header** | | | | | |
| 34+D+S+N | 1 | Version/ Traffic Class | 6Xh | Compare | Check IPv6 |
| 35+D+S+N | 3 | Traffic Class/Flow Label | - | Ignore | |
| 38+D+S+N | 2 | Payload Length | - | Ignore | |
| 40+D+S+N | 1 | Next Header | IPv6 extension header or 0x11 | Check | IPv6 extension headers: |
| 41+D+S+N | 1 | Hop Limit | - | Ignore | |
| 42+D+S+N | 16 | Source Address | - | Ignore | |
| 58+D+S+N | 16 | Destination Address | | Ignore | |
| 74+D+S+N | B | Possible IPv6 Next Headers | - | Ignore | |
| **UDP header** | | | | | |
| 74+D+S+N+B | 2 | Source Port | Not (0x801) | Check | Not NFS packet |
| 76+D+S+N+B | 2 | Destination Port | Not (0x801) | Check | Not NFS packet |
| 78+D+S+N+B | 2 | Length | - | Ignore | |
| 80+D+S+N+B | 2 | Checksum | - | Ignore | |

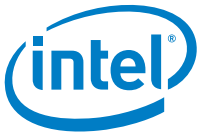
In this case the packet is split after (82+D+S+N+B) bytes.

N = (IP HDR length - 5) * 4.

#### A.2.1.4.5 Type 1.4.3: Ethernet (VLAN/SNAP) Ipv4 Ipv6 TCP

This type contain only Ethernet header, Ipv4 header, IPv6 header and TCP header.

| Offset | # of bytes | Field | Value | Action | Comment |
|---|---|---|---|---|---|
| 0 | 6 | Destination Address | | Ignore | MAC Header – processed by main address filter, or broadcast filter. |
| 6 | 6 | Source Address | | Ignore | |
| 12 | S=(0/4) | Possible VLAN Tag | 8100 | Check | |
| 12+S | D=(0/8) | Possible Len/LLC/SNAP Header | | Check | |



| Offset | # of bytes | Field | Value | Action | Comment |
|---|---|---|---|---|---|
| 12+D+S | 2 | Type | 0800h | Compare | IP |
| **Ipv4 header** | | | | | |
| 14+D+S | 1 | Version/ HDR length | 4Xh | Compare | Check IPv4 and header length |
| 15+D+S | 1 | Type of Service | - | Ignore | |
| 16+D+S | 2 | Packet Length | - | Ignore | |
| 18+D+S | 2 | Identification | - | Ignore | |
| 20+D+S | 2 | Fragment Info | 0x00 | Compare | |
| 22+D+S | 1 | Time to live | - | Ignore | |
| 23+D+S | 1 | Protocol | 0x2A | Compare | Ipv6 |
| 24+D+S | 2 | Header Checksum | - | Ignore | |
| 26+D+S | 4 | Source IP Address | - | Ignore | |
| 30+D+S | 4 | Destination IP Address | - | Ignore | |
| 34+D+S | N | Possible IP Options | | Ignore | |
| **Ipv6 header** | | | | | |
| 34+D+S+N | 1 | Version/ Traffic Class | 6Xh | Compare | Check IPv6 |
| 35+D+S+N | 3 | Traffic Class/Flow Label | - | Ignore | |
| 38+D+S+N | 2 | Payload Length | - | Ignore | |
| 40+D+S+N | 1 | Next Header | Ipv6 extension header<br>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 | |

| Offset | # of bytes | Field | Value | Action | Comment |
|---|---|---|---|---|---|
| 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.

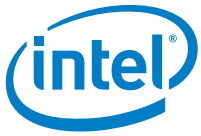
- T = D+S+N+B
- N = (IP HDR length - 5) * 4.
- F = (TCP HDR length - 5)*4

## 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 | S=(0/4) | Possible VLAN Tag | 8100 | Check | |
| 12+S | D=(0/8) | Possible Len/LLC/SNAP Header | | Check | |
| **IPv6 header** | | | | | |
| 12+D+S | 2 | Type | 86DDh | Compare | IP |
| 14+D+S | 1 | Version/ Traffic Class | 6Xh | Compare | Check IPv6 |
| 15+D+S | 3 | Traffic Class/Flow Label | - | Ignore | |
| 18+D+S | 2 | Payload Length | - | Ignore | |



| Offset | # of bytes | Field | Value (hex) | Action | Comment |
|---|---|---|---|---|---|
| 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 be 0x11/0x06 (TCP/UDP).

#### A.2.2.1.1 Type 2.2: Ethernet (VLAN/SNAP) Ipv6 UDP

This type contains only Ethernet header, Ipv6 header, and UDP header.

| Offset | # of bytes | Field | Value (hex) | Action | Comment |
|---|---|---|---|---|---|
| 0 | 6 | Destination Address | | Ignore | MAC Header – processed by main address filter, or broadcast filter. |
| 6 | 6 | Source Address | | Ignore | |
| 12 | S=(0/4) | Possible VLAN Tag | | Check | |
| 12+S | D=(0/8) | Possible Len/LLC/SNAP Header | | Check | |
| **IPv6 header** | | | | | |
| 12+D+S | 2 | Type | 86DDh | Compare | IP |
| 14+D+S | 1 | Version/ Traffic Class | 6Xh | Compare | Check IPv6 |
| 15+D+S | 3 | Traffic Class/Flow Label | - | Ignore | |
| 18+D+S | 2 | Payload Length | - | Ignore | |
| 20+D+S | 1 | Next Header | IPv6 next header types<br>Or<br>0x11 | Check | |
| 21+D+S | 1 | Hop Limit | - | Ignore | |
| 22+D+S | 16 | Source Address | - | Ignore | |
| 38+D+S | 16 | Destination Address | | Ignore | |

| Offset | # of bytes | Field | Value (hex) | Action | Comment |
|---|---|---|---|---|---|
| 54+D+S | N | Possible IPv6 Next Headers | - | Ignore | |
| **UDP header** | | | | | |
| 54+D+S+N | 2 | Source Port | Not (0x801) | | Not NFS packet |
| 56+D+S+N | 2 | Destination Port | Not (0x801) | | Not NFS packet |
| 58+D+S+N | 2 | Length | - | | |
| 60+D+S+N | 2 | Checksum | - | | |

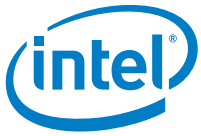
In this case the packet is split after (62+D+S+N) bytes.

- 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 TCP header.

| Offset | # of bytes | Field | Value (hex) | Action | Comment |
|---|---|---|---|---|---|
| 0 | 6 | Destination Address | | Ignore | MAC Header – processed by main address filter, or broadcast filter. |
| 6 | 6 | Source Address | | Ignore | |
| 12 | S=(0/4) | Possible VLAN Tag | | Check | |
| 12+S | D=(0/8) | Possible Len/LLC/SNAP Header | | Check | |
| **IPv6 header** | | | | | |
| 12+D+S | 2 | Type | 86DDh | Compare | IP |
| 14+D+S | 1 | Version/ Traffic Class | 6Xh | Compare | Check IPv6 |
| 15+D+S | 3 | Traffic Class/Flow Label | - | Ignore | |
| 18+D+S | 2 | Payload Length | - | Ignore | |
| 20+D+S | 1 | Next Header | IPv6 next header types<br>Or<br>TCP | Check | |
| 21+D+S | 1 | Hop Limit | - | Ignore | |
| 22+D+S | 16 | Source Address | - | Ignore | |
| 38+D+S | 16 | Destination Address | | Ignore | |



| Offset | # of bytes | Field | Value (hex) | Action | Comment |
|---|---|---|---|---|---|
| 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 82599).

### A.2.4 Type 4: NFS Packets

NFS headers can come in all the frames that contain UDP/TCP header. The NFS (and RPC headers) are extensions to these types of packets: 1.2, 1.3, 1.4.2, 1.4.3, 2.2, 2.3 that were presented in the previous sections. In this section only the NFS (and RPC) header is described and to its length should be added the length of the primary type of the packet.

the 82599 starts looking within the UDP/TCP payload to check whether it contain an NFS header if either the source or destination port of the TCP/UDP equal to 0x801.

```
Destination port equal to 0x801 => NFS write request => NFS write
request (as received by the NFS server).

Source port equal to 0x801       => NFS read request => read reply (as
received by the NFS client).
```

The VSZ/CSZ fields are 4 bytes long but there actual values are less than 2 words by definition, so hardware only checks the lower 2 bytes of these size fields.

RPC read requests are not described in this document since they contain only headers and no data -> no need to split.

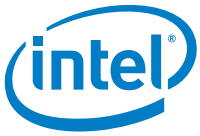
NFS over TCP is problematic - in fact, RPC header may appear in the middle of the frame. It remains to be checked if the software can support always putting RPC right next to the UDP/TCP header, but it doesn't have to.

## A.2.4.1 Type 4.1: NFS Write Request

In all the write requests the destination port of the TCP/UDP header must be 0x801.

### A.2.4.1.1 Type 4.1.1: NFS Write Request (NFSv2)

| Offset | # of bytes | Field | Value (hex) | Action | Comment |
|---|---|---|---|---|---|
| **RPC header** | | | | | |
| 0 | D =(0/4) | Record Header | - | Ignore | If the previous header was a TCP header then this field contains 4 bytes. |
| 0+D | 4 | Message type | 0x00 | Compare | |
| 4+D | 4 | RPC version | 0x02 | Compare | |
| 8+D | 4 | RPC program | 0x18A63 | Compare | |
| 12+D | 4 | Program version | 0x02 | Compare | |
| 16+D | 4 | Procedure | 0x08 | Compare | |
| 20+D | 4 | Credentials Size (CSZ) | <400 | Check | |
| 24+D | B | Credentials Data | - | Ignore | B = (CSZ pad 4) |
| 24+D+B | 4 | Verifier Flavor | - | Ignore | |
| 28+D+B | 4 | Verifier Size (VSZ) | <400 | Check | |
| 32+D+B | F | Verifier Data | | Ignore | F = (VSZ pad 4) |
| **NFS header** | | | | | |
| 32+D+B+F | 32 | handle | | Ignore | |
| 64+D+B+F | 4 | begin offset | | Ignore | |
| 68+D+B+F | 4 | Offset | | Ignore | |
| 72+D+B+F | 4 | Total count | | Ignore | |
| 76+D+B_F | 4 | Data len | | Ignore | |



In this case the packet is split after (80+D+B+F) bytes should be added to the UDP/TCP type that was already parsed.

#### A.2.4.1.2 Type 4.1.2: NFS Write Request (NFSv3)

| Offset | # of bytes | Field | Value (hex) | Action | Comment |
|---|---|---|---|---|---|
| **RPC header** | | | | | |
| 0 | D =(0/4) | Record Header | - | Ignore | If the previous header was a TCP header than this field contains 4 bytes. |
| 0+D | 4 | Message type | 0x00 | Compare | |
| 4+D | 4 | RPC version | 0x02 | Compare | |
| 8+D | 4 | RPC program | 0x18A63 | Compare | |
| 12+D | 4 | Program version | 0x03 | Compare | |
| 16+D | 4 | Procedure | 0x07 | Compare | |
| 20+D | 4 | Credentials Size (CSZ) | <400 | Check | |
| 24+D | B | Credentials Data | - | Ignore | B = (CSZ padded to 4) |
| 24+D+B | 4 | Verifier Flavor | - | Ignore | |
| 28+D+B | 4 | Verifier Size (VSZ) | <400 | Check | |
| 32+D+B | F | Verifier Data | | Ignore | F = (VSZ padded to 4) |
| **NFS header** | | | | | |
| 32+D+B+F | 4 | Fhandle_size | <64 | Check | |
| 36+D+B+F | S | fhandle | | Ignore | S = (Fhandle_size padded to 4) |
| 36+D+B+F+S | 8 | Offset | | Ignore | |
| 44+D+B+F+S | 4 | Count | | Ignore | |
| 48+D+B+F+S | 4 | Stable_how | | Ignore | |
| 52+D+B+F+S | 4 | Data len | | Ignore | |

In this case the packet is split after (56+D+B+F+S) bytes should be added to the UDP/TCP type that was already parsed.

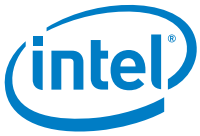
#### A.2.4.1.3 Type 4.1.3: NFS Write Request (NFSv4)

| Offset | # of bytes | Field | Value (hex) | Action | Comment |
|---|---|---|---|---|---|
| **RPC header** | | | | | |
| 0 | D =(0/4) | Record Header | - | Ignore | If the previous header was a TCP header than this field contains 4 bytes. |
| 0+D | 4 | Message type | 0x00 | Compare | |
| 4+D | 4 | RPC version | 0x02 | Compare | |
| 8+D | 4 | RPC program | 0x18A63 | Compare | |
| 12+D | 4 | Program version | 0x04 | Compare | |
| 16+D | 4 | Procedure | 0x26 | Compare | |
| 20+D | 4 | Credentials Size (CSZ) | <400 | Check | |
| 24+D | B | Credentials Data | - | Ignore | B = (CSZ pad 4) |
| 24+D+B | 4 | Verifier Flavor | - | Ignore | |
| 28+D+B | 4 | Verifier Size (VSZ) | <400 | Check | |
| 32+D+B | F | Verifier Data | | Ignore | F = (VSZ pad 4) |
| **NFS header** | | | | | |
| 32+D+B+F | 8 | State id | | Ignore | |
| 40+D+B+F | 8 | Offset | | Ignore | |
| 48+D+B+F | 4 | Stable_how | | Ignore | |
| 52+D+B+F | 4 | Data len | | Ignore | |

In this case the packet is split after (100+D+F) bytes should be added to the UDP/TCP type that was already parsed.

#### A.2.4.1.4 Type 4.2.1: NFS Read Response (NFSv3)

| Offset | # of bytes | Field | Value (hex) | Action | Comment |
|---|---|---|---|---|---|
| **RPC header** | | | | | |
| 0 | D =(0/4) | Record Header | - | Ignore | If the previous header was a TCP header than this field contains 4 bytes. |
| 0+D | 4 | XID | | Ignore | |
| 4+D | 4 | Message type | 0x01 | Compare | |



| Offset | # of bytes | Field | Value (hex) | Action | Comment |
|---|---|---|---|---|---|
| 8+D | 4 | Reply status | 0x00 | Ignore | '0' means OK and only if this value is '0' is there additional data. |
| 12+D | 4 | Verifier Flavor | - | Ignore | |
| 16+D | 4 | Verifier Size (VSZ) | <400 | Check | |
| 20+D | F | Verifier Data | - | Ignore | F = (VSZ pad 4) |
| 20+D+F | 4 | Accept status | 0x00 | Ignore | '0' means OK |
| **NFS header** | | | | | |
| 24+D+F | 4 | Status | 0x00 | Ignore | |
| 28+D+F | 68 | Attributes | - | Ignore | |
| 96+D+F | 4 | Data len | - | Ignore | |

In this case the packet is split after (40+D+F+S) bytes should be added to the UDP/TCP type that was already parsed.

#### A.2.4.1.5 Type 4.2.1: NFS Read Response (NFSv4)

| Offset | # of bytes | Field | Value (hex) | Action | Comment |
|---|---|---|---|---|---|
| **RPC header** | | | | | |
| 0 | D =(0/4) | Record Header | - | Ignore | If the previous header was a TCP header than this field contains 4 bytes. |
| 0+D | 4 | XID | - | Ignore | |
| 4+D | 4 | Message type | 0x01 | Compare | |
| 8+D | 4 | Reply status | 0x00 | Ignore | '0' means OK and only if this value is '0' is there additional data. |
| 12+D | 4 | Verifier Flavor | - | Ignore | |
| 16+D | 4 | Verifier Size (VSZ) | <400 | Check | |
| 20+D | F | Verifier Data | - | Ignore | F = (VSZ pad 4) |
| 20+D+F | 4 | Accept status | 0x00 | Ignore | '0' means OK |
| **NFS header** | | | | | |
| 24+D+F | 4 | Status | 0x00 | Ignore | |
| | 4 | Attr_follow | - | Check | |

| Offset | # of bytes | Field | Value (hex) | Action | Comment |
|---|---|---|---|---|---|
| 28+D+F | S | Attributes | - | Ignore | Attr_flow=1? S=84: S=0 |
| 28+D+F+S | 4 | Count | - | Ignore | |
| 32+D+F+S | 4 | Eof | - | Ignore | |
| 36+D+F+S | 4 | Data len | - | Ignore | |

In this case the packet is split after (36+D+F) bytes should be added to the UDP/TCP type that was already parsed.

# A.3 IPsec Formats Run Over the Wire

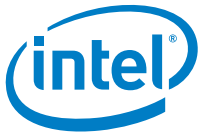
This section describes the IPsec packet encapsulation formats run over the wire by IPsec packets concerned with the off load in either Tx or Rx direction.

The following legend is valid for the figures Figure A-12 through Figure A-17 of this appendix.

| |
|---|
| Shaded fields correspond to the portion of the data that is protected by the integrity check. |
| Yellow colored fields are mutable fields that might be changed when traveling between the source and the destination and shall thus be zeroed when computing ICV or when encrypting/decrypting. |
| Cyan colored fields correspond to the portion of data that is protected for both integrity and confidentiality. |
| Non-colored fields are not protected either for integrity or for confidentiality. |

## A.3.1 AH Formats

- IPv4 header:
  - IP total length (2 bytes) - Total IP packet length in bytes, including IP header, AH header, TCP/UDP header, and TCP/UDP payload.
  - Protocol (1 byte) - AH protocol number, i.e. value 51.
- IPv6 header:
  - IP payload length (2 bytes) - IP payload length in bytes, including AH header, TCP/UDP header, and TCP/UDP payload.
  - Next header (1 byte) - AH protocol number, i.e. value 51.
- AH header:
  - Next header (1 byte) - Layer4 protocol number, 6 for TCP, 17 for UDP, etc.
  - AH length (1 byte) - Authentication Header length in 32-bits Dwords units, minus “2”, i.e. for AES-128 its value is 7 for IPv4 and 8 for IPv6.



  - — Reserved (2 bytes) - must be set to zero.
  - — SPI (4 bytes) - arbitrary 32-bits Security Parameters Index allocated by the receiver to identify the SA to which the incoming packet is bound. It is required that the local OS will allocate SPIs in a unique manner per local IP address.
  - — SN (4 bytes) - unsigned 32-bit Sequence Number that contains a counter value that increases by one for each Ethernet frame sent. It is initialized to 0 by the sender (and the receiver) when the SA is established, i.e. the first packet sent using a given SA will have a sequence number of 1.
  - — IV (8 bytes) - Initialization Vector to be used 'as is' in the nonce input to AES-128 crypto engine, but it must be zeroed prior to using it in the AAD input to the engine.
  - — ICV (16 bytes) - Integrity Check Value for this packet, authentication tag output of the AES-128 crypto engine. As being part of the AH header, this field is also included in the AAD input to the crypto engine, and it should be zeroed prior to the computation.
  - — ICV Padding (4 bytes) - *explicit* padding bytes appended to the ICV field in IPv6, as it is required to maintain the (Authentication) extension header length as a multiple of 64-bits. By *explicit* we mean that these bytes are sent over the wire. It is formed by 4 arbitrary bytes that need not be random to achieve security. For TSO, it will be replicated from the header provided by the driver in every frame.
- L4 header (for example - TCP/UDP): Length (in bytes) depend on the protocol.
- L4 payload (for example - TCP/UDP): Can be any length in bytes

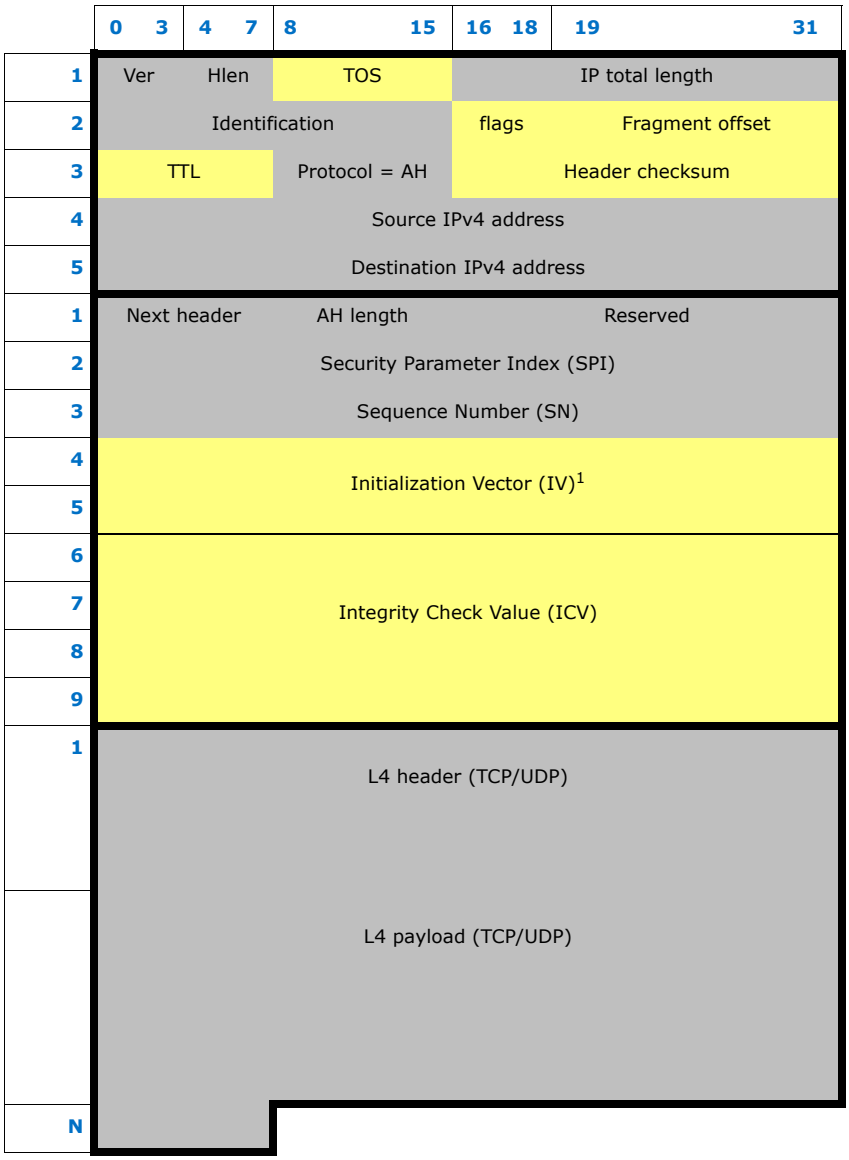



**Figure A-12 AH packet over IPv4**

1. IV field has been colored in Yellow as it must be zeroed in the AAD input to AES-128 crypto engine, in spite of this it is NOT zeroed in the nonce input to the engine.

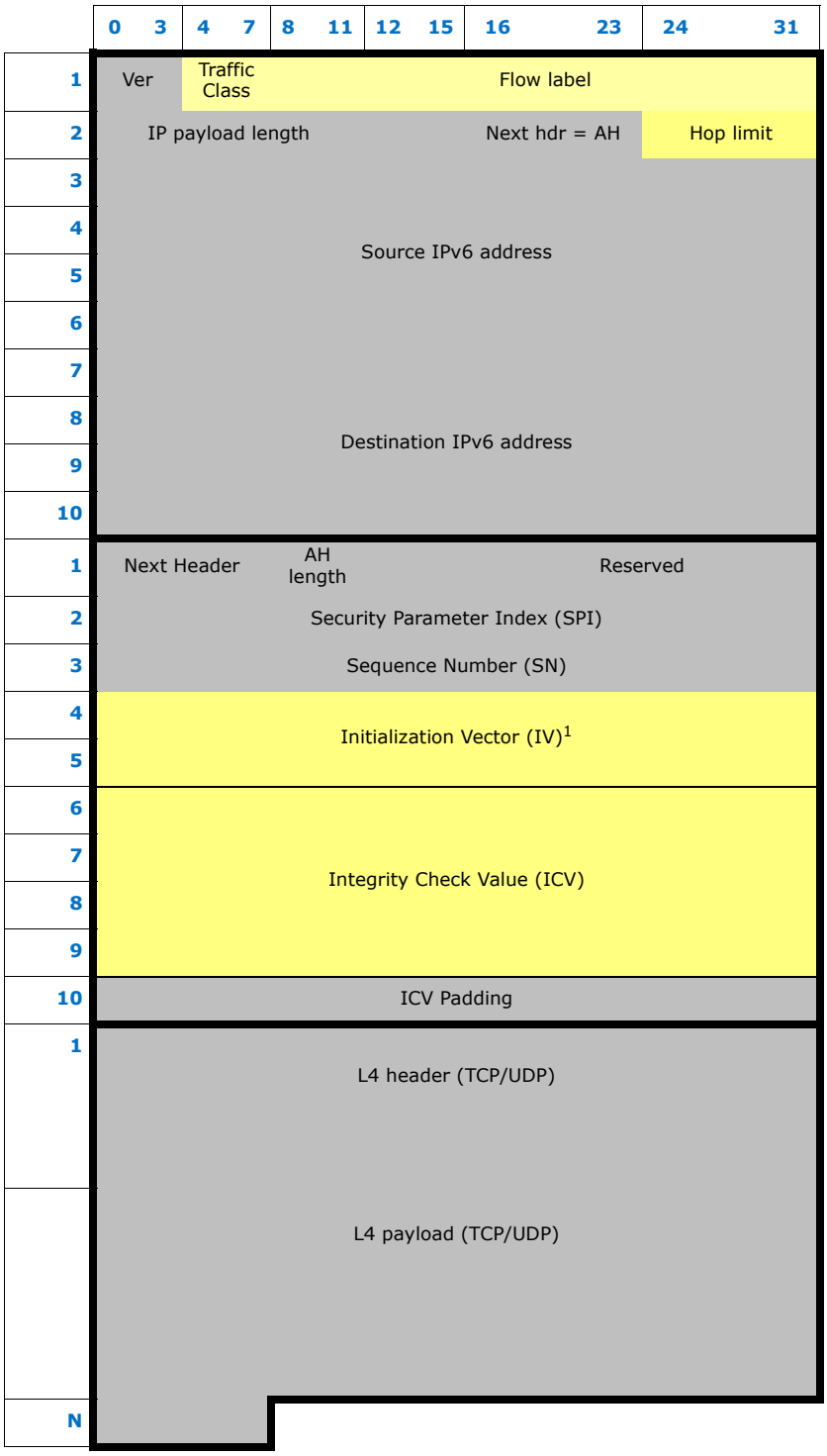



**Figure A-13 AH packet over IPv6**

1. IV field has been colored in Yellow as it must be zeroed in the AAD input to AES-128 crypto engine, in spite of this, it is NOT zeroed in the nonce input to the engine.

## A.3.2 ESP Formats

- IPv4 header:
  - IP total length (2 bytes) - Total IP packet length in bytes, including IP header, ESP header, TCP/UDP header, TCP/UDP payload, ESP trailer, and ESP ICV if present.
  - Protocol (1 byte) - ESP protocol number, i.e. value 50.
- IPv6 header:
  - IP payload length (2 bytes) - IP payload length in bytes, including ESP header, TCP/UDP header, TCP/UDP payload, ESP trailer, and ESP ICV if present.
  - Next header (1 byte) - ESP protocol number, i.e. value 50.
- ESP header:
  - SPI (4 bytes) - arbitrary 32-bit Security Parameters Index allocated by the receiver to identify the SA to which the incoming packet is bound. It is required that the local OS will allocate SPIs in a unique manner per local IP address.
  - SN (4 bytes) - unsigned 32-bit Sequence Number that contains a counter value that increases by one for each Ethernet frame sent. It is initialized to 0 by the sender (and the receiver) when the SA is established, i.e. the first packet sent using a given SA will have a sequence number of 1.
  - IV (8 bytes) - Initialization Vector to be used in the nonce input field of the AES-128 crypto engine, and for authenticated-only ESP packets it is used also in the AAD input.
- L4 header (for example - TCP/UDP):
  - Length (in bytes) depend on the protocol.
  - TCP/UDP checksum computed from the TCP/UDP header up to the end of TCP payload, excluding ESP trailer.
  - TCP/UDP header is encrypted if ESP encryption is required.
- L4 payload (for example - TCP/UDP): Can be any length in bytes. It is encrypted if ESP encryption is required.
- ESP trailer:
  - Padding (0-255 bytes) - unsigned 1-byte integer values, with its content started by 1 and making up a monotonically increasing sequence: 1, 2, 3,... Though in Tx it will only be 0-15 bytes long, in Rx it might be longer, if the sender's policy is to hide the packet length.
  - Padding length (1 byte) - Number of explicit padding bytes (Padding length and Next header bytes excluded) required to get 4-bytes alignment of the ESP header, TCP/UDP header, TCP/UDP payload, and ESP trailer. By explicit we mean that these bytes are sent over the wire. A remote IPsec implementation may also add more padding bytes (up to 255-bytes) than the minimum required for getting the 4-bytes alignment with the aim of hiding the packet length.
  - Next header (1 byte) - Layer4 protocol number, 6 for TCP, 17 for UDP, etc.
  - ESP trailer will be encrypted if ESP encryption is required.

- ESP ICV (16 bytes) - Integrity Check Value for this packet, authentication tag output of the AES-128 crypto engine.

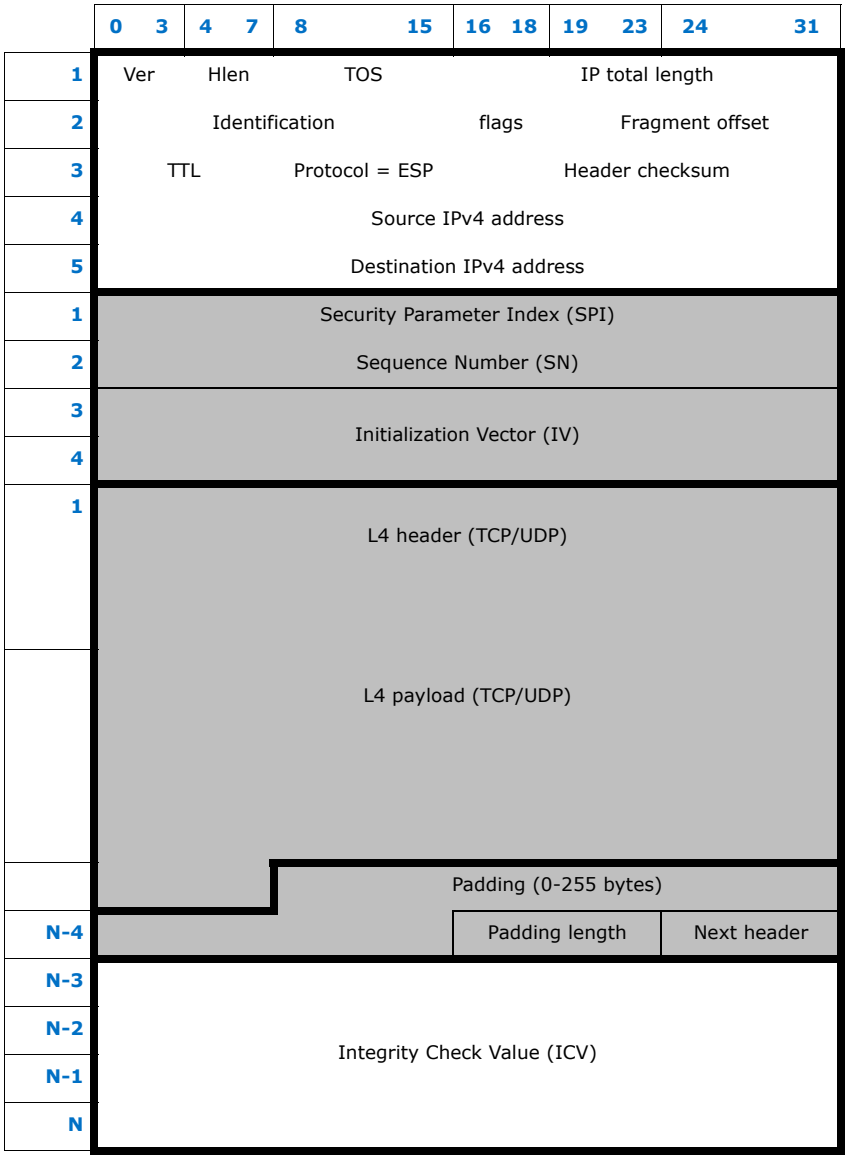



**Figure A-14 Authenticated only ESP packet over IPv4**

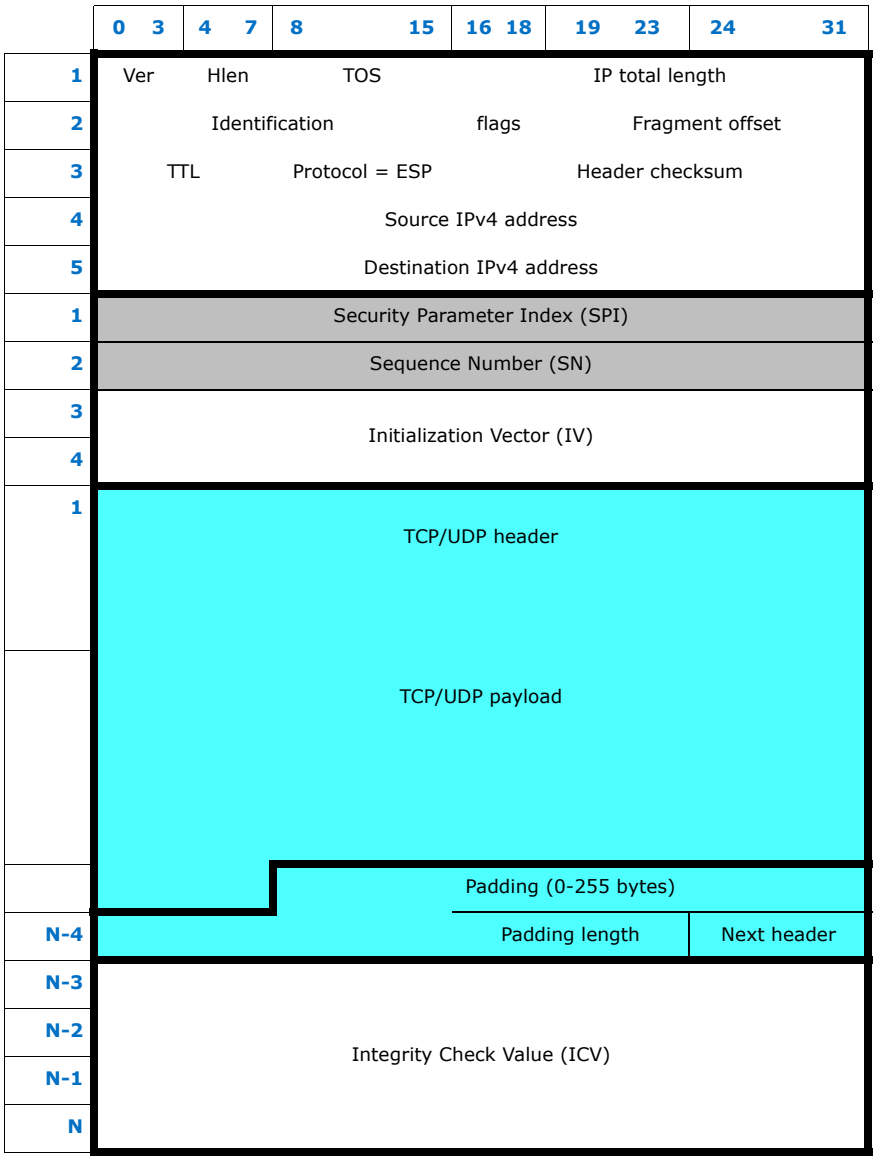



**Figure A-15 Authenticated & encrypted ESP packet over IPv4**

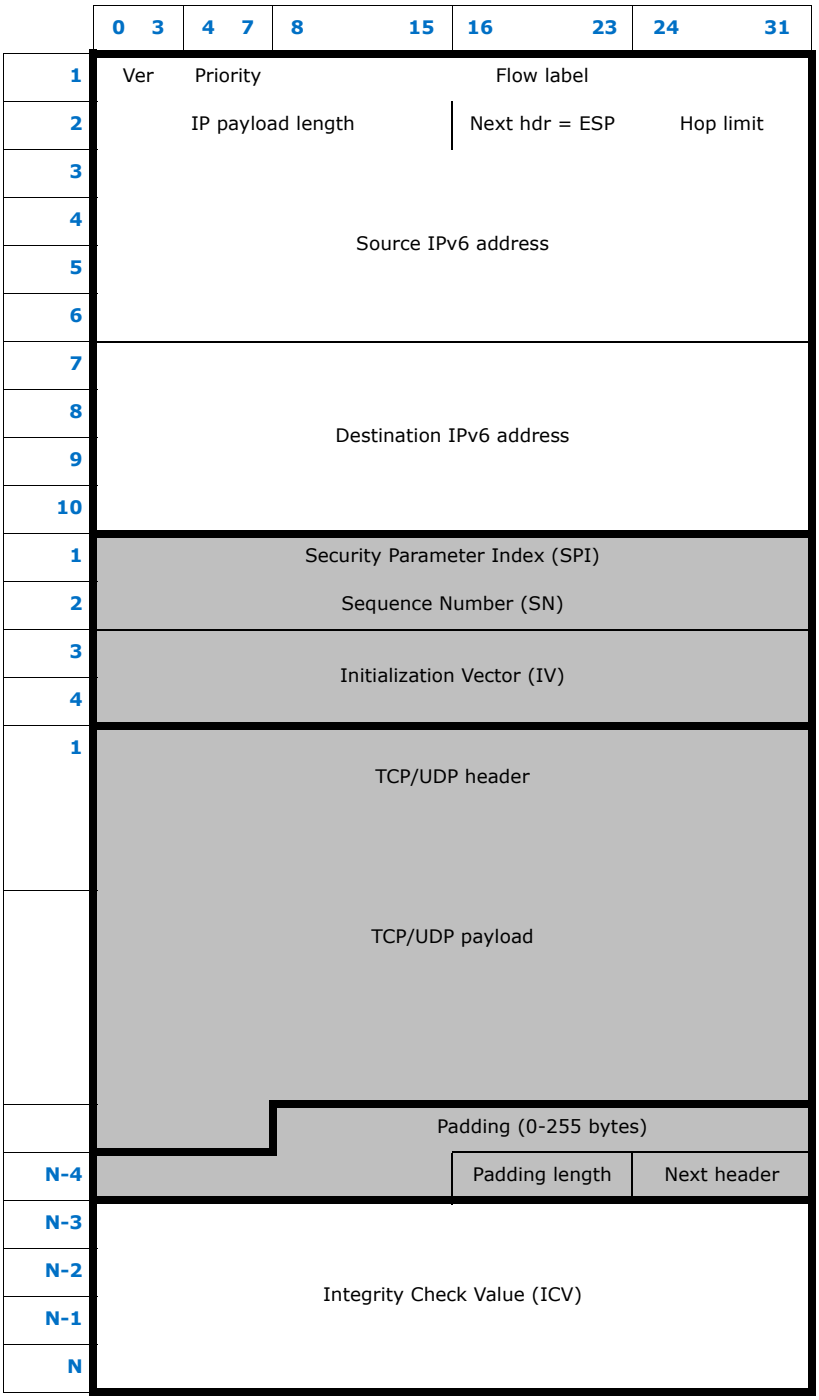



**Figure A-16 Authenticated only ESP packet over IPv6**

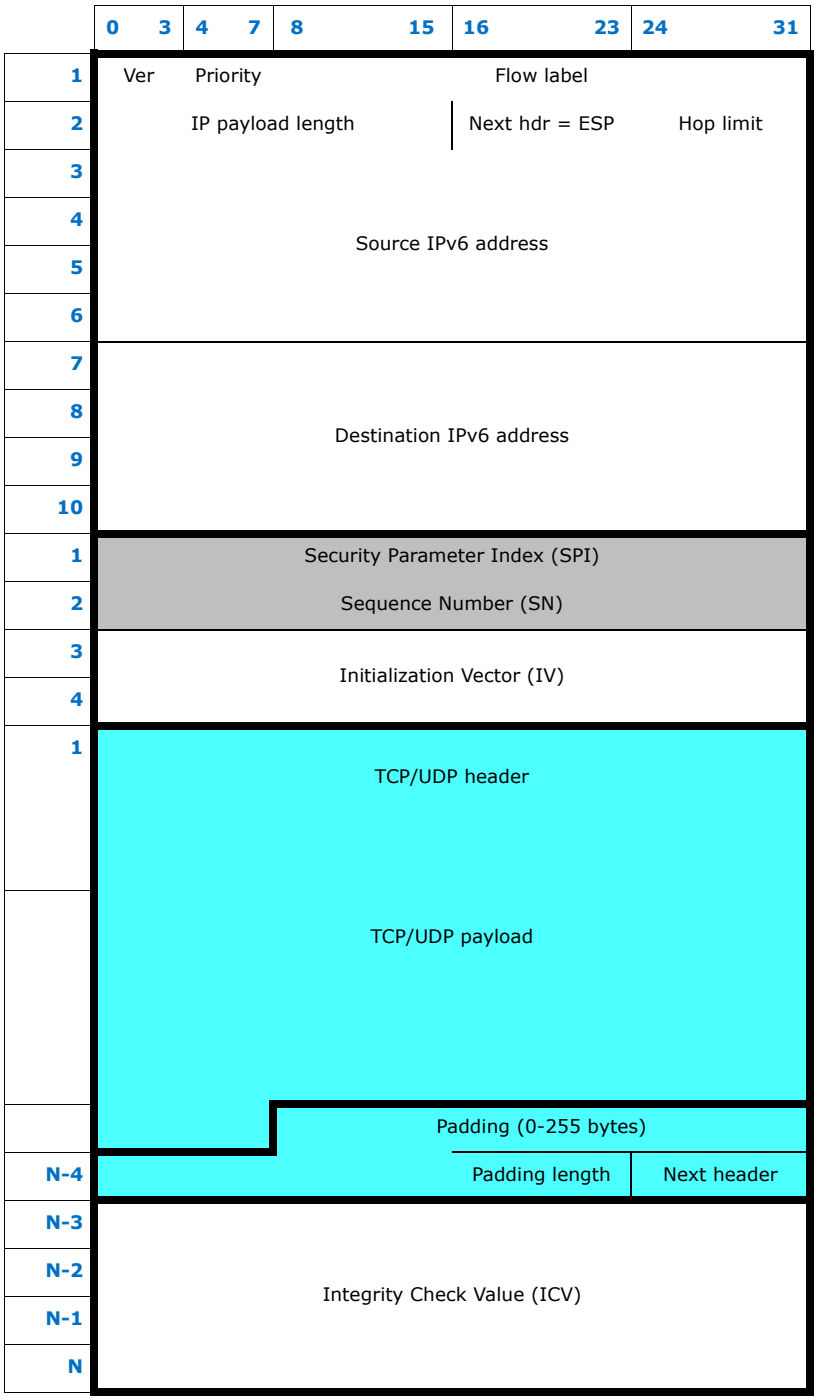



**Figure A-17 Authenticated and encrypted ESP packet over IPv6**

For authenticated and encrypted ESP packets, though it is used in the Nonce input to the AES-128 crypto engine, the IV field was left non-colored because it is not a part of the ADD or the Plaintext input fields. Refer to Section 7.12.7.

# A.4 BCN Frame Format

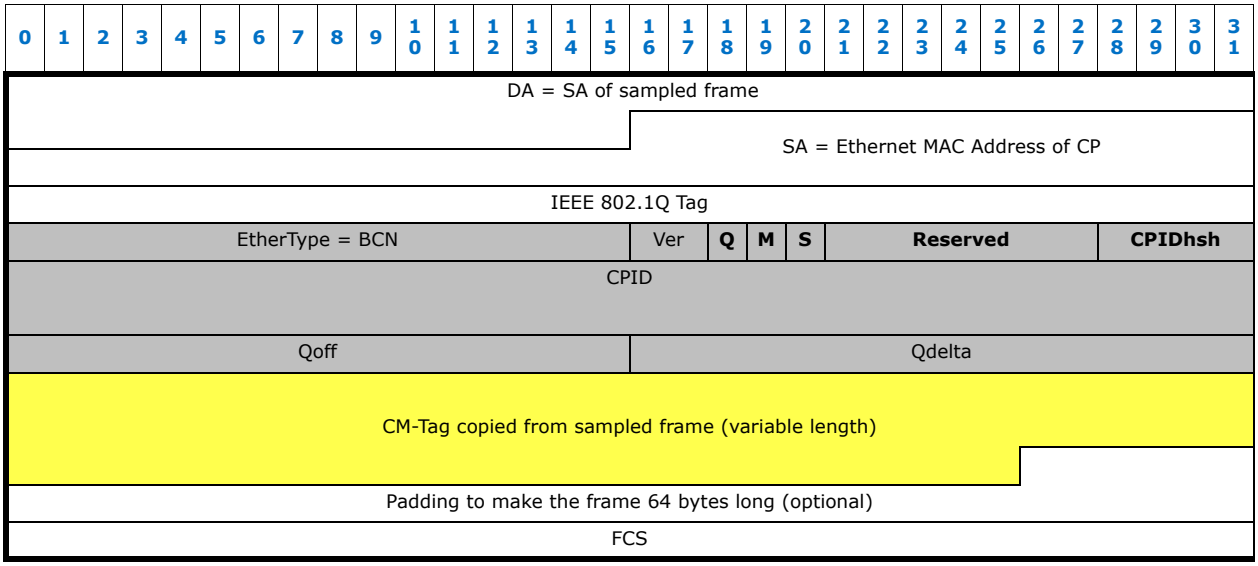



**Figure A-18 BCN frame format**

- 802.1Q Tag is copied from the sampled frame. The 802.1p priority field is set either to the priority of the sampled frame or to a configurable priority. It is preferable to use highest priority or the priority value corresponding to the network management/ control traffic in order to minimize the latency experienced by BCN frames.
- EtherType = TBD, is programmable via the ETQF register, because IEEE802.1au Standard has not assigned a value yet. It is different from the EtherType included in the CM-tags.
- Ver stands for the Version of the BCN protocol.
- Special congestion conditions, Q,M, and S bits:
  - Q-bit indicates that the Qdelta field has saturated.
  - M-bit indicates that a Mild congestion threshold was passed.
  - S-bit indicates that a Severe congestion threshold was passed.
- CPIDhsh is a hash tag computed by the Congestion Point over its own CPID. It will be used by the CP when returned in a rate-limited frame to determine whether a positive BCN control frame shall be sent for that flow - if the congested conditions disappeared. If the CPIDhsh hash tag does not match with the congestion point's CPID, for sure the rate-limited frame was not associated with that congestion point, and thus no positive BCN frame will be sent for that flow by the congested point.
- CPID stands for Congestion Point IDentifier, which should include the congestion point Ethernet MAC Address, as well as a local identifier for the local congestion entity, usually a queue in the switch.

- Qoff and Qdelta contains the queue filling status at the congestion point, where Qoff is the algebraic offset of the current queue length with respect to the equilibrium threshold Qeq, and Qdelta is the change in length of the queue since the last sampled frame. Both are expressed in pages of 64-bytes.
- FCS is the Frame Check Sequence of Ethernet frames.

**Note:** The shaded fields shall be inserted just after the 802.1Q VLAN tag, and thus it can be located as follow:

1. MAC DA, SA
2. LinkSec tag - all data after this tag is encrypted.
3. Double VLAN - follows the same rules described above for 802.1Q tag.
4. 802.1Q VLAN
5. Shaded fields

# A.5 FCoE Framing

Related Standards

- FC-FS-2 - FRAMING AND SIGNALING-2 (FC-FS-2) Rev 1.00
- FCoE - Fibre Channel over Ethernet Draft Presented at the T11 on May 2007

## A.5.1 FCoE Frame Format

FC over Ethernet packets encapsulate FC frames as shown in Figure A-19 and Figure A-20. Maximum expected FCoE frame size is 2164 bytes. This size does not include FC extension headers (not expected in FCoE), without optional LinkSec encapsulation (expected to be off-loaded by the hardware) and without BCN tag (not expected in receive).

All fields in the FCoE frame are treated as any other field in the network. The MS Byte is first on the wire and the LS bit on each byte is first on the wire. This rule applies for all fields including those ones that span on non complete byte boundaries such as the FCoE VER field.

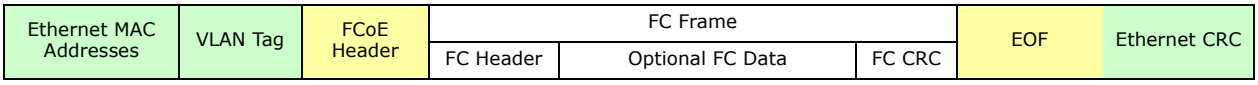
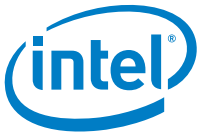



**Figure A-19 Ethernet Encapsulation to FC frames**



| | msb 31 lsb 24<br>First byte on the wire | msb 23 lsb 16 | msb 15 lsb 12 | 11 lsb 8 | msb 7 ... . . lsb 0<br>Last byte on the wire |
|---|---|---|---|---|---|
| 0 | Destination Ethernet MAC Address | | | | |
| 4 | | | | | |
| 8 | Source Ethernet MAC Address | | | | |
| 12 | 802.1Q Tag (VLAN + Priority) | | | | |
| 16 | FCoE Ethernet Type = 0x8906 | | Ver | Reserved | |
| 20 | Reserved | | | | |
| 24 | Reserved | | | | |
| 28 | Reserved | | | | SOF |
| 32 | Routing Control (R_CTL) | Destination Identification (D_ID) | | | |
| 36 | Class Specific Control (CS_CTL) | Source Identification (S_ID) | | | |
| 40 | TYPE | Frame Control (F_CTL) | | | |
| 44 | Sequence ID (SEQ_ID) | Data Field Control (DF_CTL) | Sequence Count (SEQ_CNT) | | |
| 48 | Originator Exchange ID (OX_ID) | | Responder Exchange ID (RX_ID) | | |
| 52 | Parameter (PARAM) | | | | |
| 0...N | Optional FC Data... always 4 byte align (including optional FC padding) | | | | |
| 56 +N | Fibre Channel CRC (FC_CRC) | | | | |
| N+8 | EOF | Reserved | | | |
| M | Ethernet CRC | | | | |

**Figure A-20 FCoE Packet Structure**

### A.5.1.1 Ethernet MAC Addresses

L2 destination and Source Ethernet MAC Addresses (each of them is 6 bytes long). The Ethernet MAC Address of the target is assumed to be assigned by the network. It could be done by the FCoE repeater or LAN administrator. The mechanism that is used for Ethernet MAC Address assignment and Ethernet MAC Address detection is outside of the scope of this document.

### A.5.1.2 802.1Q Tag

802.1Q tagging is mandatory for FCoE usage. FCoE assumes Reedtown functionality which provides class-based FC and BCN. These functions require 802.1Q tag presence to define packet priority.

### A.5.1.3 FCoE Header

The FCoE header is composed of a new FCoE Ethernet type, FCoE version tag and the Start Of Frame tag.

FCoE Ethernet Type

Equals 0x8906

Version (Ver)

A 4 bit field that indicates the FCoE protocol version number. the 82599 supports FCoE version as defined by FCRXCTRL.FCOEVER.

Start of Frame (SOF)

The FCoE Start of frame is a subset of the FC-FS-2 SOF codes as defined in the Table below:

**Table A-1 E_SOF Mapping**

| FC SOF | FCoE SOF Code | FC Traffic Class | Comment |
|---|---|---|---|
| SOFf | 0x28 | F | Fabric start of frame. Not expected in an FCoE. |
| SOFi2 | 0x2D | 2 | used in the first frames in a sequence |
| SOFn2 | 0x35 | 2 | used in all but first frame in a sequence |
| SOFi3 | 0x2E | 3 | used in the first frames in a sequence |
| SOFn3 | 0x36 | 3 | used in all but first frame in a sequence |

### A.5.1.4 FCoE Packet Encapsulation Trailer

The FCoE trailer is composed of an End of frame, optional padding and Ethernet CRC.

End of Frame (EOF):

The FCoE End of frame maps the FC-FS-2 EOF codes as defined in Table A-2:

**Table A-2 EOF Mapping**

| FC EOF | FCoE EOF Code | FC Traffic Class | Comment |
|---|---|---|---|
| EOFn | 0x41 | 2, 3, 4, F | normal EOF |
| EOFt | 0x42 | 2, 3, 4, F | EOF Terminate used to close a sequence |
| EOFni | 0x49 | 2, 3, 4, F | EOF Invalid indicating that the frame content is invalid. |
| EOFa | 0x50 | 2, 3, 4, F | EOF Abort |

Ethernet CRC:

The IEEE 802.3 CRC as defined by the following polynomial:
$X_{32}+X_{26}+X_{23}+X_{22}+X_{16}+X_{12}+X_{11}+X_{10}+X_8+X_7+X_5+X_4+X_2+X+1$

## A.5.2 FC Frame Format

**Note:** This section is provided as a background on FC and is not required for the HW 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 the Figure A-21 below while relevant fields are detailed in this section.

**Figure A-21 FC Frame Format**

### A.5.2.1 FC SOF and EOF

FC Start of frame (SOF) delimiter and End of frame (EOF) delimiter. In FCoE frames, the SOF and EOF fields in the FC frame are extracted and reflected in the FCoE encapsulation. The SOF and EOF codes that are reflected in the FCoE framing are shown in Table  and Table .

### A.5.2.2 FC CRC

The Cyclic Redundancy Check (CRC) is a four byte field that follows the Data Field. It enables end to end integrity checking on the whole FC frame. The HW adds this field if the FCoE bit is set in the transmit context descriptor. The FC CRC off load is described in more detail in Section 7.13.3.2.

### A.5.2.3 FC Header

The FC header fields are provided in the header buffer by the FCoE driver. The FC header includes fields that are modified by the HW as part of Large Send off load. These fields are indicated in this section as “Dynamic” while fields that are not modified by the HW are indicated as “Static”.

#### Routing Control (R_CTL)

The R_CTL is a one-byte Static field that contains routing and information bits to categorize the frame function.

#### Class Specific Control (CS_CTL)

This is a 1 byte Static field that defines either the Class specific control or priority according to bit 17 in the Frame control (F_CTL) field.

#### Destination Identification (D_ID)

The D_ID is a 3 byte Static field that defines the FC destination address.

#### Source Identification (S_ID)

The S_ID is a 3 byte Static field that defines the FC source address.

#### Data Structure Type (TYPE)

The TYPE is a 1 byte Static field that identifies the protocol of the frame content for data frames.

#### Frame Control (F_CTL)

The F_CTL is a 3 byte Dynamic field that contains control information relating to the frame content. The F_CTL is further described in Section A.5.2.3.1. Section 7.13.2.7 describes how the F_CTL field is modified during large send.

#### Data Field Control (DF_CTL)

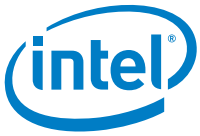
The DF_CTL is a 1 byte Dynamic field that specifies the presence of optional headers at the beginning of the Data_Field. The Optional headers supported by large send are present only on the first frame in the FC sequence. Section 7.13.2.7.1 describes how the DF_CTL field is modified during large send.

#### Sequence ID (SEQ_ID)

The SEQ_ID is a 1 byte Static number associated with a sequence. A sender must assign SEQ_ID numbers so that the recipient would always be able to distinguish between consecutive sequences. SEQ_ID do not have to be sequential and do not have to be unique even within the same IO exchange as long as it is guaranteed that the recipient would be able to distinguish between the them.

#### Sequence Count (SEQ_CNT)

The SEQ_CNT is a 2 byte Dynamic field that indicates the sequential order of Data frame transmission within a single Sequence or multiple consecutive Sequences for the same Exchange. The SEQ_CNT of the first Data frame of the first Sequence of the Exchange transmitted by either the Originator or Responder is ‘0’. The SEQ_CNT of subsequent Data frames in the Sequence is incremented by one for each data frame. The SEQ_CNT of the first Data frame in each sequence other than the first one can start at ‘0’ or be incremented by 1 from the last used SEQ_CNT.



#### Originator Exchange ID (OX_ID)

The OX_ID is a two-byte **S**tatic field that identifies the Exchange_ID assigned by the Originator. If the Originator is enforcing uniqueness via the OX_ID mechanism, it shall set a unique value for OX_ID other than FFFFh. A value of FFFFh indicates that the OX_ID is unassigned and that the Originator is not enforcing uniqueness via the OX_ID. the 82599 supports large receive and direct data placement only if OX_ID is used to identify uniqueness.

#### Responder Exchange ID (RX_ID)

The RX_ID is a two byte Static field assigned by the Responder that shall provide a unique, locally meaningful exchange identifier at the Responder. The Responder of the Exchange shall set a unique value for RX_ID other than FFFh.

#### Parameter (PARAM)

The Parameter field is a Dynamic field which is based on frame type. For Data frames with the relative offset present bit set to 1, the Parameter field specifies relative offset. The offset defines the relative displacement of the first byte of the Payload of the frame from the base address as specified by the ULP. Relative offset is expressed in terms of bytes.

### A.5.2.3.1 Frame Control (F_CTL)

The Frame Control (F_CTL) is a three-byte field that contains control information relating to the frame content. If an error in bit usage is detected, the SW initiates a reject frame (P_RJT) in response with an appropriate reason code (FCoE SW driver responsibility). The F_CTL format is shown in Table A-3 below.

When a bit(s) is designated as “Static” it is provided by the FCoE driver as part of the large send header. The HW keeps this bit(s) as is in all preceding frames of the large send.

When a bit(s) is designated as “Dynamic” it is provided by the FCoE driver as part of the large send header. The HW may change it in some of the packets in the large send as described in Section 7.13.2.7. Exact setting of these bit(s) is described in the Table A-3 below.

When a bit(s) is designated as meaningful under a set of conditions, that bit shall be ignored if those conditions are not present.

**Table A-3 F_CTL Format**

| Control Field | bit | Type | Description. |
|---|---|---|---|
| Exchange Context | 23 | Static | 0b = Originator of Exchange<br>1b = Responder of Exchange |
| Sequence Context | 22 | Static | 0b = Sequence Initiator<br>1b = Sequence Recipient |
| First Sequence | 21 | Static | 0b = Sequence other than first of Exchange<br>1b = First Sequence of Exchange |

## Table A-3 F_CTL Format (Continued)

| Control Field | bit | Type | Description. |
|---|---|---|---|
| Last Sequence | 20 | Static | 0b = Sequence other than last of Exchange<br>1b = Last Sequence of Exchange Must be set on the last data frame of the last sequence. However it can be set on any preceding frames. Once it is set, it must be set on all frames till the last one of that exchange. |
| End Sequence | 19 | Dynamic | 0b = Data frame other than last of Sequence<br>1b = Last Data frame of Sequence |
| End Connection | 18 | Static | Relevant for Class 1 or 6 |
| CS CTL / Priority Enable | 17 | Static | Defines the meaning of the CS_CTL byte in the FC header to be either CS_CTL or Priority indication.<br>0b = 0Word 1, Bits 31-24 = CS_CTL<br>1b = Word 1, Bits 31-24 = Priority |
| Sequence Initiative | 16 | Dynamic | This bit is used to transfer initiative from a sender to a recipient. This bit is meaningful only on the last frame of a sequence (when bit 19 - "End Sequence" is set).<br>0b = Hold Sequence initiative<br>1b = Transfer Sequence initiative |
| X_ID reassigned | 15 | Static | Obsolete. |
| Invalidate X_ID | 14 | Static | Obsolete. |
| ACK Form | 13:12 | Static | ACK Form is meaningful on all Class 1, Class 2, or Class 6 Data frames of a Sequence and on all connect request frames. ACK_Form is not meaningful on Class 1, Class 2, or Class 6 Link_Control frames, or any Class 3 frames.<br>00b = No assistance provided<br>10b = Reserved<br>01b = Ack_1 Required<br>11b = Ack_0 Required |
| Data Compression | 11 | Static | Obsolete. |
| Data Encryption | 10 | Static | Obsolete. |
| Retransmitted Sequence | 9 | Static | Meaningful in Class 1 and 6 and only.<br>0b = Original Sequence transmission<br>1b = Sequence retransmission |
| Unidirectional Transmit | 8 | Static | Relevant for Class 1. |
| Continue Sequence Condition | 7:6 | Dynamic | Last Data frame - Sequence Initiator<br>00b = No information<br>01b = Sequence to follow-immediately<br>10b = Sequence to follow-soon<br>11b = Sequence to follow-delayed<br>It is meaningful only when the "End Sequence" is set to '1' and the "Sequence Initiative" bit is cleared '0'. |

**Table A-3 F_CTL Format (Continued)**

| Control Field | bit | Type | Description. |
|---|---|---|---|
| Abort Sequence Condition | 5:4 | Static | ACK frame - Sequence Recipient<br>00b = Continue sequence<br>01b = Abort Sequence, Perform ABTS<br>10b = Stop Sequence<br>11b = Immediate Sequence re-XMT requested<br>Data frame (1st of Exchange) - The Abort Sequence shall be set to a value by the Sequence Initiator on the first Data frame of an Exchange to indicate that the Originator is requiring a specific error policy for the Exchange.<br>00b = Abort, Discard multiple Sequences<br>01b = Abort, Discard a single Sequence<br>10b = Process policy with infinite buffers<br>11b = Discard multiple Seq with immediate re-XMT |
| Relative offset present | 3 | Static | 0b = Parameter field defined for some frames<br>1b = Parameter Field = relative offset |
| Exchange reassembly | 2 | Static | Reserved for Exchange reassembly |
| Fill Bytes | 1:0 | Dynamic | Defines the number of FC padding bytes to fill in the last frame in the sequence. The value of these bytes is 0x00.<br>When FCoE offload is enabled (TUCMD.FCoE bit is set in the transmit context descriptor), the hardware can pad the FC payload according to the buffer size indicated by software (by the PAYLEN field in the transmit data descriptor). |

## A.5.2.4 FC Extended Headers

**Note:** The following section is provided as a background on FC and is not required for the HW implementation or the SW implementation since in FCoE frames, extended headers are not expected. The following quote is from the FCoE spec Rev 0.7 "No Extended Headers are relevant for the operations of a Reedtown NIC".

The diagrams in Figure A-22, Figure A-23 and Table A-4 show the Fibre Channel frame structure with Extended headers and lists the existing headers. Extended headers are identified by the R_CTL value as shown in the diagrams below. The LAN controller does not support these Extended headers since they are not expected in FCoE usage.

| FC Frame | | | | |
|---|---|---|---|---|
| Extended Headers [Optional] | FC Header | Optional FC Header | [optional] FC Data (including optional padding) | FC CRC |

**Figure A-22 FC Frame format with Extended Headers**

| | 31 24<br>Last bytes on the wire | 23 16 | 15 8 | 7 0<br>First bytes on the wire |
|---|---|---|---|---|
| 0 | Extended Header Specific Fields | | | R_CTL |
| 4... | | | | |

**Figure A-23 Extended Header Format**

**Table A-4 FC Extended Headers**

| R_CTL | Extended Header | Length |
|---|---|---|
| 0x50 | VFT_Header (Virtual Fabric Tagging Header) | 8 Bytes |
| 0x51 | IFR_Header (Inter-Fabric Routing Header) | 8 Bytes |
| 0x52 | Enc_Header (Encapsulation Header) | 24 Bytes |
| 0x53…0x5F | Reserved | - |

### A.5.2.5 FC Optional Headers

**Note:** Most of the following section is provided as a background on FC and is not required for the HW implementation. The reader may skip the detailed explanation of the Optional headers provided below and concentrate in the tables and figures that follow the text.

FCoE frames may include FC Optional headers. These headers (if they exist) would always show in the first frame in a sequence. While FC implementation may include the FC Optional headers only on the first frame in a sequence. In Large send functionality, the FC Optional headers may show only on the first frame as shown in Figure 7-47 and Table A-5 (in Section 7.13.2.7.1).

The following diagrams Table A-5, Figure A-24 and Figure A-25 show the Fibre Channel frame structure with Optional headers and lists the Optional headers. The Optional headers (that are present) are always ordered as shown in Figure A-24 and Figure A-25. 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 |

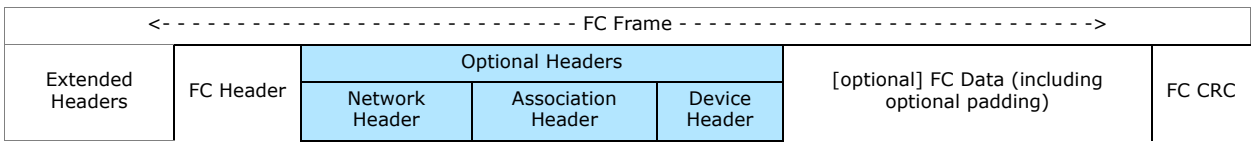



**Figure A-24 FC Frame format with Optional Headers (without ESP Header)**

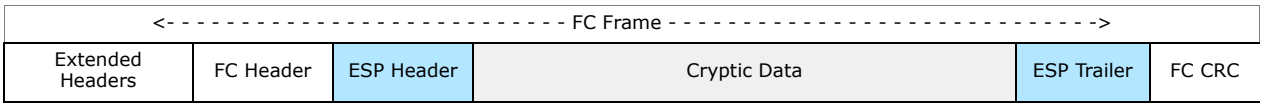



**Figure A-25 FC Frame format with Optional Headers (with ESP Header)**

### ESP Header:

This is the first Optional header which covers the whole FC frame other than the FC header which is transmitted on the clear (as plain text). When an ESP header is present there is also the ESP trailer. If required, the SW is responsible for the cryptic calculation and preparing the ESP header and trailer. Its presence is indicated by bit 6 in the DF_CTL field being set to one. The hardware does not support Large Send off load when ESP Optional header is used. When present, the ESP header and trailer are present in all frames of the exchange.

### Network Header:

The Network Header, if used, shall be present only in the first Data frame of a Sequence. A bridge or a gateway node that interfaces to an external Network may use the Network Header. The Network Header, is an optional header 16 Bytes long within the FC Data Field content. Its presence is indicated by bit 5 in the DF_CTL field being set to one. The Network Header may be used for routing between Fibre Channel networks of different Fabric address spaces, or Fibre Channel and non-Fibre Channel networks. The Network Header contains Name Identifiers for Network Destination Address and Network Source Address.

### Association Header:

The Association Header, if used, shall be present only in the first Data frame of a Sequence. The Association Header is an optional header of 32 Bytes long within the Data Field content. Its presence is indicated by bit 4 in the DF_CTL field being set to one. The Association Header may be used to identify a specific Process or group of Processes within a node associated with an Exchange. When an Nx_Port has indicated during Login that an Initial Process Associator is required to communicate with it, the Association Header should be used by that Nx_Port to identify a specific Process or group of Processes within a node associated with an Exchange. the 82599 does not use the Association for any filtering purposes but rather uses the OX_ID.

### Device header:

The Device Header, if present, shall 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, shall be 16, 32, or 64 bytes in size as defined by bits 1:0 in the DF_CTL field. The contents of the Device Header are controlled at a level above FC-2. Upper layer protocol (ULP) may use a Device Header, requiring the Device Header to be supported. The Device Header may be ignored and skipped, if not needed. If a Device Header is present for a ULP that does not require it, the related FC-4 may reject the frame with the reason code of “TYPE not supported”.

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

# Appendix B LESM - Link Establishment State Machine for the 82599

## B.1 Background

"Legacy" XAUI-based switches developed prior to the IEEE 802.3ap standard TX only in one lane (Lane 0) during link detection. Typically, these devices only transition to a XAUI-like 10 GbE link when all four pairs of their receivers is active.

Additionally, IEEE 802.3ap compliant devices such as the Intel 82599 controller are required to transmit auto-negotiation only on Lane 0 per Clause 73.3, and the Intel device only parallel-detects a XAUI-like 10 GbE link when all four pairs of their receivers are active. Therefore, a speedlock condition can occur when the 82599 device is connected to a legacy XAUI-based switch, since both devices are capable of 10 GbE XAUI-like parallel detection, but only the lane 0 transmitters on each device are active -- one device needs to turn on all four transmitters for the other device to see 10 GbE XAUI-like mode. Otherwise, either no link, or a 1 GbE link is observed in the system, depending on the specific behavior of the switch link state machine.

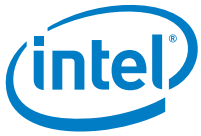
The LESM (link establishment state machine) was developed by Intel to break the speedlock condition described above. The feature can be implemented in the 82599 controller with on-chip firmware, and is used to switch the link-mode-select setting in the AUTOC register to try a different configuration after timeout. For example, after trying CL 73 AN and Parallel Detect, it may change to XAUI-mode (which turns on all four lane transmitters) and check link status.

# B.2 Location in the NVM

The LESM module in NVM includes the parameters used by LESM and is pointed by the Firmware Module offset 0x2 (see Section 6.4) and is placed instead the unused No Manageability Patch.

**Table B-1 Firmware module**

| Global MNG Word Offset | Description |
|---|---|
| 0x0 | Test Configuration Pointer - Section 6.4.1 |
| 0x1 | Reserved |
| 0x2 | LESM Module Pointer |
| 0x3 | Common Firmware Parameters - Section 6.4.2 |
| 0x4 | Pass Through Patch Configuration Pointer (Patch structure identical to the Loader Patch) - Section 6.4.3 |
| 0x5 | Pass Through LAN 0 Configuration Pointer - Section 6.4.3 |
| 0x6 | SideBand Configuration Pointer - Section 6.4.4 |
| 0x7 | Flexible TCO Filter Configuration Pointer - Section 6.4.5 |
| 0x8 | Pass Through LAN 1 Configuration Pointer - Section 6.4.3 |
| 0x9 | NC-SI Microcode Download Pointer - Section 6.4.6 |
| 0xA | NC-SI Configuration Pointer - Section 6.4.7 |



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

# LEGAL

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors which may cause deviations from published specifications.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

* Other names and brands may be claimed as the property of others.

© 2006-2016 Intel Corporation.

# Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Intel:

JL82599ES S LGWF HL82599ES S LGWK JL82599EB S LGWH JL82599EC S LJ2Y JL82599EN S LJFU JL82599ES S LGWE JL82599EN S LJFT JL82599EB S LGWG JL82599EB S R1W6 JL82599EB S R1W7 JL82599EN S R1ZR JL82599EN S R1ZS JL82599ES S R1VM JL82599ES S R1VN