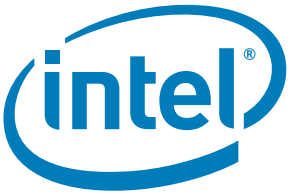


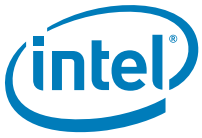
Intel® I/O Controller Hub 8/9/10 and 82566/82567/82562V Software Developer's Manual

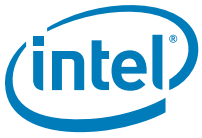
July 2009

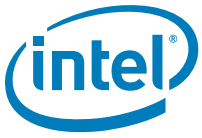
322409-001

Revision 1.8

Contents







Revision History

Date	Revision	Description
July 2009	1.8	• Initial Public Release.
October 2008	1.7	• Updated sections 10.4.36, 10.4.45, and 10.4.59.
July 2008	1.6	• Removed section 9.12.1.
June 2008	1.5	• Added GLCI diagnostics register (section 10.4.8.38).
April 2008	1.4	• Updated Table 60 (PHYADD bits 25:21 description) for the **82566** and **82567**. • Added the PHY LEDs Control Register (section 10.4.8.39) for the **82567** only.
March 2008	1.3	• Removed all references to PCI Express* (PCIe*). • Removed all references to SMBus.
March 2008	1.2	Updated Section 5.0: Content for this section can now be found in the ICH8, ICH9, and ICH10 NVM map and information guides. For the **82567** only: • Updated section 6.3.3.1.8 (Added ACPI_En bit description). • Updated section 10.6.1 (Added ACPI_EN bit description).
Januray 2008	1.1	For the **ICH8** only: • Updated NVM images in sections 5.5.3 through 5.5.5. • Updated bit descriptions for NVM words 0Fh, 13h, 14h, 15h, 16h, 32h, and 33h.
December 2007	1.0	Added ICH10 information.
October 2007	0.8	Updated to reflect the ICH9M/82567, stepping A2, revision 1.6 NVM image.
June 2007	0.5	Major revision all sections.
January 2007	0.25	Initial release (Intel Confidential)

1.0 Introduction

1.1 Scope

This document serves as a software developer's manual for the **ICH8/ICH9/ICH10** (MAC) GbE LAN Controller and the **82566/82567/82562V** Physical Layer Transceiver (PHY). It can be used a reference for software device driver developers, board designers, test engineers, or anyone else who might need specific technical or programming information.

1.2 Overview

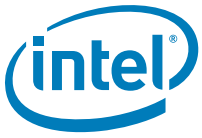
The **ICH8/ICH9/ICH10** MAC and **82566/82567/82562V** PHY are highly integrated, high-performance Ethernet LAN controllers for 1000 Mb/s, 100 Mb/s and 10 Mb/s data rates. They are optimized for LAN on Motherboard (LOM) designs, enterprise networking, and Internet appliances that use the PCI architecture (Revision 1.0a).

The Ethernet LAN controllers support all IEEE 802.3 receive and transmit MAC functions as well as physical-layer circuitry for 1000Base-T, 100Base-TX, and 10Base-T applications (IEEE 802.3, 802.3q, 802.3u, 802.3x, 802.3ab).

Table 1 lists the PHYs that are associated with each MAC.

Table 1. MAC/PHY Association

MAC	PHY
ICH8	82566 and 82562V
ICH8M	82566 and 82562V
ICH9	82566, 82567, and 82562V
ICH9M	82567
ICH10	82567



1.3 Ethernet LAN Controller Features

This section describes the features of the Ethernet LAN controllers.

1.3.1 ICH8/ICH9/ICH10 MAC Features

- Compliance with industry standards:
  - 802.3, 802.3q, 802.3u, 802.3x, 802.3ab
- Ethernet:
  - Multi-speed operation: 10/100/1000 Mb/s
  - Full duplex operation at all supported speeds; half duplex at 10/100 Mb/s
  - Flow control support: send/receive pause frames & receive FIFO thresholds
  - 802.1q VLAN support
  - MAC Address filters: eight perfect match filters; multicast hash filtering, broadcast filter and promiscuous mode

Note: There is no support for gigabit half-duplex operation.

- Host Interface:
  - 64-bit address master support for systems using more than 4 GB of physical memory
  - Programmable host memory receive buffers (256 bytes to 16 KB)
  - Intelligent interrupt generation features to enhance driver performance
  - IPv4 and IPv6 checksum offload support (receive, transmit, and large send).
- LAN Connected Device (LCD) Interface:
  - LCI-2 interface to external PHY (**82566/82567** or **82562V**)
  - GLCI-2 interface to external PHY (**82566/82567**)
- Intel® Active Management Technology
- Pre-Driver Configuration:
  - Magic Packet* wake-up enable with unique MAC address
  - Configurable LED operation for OEM customization of LED displays

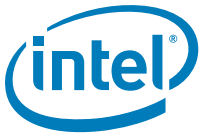
- Performance
  - Configurable receive and transmit data FIFO; programmable in 1 KB increments
  - TCP segmentation capability compatible with Large Send offloading features
  - Fragmented UDP checksum offload for packet reassembly
  - Split header support to eliminate payload copy from user space to host space
  - Receive Side Scaling (RSS) with two hardware receive queues
- Software Interface:
  - Descriptor ring management hardware for transmit and receive
  - Software controlled bit (resets everything except the configuration registers)
  - Message Signaled Interrupts (MSI)
- Power Management:
  - ACPI register set and power down functionality supporting D0 and D3 states
  - Full wakeup support (APM and ACPI)
  - Auto Connect Battery Saver (ACBS) for power savings when network cable is disconnected
- Diagnostics:
  - Statistics for management and RMON
  - Additional statistics to enhance new interrupt features
  - MAC loopback
- Other:
  - 16 KB packet buffer (**ICH8**)
  - 24 KB packet buffer (**ICH9/ICH10**)
  - LAN disable function provided via ICH fuse option and Function Disable SUS Well register (RTC Well register for **ICH9/ICH10**). Refer to the *Intel(R) I/O Controller Hub 8 (ICH8) Family External Design Specification (EDS) - Volumes 1 and 2, Intel(R) I/O Controller Hub 9 (ICH9) Family External Design Specification (EDS) - Volumes 1 and 2*), or the *Intel(R) I/O Controller Hub 10 (ICH10) Family External Design Specification (EDS) - Volumes 1 and 2*).

1.3.2 ICH9/ICH10 MAC Features

- Virtual Technology:
  - Warm Function Reset - Function Level Reset (FLR)
- Four additional programmed perfect match filters and all nodes multicast (total of 12 perfect match filters)
- Jumbo frame support

1.3.3 ICH10 MAC Features

- Virtual Technology:
  - Virtual Machine Devices (VMDq) support
- LinkSec offload compliant with 802.3ae specification
- TimeSync offload compliant with 802.1as specification



1.3.4 PHY Features (82566 and 82562V)

- Operates with the ICH8/ICH9 I/O hub
- FCMMAP 2L 10 x 10 mm package
- IEEE 802.3ab compliant
- Line Length >140 m
- Operates with worst case cable
- Supports carrier extension (half duplex)
- Auto-negotiation with support for next page
- Smart speed
- Automatic MDI crossover capable
- PMA loopback capable (no echo cancel)
- Advanced Power Management:
  - Low power link up; differentiates between D0a/non-D0a
  - Smart power down; link disconnect with back-to-back option
- Advanced cable diagnostics:
  - TDR
  - Channel frequency response
- Extended configuration load sequence
- Automatic resolution of full duplex/half duplex mismatch in 10/100 forced configurations

1.3.5 PHY Features (82567)

- Operates with the ICH9M I/O hub
- 56-pin QFN package
- IEEE 802.3ab compliant; IEEE 802.3u compliant auto-negotiation
- Reduced power consumption during normal operation and power down modes
- Integrated MDI interface termination resistors
- Energy detect and energy detect+ low power modes
- Three loopback modes for diagnostics
- Downshift mode for two-pair cable installations
- Fully integrated digital adaptive equalizers, echo cancellers, and crosstalk cancellers
- Advanced digital baseline wander correction
- Automatic MDI/MDI-X crossover at all speeds of operation
- Automatic polarity correction
- Software programmable LED modes, including LED testing
- MDC/MDIO management interface

1.3.6 Additional Features

- Dual interconnect between MAC and PHY
  - — LCI for 10/100 Mb/s operation control traffic
  - — GLCI for 1000 Mb/s operation
- Three LED outputs
- Variable-rate reference clock to MAC
- Multi-voltage regulation modes
  - — External voltage regulation
  - — For the **82566** and **82562V**, internal linear regulators with external regulation transistors (nominal 1.8 V and 1.0 V programmable)
  - — For the **82567**, fully integrated linear regulator (nominal 1.0 V programmable)
  - — For the **82567**, fully integrated switched voltage regulator (nominal 1.8 V, programmable)

1.4 Conventions

This document uses notes that call attention to important comments:

Note: Indicates details about the hardware's operations that are not immediately obvious. Read these notes to get information about exceptions, unusual situations, and additional explanations of some product features.

1.4.1 Register and Bit References

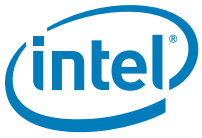
This document refers to device register names with all capital letters. To refer to a specific bit in a register the convention REGISTER.BIT is used. For example CTRL.FD refers to the *Full Duplex Mode* bit in the Device Control (CTRL) register.

1.4.2 Byte and Bit Designations

This document uses "B" to abbreviate quantities of bytes. For example, a 4 KB represents 4096 bytes. Similarly, "b" is used to represent quantities of bits. For example, 100 Mb/s represents 100 megabits per second.

1.4.3 Numbering

All numbers are in decimal unless otherwise indicated. Hexadecimal numbers are preceded by an h. If not clear from the context, binary numbers are followed by a b.



1.5 Memory Alignment Terminology

Some data structures have special memory alignment requirements. This implies that the starting physical address of a data structure must be aligned as specified in this document. The following terms are used for this purpose:

- **BYTE** alignment: Implies that the physical addresses can be odd or even. Examples: 0FECBD9A1h, 02345ADC6h.
- **WORD** alignment: Implies that physical addresses must be aligned on even boundaries. For example, the last nibble of the address can only end in 0, 2, 4, 6, 8, Ah, Ch, or Eh (0FECBD9A2h).
- **DWORD** (Double-Word) alignment: Implies that the physical addresses can only be aligned on 4-byte boundaries. For example, the last nibble of the address can only end in 0, 4, 8, or Ch (0FECBD9A8h).
- **QWORD** (Quad-Word) alignment: Implies that the physical addresses can only be aligned on 8-byte boundaries. For example, the last nibble of the address can only end in 0 or 8 (0FECBD9A8h).
- **PARAGRAPH** alignment: Implies that the physical addresses can only be aligned on 16-byte boundaries. For example, the last nibble must be a 0 (02345ADC0h).

1.6 Alignment and Byte Ordering

It should be noted that the data stream in Ethernet has no notion of byte alignment. All data on the wire is referenced as bit ordered. Data is presented on the wire as Least Significant Bit (LSB) first. This is true for any BYTE, WORD, DWORD, or QWORD representation.

1.7 Register Byte Ordering

This section defines the structure of registers that contain fields carried over the network. Some examples are L2, L3, L4 fields and for the **ICH10** LinkSec fields.

The following example is used to describe byte ordering over the wire (hex notation):

Last First

...,06, 05, 04, 03, 02, 01, 00

where each byte is sent with the Least Significant Bit (LSB) first. That is, the bit order over the wire for this example is

Last First

..., 0000 0011, 0000 0010, 0000 0001, 0000 0000

The general rule for register ordering is to use host ordering (also called little endian). Using the previous example, a 6-byte field (such as MAC address) is stored in a CSR in the following manner:

	Byte 3	Byte 2	Byte 1	Byte0
DW address (N)	0x03	0x02	0x01	0x00
DW address (N+4)	...	...	0x05	0x04

The following exceptions use network ordering (also called big endian). Using the previous example, a 16-bit field (such as EtherType) is stored in a CSR in the following manner:

	Byte 3	Byte 2	Byte 1	Byte0
(DW aligned)	...	...	0x00	0x01
or				
(Word aligned)	0x00	0x01	...	...

The following exceptions use network ordering:

- All ETherType fields

Note: Normal notation is to use network ordering. For example: using a MAC address of 00-A0-C9-00-00-00. The order on the network is 00, then A0, then C9, etc. However, the host ordering presentation would be

	Byte 3	Byte 2	Byte 1	Byte0
DW address (N)	00	C9	A0	00
DW address (N+4)	...	...	00	00

1.8 CSR Register Conventions

All registers in the MAC are defined to be 32 bits, so write cycles should be accessed as 32-bit double words. However, there are exceptions to this rule:

- Register pairs where two 32-bit registers make up a larger logical size
- **Reserved bit positions.** Some registers contain certain bits that are marked as “reserved.” These bits should never be set to a value of 1b by software. Reads from registers containing reserved bits can return indeterminate values in the reserved bit positions unless read values are explicitly stated. When read, these reserved bits should be ignored by software.
- **Reserved and/or undefined addresses.** Any register address not explicitly declared in this document should be considered to be reserved and should not be written. Writing to reserved or undefined register addresses can cause indeterminate behavior. Reads from reserved or undefined configuration register addresses can return indeterminate values unless read values are explicitly stated for specific addresses. Reserved fields within defined registers are defined as Read-Only (RO). When writing to these registers the RO fields should be set to their initial value. For **ICH9/ICH10**, reading from reserved fields might return indeterminate values.

- **Initial values.** Most registers define the initial hardware values prior to being programmed. In some cases, hardware initial values are undefined and are listed as such via the text “undefined,” “unknown,” or “X.” Some such values might need setting through NVM configuration or software in order for proper operation to occur; this need is dependent on the function of the bit. Other registers might cite a hardware default that is overridden by a higher precedence operation. Operations that might supersede hardware defaults can include a valid NVM load, completion of a hardware operation (such as hardware auto-negotiation), or writing of a different register whose value is then reflected in another bit.

For registers that should be accessed as 32-bit double words, partial writes (less than a 32-bit double word) is ignored. Partial reads return all 32 bits of data regardless of the byte enables.

Note: Partial reads to read-on-clear registers (like ICR) can have unexpected results since all 32 bits are actually read regardless of the byte enables. Partial reads should not be performed.

Note: All statistics registers are implemented as 32-bit registers. Though some logical statistics registers represent counters in excess of 32-bits in width, registers must be accessed using 32-bit operations. For example, independent access to each 32-bit field.

Refer to the special notes for Multicast Table Arrays in their specific register definitions.

1.9 Related Documents

- *Intel(R) I/O Controller Hub 8 (ICH8) Family External Design Specification (EDS) - Volumes 1 and 2.*
- *Intel(R) I/O Controller Hub 9 (ICH9) Family External Design Specification (EDS) - Volumes 1 and 2.*
- *Intel(R) I/O Controller Hub 10 (ICH10) Family External Design Specification (EDS) - Volumes 1 and 2.*
- *IEEE Std. 802.3, 2000 Edition. Incorporates various IEEE standards previously published separately.*
- *PCI Local Bus Specification, Revision 2.3, PCI Local Bus Special Interest Group.*
- *Advanced Configuration and Power Interface (ACPI) Specification, Revision 2.0*
- *PCI Bus Power Management Interface Specification, Revision 1.1*
- *Information Technology - Telecommunication & Information Exchange Between Systems - LAN/MAN - Specific Requirements - Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, IEEE Standard No.: 802.3-2002.*
- *IEEE 1149.1-2001 (JTAG).*

2.0 Architecture Overview

2.1 Introduction

This section provides an overview of the Ethernet LAN controllers. The following sections give detailed information about the Ethernet LAN controller's functionality, register description, and initialization sequence. All major interfaces of the Ethernet LAN controllers are described in detail.

2.2 ICH8/ICH9/ICH10 (MAC) GbE LAN Controller

The **ICH8/ICH9** GbE MAC (MAC) provides an 802.3 Carrier Sense Multiple Access/ Collision Domain (CSMA/CD) Ethernet LCI-2 interface to an external **82566** or **82562V** PHY to provide wired LAN connectivity.

The MAC provides a system interface via a PCI function. A full memory-mapped or IO-mapped interface is provided to the network driver along with Direct Memory Addressing (DMA) mechanisms for high performance data transfers. Refer to the appropriate ICH documentation for more details.

2.3 82566/82567/82562V PHY

The **82566/82567/82562V** is a single port Gigabit Ethernet Physical Layer Transceiver (PHY) that connects to its MAC through a dedicated interconnect. The PHY is based on Intel's Gigabit PHY technology and supports operation at 1000/100/10 Mb/s data rates. The physical layer circuitry provides a standard IEEE 802.3 Ethernet interface for 1000BASE-T, 100BASE-TX, and 10BASE-T applications (802.3, 802.3u, and 802.3ab).

The **82566/82562V** operates with the **ICH8/ICH9** chipset that incorporates the GbE MAC.

The **82567** operates with the **ICH9/ICH10** chipset that incorporates the GbE MAC.

The PHY interfaces with its MAC through two interfaces: GLCI and LCI. GLCI is a high speed serial interface based on 802.3 SerDes. LCI is a lower speed interface based on the 82562ET LCI interface. The PHY operates in a dual interface mode using GLCI for 1000 Mb/s traffic and LCI for all other traffic types.

Figure 1 shows the major components of the PHY architecture.

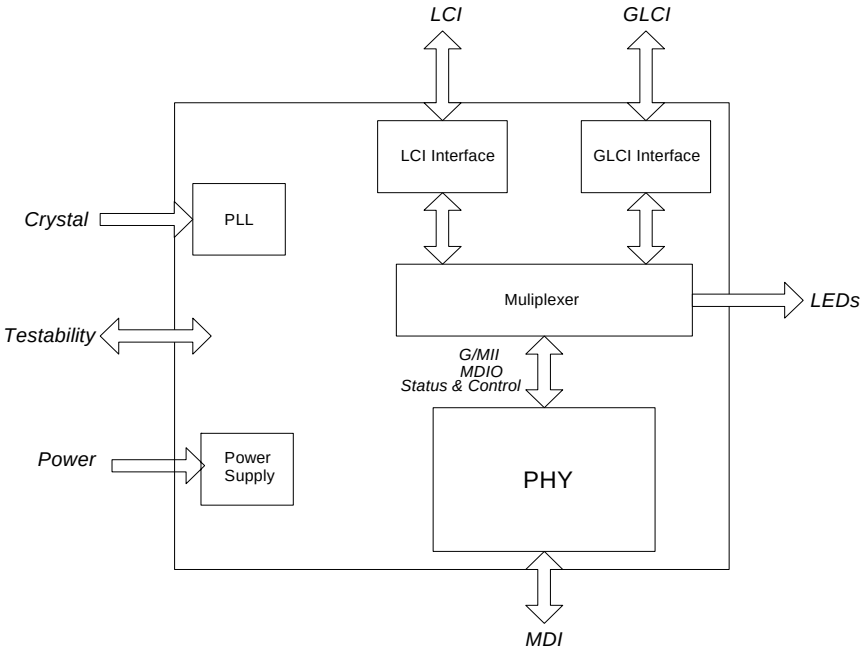


Figure 1. PHY Block Diagram

2.4 Interface Flows

The PHY's main interface are GLCI and LCI on the host side and the MDI interface on the link side. Transmit traffic is received from the MAC as either GLCI or LCI packets on the host interconnect and transmitted as Ethernet packets on the MDI link. Receive traffic arrives as Ethernet packets on the MDI link and transferred to the MAC through either the GLCI or LCI interconnects.

Note: If the GLCI is unable to achieve link between the MAC and PHY, the lock loss function is used to establish the link. If the GLCI link cannot be established, then the speed drops to 100 Mb/s.

The host interface can operate with GLCI and LCI interfaces running in parallel. In GLCI and LCI mode, all traffic is routed via LCI, other than 1000 Mb/s traffic that goes through GLCI.

The MAC and system software control the PHY functionality through two mechanisms:

PHY configuration registers are mapped into the MDIO space and can be accessed by the MAC through the LCI interconnect. The MDIO traffic is embedded in specific fields in LCI packets and is carried by special packets in the GLCI interconnect.

Some parameters in the PHY are controlled or monitored directly through dedicated bits in the LCI or GLCI packets.

Management traffic is carried through LCI in GLCI and LCI mode.

2.4.1 System Interface

The MAC supports a one lane PCI bus interface operating at 2.5 GHz. 32 KB of on-chip buffering mitigates instantaneous receive bandwidth demands and eliminates transmit under-runs by buffering the entire outgoing packet prior to transmission.

2.4.2 NVM Interface

The LAN NVM shares space on an SPI Flash device (or devices) along with the BIOS, manageability firmware, and a Flash descriptor region. It is programmed through the **ICH8/ICH9/ICH10**. This combined image is shown in Section 5.0. The Flash descriptor region is used to define vendor specific information and the location, allocated space, and read and write permissions for each region. The Manageability Engine (ME) region contains the code and configuration data for ME functions such as Intel® Active Management Technology, ASF, and Advanced Fan Speed Control. The system BIOS is contained in the BIOS region.

Note: The ME region and BIOS region are beyond the scope of this section and a more detailed explanation of these areas can be found in the *Intel® I/O Controller Hub 8 (ICH8) Family External Design Specification (ICH8 EDS)*, *Intel® I/O Controller Hub 9 (ICH9) Family External Design Specification (ICH9 EDS)*, or the *Intel® I/O Controller Hub 10 (ICH10) Family External Design Specification (ICH10 EDS)*.

2.5 DMA Addressing

In appropriate systems, all addresses mastered by the MAC are 64 bits in order to support systems that have larger than 32-bit physical addressing. Providing 64-bit addresses eliminates the need for special segment registers.

Note: The MAC only supports little endian data arrangement on the host bus.

The following example illustrates data-byte ordering. Bytes for a receive packet arrive in the order shown from left to right.

```
01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e
```

2.5.1 Byte Ordering

As explained in Section 3.2.2, the incoming data can be offset. The left side of the boxes shows the byte addresses for the major words while the tops of the boxes show the byte numbers and bit numbers for the host bus. Data byte position is identical in both modes. Accomplishing this requires a swap of the data from the least-significant byte position to the most-significant byte position.

Table 2. Data Ordering (64-Bit)

		63							0
		7	6	5	4	3	2	1	0
Byte Address	0	08	07	06	05	04	03	02	01
	8	10	0f	0e	0d	0c	0b	0a	09
	10	18	17	16	15	14	13	12	11
	18	20	1f	1e	1d	1c	1b	1a	19

2.6 Ethernet Addressing

Several registers store Ethernet addresses in the MAC. Two 32-bit registers make up the address: one is called high and the other is called low. For example, the Receive Address register is comprised of Receive Address High (RAH) and Receive Address Low (RAL). The least significant bit of the least significant byte of the address stored in the register (for example, bit 0 of RAL) is the multicast bit. The LS byte is the first byte to appear on the wire. This notation applies to all address registers, including the flow control registers.

Figure 2 shows the bit/byte addressing order comparison between what is on the wire and the values in the unique receive address registers.

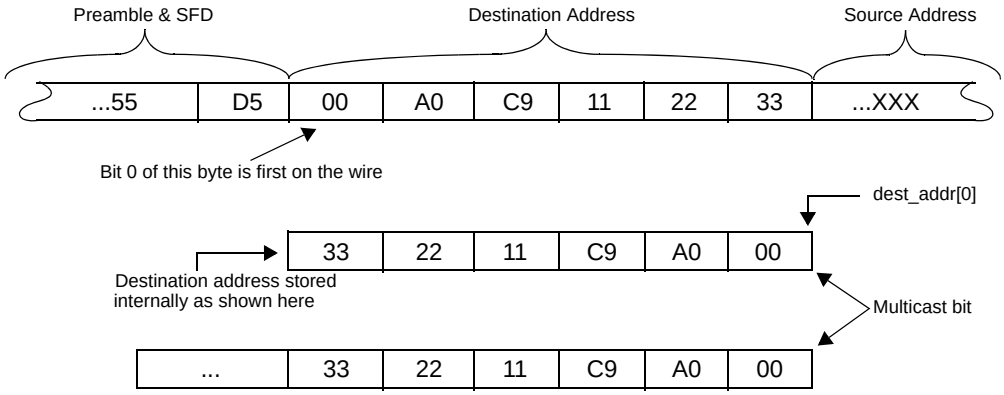


Figure 2. Example of Address Byte Ordering

The address byte order numbering shown in Figure 2 maps to Table 3. Byte #1 is first on the wire.

Table 3. Intel® Architecture Byte Ordering

IA Byte #	1 (LSB)	2	3	4	5	6 (MSB)
Byte Value (Hex)	00	A0	C9	11	22	33

Note: The notation in this manual follows the convention shown in Table 3. For example, the address in Table 3 indicates 00_A0_C9_11_22_33h, where the first byte (00_) is the first byte on the wire, with bit 0 of that byte transmitted first.

2.7 Interrupt Control and Tuning

The MAC provides a complete set of interrupts that allow for efficient software management. The interrupt structure is designed to accomplish the following:

- Make accesses thread-safe by using set and clear-on-read rather than read-modify-write operations.
- Correlate between related bits in different registers (for example, CPU vector and ICR)
- Minimize the number of interrupts needed relative to work accomplished.

- Minimize the processing overhead associated with each interrupt.

The interrupt logic consists of the following interrupt registers. More detail about these registers is given in Section 10.0.

- Interrupt Cause Set/Read Registers
  The Read register records the cause of the interrupt. All bits set at the time of the read are auto-cleared (a few cause bits are also cleared when the CPU vector is cleared). The cause bit is set for each bit written as a 1b in the Set register. If there is a race between hardware setting a cause and software clearing an interrupt, the bit remains set. No race condition exists on writing the Set register. Set provides for software posting of an interrupt. Reads are auto-cleared to avoid expensive write operations. Most systems have write buffering that minimizes overhead but typically requires a read operation to guarantee that the write operation has been flushed from the posted buffers. Without auto-clear, the cost of clearing an interrupt can be as high as two reads and one write.
- Interrupt Mask Set (Read)/Clear Registers
  Interrupts appear only if the interrupt cause bit is a 1b and the corresponding interrupt mask bit is a 1b. Software blocks submission of the interrupt MSI by clearing the bit in the mask register. The cause bit stores the interrupt event regardless of the state of the mask bit. The Clear and Set operations make this register more thread-safe by avoiding a read-modify-write operation on the mask register. The mask bit is set to a 1b for each bit written in the Set register, and cleared for each bit written in the Clear register. Reading the Set register returns the current value.
- Interrupt Throttling Register
  The frequency of interrupts from the MAC can be reduced when inter-interrupt interval value is non-zero. The MAC asserts pending interrupts only at regularly scheduled intervals. When inter-interrupt interval value is zero, the MAC immediately asserts pending interrupts.
- Interrupt Canceling
  When Interrupt throttling is functioning, the frequency of interrupts can be further reduced if the receive or transmit interrupt inter value is 0b the opposite inter value is cleared to form an interrupt for both transmit and receive at the same time. This prevents interrupts from being too close because the inter values are not correlated.

Three actions minimize the number of interrupts:

1. Reducing the frequency of all interrupts.
2. Accepting multiple receive packets before signaling an interrupt.
3. Eliminating or reducing the need for interrupts on transmit.

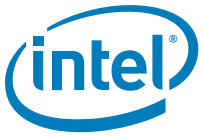
One interrupt register consolidates all interrupt information eliminating the need for multiple accesses. However, a software device driver might choose to have finer details by also reading the CPU Vector register.

Note: The MAC supports Message Signaled Interrupts per the PCI specification.

2.8 Hardware Acceleration Capability

The MAC provides the ability to offload IP, TCP, and UDP checksum for transmit. The functionality provided by these features can significantly reduce processor utilization by shifting the burden of the functions from the software device driver to the hardware. Features include:

- Receive and transmit checksum offloading



- TCP segmentation
- Receive fragmented UDP checksum offload

These features are briefly outlined in the following sections. More detail about all of the hardware acceleration capabilities is provided in Section 3.0.

2.9 ICH9/ICH10 Jumbo Frame Support

The MAC supports jumbo frames to increase performance and decrease CPU utilization. By default, the MAC might receive packets with a maximum size of 1522 bytes. If large frame reception is enabled by the RCTL register, The MAC supports jumbo packet reception of up to 9 KB. On the transmit size, jumbo packets are always supported. It is the responsibility of the software device driver to initiate jumbo packets only in full-duplex mode and when it is configured to do so.

Note: The packet buffer size must be at least the size of the maximum packet size. Also, in order to reach wire speed, the transmit packet buffer size must be at least twice the size of the maximum packet size.

Note: The hardware supports 16 KB jumbo frames, but the accepted maximum size for jumbo frames in the industry is actually 9 KB. Validation to 9 KB is all that is supported given the smaller packet buffer FIFOs.

2.9.1 Checksum Offloading

The MAC provides the ability to offload the IPv4, TCP, and UDP checksum requirements from the software device driver. For common frame types, the hardware automatically calculates, inserts, and checks the appropriate checksum values normally handled by software.

Note: IPv6 headers do not have a checksum.

For transmits where the MAC is doing non-TCP segmentation, every transmitted Ethernet packet can have two checksums calculated and inserted by the MAC. Typically these would be the IPv4 and either TCP or UDP checksums. The software device driver specifies which portions of the packet are included in the checksum calculations, and where the calculated values are inserted, via descriptor(s). Refer to Section 3.3.5 for details.

For receives, the hardware recognizes the packet type and performs the checksum calculations as well as error checking automatically. Checksum and error information is provided to software via the receive descriptor(s). Refer to Section 3.2.11 for details.

2.9.2 TCP Segmentation

The MAC implements a TCP segmentation capability for transmits that allows the software device driver to offload packet segmentation and encapsulation to the hardware. The software device driver can send the MAC the entire IP (IPv4 or IPv6), TCP or UDP message sent down by the Network Operating System (NOS) for transmission. The MAC segments the packet into legal Ethernet frames and transmit them on the wire. By handling the segmentation tasks, the hardware alleviates the software from handling some of the framing responsibilities. This reduces the overhead on the CPU for the transmission process thus reducing overall CPU utilization. See Section 3.5 for details.

2.9.3 Fragmented UDP Checksum

The MAC provides the ability to offload inbound fragmented UDP packet reassembly. The MAC provides the partial checksum calculation for each incoming UDP fragment so that the software device driver is required to sum the partial checksum words for each fragment to produce the complete checksum. The fragmented UDP checksum offload is provided to IPv4 packets. For more details see Section 3.0.

2.10 Transmit and Receive Data Interface

The MAC provides a data transmit and receive interface for 10/100/1000 Mb/s operation along with the proper side band signaling that is needed according to the interface mode: LCI/GLCI.

2.11 Buffer and Descriptor Structure

Software allocates the transmit and receive buffers, and also forms the descriptors that contain pointers to, and the status of, those buffers. A conceptual ownership boundary exists between the driver software and the hardware of the buffers and descriptors.

The software gives the hardware ownership of a queue of buffers for receives. These receive buffers store data that the software then owns once a valid packet arrives.

For transmits, the software maintains a queue of buffers. The driver software owns a buffer until it is ready to transmit. The software then commits the buffer to the hardware; the hardware then owns the buffer until the data is loaded or transmitted in the transmit FIFO.

Descriptors store the following information about the buffers:

- The physical address
- The length
- Status and command information about the referenced buffer

Descriptors contain an end-of-packet field that indicates the last buffer for a packet. Descriptors also contain packet-specific information indicating the type of packet, and specific operations to perform in the context of transmitting a packet, such as those for VLAN or checksum offload.

Section 3.0 provides detailed information about descriptor structure and operation in the context of packet transmission and reception.

Note: Any programming of the transmit and receive setting should be made while the MAC is in the idle state. If an existing setting is required while the MAC is active, software should terminate activity by clearing the *Enable* bit in the Receive Control or Transmit Control registers.

2.12 Multiple Transmit Queues

The MAC supports two transmit descriptor rings. Each ring functionality is implemented according to the description in Section 3.0. When software enables the two transmit queues it also must enable the *Multiple Request Support* field in the Transmit Control register. The priority between the queues can be set and specified in the memory space.

2.12.1 Quality of Service (QoS)

Supporting 802.1p, while classifying packets into different priority queues. When adding a priority value between queues it might be possible to support QoS by classifying packets to queues based on their priority.

2.12.2 Resource Locking Prevention

TARC0 and TARC1 registers enable the tuning of the arbitration parameters, which control the transmission of both transmission queues (see Section 10.0).

2.13 ICH9/ICH10 Virtual Technology Support

Virtual Machine Devices (VMD) are I/O devices specifically targeted for sharing in a virtual system. In a virtual system, multiple operating systems are loaded and each executes as though the entire system's resources are available. The following section describes the various aspects relating to the required support in a Virtual System (VM).

2.13.1 Function Level Reset (FLR)

An operating system in a VM must have a complete control over a device, including its initialization without interfering with normal platform operation. In this case, the MAC provides a software interface that enables the operating system to reset the entire device as if a PCI reset was asserted. This interface is called FLR (see Section 6.0).

2.13.2 ICH10 Receive Queuing for Virtual Machine Devices (VMDq)

VMDq is a mechanism to share I/O resources among several consumers. For example, in a virtual system, multiple operating systems are loaded and each executes as though the 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. Virtual Machine Devices (VMDs) 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 operating system or Virtual Machine (VM). **ICH9**'s two queues can be accessed by two VMs or more if configured properly. When **ICH10** is enabled for multiple queue direct access for VMs, it becomes a VMDq device.

Note: Most configuration and resources are shared across queues. System software must resolve any conflicts in configuration between the VMs.

Note: VMDq and Receive-Side Scaling (RSS) are mutually exclusive and should not be enabled at the same time.

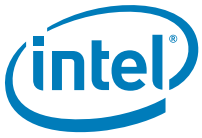
When enabled, VMDq assigns a 1-bit VMDq output index to each received packet. The VMDq output index is used to associate the packet to a receive queue.

2.14 ICH10 Multiple Receive Queues and Receive-Side Scaling (RSS)

The **ICH10** provides two hardware receive queues and filters each receive packet into one of the queues based on criteria that is described in Section 3.2.9. Classification of packets into receive queues enables Receive Side Scaling (RSS).

2.15 ICH10 LinkSec Encryption/Authentication Scheme

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 **ICH10**.



2.16 ICH10 Time Sync (IEEE1588 and 802.1as)

Measurement and control applications are increasingly using distributed system technologies such as network communication, local computing, and distributed objects. Many of these applications can be enhanced by having an accurate system-wide sense of time; achieved by having local clocks in each sensor, actuator, or other system device(s) synchronized and coordinated. Without a standardized protocol for synchronizing these clocks, it is unlikely that the benefits can be realized in the multi-vendor system component market. Existing protocols for clock synchronization are not optimized for these applications. For example, Network Time Protocol (NTP) targets large distributed computing systems with millisecond synchronization requirements. The 1588 standard specifically addresses the needs of measurement and control systems:

- Spatially localized
- Microsecond to sub-microsecond accuracy
- Administration free
- Accessible for both high-end devices and low-cost, low-end devices

The time sync mechanism activation is only possible in full duplex mode. There are no limitations on the wire speed although the wire speed might affect the accuracy.

3.0 Receive and Transmit Description

3.1 Introduction

This section describes the packet reception, packet transmission, transmit descriptor ring structure, TCP segmentation, and transmit checksum offloading for the Ethernet LAN controllers.

Note: Any programming of the transmit and receive settings must be made while the MAC is in an idle state. If an existing setting is required while the MAC is active, software should terminate activity by clearing the *Enable* bit in the Receive Control or Transmit Control registers.

3.2 Packet Reception

In general, packet reception consists of recognizing the presence of a packet on the wire, performing address filtering, storing the packet in the receive data FIFO, transferring the data to one of the two receive queues in host memory, and updating the state of a receive descriptor.

Note: The maximum supported received packet size is 16383 bytes for **ICH8** (9018 bytes for **ICH9/ICH10**).

3.2.1 Packet Address Filtering

Hardware stores incoming packets in host memory subject to the following filter modes. If there is insufficient space in the receive FIFO, hardware drops them and indicates the missed packet in the appropriate statistics registers. Figure 3 shows the address filtering as well as other filters that are described in the sections that follow.

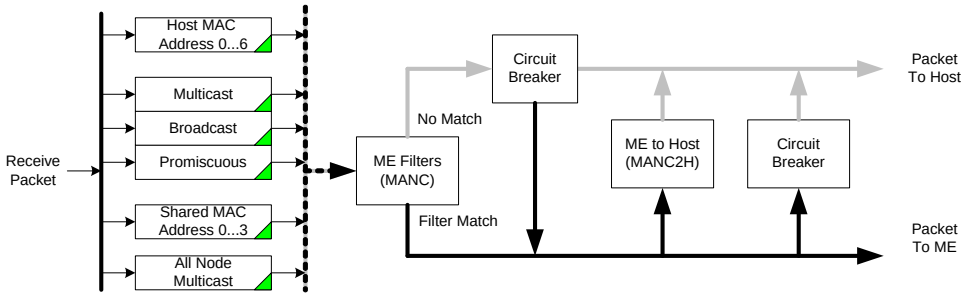


Figure 3. Receive Packet Filtering

Note: The ME and circuit breaker areas are beyond the scope of this section and a more detailed explanation of these areas can be found in the *Intel® I/O Controller Hub 8 (ICH8) Family External Design Specification (ICH8 EDS)*, *Intel® I/O Controller Hub 9 (ICH9) Family External Design Specification (ICH9 EDS),* or the *Intel® I/O Controller Hub 10 (ICH10) Family External Design Specification (ICH10 EDS).*

The following filter modes are supported:

- Exact Unicast/Multicast:
  - Dedicated Host Exact Unicast/Multicast: The destination address must exactly match one of seven stored addresses (accessed via the RAL and RAH registers). These addresses can be unicast or multicast.
  - **ICH9/ICH10** Dedicated Shared Exact Unicast/Multicast: The destination address must exactly match one of four stored addresses (accessed via the SHRAL and SHRAH registers). These addresses can be unicast or multicast. The SHRAL and SHRAH programming might be owned by either the host or the ME as defined by the *LockMAC* field in the FWSM register. By hardware default, the host owns all filters. However, if the ME is present, the ME can gate some of the SHRAL and SHRAH registers for ME programming.
  - **ICH9/ICH10** All Node Multicast: There is a dedicated hardwired filter for all node multicast packets with a destination MAC address equal to 33:33:00:00:00:01. The filter is enabled by the Shared MAC Address register SHRAH[3] and it follows the same filtering rules as all other SHARL and SHARH filters.
- Promiscuous Unicast: Receives all unicast filtering.
- Multicast: The upper bits of the incoming packet's destination address index a bit vector that indicates whether to accept the packet; if the bit in the vector is one, accept the packet, otherwise, reject it. The MAC provides a 4096 bit vector. Software provides a choice of four bits that can be used for indexing. These are [47:36], [46:35], [45:34], or [43:32] of the internally stored representation of the destination address.
- Promiscuous Multicast: Receive all multicast packets.

Note: When a promiscuous bit is set and a multicast packet is received, the *PIF* bit of the packet status is not set (not applicable to **ICH10**).

- Broadcast: The destination address of the incoming packets is broadcast address (FF:FF:FF:FF:FF:FF) while broadcast reception is enabled by the *Broadcast Accept Mode* bit in the RCTL register.
- Packet Filtering to Host vs. ME: Packets that matched any of the above MAC address filters can be routed to either the host, ME or both. Direction is made according to MANC setting, MANC2H setting, and additional registers that enable specific filters.

Normally, only good packets are received. A good packet is defined as one without a:

- CRC error
- Symbol error
- Sequence error
- Length error
- Alignment error
- Receive error being signaled by the PHY

However, if the store–bad–packet bit is set in the Device Control register (RCTL.SBP), then bad packets that 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 (RCTL.UPE/MPE) and the store–bad–packet bit (RCTL.SBP).

Note: All packets must have a valid SFD (RX_DV with no RX_ER in the 10/100/1000Base-T interface) in order to be recognized by the MAC). As a result, a CRC error is not counted in frames lacking an SFD.

3.2.2 Receive Data Storage

Memory buffers pointed to by descriptors store packet data. Hardware supports seven receive buffer sizes:

- 256 bytes
- 512 bytes
- 1024 bytes
- 2048 bytes
- 4096 bytes
- 8192 bytes
- 16384 bytes
- [1]

1. FLXBUF x 1024 bytes while FLXBUF = 1, 2, 3, . . . 15.

Buffer size is selected by bit settings in the Receive Control register (RCTL.BSIZE & RCTL.BSEX, RCTL.DTYP, RCTL, and FLXBUF). See Section 10.0 for details.

In legacy mode, the MAC places no alignment restrictions on packet buffer addresses. This is desirable in situations where the receive buffer was allocated by higher layers in the networking software stack, as these higher layers may have no knowledge of the MAC's buffer alignment requirements.

Although alignment is completely unrestricted, it is highly recommended that software allocate receive buffers on at least cache-line boundaries whenever possible.

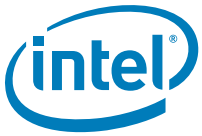
3.2.3 Legacy Receive Descriptor Format

A receive descriptor is a data structure that contains the receive data buffer address and fields for hardware to store packet information. If the RFCTL.EXSTEN bit is cleared and the RCTL.DTYP = 00b, the MAC uses the Legacy Rx Descriptor as shown in Table 4. The shaded areas indicate fields that are modified by hardware upon packet reception.

Table 4. Receive Descriptor (RDESC) Layout

	63 48	47 40	39 32	31 16	15 0
0	Buffer Address [63:0]				
8	VLAN Tag	Errors	Status 0	Packet Checksum (See Note)	Length

Note: The checksum indicated here is the unadjusted "16 bit ones complement" of the packet. A software assist might be required to back out appropriate information prior to sending it to upper software layers. The packet checksum is always reported in the first descriptor (even in the case of multi-descriptor packets).



3.2.3.1 Length Field (16-Bit; Offset 0)

Upon receipt of a packet for the MAC, hardware stores the packet data into the indicated buffer and writes the length, packet checksum, status, errors, and status fields. 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.

3.2.3.2 Packet Checksum (16-Bit; Offset 16)

For standard 802.3 packets (non-VLAN) the packet checksum is by default computed over the entire packet from the first byte of the DA through the last byte of the CRC, including the Ethernet and IP headers. Software can modify the starting offset for the packet checksum calculation by means of the Receive Control register. This register is described in Section 10.0. To verify the TCP/UDP checksum using the packet checksum, software must adjust the packet checksum value to back out the bytes that are not part of the true TCP checksum. When operating with the legacy Rx descriptor, the RXCSUM.IPPCSE and the RXCSUM.PCSD should be cleared (the default value).

For packets with VLAN header the packet checksum includes the header if VLAN striping is not enabled by the CTRL.VME. If VLAN header strip is enabled, the packet checksum and the starting offset of the packet checksum exclude the VLAN header.

3.2.3.3 Status 0 Field (8-Bit; Offset 32)

Status information indicates whether the descriptor has been used and whether the referenced buffer is the last one for the packet. Refer to Table 5 for the layout of the status field. Error status information is shown in Table 6.

Table 5. Receive Status (RDESC.STATUS) Layout

Receive Descriptor Status Bits	Description
PIF (bit 7)	Passed In-Exact Filter Hardware supplies the PIF field to expedite software processing of packets. Software must examine any packet with PIF set to determine whether to accept the packet. If PIF is clear, then the packet is known to be for this station, so software need not look at the packet contents. Packets passing only the Multicast Vector has PIF set. Note: This is a reserved bit for **ICH10**.
IPCS (bit 6)	IPv4 Checksum Calculated on Packet 0b = Do not perform IP checksum 1b = Perform IP checksum Pass/fail information regarding the checksum is indicated in the error bit (IPE) of the descriptor receive errors (RDESC.ERRORS) IPv6 packets do not have the IPCS bit set. Reads as 0b.
TCPCS (bit 5)	TCP Checksum Calculated on Packet 0b = Do not perform TCP/UDP checksum; 1b = Perform TCP/UDP checksum Pass/Fail information regarding the checksum is indicated in the error bit (TCPE) of the descriptor receive errors (RDESC.ERRORS). IPv6 packets may have this bit set if the TCP/UDP packet was recognized. Reads as 0b.
UDPCS (bit 4)	UDP Checksum Calculated on Packet

Receive Descriptor Status Bits	Description
VP (bit 3)	Packet is 802.1Q Indicates whether the incoming packet's type matches a VLAN (802.1q) type). It is set if the packet type matches CTRL.VME. For a further description of 802.1q VLANs, see Section 8.0. Reads as 0b.
Reserved (bit 2)	Reserved
EOP (bit 1)	End of Packet EOP indicates whether this is the last descriptor for an incoming packet.
DD (bit 0)	Descriptor Done Indicates whether hardware is done with the descriptor. When set along with EOP, the received packet is complete in main memory.

Note: See Table 12 for a description of supported packet types for receive checksum offloading. IPv6 packets do not have the IPCS bit set, but might have the TCPCS bit set if the MAC recognized the TCP or UDP packet.

3.2.3.4 Error Field (8-Bit; Offset 40)

Most error information appears only when the Store Bad Packets bit (RCTL.SBP) is set and a bad packet is received. Refer to Table 6 for a definition of the possible errors and their bit positions.

The error bits are valid only when the EOP and DD bits are set in the descriptor status field (RDESC.STATUS)

Table 6. Receive Errors (RDESC.ERRORS) Layout

Receive Descriptor Error bits	Description
RXE (bit 7)	RX Data Error Indicates that a data error occurred during the packet reception. In 10/100/1000Base-T mode, the assertion of I_RX_ER during data reception indicates a data error. This bit is valid only when the EOP and DD bits are set; it is not set in descriptors unless RCTL.SBP (Store Bad Packets) control bit is set.
IPE (bit 6)	IPv4 Checksum Error When set, indicates that IP checksum error is detected in the received packet. Valid only when the IP checksum is performed on the receive packet as indicated via the IPCS bit in the RDESC.STATUS field. If receive IP checksum offloading is disabled (RXCSUM.IPOFL), the IPE bit is set to 0b. It has no effect on the packet filtering mechanism. Reads as 0b.

Receive Descriptor Error bits	Description
TCPE (bit 5)	TCP/UDP Checksum Error When set, indicates that TCP/UDP checksum error is detected in the received packet. Valid only when the TCP/UDP checksum is performed on the receive packet as indicated via TCPCS bit in RDESC.STATUS field. If receive TCP/UDP checksum offloading is disabled (RXCSUM.TUOFL), the TCPE bit is set to 0b. It has no effect on the packet filtering mechanism. Reads as 0b.
RSV (bits 4:1)	Reserved Reads as 0b.
CE (bit 0)	CRC Error or Alignment Error CRC errors and alignment errors are both indicated via the CE bit. Software might distinguish between these errors by monitoring the respective statistics registers.

3.2.3.5 VLAN Tag Field (16-Bit; Offset 48)

Hardware stores additional information in the receive descriptor for 802.1q packets. If the packet type is 802.1q and RCTL.VME = 1b, then the VLAN Tag field records the VLAN information and the four byte VLAN information is stripped from the packet data storage. Otherwise, the special field contains 0000h.

Table 7. VLAN Tag Field Layout for 802.1g Packets

VLAN Tag Field	Description
VLAN	VLAN Identifier 12 bits that records the packet VLAN ID number
CFI	Canonical Form Indicator 1 bit that records the packet's CFI VLAN field
PRI	User Priority 3 bits that records the packet's user priority field.

3.2.4 Extended Rx Descriptor

If the RFCTL.EXSTEN bit is set and the RCTL.DTYP = 00b, the MAC uses the following Extended Rx descriptor.

Descriptor Read Format:

	63	0
0	Buffer Address [63:0]	
8	Reserved	0

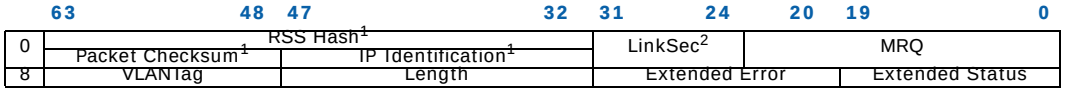
3.2.4.1 Buffer Address (64-Bit Offset 0.0)

This field contains the physical address of the receive data buffer. The size of the buffer is defined by the RCTL register (RCTL.BSIZE, RCTL.BSEX, RCTL.DTYP, RCTL, and FLXBUF fields).

3.2.4.2 DD (1-Bit Offset 8.0)

This is the location of the DD bit in the Status field. The software device driver must clear this bit before it handles the receive descriptor to the MAC. The software device driver can use this bit field later on as a completion indication of the hardware.

Descriptor Write Format:



1. Mutually exclusive by RXCSUM.PCSD.
2. **ICH10** only.

3.2.4.3 MRQ Field (32-Bit Offset 0.0)

Field	Bit(s)	Description
RSS Type	3:0	RSS Type Indicates the type of hash function used for RSS computation (see Table 8).
Reserved	7:4	Reserved
Queue[1]	12:8	Indicates the receive queue associated with the packet. It is generated by the indirection table as defined by the Multiple Receive Queues Enable field. This field is reserved when the Multiple Receive Queues Enable field of the Multiple Receive Queues Command register is set to 00b (Multiple Receive Queues are disabled).
Reserved	31:13	Reserved

1. Not applicable to **ICH10**.

3.2.4.4 RSS Type Decoding

The RSS Type field represents the hash type used by the RSS function as listed in Table 8.

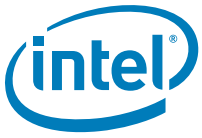


Table 8. RSS Type Decoding

Packet Type	Description
0h	No hash computation done for this packet.
1h	IPv4 with TCP hash used (NdisTcpIPv4).
2h	IPv4 hash used (NdisIPv4)
3h	IPv6 with TCP hash used (NdisTcpIPv6).
4h	IPv6 with extension header hash used (NdisIPv6Ex).
5h	IPv6 hash used (NdisIPv6).
6h - Fh	Reserved

3.2.4.5 LinkSec (8-Bit Offset 0.24); ICH10 Only

This field indicates LinkSec offload status and error reporting as follows:

31	30 29	28	27 25	24
Rsv	LSecE	Rsv	SAINDX	LSecH

LSecH (bit 24) - *LinkSec Valid* bit indicates that hardware detected LinkSec encapsulation version 0.

SAINDX (bits 27:25) - Indicates the SA index (note that the SC index = SA index/2).

LSecE (bits 30:29) - LinkSec Error code:

00b = No error
01b = No SA match
10b = Replay detection
1b1 = Bad LinkSec signature

Note: If strict mode is activated, meaning bad packets are dropped, the error code is omitted.

3.2.4.6 Packet Checksum (16-Bit Offset 0.48)

For standard 802.3 packets (non-VLAN) the packet checksum is by default computed over the entire packet from the first byte of the DA through the last byte of the CRC, including the Ethernet and IP headers. Software can modify the starting offset for the packet checksum calculation via the Receive Checksum Control Register (RXCSUM). To verify the TCP/UDP checksum using the packet checksum, software must adjust the packet checksum value to back out the bytes that are not part of the true TCP checksum. Likewise, for fragmented UDP packets, the packet checksum field can be used to accelerate UDP checksum verification by the host processor. This operation is enabled by the RXCSUM.IPPCSE bit.

For packets with VLAN header, the packet checksum includes the header if VLAN striping is not enabled by the CTRL.VME. If VLAN header strip is enabled, the packet checksum and the starting offset of the packet checksum exclude the VLAN header.

This field is mutually exclusive with the RSS Hash. It is enabled when the RXCSUM.PCSD bit is cleared.

3.2.4.7 IP Identification (16-Bit Offset 0.32)

This field stores the IP identification field in the IP header of the incoming packet. The software device driver should ignore this field when IPIDV is not set.

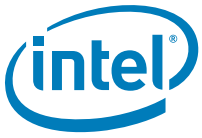
This field is mutually exclusive with the RSS Hash. It is enabled when the RXCSUM.PCSD bit is cleared.

3.2.4.8 RSS Hash (32-Bit Offset 0.32)

This field is mutually exclusive with the IP identification and the packet checksum. It is enabled when the RXCSUM.PCSD bit is set.

3.2.4.9 Extended Status (20-Bit Offset 8.0)

Field	Bit(s)	Description
Reserved	19:16	Reserved.
ACK	15	ACK Packet identification The ACK bit indicates that the received packet was an ACK packet with or without TCP payload depending on the *RFCTL.ACKD_DIS* bit.
Reserved	14:11	Reserved.
UDPV	10	Valid UDP XSUM The UDPV bit indicates that the incoming packet contains a valid (non-zero value) checksum field in an incoming Fragmented UDP IPv4 packet. It means that the Packet Checksum field contains the UDP checksum as described in this section. 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.
IPIDV	9	IP Identification Valid The IPIDV bit indicates that the incoming packet was identified as a fragmented IPv4 packet. The IPID field contains a valid IP Identification value if the RXCSUM.PCSD is cleared.
TST	8	Time Stamp Taken **Note:** This is a reserved bit for **ICH8/ICH9**.
PIF	7	Passed In-Exact Filter **Note:** This is a reserved bit for **ICH10**.
IPCS	6	IPv4 Checksum Calculated on Packet If active, hardware provides IPv4 checksum offload.
TCPCS	5	TCP Checksum Calculated on Packet If active withTCP checksum offload active, hardware provides IPv4 checksum offload. Pass/fail indication is provided in the Error field (IPE and TCPE). If IPCS is active and UDP checksum offload is active, hardware provides IPv4 checksum offload. Pass/fail indication is provided in the Error field (IPE and TCPE).
UDPCS	4	UDP Checksum Calculated on Packet If IPCS is active and fragmented UDP checksum offload is active, hardware provides IP checksum offoad for IPv4 packets. IP Pass/fail indication is provided in the IPE field. Fragmented UDP checksum is provided in the packet checksum field if the RXCSUM.PCSD bit is cleared.
VP	3	Packet is 802.1q.
Reserved	2	Reserved
EOP	1	End of Packet.
DD	0	Descriptor Done.



3.2.4.10 Extended Errors (12-Bit Offset 8.20)

Field	Bit(s)	Description
RXE	11	Rx Data Error.
IPE	10	IPv4 Checksum Error.
TCPE	9	TCP/UDP Checksum Error.
Reserved	8:5	Reserved. Should be set to 0000b.
CE	4	CRC Error or Alignment Error.
Reserved	3:0	Reserved. Should be set to 0000b.

3.2.5 Receive UDP Fragmentation Checksum

The MAC provides receive fragmented UDP checksum offload. The following setup should be made to enable this mode:

1. RXCSUM.PCSD bit should be cleared. The Packet Checksum and IP Identification fields are mutually exclusive with the RSS hash. When the PCSD bit is cleared, the Packet Checksum and IP Identification are active.
2. RXCSUM.IPPCSE bit should be set. This field enables the IP payload checksum enable that is designed for the fragmented UDP checksum.
3. RXCSUM.PCSS field must be zero. The packet checksum start should be zero to enable auto start of the checksum calculation. See table below for exact description of the checksum calculation.

The following table lists the outcome descriptor fields for the following incoming packet types:

Incoming Packet Type	Packet Checksum	IP Identification	UDPV/IPIDV	UDPCS/TCPCS
None IPv4 packet	Un-adjusted “16 bit ones complement” checksum of the entire packet (excluding VLAN header)	Reserved	0/0	0/0
Fragment IPv4 with TCP header	Same as above	Incoming IP Identification	0/1	0/0
Non-fragmented IPv4 packet	Same as above	Reserved	0b/0b	Depends on transport header and TUOFL field
Fragmented IPv4 without transport header	The unadjusted 1’s complement checksum of the IP payload	Incoming IP Identification	0b/1b	1b/0b
Fragmented IPv4 with UDP header	The un-adjusted 1’s complement checksum of the IP payload plus the pseudo header. Note that the UDP packet length is taken from the UDP header.	Incoming IP Identification	1b if the UDP header checksum is valid/1	1b/0b

Note: When the software device driver computes the “16 bit ones complement” checksum on the incoming packets of the UDP fragments, it should expect a value of FFFFh.

3.2.6 Packet Split Receive Descriptor

The MAC uses the packet split feature when the RFCTL.EXSTEN bit is set and RCTL.DTYP equals 01b. The software device driver must also program the buffer sizes in the PSRCTL register.

Table 9. Descriptor Read Format

	63:60	59:56	55:52	51:48	47:44	43:40	39:36	35:32	31:28	27:24	23:20	19:16	15:12	11:8	7:4	3:0
0	Buffer Address 0															
8	Buffer Address 1															
16	Buffer Address 2															
24	Buffer Address 3															

3.2.6.1 Buffer Addresses [3:0] (4 x 64-Bit)

The physical address of each buffer is written in the buffer addresses fields. The sizes of these buffers are statically defined by BSIZE0 to BSIZE3 in the PSRCTL register.

Note: All buffers' addresses in a packet split descriptor must be word aligned.

Packet headers cannot span across buffers. As a result, the size of the first buffer must be larger than any expected header size. Otherwise the packet will not be split.

If software sets a buffer size to zero, all buffers that follow should be set to zero as well. Pointers in the receive descriptors to buffers with a zero size should be set to anything but NULL pointers.

When configured to packet split and a given packet spans across two or more packet split descriptors, the first buffer of any descriptor (other than the first one) is not used.

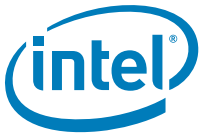
3.2.6.2 DD (1-Bit; Offset 8.0)

The software device driver can use the *DD* bit from the status field to determine when a descriptor has been used. Thus, the software device driver must ensure that the least significant bit of Buffer Address 1 is 0b. For proper operation, the buffers should be page aligned for any packet split. Any software device driver that cannot align buffers should not use this descriptor format.

Table 10. Descriptor Write-Back Format

	63:60	59:56	55:52	51:48	47:44	43:40	39:36	35:32	31:28	27:24	23:20	19:16	15:12	11:8	7:4	3:0
0	RSS Hash								MRQ							
	Packet Checksum				IP Identification											
8	VLAN Tag				Length 0				Extended Error				Extended Status			
16	Length 3				Length 2				Length 1				Header Status			
24	Reserved															

Notes: The shaded fields are mutually exclusive by RXCSUM.PCSD.
1. MRQ: Same as Extended Receive Descriptor.
2. LinkSec: Same as Extended Receive Descriptor (**ICH10 only**).
3. Packet Checksum, IP Identification, RSS Hash: Same as Extended Receive Descriptor.
4. Extended Status, Extended Errors, VLAN Tag: Same as Extended Receive Descriptor.



3.2.6.3 Length [3:1] (3 x 16-Bit; Offset 16.16), Length 0 (16-Bit; Offset 8.32)

Upon packet reception, hardware stores the packet data into one or more of the indicated buffers. The hardware writes in the length field of each buffer the number of bytes that were posted in the corresponding buffer. If no packet data is stored in a given buffer, hardware writes 0b in the corresponding length field. Length covers the data written to the receive buffer including CRC bytes.

Software is responsible for checking the *Length* fields of all buffers for data that hardware might have written to the corresponding buffers.

Table 11. Header Status (16-bit Offset 16.0)

15	14:10	9:0
HDRSP	Reserved	HLEN (Header Length)

3.2.6.4 HDRSP (Bit 15)

The HDRSP bit (when active) indicates that hardware split the headers from the packet data for the packet contained in this descriptor. Table 11 identifies the packets supported by header/data split functionality. Packets with a data portion smaller than 16 bytes are not guaranteed to be split. If the MAC is not configured to provide any offload that requires packet parsing, the *HDRSP* bit is set to 0b even if packet split was enabled. No split packets are stored linearly in the buffers of the receive descriptor.

3.2.6.5 HLEN (bits 9:0)

The HLEN field indicates the Header Length in byte count that was analyzed by the MAC.

3.2.7 Receive Descriptor Fetching

The fetching algorithm attempts to make the best use of PCI bandwidth by fetching a cache line (or more) descriptors with each burst. The following paragraphs briefly describe the descriptor fetch algorithm and the software control provided.

When the on-chip buffer is empty, a fetch happens as soon as any descriptors are made available (software writes to the tail pointer). When the on-chip buffer is nearly empty (RXDCTL.PTHRESH), a prefetch is performed whenever enough valid descriptors (RXDCTL.HTHRESH) are available in host memory and no other PCI activity of greater priority is pending (descriptor fetches and write-backs or packet data transfers).

When the number of descriptors in host memory is greater than the available on-chip descriptor storage, the chip may elect to perform a fetch which is not a multiple of cache line size. The hardware performs this non-aligned fetch if doing so results in the next descriptor fetch being aligned on a cache line boundary. This mechanism provides the highest efficiency in cases where fetches fall behind software.

Note: The MAC **never** fetches descriptors beyond the descriptor TAIL pointer.

3.2.8 Receive Descriptor Write-Back

Processors have cache line sizes that are larger than the receive descriptor size (16 bytes). Consequently, writing back descriptor information for each received packet would cause expensive partial cache line updates. Two mechanisms minimize the occurrence of partial line write backs:

- Receive descriptor packing
- Null descriptor padding

The following sections explain these mechanisms.

3.2.8.1 Receive Descriptor Packing

To maximize memory efficiency, receive descriptors are packed together and written as a cache line whenever possible. Descriptors accumulate and are written out in one of these conditions:

- RXDCTL.WTHRESH descriptors have been used (the specified max threshold of unwritten used descriptors has been reached)
- The last descriptors of the allocated descriptor ring have been used (allows the hardware to re-align to the descriptor ring start)
- The receive timer expires (RADV or RDTR)
- Explicit software flush (RDTR.FPD)

When the number of descriptors specified by RXDCTL.WTHRESH have been used, they are written back, regardless of cache line alignment. It is therefore recommended that WTHRESH be a multiple of cache line size. When a receive timer (RADV or RDTR) expires, all used descriptors are forced to be written back prior to initiating the interrupt, for consistency. Software can explicitly flush accumulated descriptors by writing the RDTR register with the high-order bit (FPD) set.

3.2.8.2 Null Descriptor Padding

Hardware stores no data in descriptors with a null data address. Software can make use of this property to cause the first condition under receive descriptor packing to occur early. Hardware writes back null descriptors with the *DD* bit set in the status byte and all other bits unchanged.

Note: Null Descriptor Padding is not supported for Packet Split descriptors.

3.2.9 Receive Descriptor Queue Structure

Figure 4 shows the structure of the two receive descriptor rings. Hardware maintains two circular queues of descriptors and writes back used descriptors just prior to advancing the head pointer(s). Head and tail pointers wrap back to base when size descriptors have been processed.

Software adds 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 hardware adjusts its internal tail pointer(s) accordingly. As packets arrive, they are stored in memory and the head pointer(s) is incremented by hardware. When the head pointer(s) is equal to the tail pointer(s), the queue(s) is empty. Hardware stops storing packets in system memory until software advances the tail pointer(s), making more receive buffers available.

The receive descriptor head and tail pointers reference 16-byte blocks of memory. Shaded boxes in the figure 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 non-zero status byte has been processed by the hardware, and is ready to be handled by the software.

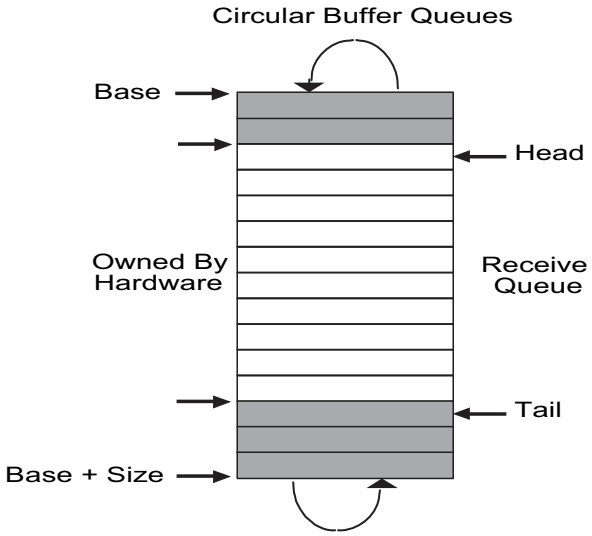


Figure 4. Receive Descriptor Ring Structure

When configured to operate using the packet split feature, the descriptor tail needs to be incremented by software by two for every descriptor ready in memory because packet split descriptors are 32 bytes and regular descriptors are 16 bytes.

Note: The head pointer points to the next descriptor that is 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 AND TAIL]**. Any descriptor not in this range is owned by software.

The receive descriptor ring is described by the following registers:

- Receive Descriptor Base Address registers (RDBA0 and RDBA1)
- These registers indicate 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. RDBA0 contains the lower 32-bits; RDBA1 contains the upper 32 bits. Hardware ignores the lower 4 bits in RDBA0.
- Receive Descriptor Length registers (RDLEN0 and RDLEN1)
- These registers determine 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 (RDH0 and RDH1)

- These registers hold a value that is an offset from the base, and indicates the in–progress descriptor. There can be up to 64 KB descriptors in the circular buffer. Hardware maintains a shadow copy that includes those descriptors completed but not yet stored in memory.
- Receive Descriptor Tail registers (RDT0 and RDT1)
- These registers hold a value that is an offset from the base, and identifies the location beyond the last descriptor hardware can process. Note that tail should still point to an area in the descriptor ring (somewhere between RDBA and RDBA + RDLEN). This is because tail points to the location where software writes the first new descriptor.

If software statically allocates buffers, and uses memory read to check for completed descriptors, it has to zero the status byte in the descriptor to make it ready for reuse by hardware. This is not a hardware requirement, but is necessary for performing an in–memory scan.

3.2.10 Receive Interrupts (Immediate and Delayed)

The following indicates the presence of new packets:

- Receiver Timer Interrupt (ICR.RXT0) due to packet delay timer (RDTR)
- Receiver Timer Interrupt (ICR.RXT0) due to absolute timer (RADV)
- Small Receive Packet Detect (ICR.SRPD)
- Receive ACK Frame Detect (ICR.ACK)
- Receive Descriptor Minimum Threshold (ICR.RXDMT)
- Receiver FIFO Overrun (ICR.RX0)

3.2.10.1 Receive Timer (ICR.RXT0) Due to Packet Delay Timer (RDTR)

Note: A pre-determined amount of time has elapsed since the last packet was received and transferred to host memory.

Every time a new packet is received and transferred to the host memory, the timer is re-initialized to the predetermined value. The timer then counts down and triggers an interrupt if no new packet is received and transferred to host memory completely before the timer expires. Software can set the timer value to 0 if it needs to be notified immediately (no interval delay) each time a new packet has been stored in memory.

Writing the absolute timer with its high order bit 1 forces an explicit flush of any partial cache lines worth of consumed descriptors. Hardware writes all used descriptors to memory and updates the globally visible value of the RXDH head pointer(s).

This timer is re-initialized when an interrupt is generated and restarts when a new packet is seen. It stays disabled until a new packet is received and transferred to host memory. The packet delay timer is also re-initialized when an interrupt occurs due to an absolute timer expiration or small packet-detection interrupt.

3.2.10.2 Receive Timer (ICR.RXT0) Due to Absolute Timer (RADV)

Note: A pre-determined amount of time has elapsed since the first packet was received after the hardware timer was written (specifically, after the last packet data byte was written to memory).

This timer is re-initialized when an interrupt is generated and restarts when a new packet is seen. It stays disabled until a new packet is received and transferred to host memory. The absolute delay timer is also re-initialized when an interrupt occurs due to a packet timer expiration or small packet-detection interrupt.

The absolute timer and the packet delay timer can be used together. The following table lists the conditions when the absolute timer and the packet delay timer are initialized, disabled, and when they start counting. The timer is always disabled if the value of RDTR = 0b.

Interrupt Timers	Starts Counting	When Re-Initialized	When Disabled
Absolute delay timer	Timer inactive and receive packet transferred to host memory.	At start.	On expiration. Due to other receive interrupt.
Packet delay timer	Timer inactive and receive packet transferred to host memory.	At start. New packet received and transferred to host memory	On expiration. Due to other receive interrupt.

Figure 5 shows the packet timer operation.

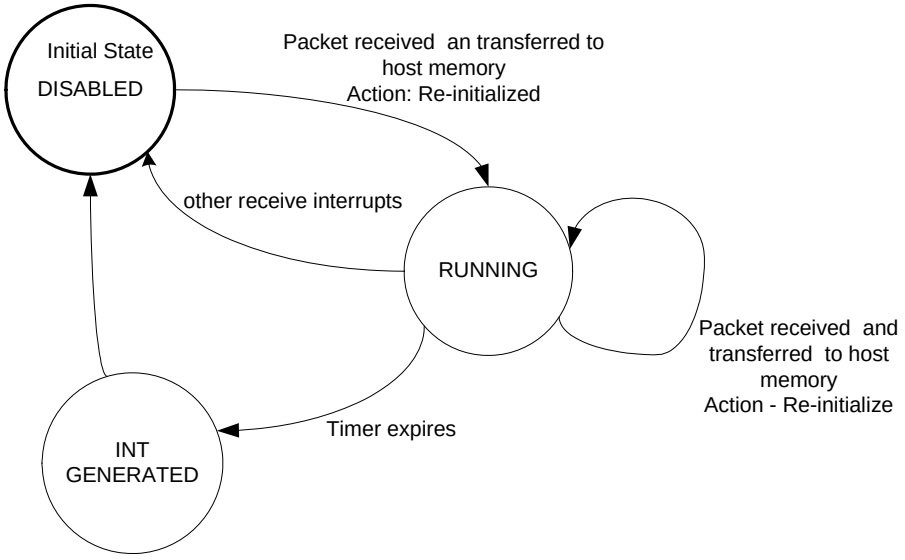
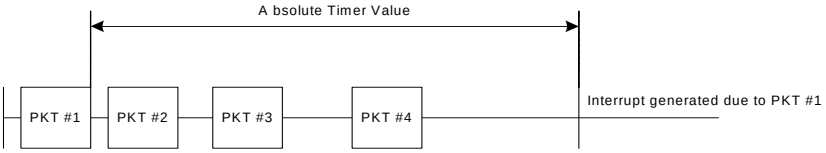


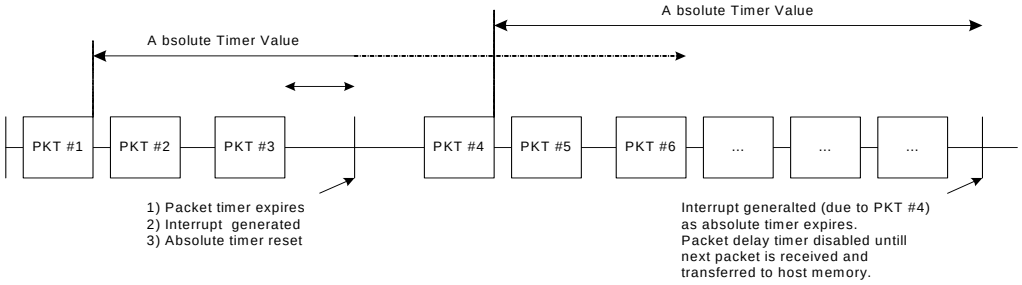
Figure 5. Packet Delay Timer Operation

The following diagrams show how the packet timer and absolute timer can be used together:

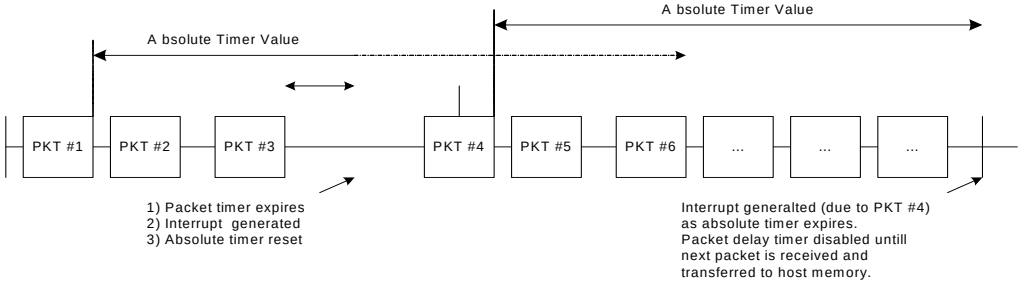
Case A: Using only an absolute timer



Case B: Using an absolute time in conjunction with the Packet timer



Case C: Packet timer expiring while a packet is transferred to host memory.
Illustrates that packet timer is re-started only after a packet is transferred to host memory.

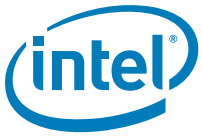


3.2.10.3 Small Receive Packet Detect (ICR.SRPD)

A receive interrupt is asserted when small-packet detection is enabled (RSRPD is set with a non-zero value) and a packet of (size < RSRPD.SIZE) has been transferred into the host memory. When comparing the size the headers, CRCs are included (if CRC stripping is not enabled). CRC and VLAN headers are not included if they have been stripped. A receive timer interrupt cause (ICR.RXT0) is also noted when the small packet detect interrupt occurs.

3.2.10.4 Receive ACK Frame Detect (ICR.ACK)

A receive ACK frame interrupt is asserted when a frame is detected to be an ACK frame. Detection of ACK frames are masked through the IMS register. When a frame is detected as an ACK frame an interrupt is asserted after the RAID.ACK_DELAY timer has expired and the ACK frame interrupts were not masked in the IMS register.



Note: The ACK Frame detect feature is only active when configured to Packet Split (RCTL.DTYP = 01b) or the Extended Status feature is enabled (RFCTL.EXSTEN is set).

3.2.10.5 Receive Descriptor Minimum Threshold (ICR.RXDMT)

The minimum descriptor threshold helps avoid descriptor under-run by generating an interrupt when the number of free descriptors becomes equal to the minimum. It is measured as a fraction of the receive descriptor ring size. This interrupt would stop and re-initialize the entire active delayed receives interrupt timers until a new packet is observed.

3.2.10.6 Receiver FIFO Overrun (ICR.RXO)

FIFO overrun occurs when hardware attempts to write a byte to a full FIFO. An overrun could indicate that software has not updated the tail pointer to provide enough descriptors/buffers, or that the PCI bus is too slow draining the receive FIFO. Incoming packets that overrun the FIFO are dropped and do not affect future packet reception. Note that this interrupt stops and re-initializes the entire active delayed receive interrupt process.

3.2.11 Receive Packet Checksum Offloading

The MAC supports the offloading of three receive checksum calculations: the packet checksum, the IPv4 header checksum, and the TCP/UDP checksum.

Note: IPv6 packets do not have IP checksums.

The packet checksum is the one's complement over the receive packet, starting from the byte indicated by RXCSUM.PCSS (0b corresponds to the first byte of the packet), after stripping. For packets with a VLAN header, the packet checksum includes the header if VLAN striping is not enabled by CTRL.VME. If the VLAN header strip is enabled, the packet checksum and the starting offset of the packet checksum exclude the VLAN header due to masking of VLAN header. For example, for an Ethernet II frame encapsulated as an 802.3ac VLAN packet with CTRL.VME and with RXCSUM.PCSS set to 14 decimal, the packet checksum includes the entire encapsulated frame, excluding the 14-byte Ethernet header (DA, SA, Type/Length) and the 4-byte q-tag. The packet checksum does not include the Ethernet CRC if the RCTL.SECRC bit is set.

Software must make the required offsetting computation (to back out the bytes that should not have been included and to include the pseudo-header) prior to comparing the packet checksum against the TCP checksum stored in the packet.

For supported packet/frame types, the entire checksum calculation can be off-loaded to the MAC. If RXCSUM.IPOFLD is set to 1b, the MAC 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, if RXCSUM.TUOFLD is set to 1b, the MAC 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.TCPCS respectively). In addition, if RFCTL.Ipv6_DIS and RFCTL.IP6Xsum_DIS are cleared to 0b and RXCSUM.TUOFLD is set to 1b, the MAC 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).

If neither RXCSUM.IPOFLD nor RXCSUM.TUOFLD is set, the checksum error bits (IPE and TCPE) is 0b for all packets.

Supported Frame Types include:

- Ethernet II
- Ethernet SNAP

Table 12. Supported Receive Checksum Capabilities

Packet Type	HW IP Checksum Calculation	HW TCP/UDP Checksum Calculation
IPv4 packets	Yes	Yes
IPv6 packets	No (n/a)	Yes
IPv6 packet with next header options: Hop-by-Hop options Destinations options Routing (with LEN = 0) Routing (with LEN > 0) Fragment Home Option	 No (n/a) No (n/a) No (n/a) No (n/a) No (n/a) No (n/a)	 Yes Yes Yes No No No
IPv4 tunnels: • IPv4 packet in an IPv4 tunnel • IPv6 packet in an IPv4 tunnel	 No Yes (IPv4)	 No Yes[1]
IPv6 tunnels: • IPv4 packet in an IPv6 tunnel • IPv6 packet in an IPv6 tunnel	 No No	 No No
Packet is an IPv4 fragment	Yes	No
Packet is greater than 1552 bytes; (LPE=1b)	Yes	Yes
Packet has 802.3ac tag	Yes	Yes
IPv4 Packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet has TCP or UDP options	Yes	Yes
IP header's protocol field contains a protocol # other than TCP or UDP.	Yes	No

1. The IPv6 header portion can include supported extension headers as described in the IPv6 Filter section.

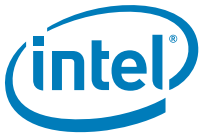
Table 12 lists the general details about what packets are processed. In more detail, the packets are passed through a series of filters (Section 3.2.11.1 through Section 3.2.11.5) to determine if a receive checksum is calculated.

3.2.11.1 MAC Address Filter

This filter checks the MAC destination address to be sure it is valid (IA match, broadcast, multicast, etc.). The receive configuration settings determine which MAC addresses are accepted. See the various receive control configuration registers such as RCTL (RTCL.UPE, RCTL.MPE, RCTL.BAM), MTA, RAL, RAH, SHRAL, and SHRAH.

3.2.11.2 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 RCTL.



3.2.11.3 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 5 (dwords). If the IPv4 header is properly decoded, the IP checksum is checked for validity. The RXCSUM.IPOFL bit must be set for this filter to pass.

3.2.11.4 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 size next header accepted is 16 dwords (64 bytes).

All of the IPv6 extension headers supported by the Ethernet controller have the same header structure:

Byte 0	Byte 1	Byte 2	Byte 3
Next Header	HDR EXT LEN		

- NEXT HEADER is a value that identifies the header type. The supported IPv6 next headers values are:
  - Hop-by-Hop = 00h
  - Destination Options = 3Ch
  - Routing = 2Bh
- HDR EXT LEN is the 8 byte count of the header length, not including the first 8 bytes. For example, a value of 3 means that the total header size including the NEXT HEADER and HDR EXT LEN fields is 32 bytes (8 + 3*8).
  - The *RFCTL.Ipv6_DIS* bit must be cleared for this filter to pass.

3.2.11.5 UDP/TCP Filter

This filter checks for a valid UDP or TCP header. The prototype next header values are 11h and 06h, respectively. The RXCSUM.TUOFL bit must be set for this filter to pass.

3.2.12 ICH9/ICH10 Early Receive DMA

The early receive DMA feature can be used to reduce the latency between the time a packet is recognized on the wire and when the last byte of packet data is transferred to host memory. Normally, an entire receive packet is stored in the MAC's packet buffer before any of the packet data is transferred to host memory. The early receive DMA feature enables data to be transferred to host memory before the entire packet is received on the wire.

Setting the Early Receive Threshold (ERT) register to a non-zero value enables this feature. The ERT register specifies the number of quad-words of packet data that must be stored in the packet buffer before a host DMA is requested. Once the threshold has been met, any available data in the packet buffer is transferred to host memory. The MAC then waits for the threshold to be met once again before performing another DMA. This process continues until either the complete packet is received or the packet is rejected due to an error. Only after the entire packet is received will any descriptors be written back. This prevents software from processing descriptors for packets that were eventually rejected.

Note: Although all descriptors are written only after the full packet is in the host memory, only the last descriptor of an early-received packet has the correct status of the packet within it.

3.2.13 Multiple Receive Queues and Receive-Side Scaling (RSS)

The MAC provides two hardware receive queues and filters each receive packet into one of the queues based on criteria described in the sections that follow. Classification of packets into receive queues enables Receive Side Scaling (RSS).

Multiple receive queues are enabled when the packet checksum bit (RXCSUM.PCSD) is set (packet checksum is disabled) and the *Multiple Receive Queues Enable* bits do not equal 00b. Multiple receive queues are mutually exclusive with UDP fragmentation and are not supported when a legacy receive descriptor format is used. Note that multiple receive queue status is not reported in the receive packet descriptor and the interrupt mechanism bypasses the interrupt scheme. Instead, a receive packet is issued directly by the interrupt logic.

When multiple receive queues are enabled, the MAC provides several types of information to software. Some are requirements of RSS while others are provided for software device driver assistance:

- RSS can only be enabled during initialization. Disabling/enabling the RSS feature during normal operation is not recommended.
- In multiple queues, the last descriptor pointed by the software device driver (the one before the Tx tail) should not be context/null.
- A dword result of the RSS hash function. This is used by the stack for flow classification and is written into the receive packet descriptor (required by RSS).
- A 4-bit RSS Type field. This conveys the hash function used for the specific packet (required by RSS).
- A 5-bit output of a redirection table. This identifies the CPU to process the packet and is written into the receive packet descriptor (for software device driver use).
- A mechanism for issuing an interrupt to one or more CPUs.

The following summarizes the process of classifying a packet into a receive queue (see Figure 6):

1. The receive packet is parsed into the header fields used by the hash operation (for example, IP addresses, TCP port, etc.)
2. A hash calculation is performed. The MAC supports a single hash function as defined by RSS. The MAC does not indicate to the software device driver which hash function is used. The 32-bit result is fed into the receive packet descriptor.
3. The seven LSBs of the hash result are used as an index into a 128-entry Redirection Table. Each entry in this table contains a 5-bit CPU number. This 5-bit value is fed into the packet receive descriptor. In addition, each entry provides a single bit queue number, which denotes the queue into which the packet is routed.

When multiple request queues are disabled, packets enter hardware queue 0. System software can enable or disable RSS at any time. While RSS is disabled, system software can update the contents of any of the RSS related registers.

When multiple request queues are enabled in RSS mode, undecoded packets enter hardware queue 0. The 32-bit tag, which is normally a result of the hash function, equals 0. The 5-bit MRQ field also equals 0.

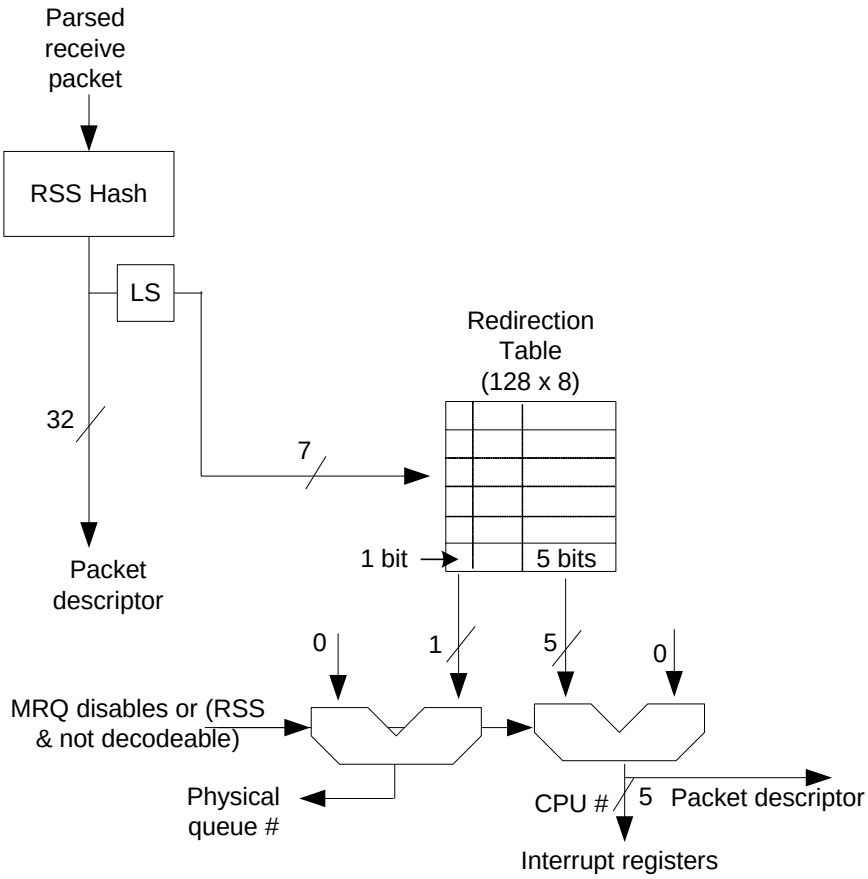


Figure 6. RSS Block Diagram

3.2.13.1 RSS Hash Function

A single hash function is defined with five variations for the following cases:

- TCP/IPv4. The MAC 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 performed.
- IPv4. The MAC parses the packet to identify an IPv4 packet. If the packet is not an IPv4 packet, RSS is not performed.
- TCP/IPv6. The MAC parses the packet to identify an IPv6 packet containing a TCP segment per the criteria described below. If the packet is not an IPv6 packet containing a TCP segment, RSS is not done. Extension headers should be parsed for a home address option field (for the source address) or the routing header type 2 field (for the destination address)[1].

1. Both extensions are defined in the IETF draft, “Mobility Support in IPv6 dated 10/29/2002.

- IPv6Ex. The MAC parses the packet to identify an IPv6 packet. Extension headers should be parsed for a *Home Address Option* field (for the source address) or the routing header type 2 field (for the destination address). The packet is not required to contain any of these extension headers to be hashed by this function. If the packet is not an IPv6 packet, RSS is not done for the packet.

Note: Each time the operating system enables IPv6Ex (*Hash Select*, bit 3) the software device driver should also enable IPv6 (*Hash Select*, bit 4). This ensures that the packet is hashed. In addition, the software device driver should convert all the received packets that are tagged as IPv6 (type 5) to IPv6Ex (type 4).

- IPv6. The MAC parses the packet to identify an IPv6 packet. If the packet is not an IPv6 packet, RSS is not done for the packet.

Two configuration bits impact the choice of the hash function as described above:

- IPv6_ExDIS bit in the Receive Filter Control Register (RFCTL). When this is set, if an IPv6 packet includes extension headers, then the TCP/IPv6Ex and IPv6Ex functions are not used.
- NEW_IPV6_EXT_DIS bit in Receive Filter Control Register (RFCTL). When this is set, if an IPv6 packet includes either a home address option or a routing header type 2, then the TCP/IPv6Ex and IPv6Ex functions are not used.

A packet is identified as containing a TCP segment if all of the following conditions are met:

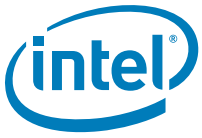
- The transport layer protocol is TCP.
- The TCP segment can be parsed (for example, IP options can be parsed or the packet is not encrypted).
- The packet is not IP fragmented (even if the fragment contains a complete TCP header).

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.
  a. Use the TCP/IPv4 function. If it does not meet the requirements, move to the next step (IPv4 function).
  b. Use the IPv4 function.
- IPv6 Packet.
  a. Use the TCP/IPv6Ex function. If it does not meet the requirements, move to the next step.
  b. Use the IPv6Ex function. If does not meet the requirements, move to the next step.
  c. Use the IPv6 function.

The following combinations are currently supported. Other combinations may be supported in future products.

1. IPv4 hash types:
   — S1a: TCP/IPv4 is enabled as defined above.
   — S1b: both TCP/IPv4 and IPv4 are enabled. The packet is first parsed according to TCP/IPv4 rules. If it is not identified as a TCP/IPv4 packet, it is parsed as an IPv4 packet.



2. IPv6 hash types:
   — S2a: IPv6 is enabled as defined above.
   — S2b: TCP/IPv6, IPv6EX, and IPv6Ex are enabled. The packet is first parsed according to TCP/IPv6 rules. If it is not identified as a TCP/IPv6 packet, it is parsed as an IPv6Ex packet. If the MAC cannot parse extensions headers (such as an unidentified extension in the packet), then the packet is parsed as IPv6.

When a packet cannot be parsed by the above rules, it enters hardware queue 0. The 32-bit tag (which is a result of the hash function) equals 0. The 5-bit MRQ field also equals zero.

In the case of tunneling (for example, IPv4-IPv6 tunnel), the external IP address (in the base header) is used.

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 hash functions below:

- Ordering is little endian in both bytes and bits. For example, the IP address 161.142.100.80 translates into A18E 6450h 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 generated and supplied through the RSS Random Key Register (RSSRK).

The algorithm works by examining each bit of the hash input from left to right. Intel 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:

```
K[0] K[1] K[2] … K[k-1]
```

K[0] is the left most byte, and the most significant bit of K[0] is the left most bit. K[k-1] is the right most byte, and the least significant bit 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;
```

The following four pseudo-code examples are intended to help clarify exactly how the hash can be performed using four cases: IPv4 with and without ability to parse the TCP header and IPv6 with and without a TCP header.

3.2.13.1.1 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);
```

3.2.13.1.2 Hash for IPv4 without TCP

```
Concatenate SourceAddress and DestinationAddress into one single byte-array

Input[8] = @12-15, @16-19

Result = ComputeHash(Input, 8)
```

3.2.13.1.3 Hash for IPv6 with TCP

```
Similar to above:

Input[36] = @8-23, @24-39, @40-41, @42-43

Result = ComputeHash(Input, 36)
```

3.2.13.1.4 Hash for IPv6 without TCP

```
Input[32] = @8-23, @24-39

Result = ComputeHash(Input, 32)
```

3.2.13.2 Redirection Table

The redirection table is a 128-entry structure, indexed by the 7 least significant bits of the hash function output. Each entry of the table contains the following:

- Bit [7]: Queue Index.
- Bits [6:5]: Reserved.
- Bits [4:0]: CPU Index.

The CPU value indexed by the hash function is written into the packet descriptor to serve as an indication of the CPU that should process the packet. The Queue Index determines the physical queue for the packet.

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 will take effect on a specific packet boundary.

3.2.14 RSS Verification Suite

This section contains the values used in the given examples. 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
```

3.2.14.1 IPv4

Destination Address/ Port	Source Address/Port	IPv4 Only	IPv4 with TCP
161.142.100.80 :1766	66.9.149.187 : 2794	323E 8FC2h	51CC C178h
65.69.140.83 : 4739	199.92.111.2 : 14230	D718 262Ah	C626 B0EAh
12.22.207.184 : 38024	24.19.198.95 :12898	D2D0 A5DEh	5C2B 394Ah
209.142.163.6 : 2217	38.27.205.30 : 48228	8298 9176h	AFC7 327Fh
202.188.127.2 :1303	153.39.163.191 : 44251	5D18 09C5h	10E8 28A2h

3.2.14.2 IPv6

The IPv6 address tuples are only for verification purposes and may not make sense as a tuple.

Destination Address/Port	Source Address/Port	IPv6 Only	IPv6 with TCP
3FFE:2501:200:1FFF::7 (1766)	3FFE:2501:200:3::1 (2794)	2CC1 8CD5	4020 7D3D
FF02::1 (4739)	3FFE:501:8::260:97FF:FE40:EFAB (14230)	0F0C 461C	DDE5 1BBF
FE80::200:F8FF:FE21:67CF (38024)	3FFE:1900:4545:3:200:F8FF:FE 21:67CF (44251)	4B61 E985	02D1 FEEF

3.3 Packet Transmission

The transmission process for regular (non-TCP segmentation packets) involves:

- 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.
- For each packet of the data block:
  - — Ethernet, IP and TCP/UDP headers are prepared by the stack.
  - — The stack interfaces with the software device driver and commands the driver to send the individual packet.
  - — The software device driver gets the frame and interfaces with the hardware.
  - — The hardware reads the packet from host memory (via DMA transfers).

— The driver returns ownership of the packet to the Network Operating System (NOS) when the hardware has completed the DMA transfer of the frame (indicated by an interrupt).

Output packets are made up of pointer–length pairs constituting a descriptor chain (so called descriptor based transmission). Software forms transmit packets by assembling the list of pointer–length pairs, storing this information in the transmit descriptor, and then updating the on–chip transmit tail pointer to the descriptor. The transmit descriptor and buffers are stored in host memory. Hardware typically 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 permits TCP or UDP checksum computation, and avoids problems with PCI underruns.

Note: When the link goes down during a Tx packet, this packet might be re-transmitted at link up.

Another transmit feature is TCP segmentation. The hardware has the capability to perform packet segmentation on large data buffers off-loaded from the Network Operating System (NOS). This feature is described in detail in Section 3.5.

3.3.1 Transmit Data Storage

Data is stored in buffers pointed to by the descriptors. Alignment of data is on an arbitrary byte boundary with the maximum size per descriptor limited only to the maximum allowed packet size (16288[1] bytes). A packet typically consists of two (or more) descriptors, one (or more) for the header and one for the actual data. Some software implementations copy the header(s) and packet data into one buffer and use only one descriptor per transmitted packet.

3.3.2 Transmit Descriptors

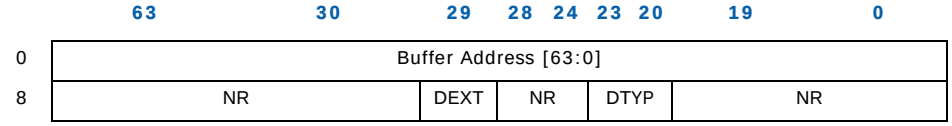
The MAC provides three types of transmit descriptor formats.

The original descriptor is referred to as the legacy descriptor format. The other two descriptor types are collectively referred to as extended descriptors. One of them is similar to the legacy descriptor in that it points to a block of packet data. This descriptor type is called the TCP/IP Data Descriptor and is a replacement for the legacy descriptor since it offers access to new offloading capabilities. The other descriptor type is fundamentally different as it does not point to packet data. It merely contains control information that is loaded into registers of the MAC and affect the processing of future packets. The following sections describe the three descriptor formats.

Note: The extended descriptor types are accessed by setting the TDESC.DEXT bit to 1b. If this bit is set, the TDESC.DTYP field is examined to control the interpretation of the remaining bits of the descriptor. Table 13 shows the generic layout for all extended descriptors. Fields marked as NR are not reserved for any particular function and are defined on a per-descriptor type basis. Note that the DEXT and DTYP fields are non-contiguous in order to accommodate legacy mode operation. For legacy mode operation, bit 29 is set to 0b and the descriptor is defined in Section 3.3.3.

1. The maximum allowable transmit packet size is the transmit allocation minus 96 bytes with an upper limit of 24 KB due to receive synchronization limitations.

Table 13. Transmit Descriptor (TDESC) Layout

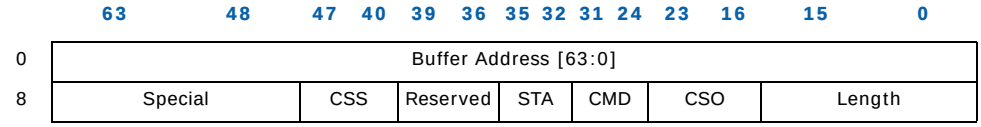


	63 30	29	28 24	23 20	19 0
0	Buffer Address [63:0]				
8	NR	DEXT	NR	DTYP	NR

3.3.3 Legacy Transmit Descriptor Format

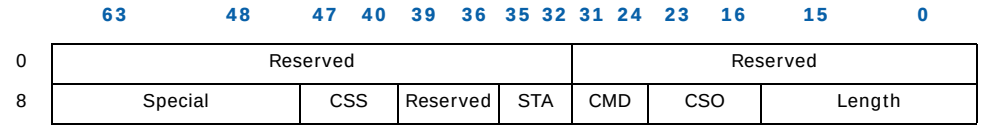
To select legacy mode operation, bit 29 (TDESC.DEXT) should be set to 0b. In this case, the descriptor format is defined as shown in Table 15. The address and length must be supplied by software. Bits in the command byte are optional, as are the Checksum Offset (CSO), and Checksum Start (CSS) fields.

Table 14. 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	Special	CSS	Reserved	STA	CMD	CSO	Length

Table 15. Transmit Descriptor Write-Back Format



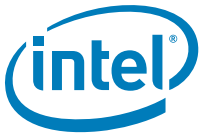
	63 48	47 40	39 36	35 32	31 24	23 16	15 0
0	Reserved				Reserved		
8	Special	CSS	Reserved	STA	CMD	CSO	Length

Table 16. Transmit Descriptor Legacy Descriptions

Transmit Descriptor Legacy	Description
Length	Length (TDESC.LENGTH) specifies the length (in bytes) that is fetched from the buffer address provided. The maximum length associated with any single legacy descriptor is 16288 bytes. Descriptor length(s) can be further limited by the size of the transmit FIFO. Due to the need to support optional checksum calculation and insertion, all buffers comprising a single packet must be able to be stored simultaneously in the transmit FIFO. For any individual packet, the sum of the individual descriptors' lengths must be at least 80 bytes less than the allocated size of the transmit FIFO. **Note:** The maximum allowable packet size for transmits changes based on the value written to the Packet Buffer Allocation register.
CSO/CSS	Checksum Offset and Start 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. Both CSO and CSS are in units of bytes.These must both be in the range of data provided to the MAC in the descriptor. This means for short packets that are padded by software, CSS and CSO must be in the range of the unpadded data length, not the eventual padded length (64 bytes). In case of 802.1Q header, the offset values depends on the VLAN insertion enable bit - CTRL.VME and the VLE bit. In case they are not set (VLAN tagging included in the packet buffers), the offset values should include the VLAN tagging. In case these bits are set (VLAN tagging is taken from the packet descriptor), the offset values should exclude the VLAN tagging. **Note:** UDP checksum calculation is not supported by the legacy descriptor. Because the CSO field is 8 bits wide, it limits the location of the checksum to 255 bytes from the beginning of the packet. When End Of Packet (EOP) is set, this indicates the last descriptor making up the packet. One or many descriptors can be used to form a packet. The MAC inserts a checksum at the offset indicated by the CSO field if the *Insert Checksum* bit (IC) is set. Checksum calculations are for the entire packet starting at the byte indicated by the CSS field. A value of 0b corresponds to the first byte in the packet. Hardware ignores IC, and CSO unless EOP is set. CSS must be set in the first descriptor for a packet. In addition, IC is ignored if CSO or CSS are out of range. This occurs if (CSS >/= length) OR (CSO length - 1).
CMD	Command Field See Section 3.3.3.1 for a detailed field description.
STA	Status Field See Section 3.3.3.2 for a detailed field description.
RSV	Reserved Should be written with 0b for future compatibility.
Special	Special Field See the notes that follow this table for a detailed field description.

Notes:

1. Even though CSO and CSS are in units of bytes, the checksum calculation typically works on 16-bit words. Hardware does not enforce even byte alignment.
2. 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.
3. Although the MAC can be programmed to calculate and insert TCP checksum using the legacy descriptor format, it is recommended that software use the TCP/IP Context Transmit Descriptor format. This descriptor format enables hardware to calculate both the IP and TCP checksums for outgoing packets. See Section 3.3.5 for more information about how this descriptor format can be used to accomplish this task.



3.3.3.1 Transmit Descriptor Command Field Format

The CMD byte stores the applicable command and has fields shown in Table 17.

Note: Software must compute an offsetting entry to back out the bytes of the header that are not included in the TCP checksum and then store it in the position where the hardware computed checksum needs to be inserted.

Table 17. Transmit Command (TDESC.CMD) Layout

TDESC.CMD	Description
IDE (bit 7)	Interrupt Delay Enable When set, activates the transmit interrupt delay timer. The MAC loads a countdown register when it writes back a transmit descriptor that has RS and IDE set. The value loaded comes from the IDV field of the Interrupt Delay (TIDV) register. When the count reaches 0, a transmit interrupt occurs if transmit descriptor write-back interrupts (IMS.TXDW) are enabled. The MAC always loads the transmit interrupt counter each time it processes a descriptor with IDE set even if it is already counting down due to a previous descriptor. If the MAC encounters a descriptor that has RS set, but not IDE, it generates an interrupt immediately after writing back the descriptor. The interrupt delay timer is cleared.
VLE (bit 6)	VLAN Packet Enable When set, indicates that the packet is a VLAN packet and the MAC should add the VLAN Ethertype and an 802.1q VLAN tag to the packet. The VLAN tag comes from the special field of the TX descriptor. The hardware inserts the FCS/CRC field in that case. When cleared, the MAC sends a generic Ethernet packet. The IFCS controls the insertion of the FCS field in that case. In order to have this capability CTRL.VME bit should also be set, otherwise VLE capability is ignored. VLE is valid only when EOP is set.
DEXT (bit 5)	Extension (0b for legacy mode). Should be written with 0b for future compatibility.
RSV (bit 4)	Reserved Should be programmed to 0b.
RS (bit 3)	Report Status This bit enables the MAC to report the status information. This is used by the software that does in-memory checks of the transmit descriptors to determine which ones are done. For example, if software queues up 10 packets to transmit, it can set the RS bit in the last descriptor of the last packet. If software maintains a list of descriptors with the RS bit set, it can look at them to determine if all packets up to (and including) the one with the RS bit set have been buffered in the output FIFO. Looking at the status byte and checking the *Descriptor Done* (DD) bit do this. If DD is set, the descriptor has been processed.

TDESC.CMD	Description
IC (bit 2)	Insert Checksum When set, the MAC needs to insert a checksum at the offset indicated by the CSO field. The checksum calculations are performed for the entire packet starting at the byte indicated by the CCS field. IC is ignored if CSO and CCS are out of the packet range. This occurs when (CSS ≥ length) OR (CSO ≥ length - 1). IC is valid only when EOP is set.
IFCS (bit 1)	Insert FCS When set, the MAC appends the FCS at the end of the packet. When cleared, software should calculate the FCS for proper CRC check. Following are cases where software must set IFCS: Short packet transmission while padding is enabled by the *TCTL.PSP* 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 Large send or TCP/IP checksum offload using the context descriptor
EOP (bit 0)	End Of Packet When set, indicates the last descriptor making up the packet. One or many descriptors can be used to form a packet.

Notes:
1. VLE, IFCS, and IC (also ILSec and TimeStamp for **ICH10**) are qualified by EOP. That is, hardware interprets these bits ONLY when EOP is set.
2. Hardware only sets the DD bit for descriptors with RS set.
3. Descriptors with the null address (0b) or zero length transfer no data. If they have the RS bit set then the DD field in the status word is written when hardware processes them.
4. Although the transmit interrupt may be delayed, the descriptor write-back requested by setting the RS bit is performed without delay unless descriptor write-back bursting is enabled.

3.3.3.2 Transmit Descriptor Status Field Format

The *Status* field stores the applicable transmit descriptor status and has the fields shown in Table 18.

The transmit descriptor status field is only present in cases where RS is set in the command field.

Table 18. Transmit Status Layout

TDESC.STATUS	Description
RSV (bit 3)	Reserved. Should be programmed to 0b.
LC (bit 2)	Late Collision Indicates that late collision occurred while working in half-duplex mode. It has no meaning while working in full-duplex mode. Note that the collision window is speed dependent: 64 bytes for 10/100 Mb/s and 512 bytes for 1000 Mb/s operation.
EC (bit 1)	Excess Collisions Indicates that the packet has experienced more than the maximum excessive collisions as defined by TCTL.CT control field and was not transmitted. It has no meaning while working in full-duplex mode.
DD (bit 0)	Descriptor Done Indicates that the descriptor is finished and is written back either after the descriptor has been processed (with RS set).

Note: The DD bit reflects status of all descriptors up to and including the one with the RS bit set.

3.3.4 Transmit Descriptor Special Field Format

The Special field is used to provide the 802.1q/802.1ac tagging information.

When CTRL.VME is set to 1b, all packets transmitted from the MAC that have VLE set in the TDESC.CMD are sent with an 802.1Q header added to the packet. The contents of the header come from the transmit descriptor special field and from the VLAN type register. The special field is ignored if the VLE bit in the transmit descriptor command field is 0b. The special field is valid only for descriptors with EOP set to 1b in TDESC.CMD.

Table 19. Special Field (TDESC.SPECIAL) Layout

TDESC.SPECIAL	Description
PRI	User Priority 3 bits that provide the VLAN user priority field to be inserted in the 802.1Q tag.
CFI	Canonical Form Indicator.
VLAN	VLAN Identifier 12 bits that provide the VLAN identifier field to be inserted in the 802.1Q tag.

3.3.5 TCP/IP Context Transmit Descriptor Format

The TCP/IP context transmit descriptor provides access to the enhanced checksum offload facility available in the MAC. This feature enables 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.

The TCP/IP context transmit descriptor does not point to packet data as a data descriptor does. Instead, this descriptor provides access to an on-chip context that supports the transmit checksum offloading feature of the MAC. A context refers to a set of registers loaded or unloaded as a group to provide a particular function. Note that only one context is active at any given time.

The context is explicit and directly accessible via the TCP/IP context transmit descriptor. The context is used to control the checksum offloading feature for normal packet transmission.

The MAC automatically selects the appropriate legacy or normal context to use based on the current packet transmission.

While the architecture supports arbitrary ordering rules for the various descriptors, there are restrictions including:

- Context descriptors should not occur in the middle of a packet.
- Data descriptors of different packet types (legacy or normal) should not be intermingled except at the packet level.

All contexts control calculation and insertion of up to two checksums. This portion of the context is referred to as the checksum context.

In addition to checksum context, the segmentation context adds information specific to the segmentation capability. This additional information includes the total payload for the message (TDESC.PAYLEN), the total size of the header (TDESC.HDRLEN), the amount of payload data that should be included in each packet (TDESC.MSS), and information about what type of protocol (TCP, IPv4, IPv6, etc.) is used. This information is specific to the segmentation capability and is therefore ignored for context descriptors that do not have the TSE bit set.

Because there are dedicated resources on-chip for the normal context, the context remains constant until it is modified by another context descriptor. This means that a context can be used for multiple packets (or multiple segmentation blocks) unless a new context is loaded prior to each new packet. Depending on the environment, it might be completely unnecessary to load a new context for each packet. For example, if most traffic generated from a given node is standard TCP frames, this context could be set up once and used for many frames. Only when some other frame type is required would a new context need to be loaded by software. After the non-standard frame is transmitted, the standard context would be setup once more by software. This method avoids the extra descriptor per packet penalty for most frames. The penalty can be eliminated altogether if software elects to use TCP/IP checksum offloading only for a single frame type, and thus performs those operations in software for other frame types.

Note: When operating with two descriptor queues, software needs to rewrite the context descriptor for each packet that requires it because software doesn't know if the second queue was modified. Hardware keeps track only for the last context descriptor that was written.

This same logic can also be applied to the segmentation context, though the environment is a more restrictive one. In this scenario, the host is commonly asked to send a message of the same type, TCP/IP for instance, and these messages also have the same total length and same Maximum Segment Size (MSS). In this instance, the same segmentation context could be used for multiple TCP messages that require hardware segmentation. The limitations of this scenario and the relatively small performance advantage make this approach unlikely; however, it is useful in understanding the underlying mechanism.

3.3.6 TCP/IP Context Descriptor Layout

The following section describes the layout of the TCP/IP context transmit descriptor.

To select this descriptor format, bit 29 (TDESC.DEXT) must be set to 1b and TDESC.DTYP must be set to 0000b. In this case, the descriptor format is defined as shown in Table 20.

Note: The TCP/IP context descriptor does not transfer any packet data. It merely prepares the checksum hardware for the TCP/IP Data descriptors that follow.

Table 20. Transmit Descriptor (TDESC) Layout – (Type = 0000b)

	63 48	47 40	39 32	31 16	15 8	7 0
0	TUCSE	TUCSO	TUCSS	IPCSE	IPCSO	IPCSS
8	MSS	HDRLEN	RSV STA	TUCMD DTYP	PAYLEN	
	63 48	47 40	39 36 35 32	31 24 23 20	19 0	

Note: The first quad word of this descriptor type contains parameters used to calculate the two checksums, which might be offloaded.

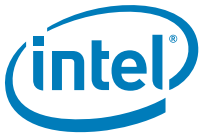


Table 21. Transmit Descriptor (TDESC) Layout

Transmit Descriptor Offload	Description
TUCSE	TCP/UDP Checksum Ending Defines the ending byte for the TCP/UDP checksum offload feature. Setting TUCSE field to 0b indicates that the checksum covers from TUCCS to the end of the packet.
TUCSO	TCP/UDP Checksum Offset Defines the offset where to insert the TCP/UDP checksum field in the packet data buffer. This is used in situations where the software needs to calculate partial checksums (TCP pseudo-header, for example) to include bytes which are not contained within the range of start and end. If no partial checksum is required, software must write a value of 0b.
TUCSS	TCP/UDP Checksum Start Defines the starting byte for the TCP/UDP checksum offload feature. It must be defined even if checksum insertion is not desired for some reason. When setting the TCP segmentation context, TUCSS is used to indicate the start of the TCP header.
IPCSE	IP Checksum Ending Defines the ending byte for the IP checksum offload feature. It specifies where the checksum should stop. A 16-bit value supports checksum offloading of packets as large as 64 KB. Setting IPCSE field to 0b indicates that the checksum covers from IPCCS to the end of the packet. In this way, the length of the packet does not need to be calculated. When executing checksum or TCP segmentation with IPv6 headers, the *IPCSE* field should be set to 0000h, IPCSS should be valid (as in IPv4 packets), and the *IXSM* bit in the data descriptor should be cleared. For proper IP checksum calculation, the *IP Header Checksum* field should be set to 0b unless some adjustment is needed by the software driver.
IPCSO	IP Checksum Offset The IPCSO field specifies where the resulting IP checksum should be placed. It is limited to the first 256 bytes of the packet and must be less than or equal to the total length of a given packet. If this is not the case, the checksum is not inserted.
IPCSS	IP Checksum Start IPCSS specifies the byte offset from the start of the transferred data to the first byte in be included in the checksum. Setting this value to 0b means the first byte of the data would be included in the checksum. Note that the maximum value for this field is 255. This is adequate for typical applications. The IPCSS value needs to be less than the total transferred length of the packet. If this is not the case, the results are unpredictable. IPCSS must be defined even if checksum insertion is not desired for some reason. When setting the TCP segmentation context, IPCSS is used to indicate the start of the IP header.
MSS	Maximum Segment Size Controls the Maximum Segment Size. This specifies the maximum TCP or UDP payload “segment” sent per frame, not including any header. The total length of each frame (or “section”) sent by the TCP Segmentation mechanism (excluding 802.3ac tagging and Ethernet CRC) is MSS bytes + HRDLEN. The one exception is the last packet of a TCP segmentation context which is (typically) shorter than “MSS+HDRLEN”. This field is ignored if TDESC.TSE is not set. **Note:** MSS must be set to a value larger than 10h.
HDRLEN	Header Length Specifies the length (in bytes) of the header to be used for each frame (or “section”) of a TCP Segmentation operation. The first HDRLEN bytes fetched from data descriptor(s) are stored internally and used as a prototype header for each section, and are pre-pended to each payload segment to form individual frames. For UDP packets this is normally equal to “UDP checksum offset + 2”. For TCP packets it is normally equal to “TCP checksum offset + 4 + TCP header option bytes”. This field is ignored if TDESC.TSE is not set.

Transmit Descriptor Offload	Description
RSV	Reserved Should be programmed to 0b for future compatibility.
STA	TCP/UDP Status field Provides transmit status indication. Section 3.3.6.2 provides the bit definition for the TDESC.STA field.
TUCMD	TCP/UDP command field The command field provides options that control the checksum offloading, along with some of the generic descriptor processing functions. Section 3.3.6.1 provides the bit definitions for the TDESC.TUCMD field.
DTYP	Descriptor Type Set to 0000b for TCP/IP context transmit descriptor type.
PAYLEN	The packet length field (TDESC.PAYLEN) is the total number of payload bytes for this TCP Segmentation offload context (i.e., the total number of payload bytes that could be distributed across multiply frames after TCP segmentation is performed). Following the fetch of the prototype header, PAYLEN specifies the length of data that is fetched next from data descriptor(s). This field is also used to determine when “last-frame” processing needs to be performed. Typically, a new data descriptor is used to denote the start of the payload data buffer(s), but this is not required. PAYLEN specification should not include any header bytes. There is no restriction on the overall PAYLEN specification with respect to the transmit FIFO size, once the MSS and HDRLEN specifications are legal. This field is ignored if TDESC.TSE is not set. Refer to Section 3.5 for details on the TCP Segmentation off-loading feature.

Notes:
1. A number of the fields are ignored if the TCP Segmentation enable bit (TDESC.TSE) is cleared, denoting that the descriptor does not refer to the TCP segmentation context.
2. Maximum limits for the HDRLEN and MSS fields are dictated by the lengths variables. However, there is a further restriction that for any TCP Segmentation operation, the hardware must be capable of storing a complete section (completely-built frame) in the transmit FIFO prior to transmission. Therefore, the sum of MSS + HDRLEN must be at least 80 bytes less than the allocated size of the transmit FIFO.

3.3.6.1 TCP/UDP Offload Transmit Descriptor Command Field

The command field (TDESC.TUCMD) provides options to control the TCP segmentation, along with some of the generic descriptor processing functions.

Table 22. Command Field (TDESC.TUCMD) Layout

TDESC.TUCMD	Description
IDE (bit 7)	Interrupt Delay Enable IDE activates the transmit interrupt delay timer. Hardware loads a countdown register when it writes back a transmit descriptor that has the RS bit and the IDE bit set. The value loaded comes from the IDV field of the Interrupt Delay (TIDV) register. When the count reaches 0, a transmit interrupt occurs. Hardware always loads the transmit interrupt counter whenever it processes a descriptor with IDE set even if it is already counting down due to a previous descriptor. If hardware encounters a descriptor that has RS set, but not IDE, it generates an interrupt immediately after writing back the descriptor. The interrupt delay timer is cleared.
SNAP (Bit 6)	
DEXT(Bit 5)	Descriptor Extension Must be 1b for this descriptor type.
RSV (Bit 4)	Reserved. Set to 0b for future compatibility.

TDESC.TUCMD	Description
RS (Bit 3)	Report Status RS tells the hardware to report the status information for this descriptor. Because this descriptor does not transmit data, only the DD bit in the status word is valid. Refer to Section 3.3.6.2 for the layout of the status field.
TSE (Bit 2)	TCP Segmentation Enable TSE indicates that this descriptor is setting the TCP segmentation context. If this bit is not set, the checksum offloading context for normal (non-"TCP Segmentation") packets is written. When a descriptor of this type is processed the MAC immediately updates the context in question (TCP Segmentation or checksum offloading) with values from the descriptor. This means that if any normal packets or TCP Segmentation packets are in progress (a descriptor with EOP set has not been received for the given context), the results are likely to be undesirable.
IP (Bit 1)	Packet Type (IPv4 = 1b, IPv6 = 0b) Identifies what type of IP packet is used in the segmentation process. This is necessary for hardware to know where the *IP Payload Length* field is located. For **ICH8**, this does not override the checksum insertion bit, IXSM.
TCP (bit 0)	Packet Type (TCP = 1b) Identifies the packet as either TCP or UDP (non-TCP). This affects the processing of the header information.

Notes:
1. The IDE, DEXT, and RS bits are valid regardless of the state of TES. All other bits are ignored if TSE = 0b.
2. The TCP Segmentation feature also provides access to a generic block send function and may be useful for performing "segmentation offload" in which the header information is constant.

3.3.6.2 TCP/UDP Offload Transmit Descriptor Status Field

Four bits are reserved to provide transmit status, although only one is currently assigned for this specific descriptor type. The status word is only written back to host memory in cases where the RS is set in the command.

Table 23. Transmit Status Layout

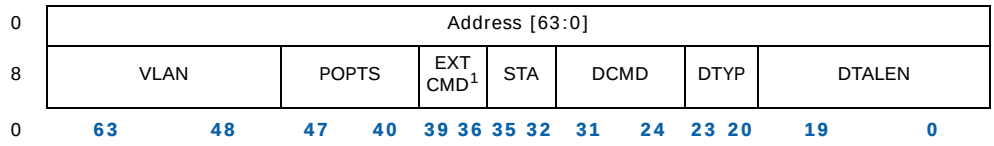
TDESC.STA	Description
RSV	Reserved Reserved for future use. Reads as 0b.
DD (bit 0)	Descriptor Done Indicates that the descriptor is finished and is written back after the descriptor has been processed.

3.3.7 TCP/IP Data Descriptor Format

The TCP/IP data descriptor is the companion to the TCP/IP context transmit descriptor described in the previous section. This descriptor type provides similar functionality to the legacy mode descriptor but also integrates the checksum offloading and TCP Segmentation feature.

To select this descriptor format, bit 29 in the command field (TDESC.DEXT) must be set to 1b and TDESC.DTYP must be set to 0001b. In this case, the descriptor format is defined as shown in Table 24.

Table 24. Transmit Descriptor (TDESC) Layout – (Type = 0001b)

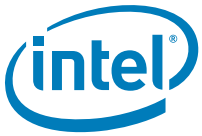


1. Reserved for **ICH8** and **ICH9**.

The first Qword of this descriptor type contains the address of a data buffer in host memory that contains a portion of a transmit packet.

The second Qword of this descriptor contains information abut the data pointed to by this descriptor as well as descriptor processing options.

Transmit Descriptor	Description
Address	Data buffer address Address of the data buffer in the host memory which contains a portion of the transmit packet.
DTALEN	Data Length Field Total length of the data pointed to by this descriptor, in bytes. For data descriptors not associated with a TCP Segmentation operation (TDESC.TSE not set), the descriptor lengths are subject to the same restrictions specified for legacy descriptors (the sum of the lengths of the data descriptors comprising a single packet must be at least 80 bytes less than the allocated size of the transmit FIFO.)
DTYP	Data Type Set to 0001b to identify this descriptor as a TCP/IP data descriptor.
DCMD	Descriptor Command Field Provides options that control some of the generic descriptor processing features. Refer to Section 3.3.7.1 for bit definitions of the DCMD field.
STA	TCP/IP Status field Provides transmit status indication. Section 3.3.7.2 provides the bit definition for the TDESC.STA field.
RSV	Reserved Set to 0b for future compatibility.
POPTS	Packet Option Field Provides a number of options which control the handling of this packet. This field is ignored except on the first data descriptor of a packet. Section 3.3.7.4 provides the bit definition for the TDESC.POPTS field.
VLAN	VLAN field The field is used to provide 802.1q tagging information. This field is only valid in the last descriptor of the given packet (qualified by the EOP bit).
EXTCMD	**ICH10** only.



3.3.7.1 TCP/IP Data Descriptor Command Field

The Command field provides options that control checksum offloading and TCP segmentation features along with some of the generic descriptor processing features.

Table 25. Command Field (TDESC.DCMD) Layout

TDESC.DCMD	Description
IDE (bit 7)	Interrupt Delay Enable When set, activates the transmit interrupt delay timer. Hardware loads a countdown register when it writes back a transmit descriptor that has RS and IDE set. The value loaded comes from the IDV field of the Interrupt Delay (TIDV) register. When the count reaches 0, a transmit interrupt occurs if enabled. Hardware always loads the transmit interrupt counter whenever it processes a descriptor with IDE set even if it is already counting down due to a previous descriptor. If hardware encounters a descriptor that has RS set, but not IDE, it generates an interrupt immediately after writing back the descriptor. The interrupt delay timer is cleared.
VLE (bit 6)	VLAN Enable Indicates that the packet is a VLAN packet (hardware should add the VLAN Ethertype and an 802.1q VLAN tag to the packet). The VLAN data comes from the special field of the TX descriptor. The hardware in that case appends the FCS/CRC. Note: If VLE is set to enable VLAN tag insertion, the CTRL.VME bit should also be set. 0b = Send generic Ethernet packet. IFCS controls insertion of FCS in normal Ethernet packets. 1b = Send 802.1Q packet. The *Ethernet Type* field comes from the VET register and the VLAN data comes from the special field of the TX descriptor (hardware always appends the FCS/CRC).
DEXT (Bit 5)	Descriptor Extension Identifies this descriptor as one of the extended descriptor types.and must be set to 1b.
ILSec (bit 4)	Includes LinkSec encapsulation and LinkSec processing 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. **Note:** This is a reserved bit for **ICH8** and **ICH9**. Should be programmed to 0b.
RS (bit 3)	Report Status When set, tells the hardware to report the status information for this descriptor as soon as the corresponding data buffer has been fetched and stored in the MAC controller's internal packet buffer. In some cases the MAC provides the status indication on the descriptors even if the RS bit is cleared.
TSE (bit 2)	TCP Segmentation Enable TSE indicates that this descriptor is part of the current TCP Segmentation context. If this bit is not set, the descriptor is part of the "normal" context.
IFCS (Bit 1)	Insert IFCS Controls the insertion of the FCS/CRC field in normal Ethernet packets. IFCS is only valid in the last descriptor of the given packet (qualified by the EOP bit).
EOP (Bit 0)	End Of Packet The EOP bit indicates that the buffer associated with this descriptor contains the last data for the packet or for the given TCP Segmentation context. In the case of a TCP Segmentation context, the DTALEN length of this descriptor should match the amount remaining of the original PAYLEN. If it does not, the TCP Segmentation context is terminated but the end of packet processing may be incorrectly performed. These abnormal termination events are counted in the TSCTFC statistics register.

Note: LinkSec offload does not support the case in which IFCS is not set while ILSec is set.

Note: The VLE, IFCS, and VLAN fields are only valid in certain descriptors. If TSE is enabled, the VLE, IFCS, and VLAN fields are only valid in the first data descriptor of the TCP segmentation context. If TSE is not enabled, then these fields are only valid in the last descriptor of the given packet (qualified by the EOP bit).

IDE activates the transmit interrupt delay timer. Hardware loads a countdown register when it writes back a transmit descriptor that has RS and IDE set. The value loaded comes from the IDV field of the Interrupt Delay (TIDV) register. When the count reaches zero, a transmit interrupt occurs. Hardware always loads the transmit interrupt counter each time it processes a descriptor with IDE set even if it is already counting down due to a previous descriptor. If hardware encounters a descriptor that has RS set, but not IDE, it generates an interrupt immediately after writing back the descriptor. The interrupt delay timer is cleared.

3.3.7.2 TCP/IP Data Descriptor Status Field

Four bits are reserved to provide transmit status, although only the DD is valid. The status word is only written back to host memory in cases where the RS bit is set in the command field. The DD bit indicates that the descriptor is finished and is written back after the descriptor has been processed.

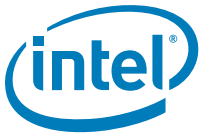
Table 26. Transmit Status Layout

TDESC.STA	Description
RSV (bit 3)	Reserved Should be programmed to 0b.
RSV (bit2)	Reserved Should be programmed to 0b.
RSV (bit 1)	Reserved Should be programmed to 0b.
DD (bit 0)	Descriptor Done Indicates that the descriptor is done and is written back either after the descriptor has been processed (with RS set).

3.3.7.3 ICH10 Extended Command Field

ExtCMD is an extension to the command byte and contains the following fields.

TDESC.ExtCMD	Description
RSV (bit 3)	Reserved Should be programmed to 0b.
RSV (bit 2)	Reserved Should be programmed to 0b.
RSV (bit 1)	Reserved Should be programmed to 0b.
TimeStamp (bit 0)	Time Stamp Indication to stamp the transmitted packet time for TimeSync.



3.3.7.4 TCP/IP Data Descriptor Option Field

The POPTS field provides a number of options which control the handling of this packet. This field is ignored except on the first data descriptor of a packet or segmentation context.

Table 27. Packet Options Field (TDESC.POPTS) Layout

TDESC.POPTS	Description
RSV (bits 7:2)	Reserved Should be written with 0b for future compatibility.
TXSM (bit1)	Insert TCP/UDP Checksum Controls the insertion of the TCP/UDP checksum. If not set, the value placed into the checksum field of the packet data is not modified, and is placed on the wire. When set, TCP/UDP checksum field is modified by the hardware. Valid only in the first data descriptor for a given packet or TCP segmentation context.
IXSM (bit 0)	Insert IP Checksum Controls the insertion of the IP checksum. If not set, the value placed into the checksum field of the packet data is not modified and is placed on the wire. When set, the IP checksum field is modified by the hardware. Valid only in the first data descriptor for a given packet or TCP segmentation context.

Note: For proper values of the IP and TCP checksum, software must set IXSM and TXSM when using transmit segmentation.

Note: Software should not set the IXSM field for IPv6 packets since IPv6 does not include IP checksum.

3.3.7.5 TCP/IP VLAN Field

The VLAN field is used to provide the 802.1q/802.3ac tagging information. Note that the special field is ignored if the VLE bit is set to 0b.

When CTRL.VME is set to 1b, all packets transmitted from the MAC that has VLE set in the DCMD field is sent with an 802.1Q header added to the packet. The contents of the header come from the transmit descriptor special field and from the VLAN type register. The special field is ignored if the VLE bit in the transmit descriptor command field is 0b. The VLAN field is valid only when EOP is set.

Table 28. Special Field (TDESC.SPECIAL) Layout

TDESC.SPECIAL	Description
PRI	User Priority Three bits that provide the VLAN user priority field to be inserted in the 802.1Q tag.
CFI	Canonical Form Indicator
VLAN ID	VLAN Identifier 12 bits that provide the VLAN identifier field to be inserted in the 802.1Q tag.

3.4 Transmit Descriptor Ring Structure

The transmit descriptor ring structure is shown in Figure 7. A pair of hardware registers maintains the transmit queue. New descriptors are added to the ring by writing descriptors into the circular buffer memory region and moving the ring's tail pointer. The tail pointer points one entry beyond the last hardware owned descriptor (but at a point still within the descriptor ring). Transmission continues up to the descriptor where head equals tail at which point the queue is empty.

Descriptors passed to hardware should not be manipulated by software until the head pointer has advanced past them.

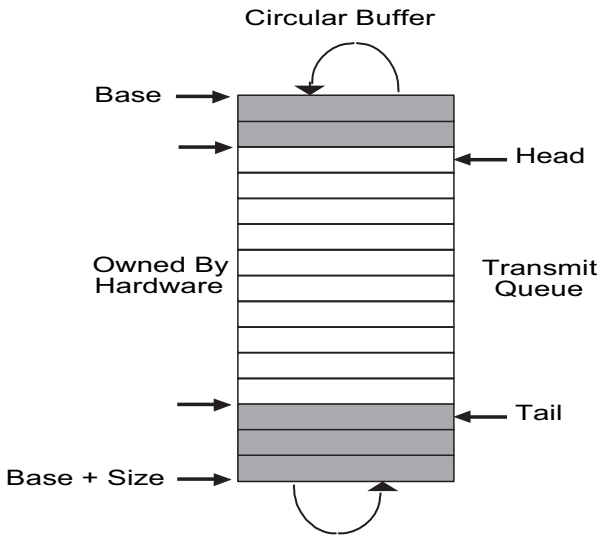


Figure 7. Transmit Descriptor Ring Structure

Shaded boxes in Figure 7 represent descriptors that have been transmitted but not yet reclaimed by software. Reclaiming involves freeing up buffers associated with the descriptors.

The transmit descriptor ring is described by the following registers:

- Transmit Descriptor Base Address registers (TDBAL and TDBAH)
  These registers indicate 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. TDBAL contains the lower 32-bits; TDBAH contains the upper 32 bits. Hardware ignores the lower 4 bits in TDBAL.
- Transmit Descriptor Length register (TDLEN)
  This register determines the number of bytes allocated to the circular buffer. This value must be 128 byte aligned.
- Transmit Descriptor Head register (TDH)
  This register holds a value which is an offset from the base, and indicates the in–progress descriptor. There can be up to 64 KB descriptors in the circular buffer. Reading this register returns the value of head corresponding to descriptors already loaded in the output FIFO.
- Transmit Descriptor Tail register (TDT)
  This register holds a value which is an offset from the base, and indicates the location beyond the last descriptor hardware can process. This is the location where software writes the first new descriptor.

The base register indicates the start of the circular descriptor queue and the length register indicates the maximum size of the descriptor ring. The lower seven bits of length are hardwired to 0b. Byte addresses within the descriptor buffer are computed as follows:

*address = base + (ptr * 16)*, where *ptr* is the value in the hardware head or tail register.

The size chosen for the head and tail registers permit a maximum of 64 KB descriptors, or approximately 16 KB packets for the transmit queue given an average of four descriptors per packet.

Once activated, hardware fetches the descriptor indicated by the hardware head register. The hardware tail register points one beyond the last valid descriptor. Software reads the head register to determine which packets (those logically before the head) have been transferred to the on-chip FIFO or transmitted.

Note: Software can determine if a packet has been sent by setting the RS bit in the transmit descriptor command field. Checking the transmit descriptor DD bit in memory eliminates a potential race condition. All descriptor data is written to the IO bus prior to incrementing the head register, but a read of the head register could pass the data write in systems performing IO write buffering. Updates to transmit descriptors use the same IO write path and follow all data writes. Consequently, they are not subject to the race condition. Other potential conditions also prohibit software reading the head pointer.

In general, hardware prefetches packet data prior to transmission. Hardware typically updates the value of the head pointer after storing data in the transmit FIFO.

The process of checking for completed packets consists of one of the following:

- Scan memory for descriptor status write-backs.
- Read the hardware head register. All packets up to but EXCLUDING the one pointed to by head have been sent or buffered and can be reclaimed.
- Take an interrupt. An interrupt condition can be generated each time a transmit queue goes empty (ICR.TXQE) or the descriptor ring has reached the threshold specified in the Transmit Descriptor Control register (ICR. TXD_LOW). These interrupts can either be enabled or masked.

3.4.1 Transmit Descriptor Fetching

The descriptor processing strategy for transmit descriptors is essentially the same as for receive descriptors except that a different set of thresholds are used. As for receives, the number of on-chip transmit descriptors buffer space is 64 descriptors.

When the on-chip buffer is empty, a fetch happens as soon as any descriptors are made available (software writes to the tail pointer). When the on-chip buffer is nearly empty (TXDCTL.PTHRESH), a prefetch is performed whenever enough valid descriptors (TXDCTL.HTHRESH) are available in host memory and no other DMA activity of greater priority is pending (descriptor fetches and write-backs or packet data transfers).

The descriptor prefetch policy is aggressive to maximize performance. If descriptors reside in an external cache, the system must ensure cache coherency before changing the tail pointer.

When the number of descriptors in host memory is greater than the available on-chip descriptor storage, the chip may elect to perform a fetch which is not a multiple of cache line size. The hardware performs this non-aligned fetch if doing so results in the next descriptor fetch being aligned on a cache line boundary. This allows the descriptor fetch mechanism to be most efficient in the cases where it has fallen behind software.

Note: The MAC never fetches descriptors beyond the descriptor tail pointer.

3.4.2 Transmit Descriptor Write-Back

The descriptor write-back policy for transmit descriptors is similar to that for receive descriptors with a few additional factors. First, since transmit descriptor write-backs are optional (controlled by RS in the transmit descriptor), only descriptors which have one (or both) of these bits set starts the accumulation of write-back descriptors. Secondly, to preserve backward compatibility with previous MACs, if the TXDCTL.WTHRESH value is 0b, the MAC writes back a single byte of the descriptor (TDESCR.STA) and all other bytes of the descriptor are left unchanged.

Since the benefit of delaying and then bursting transmit descriptor write-backs is small at best, it is likely that the threshold are left at the default value (0b) to force immediate write-back of transmit descriptors and to preserve backward compatibility.

Descriptors are written back in one of three conditions:

- TXDCTL.WTHRESH = 0b and a descriptor which has RS set is ready to be written back
- Transmit Interrupt Delay timer expires
- TXDCTL.WTHRESH > 0b and TXDCTL.WTHRESH descriptors have accumulated

For the first condition, write-backs are immediate. This is the default operation and is backward compatible. For this case, the Transmit Interrupt delay function works as described in Section 3.4.4.1.

The other two conditions are only valid if descriptor bursting is enabled (see Section 10.0). In the second condition, the Transmit Interrupt Delay timer (TIDV) is used to force timely write–back of descriptors. The first packet after timer initialization starts the timer. Timer expiration flushes any accumulated descriptors and sets an interrupt event (TXDW).

For the final condition, if TXDCTL.WTHRESH descriptors are ready for write-back, the write-back is performed.

3.4.3 Pipelined Tx Data Read Requests

The MAC supports four pipelined requests from the Tx data DMA. Other DMA engines are not enhanced in a similar manner. In general, the four requests can belong to the same packet or to consecutive packets. However, the following restrictions apply:

- All requests for a packet are issued before a request is issued for a consecutive packet
- If a request (for the next packet) requires context change, the request for the next packet is not issued until the previous request completed (no pipeline across context).

If multiple Tx queues are used, read requests can be issued from any of the supported queues, as long as the above restrictions are met. Pipelined requests can belong to the same queue or to separate queues. However, as noted above, all requests for a certain packet must be issued (from same queue) before a request is issued for a different packet (potentially from a different queue).

The PCI 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 MAC can handle completions that arrive in any order. Once all completions arrive for a given request, the MAC issues the next pending read data request.

The *MULR* bit in the Transmit Control register (TCTL) enables multiple read requests for transmit data. Its initial value is loaded from the NVM *Multiple Read Request Enable* bit.

MACs incorporate a 2 KB reorder buffer to support re-ordering of completions for four requests. Each request/completion can be up to 512 bytes long. The maximum size of a read request is defined as follows:

- When the *MULR* bit is set, maximum request size in bytes is the minimum {512, Max_Read_Request_Size}

In addition to the four pipeline requests from the Tx data DMA, The MAC can issue a single read request from each of the Tx descriptor and Rx descriptor DMA engines. The requests from the three DMA engines (Tx data, Tx descriptor and Rx descriptor) are independently issued. Each descriptor read request can fetch up to 16 descriptors (equal to 256 bytes of data).

3.4.4 Transmit Interrupts

Hardware supplies three transmit interrupts. These interrupts are initiated through the following conditions:

- Transmit queue empty (TXQE) — All descriptors have been processed. The head pointer is equal to the tail pointer.
- Descriptor done [Transmit Descriptor Write-back (TXDW)] — Set when hardware writes back a descriptor with RS set. This is only expected to be used in cases where, for example, the streams interface has run out of descriptors and wants to be interrupted whenever progress is made.

- Transmit Delayed Interrupt (TXDW) — In conjunction with IDE (Interrupt Delay Enable), the TXDW indication is delayed by a specific time per the TIDV register. This interrupt is set when the transmit interrupt countdown register expires. The countdown register is loaded with the value of the IDV field of the TIDV register, when a transmit descriptor with its RS bit and the IDE bit are set, is written back. When a Transmit Delayed Interrupt occurs, the TXDW interrupt cause bit is set (just as when a Transmit Descriptor Write-back interrupt occurs). This interrupt may be masked in the same manner as the TXDW interrupt. This interrupt is used frequently by software that performs dynamic transmit chaining, by adding packets one at a time to the transmit chain.

Note: The transmit delay interrupt is indicated with the same interrupt bit as the transmit write-back interrupt, TXDW. The transmit delay interrupt is only delayed in time as previously discussed.

- Link status change (LSC) - Set when the link status changes. When using the PHY, link status changes are determined and indicated by the PHY via a change in its LINK indication.
- Transmit Descriptor Ring Low Threshold Hit (TXD_LOW) - Set when the total number of transmit descriptors available hits the low threshold specified in the TXDCTL.LWTHRESH field in the Transmit Descriptor Control register. For the purposes of this interrupt, the number of transmit descriptors available is the difference between the Transmit Descriptor Tail and Transmit Descriptor Head values, minus the number of transmit descriptors that have been prefetched. Up to eight descriptors can be prefetched.

3.4.4.1 Delayed Transmit Interrupts

This mechanism enables software the flexibility of delaying transmit interrupts until no more descriptors are added to a transmit chain for a certain amount of time, rather than when the MAC's head pointer catches the tail pointer. This occurs if the MAC is processing packets slightly faster than the software.

A software driver usually has no knowledge of when it is going to be asked to send another frame. For performance reasons, it is best to generate only one transmit interrupt after a burst of packets have been sent.

Refer to Section 3.3.3.1 for specific details.

3.5 TCP Segmentation

The MAC implements a TCP segmentation capability for transmits that allows the software device driver to offload packet segmentation and encapsulation to the hardware. The software device driver might send the MAC the entire IP (IPv4 or IPv6), TCP or UDP message sent down by the Network Operating System (NOS) for transmission. The MAC segments the packet into legal Ethernet frames and transmits them on the wire. By handling the segmentation tasks, the hardware alleviates the software from handling some of the framing responsibilities. This reduces the overhead on the CPU for the transmission process thus reducing overall CPU utilization.

Hardware TCP segmentation is one of the off-loading options of most modern TCP/IP stacks. This is often referred to as Large Send offloading. This feature enables the TCP/IP stack to pass to the Ethernet controller software 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 software driver and hardware to carve 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.

The offloading of these processes from the software driver to the MAC saves significant CPU cycles. The software driver shares the additional tasks to support these options with the MAC.

Although the MAC's TCP segmentation offload implementation was specifically designed to take advantage of new TCP segmentation offload features, the hardware implementation was made generic enough so that it could also be used to segment traffic from other protocols. For instance this feature could be used any time it is desirable for hardware to segment a large block of data for transmission into multiple packets that contain the same generic header.

3.5.1 Assumption

The following assumption applies to the TCP segmentation implementation in the MAC:

- The RS bit operation is not changed. Interrupts are set after data in buffers pointed to by individual descriptors is transferred to hardware.

3.5.2 Transmission Process

The transmission process for regular (non-TCP segmentation packets) involves:

- 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.
- For each packet of the data block:
  - — Ethernet, IP and TCP/UDP headers are prepared by the stack.
  - — The stack interfaces with the software device driver and commands the driver to send the individual packet.
  - — The software device driver gets the frame and interfaces with the hardware.
  - — The hardware reads the packet from host memory (via DMA transfers).
- The software device driver returns ownership of the packet to the operating system when the hardware has completed the DMA transfer of the frame (indicated by an interrupt).

The transmission process for the MAC TCP segmentation offload implementation involves:

- The protocol stack receives from an application a block of data that is to be transmitted.
- The stack interfaces to the software device driver and passes the block down with the appropriate header information.
- The software device driver sets up the interface to the hardware (via descriptors) for the TCP segmentation context.
- The hardware transfers the packet data and performs the Ethernet packet segmentation and transmission based on offset and payload length parameters in the TCP/IP context descriptor including:
  - Packet encapsulation
  - Header generation & field updates including IP and TCP/UDP checksum generation
- The software device driver returns ownership of the block of data to the operating system when the hardware has completed the DMA transfer of the entire data block (indicated by an interrupt).

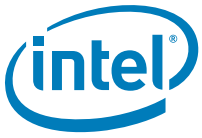
3.5.2.1 TCP Segmentation Data Fetch Control

To perform TCP segmentation in the MAC, the DMA unit must ensure that the entire payload of the segmented packet fits into the available space in the on-chip packet buffer. The segmentation process is performed without interruption. The DMA performs various comparisons between the payload and the packet buffer to ensure that no interruptions occur as well as general efficiencies in the operation of the TCP segmentation feature.

3.5.3 TCP Segmentation Performance

Performance improvements for a hardware implementation of TCP segmentation offload mean:

- The operating system stack does not need to partition the block to fit the MTU size, saving CPU cycles.
- The operating system stack only computes one Ethernet, IP, and TCP header per segment, saving CPU cycles.
- The operating system stack interfaces with the software device driver only once per block transfer, instead of once per frame.
- Interrupts are easily reduced to one per TCP message instead of one per packet.
- Fewer I/O accesses are required to command the hardware.



3.5.4 Packet Format

Typical TCP/IP transmit window size is 8760 bytes (about 6 full size frames). A TCP message can be as large as 64 KB and is generally fragmented across multiple pages in host memory. The MAC partitions the data packet into standard Ethernet frames prior to transmission. The MAC supports calculating the Ethernet, IP, TCP, and even UDP headers, including checksum, on a frame by frame basis.

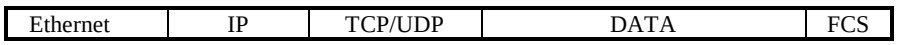


Figure 8. TCP/IP Packet Format

Frame formats supported by the Ethernet controller's TCP segmentation include:

- Ethernet 802.3
- IEEE 802.1q VLAN (Ethernet 802.3ac)
- Ethernet Type 2
- Ethernet SNAP
- IPv4 headers with options
- IPv6 headers with IP option next headers
- TCP with options
- UDP with options

Note: VLAN tag insertion is handled by hardware.

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 MAC has the ability to segment UDP traffic (in addition to TCP traffic). This process has limited usefulness.

Note: IP tunneled packets are not supported for large send operation.

3.5.5 TCP Segmentation Context Descriptor

Software indicates a TCP segmentation transmission context to the hardware by setting up a TCP/ IP Context Transmit Descriptor. The purpose of this descriptor is to provide information to the hardware to be used during the TCP segmentation offload process. The layout of this descriptor is described in Section 3.3.6.

Setting the TSE bit in the Command field to 1b indicates that this descriptor refers to the TCP 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 MAC.

The TCP segmentation prototype header is taken from the packet data itself. Software must identity the type of packet that is being sent (IP/TCP, IP/UDP, other), calculate appropriate checksum offloading values for the desired checksums, and calculate the length of the header which is prepended. The header may be up to 240 bytes in length.

TCP, IP, RS, DEXT, and IDE fields are defined in the TCP/IP context descriptor layout. Note that the RS bit should not be set in the context descriptor.

Setting the SNAP bit enables segmentation offload for SNAP packet type. When the SNAP bit is set, hardware updates the TCP/IP fields as well as the MAC header length field.

Note: When SNAP packets with a VLAN header are segmented, software must use the VLE option in the CMD field of the Data Descriptor. VLAN header insertion by software rather than use of the VLE option is not supported by the TCP segmentation offload mechanism for the SNAP packet type.

3.5.6 TCP Segmentation Data Descriptors

The TCP segmentation data descriptor is the companion to the TCP segmentation context descriptor described in the previous section. The descriptor is almost identical to the TCP/IP date descriptor. For a complete description of the descriptor please refer to Section 3.3.7.

To select this descriptor format, bit 29 (TDESC.DEXT) must be set to 1b and TDESC.DTYP must be set to 0001b.

3.5.7 TCP Segmentation Source Data

Once the TCP 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 segmentation context. The following sections describe the supported packet types and the various updates which 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:

- MAC Header (for SNAP)
  - — MAC Header LEN field should be set to 0b
- IPv4 Header
  - — Length should be set to zero
  - — 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 software device driver
- IPv6 Header
  - — Length should be set to zero

- 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 checksums as follows:

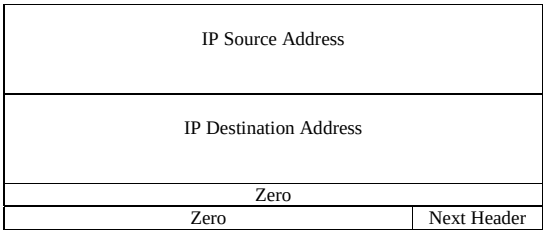


Figure 9. TCP/IPv6 Partial Pseudo-Header Checksum

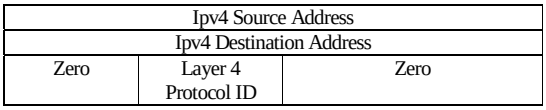


Figure 10. TCP/IPv4 Partial Pseudo-Header Checksum

- UDP Header
  - — Checksum should be set as in TCP header shown in Figure 9 and Figure 10. The MAC's DMA function fetches the ethernet, IP, and TCP/UDP prototype header information from the initial descriptor(s) and save them on-chip for individual packet header generation. The following sections describe the updating process performed by the hardware for each frame sent using the TCP segmentation capability.

3.5.8 TCP Segmentation Use of Multiple Data Descriptors

TCP segmentation enables a packet to be segmented to describe more than one data descriptor. A large packet contained in a single virtual-address buffer is better described as a series of data descriptors, each referencing a single physical address page.

The only requirement for this use is if multiple data descriptors for TCP segmentation follows this guideline:

- If multiple data descriptors are used to describe the IP/TCP/UDP header section, each descriptor must describe one or more complete headers; descriptors referencing only parts of headers are not supported.

Note: It is recommended that the entire header section, as described by the TCP Context Descriptor HDRLEN field, be coalesced into a single buffer and described using a single data descriptor. If all layer headers (L2-L4) are not coalesced into a single buffer, each buffer should not cross a 4 KB boundary, or be larger than Max_Read_Request_Size.

3.5.9 IP and TCP/UDP Headers

This section outlines the format and content for the IP, TCP and UDP headers. The MAC requires baseline information from the software device driver in order to construct the appropriate header information during the segmentation process.

Header fields that are modified by the MAC are highlighted in the figures that follow.

Note: The IPv4 header is first shown in the traditional (RFC 791) representation, and because byte and bit ordering is confusing in that representation, the IP header is also shown in little-endian format. The actual data is fetched from memory in little-endian format.

1								2								3			
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Version				IP Hdr Length				TYPE of service								Total length					
Identification																Flags			Fragment Offset		
Time to Live								Layer 4 Protocol ID								Header Checksum					
Source Address																					
Destination Address																					
Options																					

Figure 11. IPv4 Header (Traditional Representation)

Byte3								Byte2								Byte1								Byte0							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
LSB Total length MSB																TYPE of service								Version				IP Hdr Length			
Fragment Offset Low								RES				NF				MF				Fragment Offset High											
LSB Identification MSB																															
Header Checksum																				Layer 4 Protocol ID								Time to Live			
Source Address																															
Destination Address																															
Options																															

Figure 12. IPv4 Header (Little-Endian Order)

Flags Field Definition:

The Flags field is defined below. Note that hardware does not evaluate or change these bits.

- MF - More Fragments
- NF - No Fragments
- Reserved

Note: The IPv6 header is first shown in the traditional (RFC 2460), big-endian representation. The actual data is fetched from memory in little-endian format.

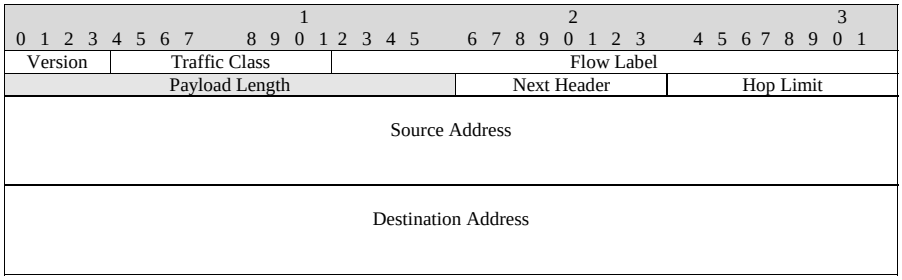


Figure 13. IPv6 TCP Header (Traditional Representation)

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 3.5.

Note: TCP requires the use of checksum; its optional for UDP.

The TCP header is first shown in the traditional (RFC 793) representation. Because byte and bit ordering is confusing in that representation, the TCP header is also shown in little-endian format. The actual data is fetched from memory in little-endian format.

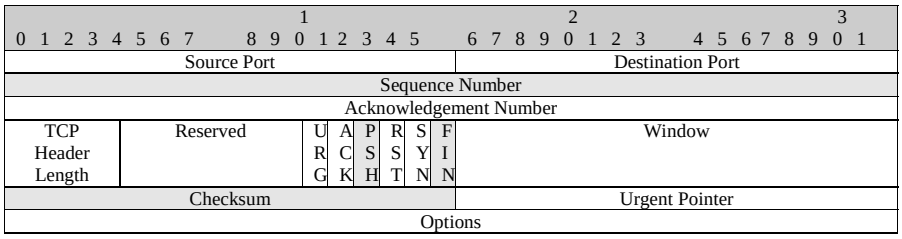
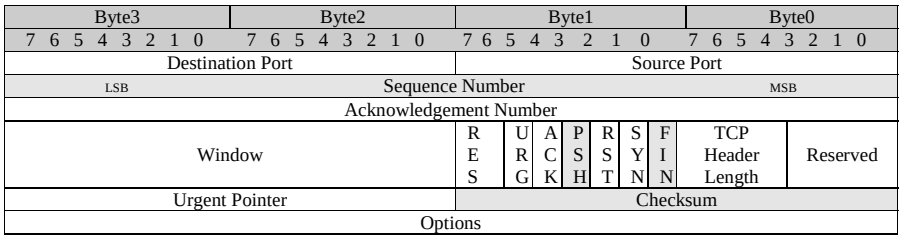


Figure 14. TCP Header (Traditional Representation)



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 conceptually prefixed to the TCP Header (see Figure 15 and Figure 16). The IPv4 pseudo header contains the IPv4 Source Address, the IPv4 Destination Address, the IPv4 Protocol field, and TCP Length.

The IPv6 pseudo header contains the IPv6 Source Address, the IPv6 Destination Address, the IPv6 Payload Length, and the IPv6 Next Header field. Software pre-calculates the partial pseudo header sum, which includes IPv4 SA, DA and protocol types, but not the TCP length, and stores this value into the TCP checksum field of the packet.

Note: The Protocol ID field should always be added 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 = Payload + HDRLEN - TUCSS

“Payload” is normally MSS except for the last packet where it represents the remainder of the payload.

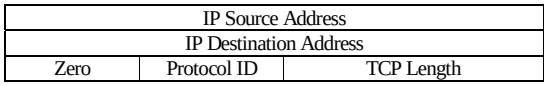


Figure 15. TCP Pseudo Header Content (Traditional Representation)

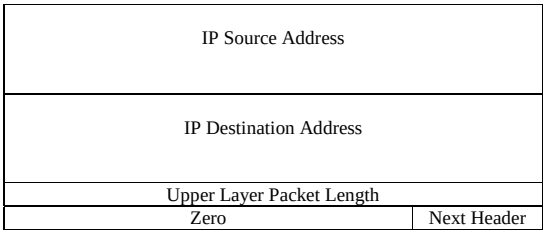


Figure 16. TCP Pseudo Header Content for IPv6

Note: The IP Destination address is the final destination of the packet. Therefore, if a routing header is used, the last address in the route list is used in this calculation. The upper-layer packet length is the length of the TCP header and the TCP payload.

The UDP header is always 8 bytes in size with no options.

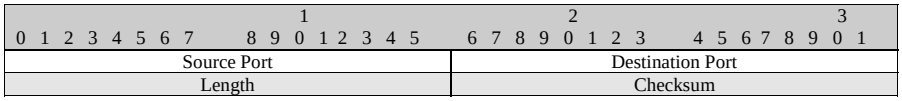


Figure 17. UDP Header (Traditional Representation)

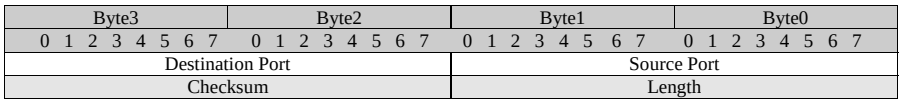


Figure 18. UDP Header (Little-Endian Order)

UDP pseudo header has the same format as the TCP pseudo header. The IPv4 pseudo header conceptually prefixed to the UDP header contains the IPv4 source address, the IPv4 destination address, the IPv4 protocol field, and the UDP length (same as the TCP Length previously discussed). The IPv6 pseudo header for UDP is the same as the IPv6 pseudo header for TCP. This checksum procedure is the same as is used in TCP.

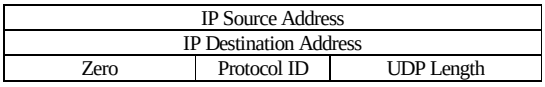


Figure 19. UDP Pseudo Header Diagram for Ipv4

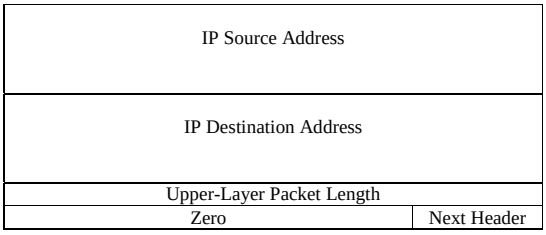


Figure 20. TCP Pseudo Header Diagram for IPv6

Note: The IP Destination Address is the final destination of the packet. Therefore, if a routing header is used, the last address in the route list is used in this calculation. The upper-layer packet length is the length of the UDP header and UDP payload.

Unlike the TCP checksum, the UDP checksum is optional. Software must set the TXSM bit in the TCP/IP Context Transmit Descriptor to indicate that a UDP checksum should be inserted. Hardware does not overwrite the UDP checksum unless the TXSM bit is set.

3.5.10 Transmit Checksum Offloading

The MAC provides a mechanism that off loads checksum calculation from the host processor of the IP and TCP or UDP.

To take advantage of the MAC's enhanced checksum offload capability, a checksum context must be initialized. For the normal transmit checksum offload feature this is performed by providing the device with a TCP/IP Context Descriptor with TSE=0b. Setting TSE=0b indicates that the normal checksum context is being set, as opposed to the segmentation context.

Enabling the checksum off loading capability without first initializing the appropriate checksum context leads to unpredictable results.

Once the checksum context has been set, that context is used for all normal packet transmissions until a new context is loaded. Also, since checksum insertion is controlled on a per packet basis, there is no need to clear/reset the context.

The MAC is capable of performing two transmit checksum calculations. Typically these would be used for TCP/IP and UDP/IP packet types; however, the mechanism is general enough to support other checksums as well. Each checksum operates independently and provides identical functionality.

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 offload feature only for a particular traffic type, thereby avoiding all context descriptors except for the initial one.

3.5.11 IP/TCP/UDP Header Updating

IP/TCP/UDP header is updated for each outgoing frame based on the IP/TCP header prototype which hardware transfers from the first descriptor(s) and stores on chip. The IP/TCP/UDP headers are fetched from host memory into an on-chip 240 byte header buffer once for each TCP segmentation context (for performance reasons, this header is not fetched again for each additional packet that is derived from the TCP segmentation process). 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 which fields are modified by hardware during the TCP segmentation process by the MAC. Figure 21 shows the overall data flow.

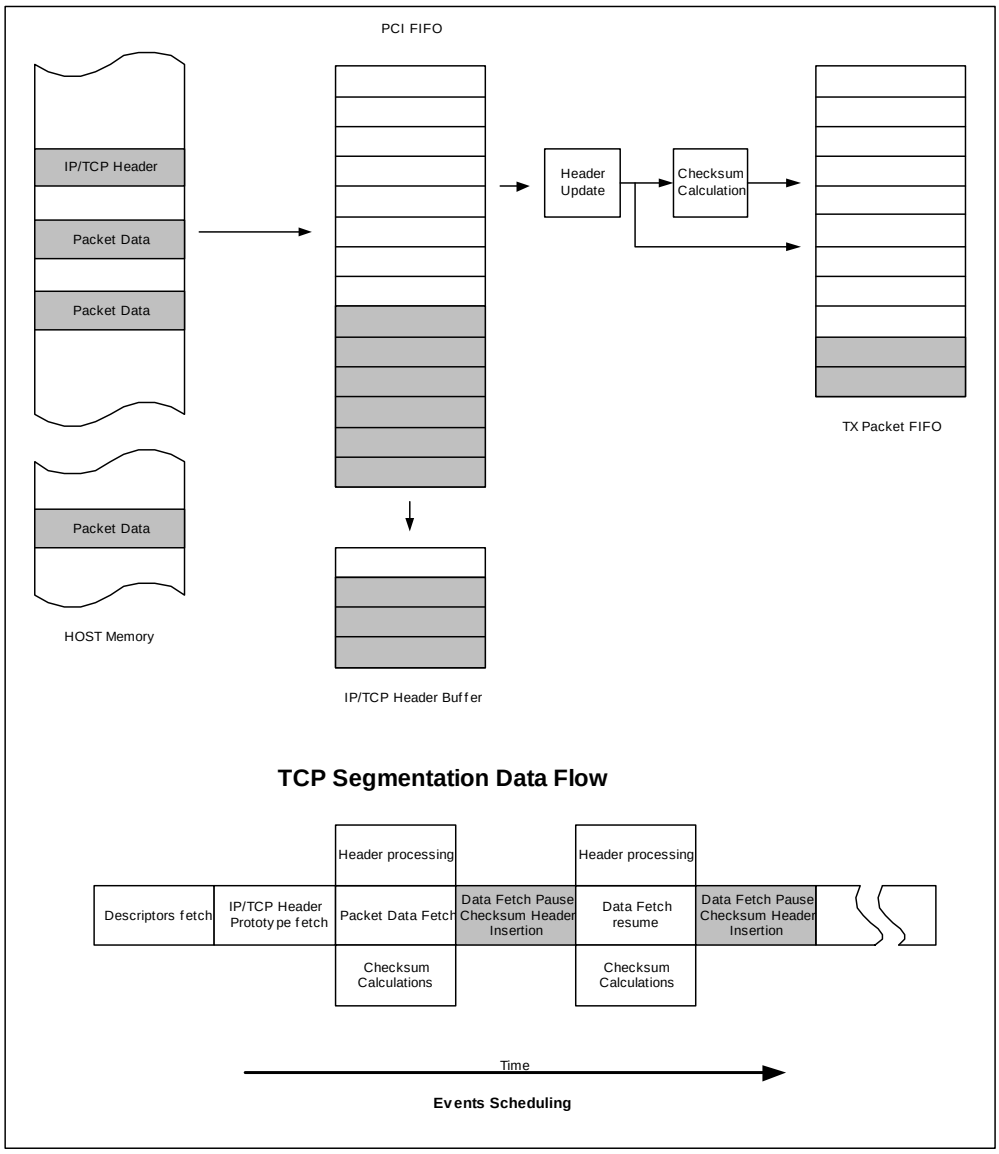


Figure 21. Overall Data Flow

3.5.11.1 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 + HDRLEN - 14

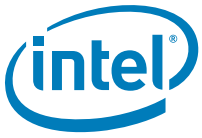
- IPv4 Header
  - IP Total Length = MSS + HDRLEN - IPCSS
  - IP Checksum
- IPv6 Header
  - Payload Length = MSS + HDRLEN - IPCSS - IPv6Size (while IPv6Size = 40 Bytes
- TCP Header
  - Sequence Number: The value is the Sequence Number of the first TCP byte in this frame.
  - If FIN flag = 1b, it is cleared in the first frame.
  - If PSH flag =1b, it is cleared in the first frame.
  - TCP Checksum
- UDP Header
  - UDP length: MSS + HDRLEN - TUCSS
  - UDP Checksum

3.5.11.2 TCP/IP/UDP 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:

Note: 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 + HDRLEN – 14
- IPv4 Header
  - IP Identification: incremented from last value (wrap around)
  - IP Total Length = MSS + HDRLEN – IPCSS
  - IP Checksum
- IPv6 Header
  - Payload Length = MSS + HDRLEN - IPCSS - IPv6Size (while IPv6Size = 40 bytes)
- TCP Header
  - Sequence Number update: Add previous TCP payload size to the previous sequence number value. This is equivalent to adding the MSS to the previous sequence number.
  - If FIN flag = 1b, it is cleared in these frames.
  - If PSH flag =1b, it is cleared in these frames.
  - TCP Checksum
- UDP Header
  - UDP Length: MSS + HDRLEN – TUCSS
  - UDP Checksum



3.5.11.3 TCP/IP/UDP Header For The Last Frame

Hardware makes the following changes to the headers for the last frame of a TCP segmentation context:

Note: Last frame payload bytes = PAYLEN – (N * MSS)

- MAC Header (for SNAP packets)
  - Type/LEN field = last frame payload bytes + HDRLEN – 14
- IPv4 Header
  - IP Total Length = (last frame payload bytes + HDRLEN) – IPCSS
  - IP Identification: incremented from last value (wrap around)
  - IP Checksum
- IPv6 Header
  - Payload Length = last frame payload bytes + HDRLEN - IPCSS - IPv6Size (while IPv6Size = 40 bytes
- TCP Header
  - Sequence Number update: Add previous TCP payload size to the previous sequence number value. This is equivalent to adding the MSS to the previous sequence number.
  - If FIN flag = 1b, set it in this last frame
  - If PSH flag =1b, set it in this last frame
  - TCP Checksum
- UDP Header
  - UDP length: (last frame payload bytes + HDRLEN) - TUCSS
  - UDP Checksum

3.5.12 Limitations and Software Considerations

Packet headers should be arranged in a way such that either the whole LSO header is located in a single buffer or each one of the networking layers (L2, L3 and L4) is located in a separate buffer that doesn't cross a 4 KB boundary and is smaller than Max_Read_Request_Size.

3.6 IP/TCP/UDP Transmit Checksum Offloading

The previous section on TCP segmentation offload describes the IP/TCP/UDP checksum offloading mechanism used in conjunction with TCP segmentation. The same underlying mechanism can also be applied as a standalone feature. The main difference in normal packet mode (non-TCP segmentation) is that only the checksum fields in the IP/TCP/UDP headers need to be updated.

Before taking advantage of the MAC's enhanced checksum offload capability, a checksum context must be initialized. For the normal transmit checksum offload feature, this task is performed by providing the MAC with a TCP/IP Context Descriptor with TSE = 0b. Setting TSE = 0b indicates that the normal checksum context is being set, as opposed to the segmentation context. For additional details on contexts, refer to Section 3.3.5.

Note: Enabling the checksum offloading capability without first initializing the appropriate checksum context leads to unpredictable results.

Once the checksum context has been set, that context, is used for all normal packet transmissions until a new context is loaded. Also, since checksum insertion is controlled on a per packet basis, there is no need to clear/reset the context.

The MAC is capable of performing two transmit checksum calculations. Typically, these would be used for TCP/IP and UDP/ID packet types, however, the mechanism is general enough to support other checksums as well. Each checksum operates independently and provides identical functionality. Only the IP checksum case is discussed as follows.

Three fields in the TCP/IP Context Descriptor set the context of the IP checksum offloading feature:

- IPCSS

  This field specifies the byte offset form the start of the transferred data to the first byte to be included in the checksum. Setting this value to 0b means that the first byte of the data is included in the checksum. The maximum value for this field is 255. This is adequate for typical applications.

Note: The IPCSS value needs to be less than the total DMA length to a packet. If this is not the case, the result will be unpredictable.

- IPCSO

  This field specifies where the resulting checksum should be placed. Again, this is limited to the first 256 bytes of the packet and must be less than or equal to the total length of a given packet. If this is not the case, the checksum is not inserted.

- IPCSE

  This field specifies where the checksum should stop. A 16-bit value supports checksum offloading of packets as large as 64 KB. Setting the IPCSE field to all zeros means End-of-Packet. In this way, the length of the packet does not need to be calculated.

As previously mentioned, 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 of the offload feature only for a particular traffic type, thereby avoiding all context descriptors except for the initial one.

3.7 LinkSec (ICH10 Only)

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 MAC. The LinkSec implementation supports the following:

- AES-GCM 128-bit offload engine in the Tx and Rx data path that support GbE wire speed.
- Both host and ME traffic can be processed by the AES-GCM engines.
- Supports single, secured Connectivity Association (CA):
  - Single Secure Connection (SC) on transmit data path.
  - Up to four SCs on receive data path.
  - Each SC supports two Security Associations (SAs) for seamless re-keying.
- Both the ME and host can act as the key agreement entity (KaY – in 802.1ae specification terminology). For example, control and access the offloading the engine (SecY in 802.1ae specification terminology)
  - Arbitration semaphores that indicate whether the ME or the host acts as the KaY.

— Tamper resistance - When the ME 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.

- Provides statistic counters as listed in Section 3.7.4.8.
- Supports replay protection with a replay window equal to zero.
- Receive memory structure:
  — New LinkSec offload receive status indication in the receive descriptors. LinkSec offload should not be used with the legacy receive format; use the Extended Receive Descriptor format instead (Section 3.2.4).
  — LinkSec Header/tag can be posted to the KaY for debug.
- Supports VLAN header location according to IEEE 802.1ae (first header inner to the LinkSec tag)

Section 22 shows a top level diagram of the LinkSec logic while the following sections provide further description of the LinkSec offload functionality.

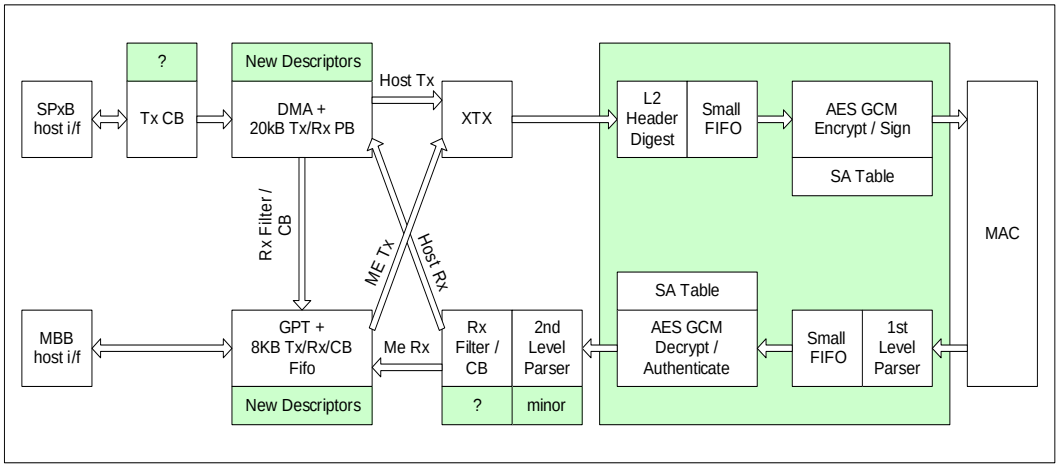


Figure 22. LinkSec Block Diagram

3.7.1 Packet Format

LinkSec defines frame encapsulation format as shown:

MAC DA SA	VLAN (optional)	Legacy Type / Len	LLC Data (Might Include IP/TCP and Higher Level Payload)	CRC
	← - User Data - →			

Figure 23. Legacy Frame Format

MAC DA SA	LinkSec Header (SecTag)	User Data (Optional Encrypted)	LinkSec ICV (tag)	CRC

Figure 24. LinkSec Encapsulation

3.7.1.1 LinkSec Header (SecTag) Format

LinkSec Ethertype	TCI and AN	SL	PN	SCI (Optional)
2 bytes	1 byte	1 byte	4 bytes	8 bytes

Figure 25. Sectag Format

3.7.1.2 LinkSec Ethertype

The MACsec Ethertype contains octet 1 and octet 2 of the SecTAG and 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

Tag Type	Name	Value
802.1AE Security TAG	LinkSec EtherType	88-E5

Figure 26. LinkSec Ethertype

3.7.1.3 TCI (Tag Control Information) and AN (Association number)

Table 29. TCI and AN Description

Bit(s)	Description
7	Version number (V) The MAC supports only version 0. Packets with other version values are discarded by the MAC.
6	End Station (ES) When set, means that the sender is an end station, thus the SCI is redundant and causes the SC bit to be cleared. Should always be set to 0b.
5	Secure Channel (SC) When the SCI field is active, this bit equals 1b if the ES bit is set. Also, the SC must be cleared.
4	Single Copy Equals 0b unless the SC supports EPON. Should always be set to 0b.

Bit(s)	Description
3	Encryption (E)[1] Set to 1b when the user data is encrypted.
2	Changed Text (C) Set to 1b if the data portion is modified by the integrity algorithm. For example, if a non default integrity algorithm is used or if a packet is encrypted.[1]
1:0	Association Number (AN) A 2-bit value defined by the control channel to uniquely identify SA (Keys, etc.).

1. The combination of the E bit (equal to 1b) and the C bit (equal to 0b) is reserved for KaY packets. The LinkSec logic ignores these packets on the receive path and transfers them to the KaY as is (no LinkSec processing and no LinkSec header strip). Intel's implementation never issues a packet in which the E bit is cleared and the C bit is set although it can tolerate such packets on receive.

Table 30. Short Length

Bit(s)	Description
7:6	Reserved, should be set to 0b.
5:0	Short Length (SL) Number of octets in the secure data field from end of SecTag to beginning of ICV if it is less then 48 octets, else SL value is zero.

3.7.1.3.1 Packet Number (PN)

The LinkSec engine increments this field for each packet on the transmit side. The PN is used to generate the Initial Value (IV) for the crypto engines. When the KaY is establishing a new SA it should set the initial value of PN to one (see Section 3.7.4.1).

3.7.1.4 Secure Channel Identifier (SCI)

The SCI is composed of the MAC address and port number as shown in the following. If the SC bit in TCI is not set, the SCI is not encoded in the SecTag.

Table 31. SCI Field Description

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Source MAC Address						Port Number	

3.7.1.5 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 specificatiion.

3.7.2 LinkSec Management – KaY (Key Agreement Entity)

The Kay management is done by the host or the ME. The ownership of the LinkSec management is as follows:

1. Initialization at power up or after Wake on LAN (WoL).

- In most cases the ME wakes before the host, thus:
  - — If the ME is capable to be a KaY it establishes a SC (authentication and key exchange).

  - If the ME is not capable to be a KaY, the only way for it to communicate is through VLAN. This means that the switch must support settings that enable specific VLANs to bypass LinkSec.

- When the host is awake
  - If the ME acted as KaY, the host should authenticate itself and transfer its ability to authenticate to the ME in order for the ME to transfer ownership over the LinkSec hardware. At this stage the system works in proxy mode where the host manages the secured channel while the ME piggybacks on it.
  - If the ME wasn't KaY, the host takes ownership over the LinkSec hardware and establishes an SC (authentication and key exchange); the ME can switch from a VLAN solution to piggyback over the SC established by the host. If the ME remains on VLAN, all host traffic should have a VLAN tag.

2. Host at SX state - ME active.
   - If ME is not Kay capable then the SC should be reset by link reset or by sending a log-off packet (1af) and the ME can return to a VLAN solution (or remain in such).
   - If the ME is KaY capable, the host should notify the ME that it retires KaY ownership and the ME should retake it.
3. Host and ME at SX.
   - The active KaY should reset the secured channel by link reset or sending a log-off packet (1af) in order to enable a WoL packet to be cleared.

3.7.3 Receive Flow

3.7.3.1 Rx Receive Modes

There are four modes of operation defined for LinkSec Rx as defined by the LSECRXCTRL.LSRXEN field:

Bypass (LSRXEN = 00b) - in this mode, LinkSec is not offloaded. There is no authenticating or decrypting the incoming traffic. The LinkSec header and trailer are not removed and these packets are forwarded to the host or the Manageability Controller (MC) according to the regular L2 MAC filtering. The packet is considered as untagged (no VLAN filtering). No further offloads are done on LinkSec packets.

Check (LSRXEN = 01b) - 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 or the MC according to the regular L2 filtering. Additional offloads are possible on LinkSec packets assuming the packet was decrypted. The header is not removed from KaY packets. In this mode, hardware has a more relaxed policy then the strict mode regarding whether to forward packets or to drop them. Since this mode is mainly for debug purposes or to overcome first generation standard inconsistencies, most of the packets are forwarded to higher layers with a suitable error code. The only case where packets are dropped is if the C bit is set and the packet failed authentication. In cases where hardware failed to locate a key but still forwards the packet, the SecTag won't be removed if bit 6 of LSECRXCTRL is set, and ICV won't be included in the packet.

Strict (LSRXEN = 10b) - in this mode, incoming packets with a 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 decrypting or authenticating was successful. Additional offloads are possible on LinkSec packets. The header is not removed from KaY packets.

Disable (LSRXEN = 11b) - in this mode, LinkSec is not offloaded and LinkSec packets are dropped. There is no authenticating or decrypting of the incoming traffic.

Note: When operating in strict or check mode, it is recommended to enable the packet checksum offload. If the offload is not enabled, the packet CRC is meaningless on encrypted packets since it was calculated before decrypting.

3.7.3.2 Receive Functionality

The MAC might receive packets that contain LinkSec encapsulation as well as packets that do not include LinkSec encapsulation concurrently. This section describes the incoming packet classification.

- Examine the user data for a SecTAG.
  - If no SecTag, proceed with the packet with the LsecH bit cleared in descriptor.
- Validate frames with a SecTAG:
  - The MPDU contains at least 17 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 <= 3Fh.
  - If the C and SC bits in the TCI are cleared, the MPDU contains 24 octets plus the number of octets indicated by the SL field; if non-zero and at least 72 octets, otherwise.
  - If the C bit is cleared and the SC bit set, then the MPDU contains 32 octets plus the number of octets indicated by the SL field; if non-zero and at least 80 octets, otherwise.
  - If the C bit is set and the SC bit cleared, then the MPDU contains 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 non-zero and at least 48 additional octets, otherwise.
  - If the C and SC bits are both set, the frame contains 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 non-zero and at least 48 additional octets, otherwise.
- Extract and decode the SecTAG as specified in Section 3.7.2.
- Extract the User Data and ICV as specified Section 3.7.1.
- 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, the packet is dropped.
  - If SCI is omitted, use the default SC (if more then one SC activated SC0 is used).
  - Select SA according to AN value.
  - If no valid SC (or more then SC active) or no valid SA found, the packet is dropped.
- 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 from the IEEE P802.1ae/D5.1 document.

3.7.3.3 Receive SA Exhausting – Re-Keying

The seamless re-keying mechanism is explained in the following example.

KaY establishes SC0 and sets SA0 as the active SA by:

- Writing the key in register LinkSec RX Key
- Writing the AN in LSECRXSA[0]
- Setting the SA Valid bit in the same register

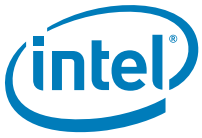
This clears the Frame received bit.

On the arrival of the first packet with SA0, the Frame received bit is automatically set. Only at this time the KaY should initiate SA1 in the same manner as was done for SA0. When a frame of SA1 arrives, SA0 retires and can be used for the next SA. For more details please see Section 10.7.13.

Same mechanism should be used for all RX SCs.

3.7.3.4 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 3.7.2. 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. If the SC bit in TCI of the frame is not set and more then one SC is valid the frame is considered as erroneous and transferred to error handling; if only one SC is valid this SC is selected in this case. 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 then 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.



3.7.3.5 Receive Statistic Counters

Detailed list and description of the LinkSec RX statistics counters can found in Section 10.7.15.

3.7.4 Transmit Data Path

The MAC might transmit packets that contain LinkSec encapsulation as well as packets that do not include LinkSec encapsulation concurrently. 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 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 the:
   a. SA Key (SAK)
   b. SCI for the SC used by the SecY to transmit
   c. PN
   d. SecTAG
   e. Sequence of octets that compose the User Data
5. Receive the following parameters from the Cipher Suite protection operation:
   a. Sequence of octets that compose the Secure Data
   b. ICV
6. Issue a request to the Transmit Multiplexer with the destination and source 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.

3.7.4.1 Transmit SA Exhausting – Re Keying

The MAC supports a single SC on the transmit data path with 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 (can be either software or firmware as defined by the LinkSec Ownership field in the FWSM register) changes the setting of the SA Select field in the LSEXTXSA register and the Active SA field gets 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 incremented above the planned value. Hardware guarantees that the PN never repeats itself, even if the KaY is slow. Once the PN reaches a value of FF...F0h, hardware clears the Enable Tx LinkSec field in the LSECTXCTRL register to 00b. When clearing the Enable Tx LinkSec field, hardware disables LinkSec offload before the PN has the chance to wraparound and then might repeat itself.

Note: Potential race conditions are possible as follows: The MAC might fetch a transmit packet (indicated as TxPacketN) from the host memory (host or ME packet). KaY can change the setting of the Tx SA Index. The TxPacketN might 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.

3.7.4.2 Transmit SA Context

After transmitting 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 TxSA and the PN from the appropriate LinkSec Tx SA PN.

3.7.4.3 Transmit Statistic Counters

Detailed list and description of the LinkSec Tx statistics counters can found in Section 10.7.14.

3.7.4.4 ME/Host Relations

The MAC supports a single CA for all the traffic that it handles. At a given time, the host might be active or inactive as well as the ME. It is expected that when only ME is enabled it acts as the KaY controlling the secured channel. The host can act as the KaY when it is functional and the control switch was executed. The following section describes the semaphore between the ME and host controlling the LinkSec setting and its tamper resistance (protection) mechanism.

3.7.4.5 Key and Tamper Protection

The LinkSec mechanism addresses hostile accesses to the organization network and traffic monitoring. In order to strengthen the capability against hostile software on the platform the hardware protects vital LinkSec context from software inside the platform. There are two levels of protection:

- Securing the LinkSec Keys
- Protecting the complete LinkSec logic from software while the firmware manages the LinkSec

3.7.4.6 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 to verify proper programming of the device. The signature is a simple byte XOR operation of the Tx and Rx keys readable in the LSECTXSUM and LSECRXSUM fields in the LSECCAP register.

3.7.4.7 Tamper Protection

In cases where the host failed authentication and cannot act as the KaY, the ME disables the host access to the 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 ME firmware can disable any read and write accesses generated by the host CPU (on the PCI interface) by setting the Lock LinkSec Logic (bit 12) bit in the FWSM register. Setting this bit can generate an interrupt to the host in case it is enabled by the host in the IMS register.

3.7.4.7.1 LinkSec Control Switch Between Firmware and Software

The following section describes the stages to switch LinkSec control ownership between the ME and the host, the owner after the switch procedure must assume all KaY responsibility.

- Only firmware has the power to switch LinkSec control unless firmware does not have the capability to act as KaY, in such a case the LinkSec ownership bit in the firmware semaphore register is cleared, and the host has the freedom to set the *Host LinkSec Connection Active* bit in H2ME register and assume responsibility. If the LinkSec *Ownership* bit is set, then the host should set the LinkSec *Request* bit prior to assuming responsibility over the LinkSec connection.
- Firmware must disable the Tx and Rx LinkSec logic before switching LinkSec control unless it was not active.
- The LinkSec owner notifies the other client (software to firmware or firmware to software) when the LinkSec connection state is changed (Connected to Unconnected). The LinkSec connection change generates an interrupt to the other client.
- When firmware owns the LinkSec it also locks host accesses (read and write) to the LinkSec CSR registers by the host.

Firmware takes LinkSec control

- Locks host software accesses to the LinkSec CSR space (set the *LLL* bit in the FWSM register)
- Closes the LinkSec channel (might require Rx filtering changes)
- Disables Tx and Rx LinkSec
- Indicates to the hardware that the firmware takes over the LinkSec control (set the *LSECO* bit in the FWSM register)
- Establishes LinkSec channel
- Indicates it to the host (set the *LSECA* bit in the FWSM register)

Firmware gives up LinkSec control

- Host software requests control over LinkSec logic (set the *LSECREQ* bit in the H2ME register and then set the *H2MEINT* bit in the CTRL register that indicates to the ME that the H2ME register contains a message)
- Firmware closes the LinkSec channel (might require Rx filtering changes)
- Firmware disables Tx and Rx LinkSec
- Firmware unlocks the host software accesses to the LinkSec CSR space (clears the *LSECL* bit in the FWSM register)
- Waits for LinkSec channel establishment by the host

3.7.4.8 LinkSec Statistics

3.7.4.8.1 Rx Statistics

After receiving a packet one and only one of the statistics in Table 32 is incremented. The precedence order of the statistics is also defined in this table.

Table 32. Rx Packet Statistics

Register Name	802.1ae Name	Priority	Notes
LSECRXUT	InPktsUntagged	1	Used in check mode. Packet is forwarded to host.
LSECRXUT	InPktsNoTag	1	Used in strict mode. Packet is dropped.
LSECRXBAD	InPktsBadTag	2	Packet is dropped in strict mode or in check mode when *C* bit is one.
LSECRXUNSCI	InPktsUnknownSCI	3	Used only in check mode. Packet is forwarded to host if *C* bit is zero.
LSECRXUNSCI	InPktsNoSCI	3	Packet is dropped in strict mode or in check mode when *C* bit is one.
LSECRXUNSA	InPktsUnusedSA	4	Used only in check mode. Packet is forwarded to host if *C* bit is zero.[1]
LSECRXNUSA	InPktsNotUsingSA	4	Packet is dropped in strict mode or in check mode when *C* bit is one.[2]
LSECRXLATE	InPktsLate	5	
N/A	InPktsOverrun	N/A	N/A
LSECRXNV[SA#]	InPktsNotValid	6	Packet is dropped in strict mode or in check mode when *C* bit is one.
LSECRXINV[SA#]	InPktsInvalid	6	Used only in check mode. Packet is forwarded to host if *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	InPktsOK	8	

1. This statistics reflects the Sum of InPktsUnusedSA for all SAs.
2. This statistics reflects the Sum of InPktsNotUsingSA for all SAs.

3.7.5 Time Sync (IEEE1588 and 802.1as)

3.7.5.1 Overview

Measurement and control applications are increasingly using distributed system technologies such as network communication, local computing, and distributed objects. Many of these applications could be enhanced by having an accurate system wide sense of time; achieved by having local clocks in each sensor, actuator, or other system device synchronized and coordinated. 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 optimized for these applications. For example, Network Time Protocol (NTP) targets large distributed computing systems with millisecond synchronization requirements. The 1588 standard specifically addresses the needs of measurement and control systems:

- Spatially localized
- Microsecond to sub-microsecond accuracy
- Administration free
- Accessible for both high-end devices and low-cost, low-end devices

The time sync mechanism activation is only possible in full duplex mode. There are no limitations on the wire speed although the wire speed might affect the accuracy.

3.7.5.2 Flow and Hardware/Software Responsibilities

The operation of a Precision Time Protocol (PTP) enabled network is divided into two stages, initialization and time synchronization.

At the initialization stage every master enabled node starts by sending sync packets that include the clock parameters of its clock. Upon receipt of a sync packet a node compares the received clock parameters to its own and if the received parameters are better, then this node moves to slave state and stops sending sync packets. When in slave state the node continuously compares the incoming sync packets to its currently chosen master and if the new clock parameters are better then the master selection is transferred to this better master clock. Eventually the best master clock is chosen. Every node has a defined time-out interval after which if no sync packet is received from its chosen master clock it moves back to master state and starts sending sync packets until a new Best Master Clock (BMC) is chosen.

The time synchronization stage is different between master and slave nodes. If a node is at a master state it should periodically send a sync packet which 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 it, the time stamp value using a Delay_Response packet. A node in 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 calculated the path delay from its master. Every sent Delay_Req packet sent by the slave is time stamped and kept. With the time stamp value received from the master with the 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 27.

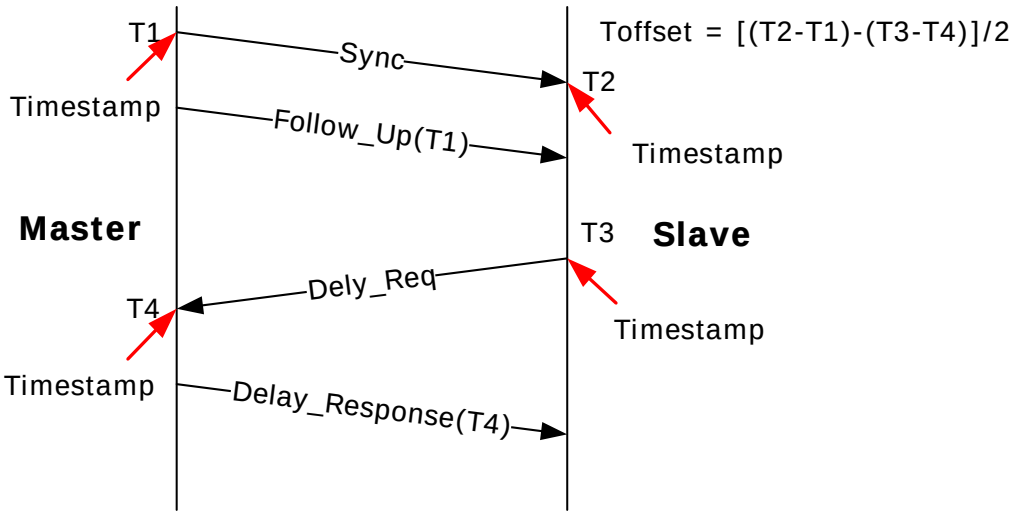


Figure 27. Sync Flow and Offset Calculation

Hardware responsibilities are:

- Identify the packets that require time stamping.
- Time stamp the packets on both Rx and Tx paths.
- Store the time stamp value for software.
- Keep the system time in hardware and give a time adjustment service to software.

Software is responsible for:

- MC protocol execution which means defining the node state (master or slave) and selection of the master clock if in slave state.
- Generate PTP packets and consume PTP packets.
- Calculate the time offset and adjust the system time using a hardware mechanism.

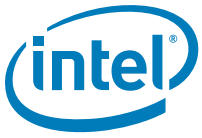
Table 33. Chronological Order of Events for Sync and Path Delay

Action	Responsibility	Node Role
Generate a sync packet with time stamp notification in 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 T1 and put it in a Follow_Up packet and send.	Software	Master
Once the Follow_Up packet arrives, store T2 from registers and T1 from the Follow_up packet.	Software	Slave
Generate a Delay_Req packet with time stamp notification in 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 the register and send back to the slave using a Delay_Response packet.	Software	Master
Once the Delay_Response packet arrives, calculate the offset using T1, T2, T3 and T4 values.	Software	Slave

3.7.5.2.1 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 *TST* bit in the ExtCMD field in the Tx advanced descriptor.

On the Rx path, hardware has two indications to transfer the packet to software, one is to indicate that this packet is a PTP packet (no matter if a timestamp was taken or not), this is also for other types of PTP packets needed for management of the protocol, this bit is set only for the L2 type of packets (the PTP packet is identified according to its Ethertype). The UDP type of PTP packets don't need such an 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 the *TST* bit in the Extended Status field of the Rx descriptor. This bit indicates to software that a time stamp was taken for this packet. Software needs to access the time stamp registers to get the time stamp values.



3.7.5.3 HW Time Sync Elements

All time sync hardware elements are reset to their initial values as defined in the registers section upon MAC reset.

3.7.5.3.1 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 using the SYSTIML and SYSTIMH registers. When in a master state the SYSTIMH and SYSTIML registers should be set once by the software according to the general system, when in slave state software should update the system time on every sync event as described in Section 3.7.5.3.3. Setting the system time is done by a direct write to the SYSTIMH register and a fine tune setting of the SYSTIM register using the adjustment mechanism described in Section 3.7.5.3.3.

Read access to the SYSTIMH and SYSTIML registers should executed 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.

After an increment event, the system time value should increment its value by the value stored in TIMINCA.*incvalue*. The increment event happens every TIMINCA.*incperiod* cycles if it is 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 if 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 a 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. For more details please refer to Section 10.7.13

Because of a PPM consideration, the clock used has a stable 48 MHz frequency that is always active at S0 state, independent of the link speed. At this frequency the recommended *incvalue* is 125 and the *incperiod* is 6 if nanosecond accuracy is required.

Note: System time registers should be implemented on a free running clock to make sure the system time is kept valid on traffic idle times (dynamic clock gating).

3.7.5.3.2 Time Stamping Mechanism

The time stamping logic is located on Tx and Rx paths at a location as close as possible to the PHY. This is to reduce delay uncertainties originating from implementation differences. The operation of this logic is slightly different on Tx and on Rx.

The Tx part decides to timestamp a packet if the Tx timestamp is enabled and the time stamp bit in the packet descriptor is set. On the Tx side only the time is captured.

On the Rx side the logic parses the traversing frame and if it is matching the message type defined in register described in Section 10.7.6 the time, sourceId and sequenceId are latched in the timestamp registers. In addition two bits in the Rx descriptor are set, one to identify that this is a PTP packet (this bit is set only for L2 packets since on the UDP packets the port number directs the packet to the application) and the second (TS) to identify that a time stamp was taken for this packet. If this PTP packet is not Sync or Delay_Req or for some reason time stamp was not taken only the first bit is set.

For more details please refer to the timestamp registers section. Figure 28 shows the exact point where the time value should be captured.

On both sides the timestamp values is locked in the registers until software access. This means that if a new PTP packet that requires time stamp has arrived before software access is not time stamped. In some cases on the Rx path a packet that was timestamped might be lost and not get to the host, to avoid a lock condition software should keep a watch dog timer to clear locking of the time stamp register. The value of such timer should be at least higher then the expected interval between two Sync or Delay_Req packets depends the state (master or slave).

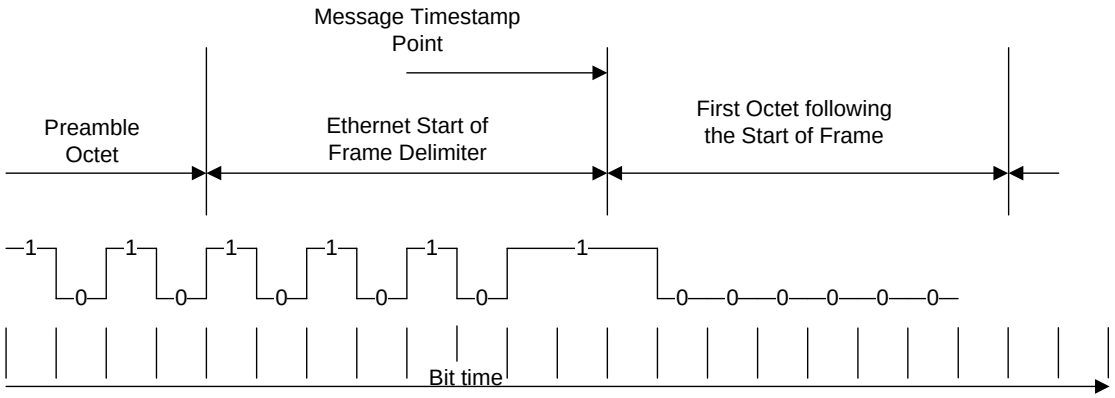


Figure 28. Time Stamp Point

3.7.5.3.3 Time Adjustment Mode of Operation

A node in the Time Sync network can be in one of two states master or slave. When a time sync entity is in a master state it synchronizes other entities to its system clock through the sending out of TimeSync, follow up, and delay response packets. Master nodes require no time adjustments. Slave nodes adjust their system clocks 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. Having all the values enables software on the slave node to calculate its offset in the following manner.

After offset calculation, the system time register should be updated. This is done by writing the calculated offset to the TIMADJL and TIMADJH registers. The order should be as follows:

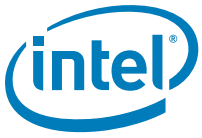
1. Write the lower portion of the offset to TIMADJL.
2. Write the high portion of the offset to TIMADJH to the lower 31 bits and the sign to the most significant bit.

After the write cycle to TIMADJH the value of TIMADJH and TIMADJL should be added to the system time.

3.7.5.4 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 layer2 V2 format.

Table 3-1. V1 and V2 PTP Message Structure



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[2]	VransportSpecific[1]	MessageId[2]
1		Reserved	VersionPTP[2]
2	VersionNetwork	VessageLength	
3			
4	Subdomain	SubdomainNumber	
5		Reserved	
6		Flags	
7			
8		CorrectionNs	
9			
10			
11			
12			
13			
14		CorrectionSubNs	
15			
16		Reserved	
17			
18			
19			
20	messageType	Reserved	
21	Source Communication Technology	Source Communication Technology	
22	Sourceuuid[2]	Sourceuuid[2]	
23			
24			
25			
26			
27			
28	Sourceportid	Sourceportid	
29			
30	SequenceId[2]	SequenceId[2]	
31			
32	Control[2]	Control	

Offset in Bytes	V1 Fields	V2 Fields
Bits	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
33	Reserved	LogMessagePeriod
34	Flags[2]	N/A
35		

1. Should be all zero.
2. Hardware interese only.

Table 3-2. PTP Message Over Layer 2

Ethernet (L2)	VLAN (Optional)	PTP Ethertype	PTP message

Table 3-3. PTP Message Over Layer 4

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 the control field in the register described at Section 10.7.6. Otherwise the byte at offset 0 (messageId) should be used for comparison to the messageId field.

The rest of the needed fields are at the same location and the same size for both V1 and V2 versions.

Table 3-4. Message decoding for V1 (the control field at offset 32)

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

Table 3-5. Message Decoding for V2 (messageId field at Offset 0)

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

MessageId	Message Type	Value (hex)
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 10.8.1 then a 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 register described at Section 10.7.2.

4.0 PCI Local Bus Interface

4.1 Introduction

The MAC contains an internal interface to the host CPU and host memory. The transactions on the internal bus are described in this section.

4.2 PCI Transaction Layer

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

4.2.1 PCI Transaction Types (Received by the Transaction Layer)

Transaction Type	Flow Control Type	Transaction Response	Data Kept From Original Packet for Response	Transaction Target
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
Memory Write Request	PH + PD	-	-	CSR
IO Read Request	NPH	CPLH + CPLD	Requester ID, TAG, Attribute	CSR
IO Write Request	NPH + NPD	CPLH	Requester ID, TAG, Attribute	CSR
Read completions	CPLH + CPLD	-	-	DMA

Note: Flow Control Types:

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

Note: Maximum payload size for reads is 128 bytes.

4.2.2 PCI Transaction Types (Transmitted by the Transaction Layer)

Transaction Type	Payload Size	Flow Control 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

Note: MAX_PAYLOAD_SIZE is 64 bytes for writes.

4.2.3 PCI Message Space (Handled as a Receiver)

The PCI host interface does not support message passing in the same capacity as PCI. There are no messages that are received that the MAC must interpret. As a result, any messages received are silently dropped.

4.2.4 PCI Message Space (Handled as a Transmitter

The PCI host interface does not support message passing in the same capacity as PCI. Any messages that are passed through the backbone are passed, but the backbone itself does not do anything with them. As a result, there is no need to pass messages.

4.2.5 PCI Data Alignment

4.2.5.1 4 KB Boundary

PCI requests must never specify an address/length combination that causes a memory space access to cross a 4 KB boundary. It is hardware's responsibility to break 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 alignment to the 4K boundaries is done in the core. The transaction layer does not do any alignment according to these boundaries.

4.2.5.2 64 Bytes[1]

PCI requests are in multiples of 64 bytes and aligned to make better use of memory controller resources. Writes, however, can be on any boundary and can cross a 64-byte alignment boundary.

4.2.5.3 Request Sizes and Alignment (ICH10 Only)

PCI requests are 128 bytes or less and are aligned to make better use of memory controller resources. Writes, however, can be on any boundary (restricted by the 4 KB boundary) and can cross a 64 byte alignment boundary.

4.2.6 PCI Configuration Request Retry Status

The MAC might have a delay in initialization due to an NVM read. If the NVM configuration read operation is not completed and the MAC receives a configuration request, the MAC responds with a Configuration Request Retry Completion Status to terminate the Request. This stalls the Configuration Request until such time that the subsystem has completed local initialization and is ready to communicate with the host.

Note that on any consecutive NVM load during active time, only memory or I/O slave accesses are retried by a Stop assertion until the NVM loads.

4.2.7 PCI Outstanding DMA Read Requests

The MAC might issue the following master read request from each of the following clients:

- Host Rx Descriptor Read (two: one per queue)
- Host Tx Descriptor Read (two: one per queue)
- Host Tx Data Read (Up to four)

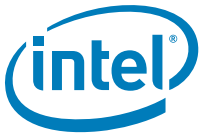
On SPXB (**ICH10** only), all requests with the same TAG are guaranteed to be completed in order. Therefore, no out of order completion handling is necessary when the same TAG IDs are used.

4.2.8 PCI Outstanding Target Reads

The MAC can receive read requests in the following manner:

- Host read (one) – configuration, memory, or I/O.

1. Not applicable to the **ICH10**.



4.2.9 PCI Transaction Attributes

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

The MAC supports the following:

- Host Traffic: Host Bus: TC = 0 and VC = 0 (default).

4.2.10 PCI Tag IDs

A client ID identifies the client and is included in the Tag field of the PCI packet header. Completions always carry the tag value included in the request to enable routing of the completion to the appropriate client.

PCI Client IDs are assigned as follows:

TAG Code in HEX	Flow: TLP TYPE – Usage
00	RX: WR REQ (data from Ethernet to main memory).
01	RX: RD REQ to read descriptor to core.
02	RX: WR REQ to write back descriptor from core to memory.
04	TX: RD REQ to read descriptor to core.
05	TX: WR REQ to write back descriptor from core to memory.
06	TX: RD REQ to read descriptor to core second queue.
07	TX: WR REQ to write back descriptor from core to memory second queue.
08	TX: RD REQ Data 0 from main memory to Ethernet.
09	TX: RD REQ Data 1 from main memory to Ethernet.
0A	TX: RD REQ Data 2 from main memory to Ethernet.
0B	TX: RD REQ Data 3 from main memory to Ethernet.
0C	RX: RD REQ to bring descriptor to core second queue.
0E	RX: WR REQ to write back descriptor from core to memory second queue.
10	MNG: RD REQ: read data.
11	MNG: WR REQ: write data.
1E	MSI.
Others	Reserved.

4.2.11 PCI Completion Timeout Mechanism

The PCI host interface does not require a completion timeout mechanism as PCI requires. No timeout mechanism is implemented in the MAC.

4.2.12 PCI Out of Order Completion Handling

On the PCI host interface, all requests with the same TAG are guaranteed to complete in order. As a result, no out of order completion handling is necessary.

4.2.13 PCI Error Events and Error Reporting

4.2.13.1 Data Parity Error

The PCI host bus does not provide parity protection, but it does forward parity errors from bridges. The MAC recognizes parity errors through the internal bus interface and sets the *Parity Error* bit in PCI Configuration space. If parity errors are enabled in configuration space, a system error is indicated on the PCI host bus to the chipset. The offending cycle with a parity error is dropped and not processed by the MAC.

4.2.13.2 Completion with Unsuccessful Completion Status

A completion with unsuccessful completion status (any status other than 000h) is dropped and not processed by the MAC. Furthermore, the request that corresponds to the unsuccessful completion is not retried. When this unsuccessful completion status is received, the *System Error* bit in the PCI configuration space is set. If the system errors are enabled in configuration space, a system error is indicated on the PCI host bus to the chipset.

4.2.14 PCI Device ID

The MAC is a PCI function device type integrated in the ICH. A device ID of a PCI device is composed of three fields: Bus Number, Device Number and Function Number. These fields are provided to the PCI device on PCI configuration cycles. The PCI device provides its ID fields on cycle completions or its own initiated cycles.

- Bus Number - The MAC captures its bus number during host configuration write cycles type 0 aimed to the MAC.
- Device Number - During nominal operation, the MAC has a predefined device number that equals 25 (19h).
- Function Number - The MAC is a single PCI function (Function 0).

Note: The bus number, device number and nominal function number are reflected in the software device driver from the BUSNUM CSR register.

- Configuration Cycles - The MAC claims every configuration cycle type 0 according to its Device and Function numbers previously described. As previously mentioned, the MAC captures its bus number during host configuration write cycles type 0.
- IO and Memory Cycles - The MAC claims I/O and memory cycles according to the address space as defined in its I/O and Memory Base Address registers. These registers are initialized by the host using configuration cycles.

4.3 PCI Configuration Registers

PCI implements a single function (Function 0) PCI configuration space with the following features:

- Mandatory PCI configuration registers
- Power management capabilities
- MSI capabilities
- FLR capabilities (**ICH9** and **ICH10**)

4.3.1 Mandatory PCI Configuration Registers

The PCI configuration registers map follows. Refer to the detailed descriptions for registers loaded from the NVM at initialization. Initial values of the configuration registers are marked in parenthesis.

Configuration registers are assigned one of the attributes listed in the following table.

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 or write and can be either set or reset.
R/WC	Read only status / Write 1b to clear the register. Writing a 0b to R/WC bits has no effect.
RWS	For **ICH8**, Read/Write with Sticky Bits: Register bits are read or write and might be either set or reset by software to the desired state. Bits are not cleared by a 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 through AUX power or PME enable) is enabled.
R/WCS	Read 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. Writing a 0b to R/WC bits has no effect. Bits are not cleared by reset and can only be reset with the AUXPWROK 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.
RWO	Read-Write-Once. Once this field is set by software, it is locked for further write accesses.

Table 34. PCI Compatible Configuration Registers

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0h	Device ID (104Bh) (10CDh for **ICH10**)		Vendor ID (8086h)	
4h	Status Register (0010h)		Command Register (0000h)	
8h	Class Code (020000h)			Revision ID (00h)
Ch	Reserved	Header Type (00h)	Latency Timer (00h)	Cache Line Size (00h)
10h	Base Address 0 (CSR Memory)			
14h	Base Address 1 (FLASH Memory)			
18h	Base Address 2 (I/O)			
1Ch	Base Address 3 (00000000h)			
20h	Base Address 4 (00000000h)			
24h	Base Address 5 (00000000h)			
28h	CardBus CIS Pointer (00000000h)			
2Ch	Subsystem ID (0000h)		Subsystem Vendor ID (8086h)	
30h	Expansion ROM Base Address (000000h)			
34h	Reserved (000000h)			Cap_Ptr (C8h)
38h	Reserved (00000000h)			
3Ch	Max_Latency (00h)	Min_Grant (00h)	Interrupt Pin (01h)	Interrupt Line (00h)

Note: The following color notation is used for reference:

	Read only fields.
	Hard coded fields (also read-only).

Interpretation of the various MAC registers is provided as follows.

Vendor ID - 2 Bytes at Offset 00h

This is a read only register that has the same value for all PCI functions. It identifies uniquely Intel products. The field can be automatically loaded from the NVM at address 0Eh during initialization depending on the *Load Vendor/Device ID* bit field in NVM word 0Ah with a default value of 8086h.

Device ID - 2 Bytes at Offset 02h

This is a read-only register with a default value of 104Bh (10CD for **ICH10**). The field can be auto-loaded from the NVM word 0Dh during initialization depending on the *Load Vendor/Device ID* field in NVM word 0Ah.

Command - 2 Bytes at Offset 04h

This is a read/write register. Note that the shaded bits are not used by this implementation and are hardwired to 0b.

Bit(s)	Initial Value	Description
15:11	0b	Reserved.
10	0b	Interrupt Disable Controls the ability of a MAC to generate a legacy interrupt message. When set, the MAC cannot generate legacy interrupt messages.
9	0b	Fast Back-to-Back Enable. Hardwired to 0b.
8	0b	SERR# Enable.
7	0b	Wait Cycle Enable Hardwired to 0b.
6	0b	Parity Error Response.
5	0b	Palette Snoop Enable Hardwired to 0b.
4	0b	MWI Enable Hardwired to 0b.
3	0b	Special Cycle Monitoring Hardwired to 0b.
2	0b	Bus Master Enable (BME) Must be set to 1b for the MAC to initiate memory read and write requests as a master. This bit does not affect a MAC's ability to generate an MSI message, error message or completion cycles as a target.
1	0b	Memory Access Enable.
0	0b	I/O Access Enable.

Status Register - 2 Bytes at Offset 06h

Shaded bits are not used by this implementation and are hardwired to 0b.

Bit(s)	Initial Value	RD/WR	Description
15	0b	R/WC	Detected Parity Error Note that this bit is never set by the MAC.
14	0b	R/WC	Signaled System Error Note that this bit is never set by the MAC.
13	0b	R/WC	Received Master Abort Note that this bit is never set by the MAC.
12	0b	R/WC	Received Target Abort Note that this bit is never set by the MAC.
11	0b	R/WC	Signaled Target Abort Note that this bit is never set by the MAC.
10:9	00b		DEVSEL Timing Hardwired to 0b.
8	0b	R/WC	Data parity reported Note that this bit is never set by the MAC.
7	0b		Fast Back-to-Back Capable Hardwired to 0b.
6	0b		Reserved.
5	0b		66 MHz Capable Hardwired to 0b.
4	1b	RO	New Capabilities This indicates that a MAC implements Extended Capabilities. The MAC sets this bit and implements a capabilities list indicating it support for PCI power management and MSI.
3	0b	RO	Interrupt Status Interrupt Status is a read-only field that indicates that an interrupt message is pending internally to the MAC.
2:0	0b		Reserved.

Revision - 1 Byte at Offset 08h

The default revision ID of the MAC (00h).

Class Code - 3 Bytes at Offset 09h

The class code is a read-only, hard-coded value that identifies the MAC functionality. For example, 020000h for an Ethernet adapter.

Cache Line Size - 1 Byte at Offset 0Ch

This field is implemented by PCI devices as a read-write field for legacy compatibility purposes but has no impact on any MAC functionality. It defaults to 0000h at reset.

Latency Timer - 1 Byte at Offset 0Dh

Not used; hardwired to 0b.

Header Type - 1Byte at Offset 0Eh

This indicates whether or not a MAC is single PCI function or multi-function device. The MAC is a single function device (00h).

Base Address Registers

The Base Address Registers (BARs) are used to map the I/O and memory spaces.

Memory Base Address Registers - 4 Bytes at Offset 10h

The internal CSR registers and memories are accessed as direct memory mapped offsets from the base address register. Note that software can only access an entire dword at a time.

Bit(s)	Reset	Type	Description
0	0	RO	Memory and I/O Space Set to 0b indicating a memory space BAR.
2:1	00b	RO	Memory Type Set to 00b indicating 32-bit BAR.
3	0	RO	Prefetchable Memory The MAC implements non-prefetchable space since it has read side effects.
16:4	0	RO	Memory Size Memory BAR 0 spaces size is 128 KB.
31:17	0	RW	Base Address Software programs this field with the base address of this region.

Flash Base Address Registers - 4 Bytes at Offset 14h

The internal registers that are used to access the LAN space in an external flash device. Accessed to these registers are direct memory mapped offsets from the base address register. Note that software can only access one dword at a time.

Bit(s)	Reset	Type	Description
0	0	RO	Memory and I/O Space Set to 0b indicating a memory space BAR.
2:1	00b	RO	Memory Type Set to 00b indicating 32-bit BAR.
3	0	RO	Prefetchable Memory The MAC implements non-prefetchable space since it has read side effects.
11:4	0	RO	Memory Size Memory BAR 0 spaces size is 4 KB.
31:12	0	RW	Base Address Software programs this field with the base address of this region.

I/O Base Address Registers - 4 Bytes at Offset 18h

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. Note that software can only access one dword at a time.

Bit(s)	Reset	Type	Description
0	1	RO	Memory and I/O Space Set to1b indicating an IO space BAR.
4:1	0	RO	IO Size I/O spaces size is 32 bytes.
31:5	0	RW	Base Address Software programs this field with the base address of this region.

Subsystem Vendor ID - 2 Bytes at Offset 2Ch

This value can be loaded automatically from NVM word 0Ch at power up or reset depending on the *Load Subsystem ID* field in NVM word 0Ah. A value of 8086h is the default for this field at power up if the NVM does not respond or is not programmed. All functions are initialized to the same value.

Subsystem ID - 2 Bytes at Offset 2Eh

This value can be loaded automatically from NVM word 0Bh at power up depending on the *Load Subsystem ID* field in NVM word 0Ah with a default value of 0000h. This value can be loaded from NVM word 0Bh.

Expansion ROM Base Address - 4 Bytes at Offset 30h

This register is used to define the address and size information for boot-time access to the optional Flash memory. No Flash memory exists and this value reports 00000000h.

Cap_Ptr - 1 Byte at Offset 34h

The *Capabilities Pointer* field provides an offset in the MAC's PCI configuration space for the location of the first item in the Capabilities Linked List. The Cap_Ptr points to the PCI Power Management Capability header at offset C8h.

Address	Item	Next Pointer
C8h : CFh	PCI Power Management	D0h
D0h : DFh	Message Signaled Interrupt	00h
E0h : EFh[1]	FLR Capability	00h

1. **ICH9/ICH10** Only.

Interrupt Line - 1 Byte at Offset 3Ch

Read/write register programmed by software to indicate which of the system interrupt request lines this MAC's interrupt pin is bound to. Note that each PCI function has its own register. This register is considered the property of the BIOS, therefore it is not initialized by a PCI reset and for the **ICH9/ICH10**, not by FLR.

Interrupt Pin - 1 Byte at Offset 3Dh

This is a read only register.The MAC implements a legacy interrupt on INTA (01h).

Max_Lat/Min_Gnt - 2 Bytes at Offset 3Eh

This field is not used and is hardwired to 0b.

4.3.2 PCI Power Management Registers

All fields are reset on full power up. All of the fields except PME_En and PME_Status are reset on exit from the D3cold state. If auxiliary power is not supplied, the PME_En and PME_Status fields are also reset on exit from the D3cold state.

The tables that follow list the organization of the PCI Power Management Register block. Note that Initial values are marked in parenthesis.

Some fields in this section depend on the Power Management Enable bits in NVM word 0Ah.

Table 35. Power Management Register Block

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
C8h	Power Management Capabilities (PMC)		Next Pointer (D0h)	Capability ID
CCh	Data	PMCSR_BSE Bridge Support Extensions	Power Management Control / Status Register (PMCSR)	

Note: The following color notation is used for reference:

	Fields identical to all functions.
	Hard coded fields and strapping option. Also read-only.

The following section describes the register definitions, whether they are required or optional for compliance, and how they are implemented in the MAC.

Capability ID - 1 Byte at Offset C8h (RO)

This field equals 01h, indicating the linked list item as the PCI Power Management registers.

Next Pointer - 1 Byte at Offset C9h (RO)

This field provides an offset to the next capability item in the capability list. It has a value of D0h, which points to MSI capability.

ICH9/ICH10 Next Pointer - 1 Byte at Offset C9h (RO)

This field provides an offset to the next capability item in the capability list. It has a value of D0h, which points to MSI capability.

Software (BIOS software) might override the initialization value by setting the value to 00h which hides the next capabilities from the operating system or set it to E0h (FLR capability) hiding just the MSI from the operating system. Once this field is set by software, it is locked from further write accesses. PCI reset or FLR initialization sets this field to its default value and re-enables its programming.

Power Management Capabilities (PMC) - 2 Bytes at Offset CAh (RO)

This field describes the device functionality at the power management states as described in the table that follows. Each device function has its own register.

Table 36. Power Management Capabilities (PMC)

Bit(s)	Default	RD/WR	Description
15:11		RO	PME_Support This 5-bit field indicates the power states in which the function might assert HOST_WAKE. It depends on the PM Enable and AUX-PWR bits in NVM word 0Ah: Condition ⇒ Functionality ⇒ Value PM Enable = 0b in NVM ⇒ No PME at all states ⇒ 00000b PM Enable and No Aux Pwr = 0b ⇒ PME at D0 and D3hot ⇒ 01001b PM Enable and Aux Pwr = 1b ⇒ PME at D0, D3hot and D3cold ⇒ 11001b
10	0b	RO	D2_Support The MAC does not support the D2 state.
9	0b	RO	D1_Support The MAC does not support D1 state.
8:6	000b	RO	Auxiliary Current This is the required current defined in the data register.
5	1b	RO	DSI The MAC requires its device driver to be executed following transition to the D0 uninitialized state.
4	0b	RO	Reserved This bit is reserved and should be set to 0b.
3	0b	RO	PME_Clock The PME clock is disabled and is hardwired to 0b.
2:0	010b	RO	Version. The MAC complies with PCI Power Management Specification, Revision 1.1.

Power Management Control/Status Register (PMCSR) - 2 Bytes at Offset CCh (R/W)

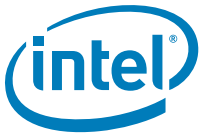
This register is used to control and monitor power management events in the device. Each device function has its own PMCSR.

Table 37. Power Management Control/Status Register

Bit(s)	Default	RD/WR	Description
15	0b (at power up)	R/WC/P	PME_Status This bit is set to 1b when the function detects a wake-up event independent of the state of the *PME Enable* bit. Writing a 1b clears this bit.
14:13	Reflects value in Data Register	RO	Data_Scale This field indicates the scaling factor that is used to interpret the value of the Data Register. For the LAN and Common functions this field equals 01b (indicating 0.1 watt units) if power management is enabled in the /NVM and the *Data_Select* field is set to 0, 3, 4, or 7 (or 8 for Function 0). Otherwise, it equals 00b. For the manageability functions, this field equals 10b (indicating 0.01 watt units) if power management is enabled in the NVM and the *Data_Select* field is set to 0, 3, 4, or 7. Otherwise, it equals 00b.
12:9	0000b	R/W	Data_Select This four-bit field is used to select which data is to be reported through the Data Register and Data_Scale field. These bits are writable only when power management is enabled through the NVM.
8	0b (at power up)	R/W	PME_En If Power Management is enabled in the NVM, writing a 1b to this register enables wakeup. If power management is disabled in the NVM, writing a 1b to this bit has no affect and will not set the bit to 1b.
7:4	0000b	RO	Reserved These bits are reserved and should return a value of 0000b for this field.
3	0b	RO	No Soft Reset When set to 0b, defines if the MAC executed an internal reset on the transition to D0. The MAC always reports 0b in this field.
2	0b	RO	Reserved. Returns a value of 0b for this field.
1:0	00b	R/W	Power State This field is used to set and report the power state of a function as follows: 00b – D0 01b – D1 (cycle ignored if written with this value) 10b – D2 (cycle ignored if written with this value) 11b – D3 (cycle ignored if power management is disabled in the NVM)

PMCSR_BSE Bridge Support Extensions - 1 Byte at Offset CEh (RO)

This register is not implemented in the MAC and its value should be set to 00h.



Data Register - 1 Byte at Offset CFh (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 and the power scale is reported in the *Data_Scale* field of the PMCSR. Data from this field is loaded from the NVM if power management is enabled in the NVM or with a default value of 00h otherwise. The values for the MAC functions are as follows:

Data_Select	Value	Data_Scale	Description
0h	NVM word 10h	01b	D0 power consumption
3h	NVM word 10h	01b	D3 power consumption
4h	NVM word 10h	01b	D0 power dissipation
7h	NVM word 10h	01b	D3 power dissipation
8h	NVM word 10h	01b	Common power

Note: For other Data_Select values, the Data Register output is reserved (0b).

4.3.2.1 Message Signaled Interrupt (MSI) Configuration Registers

This capability defines the operation of the MSI interrupt feature.

Table 38. Message Signaled Interrupt Configuration Registers

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
D0h	Message Control (0080h)		Next Pointer (00h) (E0h for **ICH10**)	Capability ID (05h)
D4h	Message Address			
D8h	Message Upper Address			
DCh	Reserved		Message Data	

Note: The following color notation is used for reference:

Hard coded fields.

Capability ID - 1 Byte at Offset D0h (RO)

This field equals 05h indicating the linked list item as being the Message Signaled Interrupt registers.

Next Pointer - 1 Byte at Offset D1h (RO)

This field provides an offset to the next capability item in the capability list. Its value of 00h indicates the end of the list.

ICH9 Next Pointer - 1 Byte at Offset D1h (R/Write Once)

This field provides an offset to the next capability item in the capability list. Its default value of E0h points to the FLR capability structure. Software (BIOS software) might override the initialization value by setting the value to 00h which hides the next capabilities from the operating system or set it to E0h (FLR capability) hiding just the MSI from the operating system. Once this field is set by software, it is locked from further write accesses. PCI reset or FLR initialization sets this field to its default value and re-enables its programming.

Message Control - 2 Bytes at Offset D2h (R/W)

The register fields are described in the table that follows.

Bits	Default	RD/WR	Description
15:8	0b	RO	Reserved Reads as 0b.
7	1b	RO	64-bit Capable A value of 1b indicates that the MAC is capable of generating 64-bit message addresses.
6:4	000b	RO	Multiple Message Enable The MAC returns 000b to indicate that it supports a single message.
3:1	000b	RO	Multiple Message Capable Indicates a single requested message.
0	0b	R/W	MSI Enable If 1b, Message Signaled Interrupts. In this case, the MAC generates an MSI for interrupt assertion instead of INTx signaling.

Message Address Low - 4 Bytes Offset D4h (R/W)

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.

Message Address High - 4 Bytes at Offset D8h (R/W)

Written by the system to indicate the upper 32-bits of the address to use for the MSI memory write transaction.

Message Data - 2 Bytes at Offset DCh (R/W)

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.

4.3.2.2 ICH9/ICH10 FLR Capability

Figure 39 lists the organization of the Function Level Reset capability structure.

The FLR capability structure depends on a *FLRCSSEL* control bit. *FLRCSSEL* is located in the General Control and Status register at offset 3410h in the Direct Media Interface described in Chapter 8 of the ICH9/ICH10 EDS.

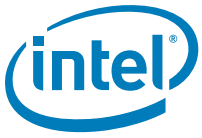


Table 39. ICH9 FLR Capability Structure

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
E0h	Capability (20h/03h)	Length (06h)	Next Pointer (00h)	Capability ID (13h/ 09h)
E4h			Device Status (00h)	Device Control (00h)

Note: The following color notation is used for reference:

Hard coded and strapping option (RO).

Capability ID - 1 Byte at Offset E0h (RO)

The value of the capability ID depends on the *FLRCSSEL* bit as follows: If *FLRCSSEL* is 0b the Capability ID equals 13h. If *FLRCSSEL* is 1b the Capability ID equals 09h, which indicates the vendor specific capability structure.

Next Pointer - 1 Byte at Offset E1h (RO)

This field provides an offset to the next capability item in the capability list. Its value of 00h indicates the end of the list.

Length - 1 Byte at Offset E2h (RO)

This field identifies the number of bytes for this capability structure. It has the value of 06h for FLR capability.

Capability - 1 Byte at Offset E3h (RO)

The *Capability* field depends on the *FLRCSSEL* bit. If *FLRCSSEL* is 1b the *Capability* field is read only and equals 20h (4 MSB = 2h defines the capability and 4 LSB = 0h defines the version). If *FLRCSSEL* is 0b the *Capability* field includes FLR enablement fields. The table below describes the *Capability* field as a function of *FLRCSSEL*.

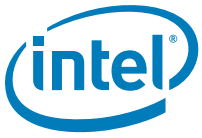
Bits	Default	R/W	Description (FLRCSSEL = 0b)
0	1b	R/W Once	TXP Capability When set to 1b, this bit indicates support for the Transactions Pending (TXP) bit. TXP must be supported if FLR is supported. This bit is read/write once. Once this field is set by software it is locked from further write accesses. A PCI reset or FLR initialization sets this field to its default value and re-enables its programming.
1	1b	R/W Once	FLR Capability When set to 1b, this bit indicates support for Function Level Reset (FLR). This bit is read/write once. Once this field is set by software it is locked from further write accesses. A PCI reset or FLR initialization sets this field to its default value and re-enables its programming.
7:2	0h	RO	Reserved
3:0	0h	RO	Capability Version 0h.
7:4	2h	RO	Vendor Specific Capability ID 2h = FLR Capability.

Device Control - 1 Byte at Offset E4h (RW)

Bits	Default	Rd/Wr	Description
0	0b	RW	Initiate FLR Used to initiate a FLR transition. Writing a 1b initiates a FLR transition. Since hardware must not respond to any cycles until a FLR completes, the value read by software from this bit is 0b.
7:1	00h	RO	Reserved.

Device Status - 1 Byte at Offset E5h (RW)

Bits	Default	Rd/Wr	Description
0	0b	RW	Transactions Pending (TXP) When set to 1b, indicates that the MAC has issued non-posted requests that are not complete. When set to 0b, indicates that completions for all non-posted requests have been received.
7:1	00h	RO	Reserved.



Note: This page intentionally left blank.

5.0 NVM Interface

5.1 Introduction

Refer to the following MAC and PHY NVM Map and Information Guides for NVM contents and Images. Refer to Table 1 for a list the PHYs that are associated with specific MACs.

- *Intel® I/O Controller Hub 10 LAN NVM Map and Information Guide*
- *Intel® I/O Controller Hub 9 LAN NVM Map and Information Guide*
- *Intel® I/O Controller Hub 9M LAN NVM Map and Information Guide*
- *Intel® I/O Controller Hub 8/8M LAN NVM Map and Information Guide*

Note: This page intentionally left blank.

6.0 Power Management

6.1 Introduction

This section details the power management functions of the MAC. The first section discusses PCI. The second section describes special considerations relating to the MAC and PHY interface aspects including LAN disable.

6.2 Host Power Management (PCI)

The MAC supports the Advanced Configuration and Power Interface (ACPI) Specification as well as Advanced Power Management (APM). This enables the host to wake up (such as from an Sx to an S0state) by a network-related activity via an internal *Host_Wake* signal. This section describes how power management is implemented using the MAC.

Implementation requirements were obtained from the following documents:

- PCI Bus Power Management Interface Specification - Revision 1.1
- ACPI Specification - Revision 2.0

Note: ACPI and APM Wake on LAN (WoL) are enabled in the NVM PCI Initialization Control Word.

Note: Power management can be disabled through bits in the Initialization Control Word, which is loaded from the NVM during power-up or after a reset. Even when disabled, the power management register set is still present. Power management support is required by the PCI specification.

6.2.1 PCI Power States

PCI (the host function) supports D0 and D3 power states defined in the PCI Power Management and PCI specifications. D0 is divided into two sub-states: D0u and D0a. The D3 state is also divided to two sub-states: D3 hot, which might be stated in this section as D3, and D3 cold, which is also named in this document as Dr. Note that the “r” index in the name indicates PCI reset.

For **ICH9/ICH10**, The MAC supports Function Level Reset (FLR) as described in Section 2.0. The FLR state flow is described in Section 6.2.5.4.2.

Figure 29 and Figure 30 show the power states and transitions between them.

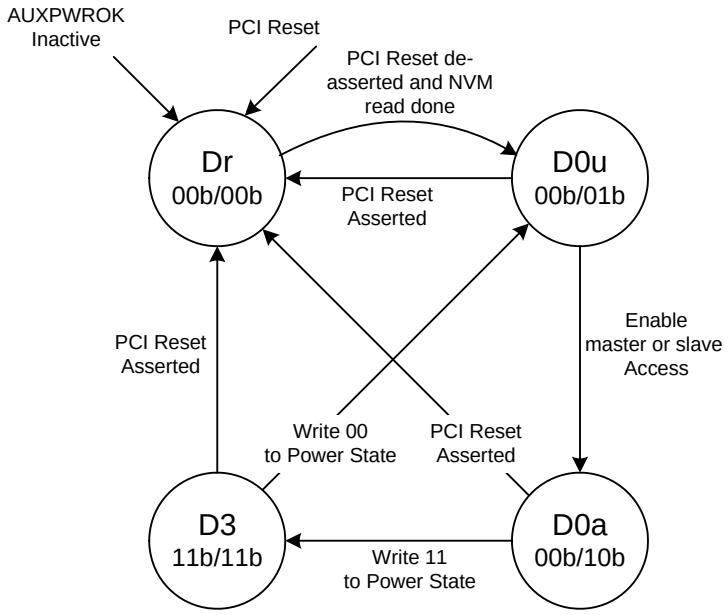


Figure 29. PCI Power States and Transitions (ICH8)

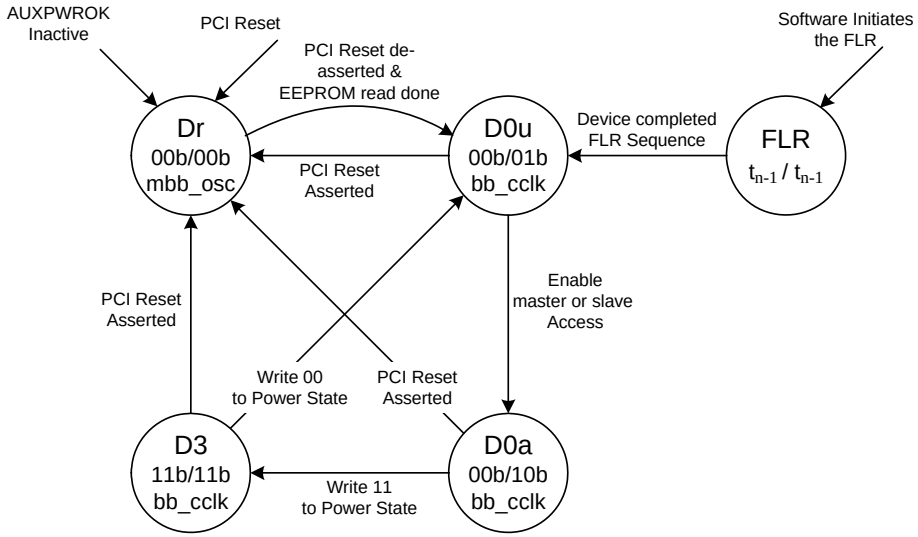


Figure 30. PCI Power States and Transitions (ICH9/ICH10)

6.2.2 Assumptions

The following assumptions apply to the implementation of power management:

- The software device driver sets the filters up prior to the system transitioning PCI to the D3 state.
- Before a transition from D0 to the D3 state, the operating system ensures the software device driver has been disabled.
- No wake-up capability, except APM wakeup if it is enabled in the NVM, is required after the system puts the MAC into the D3 state and then returns it to D0.
- If the *APMPME* bit in the Wake Up Control register (WUC.APMPME) is 1b, it is permissible to assert Host_Wake signal even when *PME_En* is 0b.

6.2.3 PCI Unit Power Reduction Measures

This section describes the power reduction techniques used by the MAC. Note that the internal PCI DMA engine has a running clock as long as the PCI function is in a functional power state as listed in Table 40.

Table 40. Clock Gating

Dynamic Clock Gating in NVM	Power MNG State	DMA Main Clock	DMA Wake Clock	DMA Free Clock
Disabled	All states	On	On	On
Enabled	D0a	Gated when all the following are inactive[1]: • Filter (regular) • DMA • GHOST • UNC	Gated when all the following are inactive[1]: • MAC Receive • Filter (early) • DMA • NVM Read • CSR Access • D/UD Change • GHOST • FUNC	On
Enabled	D0u	Gated	Same as D0a	On
Enabled	D3, Dr	Gated	Gated when all the following are inactive[1]: • MAC Receive • Filter (early)	On

1. Inactive means no request is pending from this agent.

When the *CLK_CNT_1_4* bit in the Status CSR (loaded from NVM) is set to 0b, the DMA clock runs at full frequency regardless of the link speed / state. The hardware default value is 1b.

- 1000 Mb/s link: The DMA gets the MAC clock of 62.5 MHz.
- At 10/100 Mb/s, link negotiation, or link down, the DMA gets the mosc_clk (62.5 MHz).

When the *CLK_CNT_1_4* bit is set (hardware default) it enables dynamic reduction of the DMA clock according to the functional power states.

- 1000 Mb/s link: it runs at full speed of 62.5 MHz from the MAC.

Else

- Link at 10/100 Mb/s, link negotiation, or link down, it operates at ¼ of the mosc_clk frequency.

6.2.4 Auxiliary Power Usage

The MAC uses the *AUX-PWR* NVM bit indication that the LAN Power well (AUXPWR) is connected to the VAUX platform power, and therefore advertises D3cold Wake Up support. The amount of power required for the function (including the entire NIC) is advertised in the Power Management Data register, which is loaded from the NVM.

If D3cold is supported, the *PME_En* and *PME_Status* bits of the Power Management Control/Status register (PMCSR), as well as their shadow bits in the Wake Up Control register (WUC) is 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. Note that AUX-PWR is a NVM bit configuration in the MAC.

In addition, the actual PME functionality in D3 cold is gated by the CTRL.D3WUC bit field. During nominal operation this bit must be set to 1b as its hardware default setting.

6.2.5 PCI Power States

6.2.5.1 D0 Uninitialized State

The D0u state is a low-power state used after a PCI Reset is de-asserted following power-up (cold or warm) or on D3 exit.

When entering D0u, the MAC disables wake ups and asserts a reset to the PHY while the NVM is being read. If the *APM Mode* bit in the NVM Initialization Control Word 2 is set, then APM wake up is enabled.

D0u is reached from either the Dr state (after asserting LAN_RST# or PCI Reset) or the D3hot state (by software writing a value of 00b to the *Power State* field of the PCI-PM registers).

Asserting LAN_RST# or PCI Reset (SPXB Reset for **ICH10**) means that the entire state of the MAC is cleared, other than sticky bits.

On a transition from D3 to D0u, the PCI configuration space is reset. According to the PCI Power Management Specification, Revision 1.1, Section 5.4, software “will need to perform a full re-initialization of the function including its PCI Configuration Space.”

6.2.5.2 D0 Active State

Once memory space is enabled, all internal clocks are activated and the MAC enters an active state. It can transmit and receive packets if it is properly configured by the software device driver. The PHY is enabled or re-enabled by the software device driver to operate or auto-negotiate to full line speed and power if it is not already operating at full capability. Any APM wake up previously active remains active. The software device driver can deactivate APM wake up by writing to the Wake Up Control Register (WUC). It can also activate other wake-up filters by writing to the Wake Up Filter Control Register (WUFC).

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.

6.2.5.3 D3 State (Equals PCI-PM D3hot)

When the system writes 11b to the *Power State* field of the PMCSR, the MAC transitions to D3. Any wake-up filter settings enabled before entering this reset state are maintained. Upon transition to the D3 state, the MAC clears the *Memory Access Enable* and *I/O Access Enable* bits of the PCI Command register, which disables memory access decode. In D3, the MAC only responds to PCI configuration accesses and does not generate master cycles.

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 00b to the *Power State* field of the PMCSR. Transition to the Dr state is through PCI Reset assertion.

6.2.5.3.1 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 software device driver disables scheduling of further tasks to the MAC. It masks all interrupts and does not write to the transmit descriptor tail register or to the receive descriptor tail register and operates the master disable algorithm. If wake up capability is needed, the software device driver should set up the appropriate wake up registers and the system should write a 1b to the *PME_En* bit of the PMCSR.

As a response to being programmed into D3 state, the MAC 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 are kept in the MAC (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 order to save platform power in the Sx state, the MAC does not support GbE operation in the D3 state. On entry to D3 state, the MAC sets the PHY link speed autonomously as defined in the PHY_CTRL register.

Note: The PHY link speed setting in D0 and non-D0 are enabled by the respective fields in the PHY_CTRL register. These fields are loaded from the NVM. The NVM should disable GbE speed at non D0 (D3).

For ICH8/82566 Only:

If a platform has ME firmware enabled and operating with a GbE PHY, firmware should not enable a transition to system power down (Sx, while x>0) while the MAC is in GbE mode. If the *RSPCIPHY* bit in the Firmware Semaphore (FWSM) register is cleared (PHY reset is gated) while the MAC transitions to the D3 state, then hardware does not change the PHY setting and does not re-negotiation the link with GbE disable. A system based on ICH8 does not support GbE in the Sx state. Firmware must ensure that such a setting does not happen. lists the firmware flow that guarantees the this requirement.

IF (CurrentSystemState = S0 and NextSystemState != S0 and RSPCIPHY = 0) then do the following:

- Firmware should read the PHY_CTRL CSR register (offset 00F10h), getting the following fields:
  - B2B Ena (bit 7)
  - LPLU in non D0a (bit 2)
  - LPLU in D0a (bit 1)
  - SPD Ena (bit 0)[1]
- Program the PHY register 25h in page 0 as listed in Table 41.
  - Set Restart AN to 1b so the new setting takes place
  - Set SPD_B2B_EN by the B2B Ena read from the PHY_CTRL register
  - Set Disable1000 to 1b disabling GbE mode
  - Set LPLU to the logic OR function of LPLU in non D0a and LPLU in D0a from the PHY_CTRL register
  - Set the SPD_EN by the SPD Ena read from the PHY_CTRL register[1]
  - All other bits should be set to 0b.

Table 41. GbE Disable Setting to the 82566 (Register 25h, Page 0)

Bits	Field Name	Firmware Write Value
15:11	Reserved	0h
10	Restart AN	1b
9	cdc_rst	0b
8	rst_compl	0b
7	SPD_B2B_EN	B2B Ena value from the PHY_CTRL register
6	Disable1000	1b
5	Go Disconnect	0b
4	Link Energy Detect	0b
3	Reserved	0b
2	LPLU	Logic OR function of the LPLU in non D0a and LPLU in D0a bits in the PHY_CTRL register
1	Reserved	0b
0	SPD_EN[1]	SPD Ena bit in the PHY_CTRL register

1. Not applicable to the 82567.

6.2.5.3.2 Master Disable

System software can disable master accesses on the host interface by either clearing the *Master Disable* bit in the CTRL register or by bringing the function into a D3 state. From that time on, the MAC must not issue master accesses for this function. Due to the full-duplex nature of the host interface and the pipelined design, multiple requests

1. Not applicable to the 82567

from/to several internal clients might be pending when the master disable request arrives. The protocol described in this section insures that a client does not issue master requests when entering the D3 state.

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

- *Master Disable* bit in the Device Control Register (CTRL). When the *Master Disable* bit is set, the MAC blocks new master requests. Previous master requests are not affected and proceed normally.
- *Master Enable Status* bits in the Status Register. These bits are cleared by the MAC when the *Master Disable* bit is set and no master requests are pending. When cleared, it indicates that no master requests are issued by this function as long as the *Master Disable* bit is set and the MAC can be set to D3 safely.

Note: The software device driver sets the *Master Disable* bit before it's response to an operating system request to set the MAC to the D3 state. The software device driver might need to implement a watch dog timer while it waits for the Master Enable Status. This prevents long latencies.

6.2.5.4 Dr State

A transition to the Dr state can be initiated by any of the following three instances:

- System power up - Dr state begins with the power up and ends with PCI reset de-assertion.
- Transition from a D0a state - During operation, the system might assert a PCI Reset at any time. In an ACPI system, a system transition to the G2/S5 state causes a transition from D0a to Dr state.
- Transition from a D3 state - The system transitions the MAC into the Dr state by asserting a PCI Reset. This could be a warm reset or a system state transition to the Sx state (D3 cold). The MAC does not distinguish between the two cases if it is connected to auxiliary power.

Note: The system might maintain a PCI Reset (SPXB Reset for **ICH10**) asserted for an arbitrary time. De-asserting (rising edge) of a PCI Reset (SPXB Reset for **ICH10**) causes a transition to D0u state.

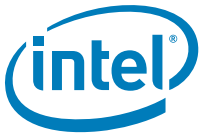
Any WakeUp filter settings that were enabled before entering this reset state are maintained.

While the MAC is in the Dr state, the DMA unit and host CSR space are fed by the MBB clock.

6.2.5.4.1 Entry to Dr State

Dr entry at platform power-up begins by asserting the internal power detection circuit (LAN_RST#). The NVM is read and determines the MAC configuration. If the *APM Enable* bit in the NVM's PCI Init Control Word is set, then APM Wake Up is enabled. PHY and MAC state is determined by the state of manageability and APM Wake. To reduce power consumption, if manageability or APM Wake is enabled, the PHY auto-negotiates to a lower link speed in Dr.

Entry to Dr state from either D0a or D0u is by asserting the PCI Reset signal. An ACPI transition to the G2/S5 state is reflected in the MAC transition from D0a to Dr state. The transition might be orderly (Shut Down option selected), in which case the software device driver might have a chance to intervene. Or, it might be an emergency transition (power button override), in which case, the software device driver is not notified.



To reduce power consumption, if any of manageability, APM Wake, or PCI-PM PME[1] is enabled, the PHY auto-negotiates to a lower link speed upon a D0a to Dr transition.

Note: Transitioning from D3 state to Dr state is done by asserting a PCI Reset signal.

6.2.5.4.2 ICH9/ICH10 FLR Flow

FLR is a transient state that should not last more than 100 ms. The MAC enters the FLR state following a software (most likely from the operating system) write access to the *Initiate FLR* bit in the FLR capability structure within the PCI configuration space. When the MAC completes the FLR sequence, it transitions to the D0u state as follows:

1. Initiate a cycle completion to the write access to the Initiate FLR bit as any other configuration cycle. Note that the steps that follow can be done in parallel.
2. If a read or write request (IO, memory or configuration cycles) arrives while an FLR is in progress, the requests must be master aborted without logging or signaling it as an error. Note that the host should not initiate any target accesses during this phase.
3. If a completion arrives while an FLR is in progress, the completion should be handled as nominal operation.
4. Before the MAC is initialized it:
   a. Sets the *Master Disable* bit in the CTRL register.
   b. Disables all interrupts. This is the same impact as setting the *Interrupt Disable* bit in the Command register in the PCI Configuration space.
   c. Clears PME status and PME enable in the PMCSR register in the PCI Configuration space.
   d. Completes all pending target read requests.
   e. Waits for the completion of all pending master read requests.
5. Initialize the MAC the same as a PCI reset and transition to the D0u state.

6.3 Host Wake-Up

The LAN wake-up mechanisms can be supported from either the MAC or the PHY.

The MAC supports two types of wake up mechanisms:

- Advanced Power Management (APM) Wake Up
- ACPI Power Management Wake Up

For **ICH10**, the indication as to whether wake-up is done from the MAC or the PHY is loaded from the NVM to the *PHY_WAKE* bit in the WUC register. Note that this bit is also configurable for software. Setting this bit to 1b indicates that the PHY supports wake-up and can be enabled and activated to perform the wake-up task

Note: PHY wake-up and ACBS are mutually exclusive and only one of the features can be enabled at a given time or configuration.

Both mechanisms use an internal Host_Wake signal to wake the system up. This signal is connected to the resume wake logic in the ICH. The wake-up steps for **ICH8/ICH9** are as follows:

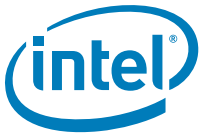
1. Host wake event occurs (no packet is delivered to host).

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

2. PME_STATUS bit is set.
3. Host_Wake signal asserted by host LAN function.
4. System wakes from Sx state to S0 state (if it was in Sx state).
5. The host LAN function transitions to D0 in one of two ways:
   a. Dr (D3cold): PCI Reset is de-asserted.
   b. D3hot: Host writes configuration to D0.
6. The host clears the PME_STATUS bit.
7. Host_Wake signal is de-asserted by host LAN function.

The wake-up steps for **ICH10** are as follows:

1. The host/ME goes into a low power state with wake-up.
2. If the PHY wake is enabled:
   a. The host/ME configures the wake-up capabilities/filters to the PHY.
   b. The host/ME activates wake-up in the PHY.
   c. The host sets the MAC to D3 and the ME to DMoff.
   d. If the ME is at DMon:
   — The ME copies the host wake filters and status (at the end) to the MAC.
   — The ME establishes a secure channel (LinkSec enabled).
   — The ME clears the PHY host active (clears wake-up status).
   — The host wake-up is done from the MAC (the host wake-up packets are encrypted if LinkSec is enabled).
   e. If the ME is at DMoff (or transitions to DMoff while the host is at Sx with wake-up):
   — The ME configures the PHY wake-up filters.
   — The ME activates the host wake-up filters.
   — The ME activates the PHY wake-up if it was active in the MAC.
   — The ME moves ICH10 to DMoff.
   — The PMC powers down the MAC.
   — The PHY waits for a wake-up event while LCI is also powered down.
3. Else, (PHY wake-up disabled):
   a. The host/ME configures the wake-up capabilities to the MAC.
   b. The host/ME activates wake-up in the MAC.
   c. The host sets the MAC to D3 and the ME to DMoff.
   d. The MAC waits for a host wake or ME wake event.
   e. The PHY post wake-up steps LCI active are as follows:
1. The host/ME wake event occurs in the PHY.
2. The PHY sends the appropriate wake indication LCI in-band message.
3. The *Host_Wake/ME_Wake* signal is asserted by the MAC.
4. System/ME wakes.
5. For host wake-up:
   a. The MAC sets the *PME_STATUS* bit.
   b. The host LAN function is transitioned to D0 in one of two ways:



— Dr (D3cold): PCI (SPXB) reset is de-asserted.
— D3hot: the host writes configuration to D0.

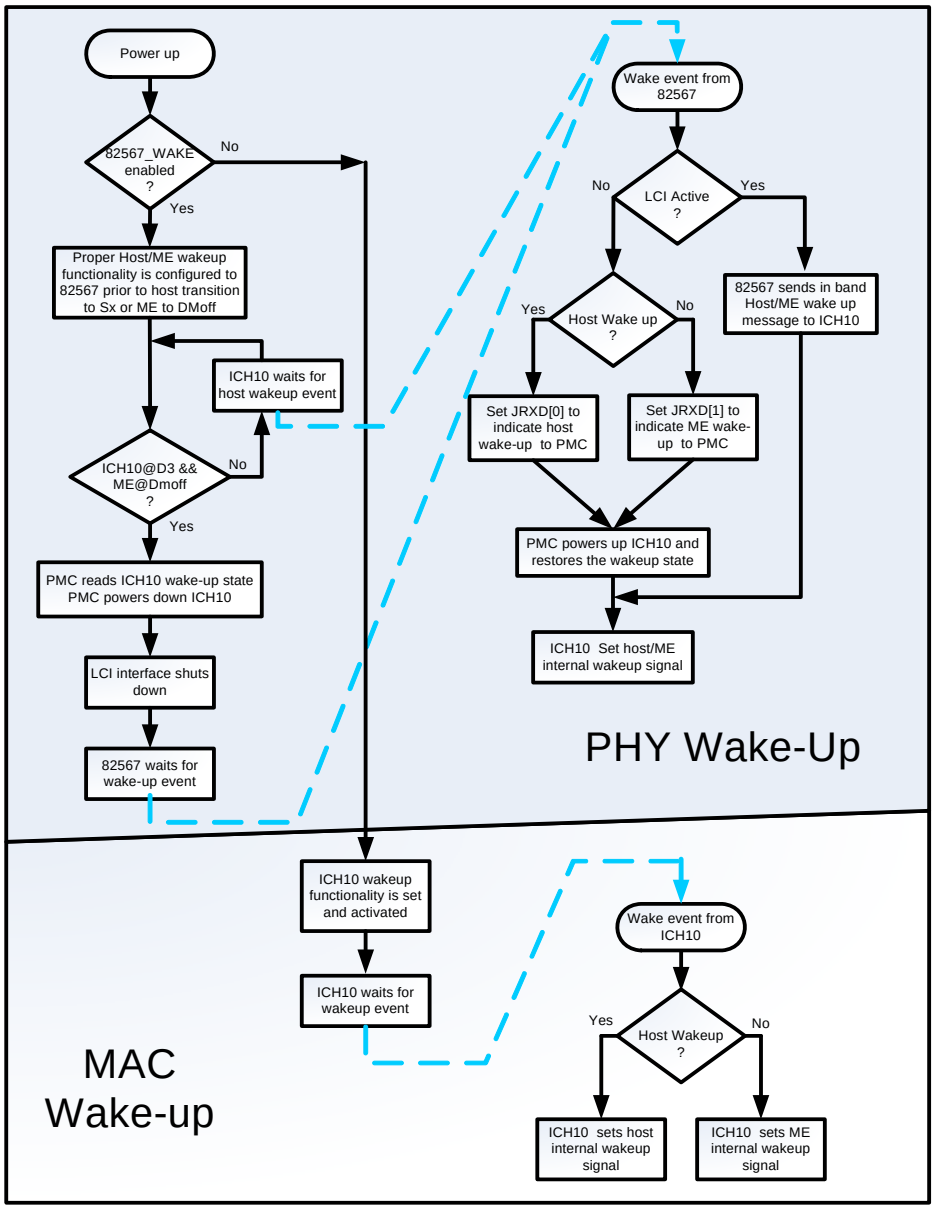
c. The host clears the *PME_STATUS* bit.
d. The *Host_Wake* signal is de-asserted by the MAC.

The PHY post wake-up steps LCI down are as follows:

1. The host/ME wake event occurs in the PHY (packet is discarded).
2. The PHY sends the appropriate wake indication to ICH10 through JRXD[1:0].
3. PMC powers up the MAC.
4. PMC updates the relevant MAC state.
5. The *Host_Wake/ME_Wake* signal is asserted by the MAC.
6. System/ME wakes.
7. For host wake-up:
   a. PMC sets the *PME_STATUS* bit.
   b. The host LAN function is transitioned to D0 in one of two ways:
   — Dr (D3cold): PCI (SPXB) reset is de-asserted.
   — D3hot: the host writes the configuration to D0.
   c. The host clears the *PME_STATUS* bit.
   d. The *Host_Wake* signal is de-asserted by the MAC.

The MAC post wake-up steps are as follows:

1. The host/ME wWake event occurs in the MAC (packet is discarded).
2. The *Host_Wake/ME_Wake* signal is asserted by the MAC.
3. System/ME wakes.
4. For host wake-up:
   a. The *PME_STATUS* bit is set.
   b. The host LAN function is transitioned to D0 in one of two ways:
   — Dr (D3cold): PCI (SPXB) reset is de-asserted.
   — D3hot: the host writes the configuration to D0.
   c. The host clears the *PME_STATUS* bit.
   d. The *Host_Wake* signal is de-asserted by the MAC.



6.3.1 Advanced Power Management Wakeup

Advanced Power Management Wake Up or APM Wake Up is a feature that receives a broadcast or unicast packet with an explicit data pattern and responds by asserting a wake-up signal to the system. In earlier generations, this was accomplished by using a special signal that ran across a cable to a defined connector on the motherboard. The MAC asserted the signal for approximately 50 ms to signal a wake up. If the MAC is configured appropriately, it uses an in-band PM_PME message to achieve this.

On power up, the MAC reads the *APM Enable* bits from the NVM Initialization Control Word 2 into the *APM Enable* (APME) bits of the Wakeup Control Register (WUC). These bits enable APM Wake Up.

When APM Wake Up is enabled, the MAC checks all incoming packets for Magic Packets*. Once the MAC receives a matching Magic Packet, it:

- Sets the *Magic Packet Received* bit in the Wake Up Status (WUS) register.
- Sets the *PME_Status* bit in the PMCSR and asserts the internal Host_Wake signal.

APM Wake Up is supported in all power states and only disabled if a subsequent NVM read results in the *APM Wake Up* bit being cleared or software explicitly writes a 0b to the *APM Wake Up* (APM) bit of the WUC register.

6.3.1.1 Link Status Change

When the *LSCWO* bit is set, wake is generated if all of the following conditions are met:

- APM Wakeup is enabled (*APME* bit is set in the WUC register)
- The *LSCWE* bit is set in the WUC register
- Link status change is detected

On detecting a link status change, the MAC sets the *PME_Status* bit in the PMCSR and issues a PM_PME message. The *Link Status Changed* (LNKC) bit in the Wake Up Status Register (WUS) is set.

When the *LSCWO* bit is set, wake is never generated on link status change if either APM Wakeup is disabled or the *LSCWE* bit is cleared. In this case, the *LNKC* bit in the Wake Up Filter Control Register reads as 0b, independent of the value written to it.

6.3.2 ACPI Power Management Wakeup

The MAC supports ACPI Power Management based wakeups. 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 software device driver programs the WUFC register indicating the wake-up packets. It also supplies the necessary data to the IPv4 Address Table (IP4AT) and the Flexible Filter Mask Table (FFMT), Flexible Filter Length Table (FFLT), and the Flexible Filter Value Table (FFVT). It also sets the *Link Status Change* (LNKC) bit in the Wake Up Filter Control Register (WUFC) to cause wake up when the link changes state.
- At configuration time, the operating system writes a 1b to the *PME_EN* bit of the PMCSR.

After enabling wake up, the operating system typically writes 11b to the lower two bits of the PMCSR placing the MAC into low-power mode.

After wake up is enabled, the MAC monitors the incoming packets. First, it filters the packets according to its standard address filtering method and 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 MAC:

- Sets the *PME_Status* bit in the PMCSR.
- If the *PME_EN* bit in the PMCSR is set, asserts the internal Host_Wake signal.
- Sets one or more of the received bits in the Wake Up Status Register (WUS). (The MAC sets more than one bit if a packet matches more than one filter.)

If enabled, a link state change wake up causes similar results, setting PME_Status, asserting the internal Host_Wake signal and setting the *Link Status Changed* (LNKC) bit in the Wake Up Status Register (WUS) when the link goes up or down.

The internal Host_Wake signal remains asserted until the operating system either writes a 1b to the *PME_Status* bit of the PMCSR or writes a 0b to the *PME_EN* bit.

After receiving a wake-up packet, the MAC ignores any subsequent wake-up packets until the software device driver clears all of the received bits in the Wake Up Status Register (WUS). It also ignores link change events until the software device driver clears the *Link Status Change* (LNKC) bit in the WUS.

6.3.3 Wake-Up Packets

The MAC supports various wakeup packets using two types of filters:

- Pre-defined filters
- Flexible filters

Each of these filters are enabled if the corresponding bit in the Wake Up Filter Control Register (WUFC) is set to 1b. If the wake-up packet passes one of the manageability filters (enabled in the MANC register) then wake-up of the system also depends on the *WUFC.NoTCO* bit being inactive.

6.3.3.1 Pre-Defined Filters

The following packets are supported by the MAC pre-defined filters:

- Directed Packet (including exact, multicast, 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 Wake Up Filter Control Register (WUFC) is set to 1b.

The explanation of each filter includes a table detailing which bytes at which offsets are compared to determine if the packet passes the filter. Both VLAN frames and LLC/Snap frames can increase the given offsets if they are present.

6.3.3.1.1 Directed Exact Packet

The MAC generates a wake-up event upon reception of any packet whose destination address matches one of the 7 valid programmed Receive Addresses if the *Directed Exact Wake Up Enable* bit is set in the Wake Up Filter Control Register (WUFC.EX).

Offset	Number of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Compare	Match any pre-programmed address as defined in the Receive Address (6:0)

6.3.3.1.2 Directed Multicast Packet

For multicast packets, the upper bits of the destination address in the incoming packet index a bit vector. This is the Multicast Table Array that indicates whether to accept the packet. If the *Directed Multicast Wake Up Enable* bit is set in the Wake Up Filter Control Register (WUFC.MC) and the indexed bit in the vector is 1b, then the MAC generates a wake-up event. The exact bits used in the comparison are programmed by software in the *Multicast Offset* field of the Receive Control Register (RCTL.MO).

Offset	Number of Bytes	Field	Value	Action
0	6	Destination Address		Compare

6.3.3.1.3 Broadcast

If the *Broadcast Wake Up Enable* bit in the Wake Up Filter Control Register (WUFC.BC) is set, the MAC generates a wake-up event when it receives a broadcast packet.

Offset	Number of Bytes	Field	Value	Action	Comment
0	6	Destination Address	FF*6	Compare	

6.3.3.1.4 Magic Packet*

The definition for a Magic Packet* can be located at http://www.amd.com/products/npd/overview/20212.html. Magic Packets are defined in the text that follows.

The MAC expects the destination address to either:

- Be the broadcast address (FF.FF.FF.FF.FF.FF).
- Match the value in the Receive Address Register 0 (RAH0, RAL0). This is initially loaded from the NVM but might be changed by the software device driver.
- Match any other address filtering enabled by the software device driver.

The MAC looks for the contents of Receive Address Register 0 (RAH0, RAL0) as the embedded IEEE address. It will consider any non-FFh byte after a series of at least 6 FFs to be the start of the IEEE address for comparison purposes. For example, it catches the case of 7 FFs followed by the IEEE address. As soon as one of the first 96 bytes after a string of FFs does not match, it continues to search for anther set of at least 6 FFs followed by the 16 copies of the IEEE address later in the packet. It should be noted that this definition precludes the first byte of the destination address from equaling FF.

If the packet destination address met one of the three criteria previously listed, the MAC searches for 16 repetitions of the same destination address in the packet's data field. Those 16 repetitions must be preceded by (in the data field) at least 6 bytes of FFh, which act as a synchronization stream. If the destination address is NOT the broadcast address (FF.FF.FF.FF.FF.FF), the MAC assumes that the first non-FFh byte following at least 6 FFh bytes is the first byte of the possible matching destination address. If the 96 bytes following the last FFh are 16 repetitions of the destination address, the MAC accepts the packet as a valid wake-up magic packet. Note that this definition precludes the first byte of the destination address from being FFh. If the destination address of the packet IS the broadcast address, and 96 bytes of FFh follow at least 6 bytes of FFh, the MAC also accepts it as a valid wake-up magic packet. If fewer than 96 bytes of FFh follow 6 leading FFh bytes, it is not yet considered a match.

A Magic Packet's destination address must match the address filtering enabled in the configuration registers with the exception that broadcast packets are considered to match even if the *Broadcast Accept* bit of the Receive Control Register (RCTL.BAM) is 0b. If APM Wake Up is enabled in the NVM, the MAC starts with the Receive Address Register 0 (RAH0, RAL0) loaded from the NVM. This enables the MAC to accept packets with the matching IEEE address before the software device driver is available.

Offset	Number of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Compare	MAC header processed by main address filter.
6	6	Source Address		Skip	
12	8	Possible LLC/SNAP Header		Skip	
12	4	Possible VLAN Tag		Skip	
12	4	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 Register (RCTL.BAM) is 0b differs from older generation Intel Gigabit Ethernet components. Previously, these devices initialized RCTL.BAM to 1b if APM was enabled in the NVM but then required that bit to equal 1b to accept broadcast Magic Packets, unless broadcast packets passed another perfect or multicast filter.

6.3.3.1.5 ARP/IPv4 Request Packet

The MAC supports the reception of ARP request packets for wakeup if the *ARP* bit is set in the Wake Up Filter Control Register (WUFC). Four IPv4 addresses are supported and are programmed in the IPv4 Address Table (IP4AT). A successfully matched packet must contain a broadcast MAC address, a Protocol Type of 0806h, an ARP OPCODE of 01h, and one of the four programmed IPv4 addresses. The MAC also handles ARP request packets with VLAN tagging on both Ethernet II and Ethernet SNAP types.

Offset	Number of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Compare	MAC header processed by main address filter.
6	6	Source Address		Skip	
12	8	Possible LLC/SNAP Header		Skip	
12	4	Possible VLAN Tag		Skip	
12	2	Type	0806h	Compare	ARP
14	2	Hardware Type	0001h	Compare	
16	2	Protocol Type	0800h	Compare	
18	1	Hardware Size	06h	Compare	
19	1	Protocol Address Length	04h	Compare	
20	2	Operation	0001h	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	May match any of 4 values in IP4AT.

6.3.3.1.6 Directed IPv4 Packet

The MAC supports the reception of Directed IPv4 packets for wakeup if the *IPv4* bit is set in the Wake Up Filter Control Register (WUFC). Three IPv4 addresses are supported and are programmed in the IPv4 Address Table (IP4AT). A successfully matched packet must contain the station MAC address, a Protocol Type of 0800h, and one of the four programmed IPv4 addresses. The MAC also handles Directed IPv4 packets with VLAN tagging on both Ethernet II and Ethernet SNAP types.

Offset	Number of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Compare	MAC Header processed by main address filter.
6	6	Source Address		Skip	
12	8	Possible LLC/SNAP Header		Skip	
12	4	Possible VLAN Tag		Skip	
12	2	Type	0800h	Compare	IP
14	1	Version/ HDR length	4Xh	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	
23	1	Protocol	-	Ignore	
24	2	Header Checksum	-	Ignore	
26	4	Source IP Address	-	Ignore	
30	4	Destination IP Address	IP4AT	Compare	May match any of 4 values in IP4AT.

6.3.3.1.7 Directed IPv6 Packet

The MAC supports reception of Directed IPv6 packets for wakeup if the *IPv6* bit is set in the Wake Up Filter Control Register (WUFC). One IPv6 address is supported and it is programmed in the IPv6 Address Table (IP6AT). A successfully matched packet must contain the station MAC address, a Protocol Type of 0800h, and the programmed IPv6 address. The MAC also handles Directed IPv6 packets with VLAN tagging on both Ethernet II and Ethernet SNAP types.

Offset	Number of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Compare	MAC Header processed by main address filter.
6	6	Source Address		Skip	
12	8	Possible LLC/SNAP Header		Skip	
12	4	Possible VLAN Tag		Skip	
12	2	Type	0800h	Compare	IP
14	1	Version / Priority	6Xh	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.

6.3.3.1.8 Flexible Filter

The MAC supports a total of four flexible filters. Each filter can be configured to recognize any arbitrary pattern within the first 128 bytes of the packet. The flexible filter is configured by software programming mask values into the Flexible Filter Mask Table (FFMT), required values into the Flexible Filter Value Table (FFVT), and the minimum packet length into the Flexible Filter Length Table (FFLT). These contain separate values for each filter. Software must also enable the filter in the Wake Up Filter Control Register (WUFC) and enable the overall wake up functionality must be enabled by setting the *PME_En* bit (*ACPI_En* bit for the **82567**) in the PMCSR or the WUC register.

When flexible filtering is enabled, the flexible filters scan incoming packets for a match. If the filter encounters any byte in the packet where the mask bit is one and the byte does not match the byte programmed in the Flexible Filter Value Table (FFVT), then the filter will fail 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 1b beyond the required length.

The following packets listed in subsequent sections are for reference purposes only. The flexible filter can be used to filter these packets.

6.3.3.1.9 IPX Diagnostic Responder Request Packet

An IPX Diagnostic Responder Request Packet must contain a valid MAC address, a Protocol Type of 8137h, and an IPX Diagnostic Socket of 0456h. It might include LLC/SNAP headers and VLAN tags. Since filtering this packet relies on the flexible filters, which use offsets directly specified by the operating system, the operating system must account for the extra offset due to the LLC/SNAP headers and VLAN tags.

Offset	Number of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Compare	
6	6	Source Address		Skip	
12	8	Possible LLC/SNAP Header		Skip	
12	4	Possible VLAN Tag		Skip	
12	2	Type	8137h	Compare	IPX
14	16	Some IPX Data	-	Ignore	
30	2	IPX Diagnostic Socket	0456h	Compare	

6.3.3.1.10 Directed IPX Packet

A valid Directed IPX Packet contains the station MAC address, a Protocol Type of 8137h, and an IPX Node Address equal to the station MAC address. It may include LLC/SNAP headers and VLAN tags. Since filtering this packet relies on the flexible filters, which use offsets directly specified by the operating system, the operating system must account for the extra offset due to the LLC/SNAP headers and VLAN tags.

Offset	Number of Bytes	Field	Value	Action	Comment
0	6	Destination Address		Compare	MAC Header processed by main address filter.
6	6	Source Address		Skip	
12	8	Possible LLC/SNAP Header		Skip	
12	4	Possible VLAN Tag		Skip	
12	2	Type	8137h	Compare	IPX.
14	10	Some IPX Data	-	Ignore	
24	6	IPX Node Address	Receive Address 0	Compare	Must match Receive Address 0.

6.3.3.1.11 IPv6 Neighbor Solicitation Message Filter

In Ipv6, a Neighbor Solicitation Message packet (type 135) is used for address resolution. A flexible filter can be used to check for a Neighborhood Solicitation Message packet.

Note: The fields checked for detection of a Neighborhood Solicitation Message packet are type, code, and addresses.

6.4 Power Management Elements

The MAC controls some power management aspects of the PHY. These aspects are discussed in the following sections.

6.4.1 PHY Link Speed Control

Normal PHY speed negotiation drives to establish a link at the highest possible speed. The PHY supports an additional mode of operation, where the PHY drives to establish a link at a low speed. The link-up process enables a link to come up at the lowest possible speed in cases where power is more important than performance. Different behavior is defined for the D0a state and the other non-D0a states. The PHY speed negotiation is defined by the *LPLU* bit fields in the PHY_CTRL register as follows:

- The LPLU in D0a bit enables the PHY to establish link at low speed at all power states.
- The LPLU in non D0a bit enables the PHY to establish link at low speed when the MAC is not in the D0a state

The MAC and its PHY consume significant power when operating at GbE speed. In an effort to reduce power for particular platforms, GbE mode can be disabled. The PHY speed negotiation is defined by the *GbE disable* bit fields in the PHY_CTRL register as follows:

- The *Global GbE Disable* bit disables 1000 Mb/s operation at all power states.
- The *GbE Disable in non D0a* bit disables 1000 Mb/s operation when the MAC is not in the D0a state

6.4.1.1 82566/82567-Specific PHY Link Speed Control

The LPLU and GbE disable fields are sent from the MAC to the PHY when link speed should be set by the MAC. The MAC accesses the PHY via the MDIC-to-PHY register 25 in bank 0 while using PHY address 00001b. Accesses are initiated as follows:

- Following a D0a to D3 state transition, the MAC uses the GbE disable and LPLU enable in non D0a if the ME enables the host to initialize the PHY (as defined by the RSPCIPHY bit in the FWSM register) and either the GbE enablement or LPLU setting is different in D0a and non D0a.
- Following a D0u to D0a state transition, the MAC uses the GbE disable and LPLU in D0a if the ME enables the host to initialize the PHY (as defined by the RSPCIPHY bit in the FWSM register) and either the GbE enablement or LPLU setting is different in D0a and non-D0a.

Note: Following any event that causes a PHY reset or PHY software reset, the MAC initializes the PHY as required by the extended configuration content in the NVM. As part of that initialization, the NVM image must include setting the *Restart AN* bit in the PHY.

Before the MAC accesses the PHY via MDIC register, it sets first the *MDIO HW Ownership* bit and wait until both software and firmware completed their accesses to the shared resources (including the PHY) by clearing the *SWFLASG* and *HWFLAG* bits.

6.4.1.2 82562V-Specific PHY Link Speed Control

LPLU and GbE disable are not relevant to the 10/100 Mb/s PHY.

Note: The Low-Power Link-Up (LPLU) feature previously described should be disabled (in both D0a state and non-D0a states) when user advertisement is anything other than 10/100/1000 Mb/s (all three). This is to avoid reaching (through the LPLU procedure) a link speed that is not advertised by the programmer.

Table 42 lists the link speed as a function of the power management state (provided by the MAC), link speed control, and GbE speed enabling (provided by the MAC).

Table 42. Power Management PHY Speed Control

Power MNG State	LPLU in Non-D0a Setting	LPLU in D0a Setting	Outcome LPLU Setting to the PHY
Non-D0a	0b	0b	0b
Non-D0a	1b	0b	1b
D0a	X	0b	0b
Any State	X (note 1)	1b	1b
Power MNG State	**GbE Disable in Non-D0a**	**GbE Disable in D0a**	**Outcome GbE Disable Setting to the PHY**
Non-D0a	0b	0b	0b
Non-D0a	1b	0b	1b
D0a	X	0b	0b
Any State	X[1]	1b	1b

LPLU	GbE Disable	PHY Speed Negotiation
0b	0b	PHY negotiates to highest speed advertised (normal operation)
0b	1b	PHY negotiates to highest speed advertised excluding 1000 Mb/s
1b	0b	PHY goes through Low Power Link Up (LPLU) procedure, starting with advertised values
1	1	PHY goes through LPLU procedure, starting with advertised values. Does not advertise 1000

1. LPLU or GbE Disable might not be set for D0a if it is not set for non-D0a.

6.4.2 PHY Power States

The PHY functionality and power state depends on the PCI states of the MAC and its setting. By default, setting maximum performance is enabled when the PCI function is in the D0a state. In non D0a state, the PHY is set to power saving mode, if LAN functionality is required, and is set to PHY power down if LAN connectivity is not required by PCI devices. Table 43 lists the PHY mode of operation as a function of PCI power states, WoL functionality, and device setting.

The PHY should also be set to power down when the LAN is disabled (see Section 6.6).

Note: For improved power saving in the MAC during PCI power down, the Dynamic Clock Gating should be enabled in the CTRL_EXT.DynCK as well.

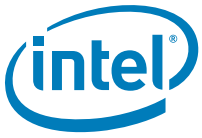


Table 43. PHY Power Operation Modes vs. PCI States

PCI Device Power State	Host Wake Enable	External PHY
D0a	X	Full Active (1)
D0u	X	LPLU, GbE Dis (1)
Dr, D3	Yes	LPLU, GbE Dis (1)
Dr, D3	No	LPLU, GbE Dis (1)
		Power Down (2)

Note: PHY operation (LPLU and GbE disable) depends on the setting of the PHY_CTRL register and PCI device power state as listed in Table 43.

Note: PHY power down is enabled by the *CTRL_EXT.PHYPDEN* bit. The PHY is set to the power down state by an in-band message or by setting the *LANPHYPC* signal, if enabled.

Note: GbE disable in non-D0a must be set.

The PHY can also be set to power down by explicit access to the *PHY Power Down* bit in the CTRL register. This bit does not depend on the state of FWSM.RSPCIPHY. If ME is functional and host software is using this bit to set the PHY to power down, it should indicate it first to ME in order to avoid conflicts.

The PHY unit in the PHY can be set to the power down by direct MDIO command. Software or firmware might set the *Power Down* bit in PHY register 0 (bank 0) to access the MDIC register. However, this method is not recommended because the ME cannot gate it.

6.4.2.1 Exiting 82566/82567 Power Down

Exiting an **82566**/**82567** power down state is possible only by initiating a reset on the LANRSTSYNC signal to the **82566**/**82567**. The MAC initiates the reset to the **82566**/**82567** using one of the following:

- The **82566**/**82567** was in power down due to explicit setting of CTRL_EXT.PHYPDEN. In this case software or firmware should clear the *CTRL_EXT.PHYPDEN* bit and then set the *CTRL.LCD_RST* bit to generate an **82566**/**82567** reset.
- The **82566**/**82567** was in power down since it was in DMoff and non D0a without WoL functionality (enabled by the CTRL_EXT.PHYPDEN). Following PCI to enter D0a by software (operating system), the MAC initiates an **82566**/**82567** reset.

The event flow for resetting the **82566** is very similar to the power-up sequence shown in Figure 31.

Exiting from **82566**/**82567** power down is done by direct MDIO access by clearing the *Power Down* bit in the PHY at register 0 (bank 0). Event flow is shown in Figure 31.

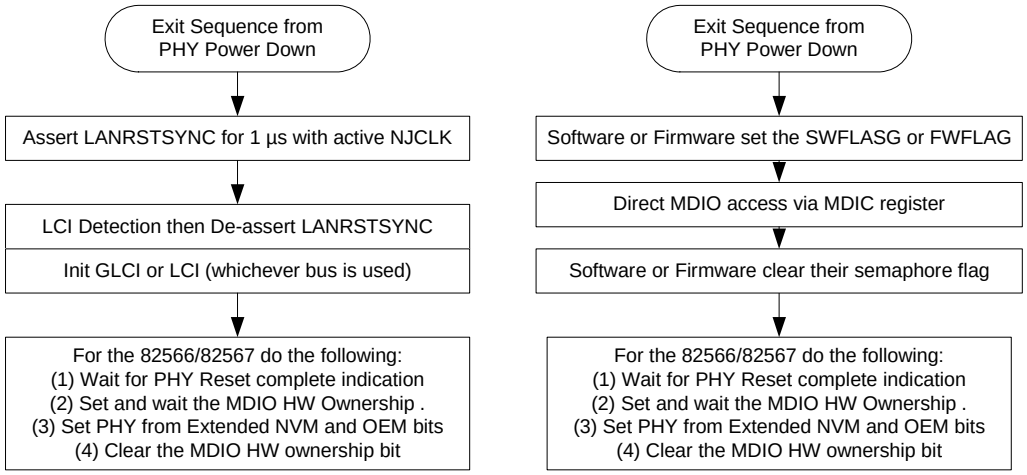


Figure 31. 82566/82567 and PHY Power Up Sequence

6.5 82566 Auto Connect Battery Saver (ACBS)

ACBS is a power saving feature that enables the **82566** to be powered down when no link is present while the system is in the S0 state. This power saving mode, if enabled, is entered upon link loss due to a cable disconnect. Note that the **82566** can be completely powered off. The voltage rails of the **82566** can be controlled through the LAN_PHY POWER CONTROL (LANPHYPC) signal or the 1.8V and 1.0V supplies can be controlled through the LCI interface. When a link partner is reconnected, the ENERGY_DETECT signal indicates to the MAC that power to the **82566** must be restored.

While in the D0 state, ACBS is controlled by the software device driver. The software device driver is responsible for initiating the 82566 transitions into ACBS mode and then the MAC is responsible for detecting that the link partner is present.

In D3 state (either D3 hot or D3 cold), the ACBS feature might be controlled autonomously by the MAC. No software or firmware intervention is required. ACBS, while in the Sx state, is enabled by the *DisDSPD* bit in the FEXTNVM register.

This feature requires an external link energy detection circuit. In addition, the ENERGY_DETECT_GP13 pin should be configured to the native mode by the GPIO_USE_SEL[31:0] register.

6.5.1 ACBS Event Flow at D0 State

The complete event flow for software and hardware is shown in Figure 32. The major events are described in the following sections.

6.5.1.1 Software Flow (Enter ACBS Mode)

ACBS is controlled by the software device driver. The software device driver is responsible for initiating the **82566** transition into ACBS mode. The MAC is responsible for detecting that ACBS mode should be terminated due to link partner availability. The event flow is as follows:

1. The software device driver is notified by the Link Status Change (LSC) interrupt that the link has been lost.

2. The software device driver then checks that the reason for the LSC is that the LAN cable was disconnected (no link partner) by reading the FEXT register cable disconnect indication.
3. If ACBS is enabled in the NVM, the software device driver might proceed and assert the *PHY_CTRL.DSPDA* bit that transitions the **82566** into the ACBS mode.
4. After asserting the *DSPDA* bit, the software device driver should verify that the cable disconnect in the FEXT register is still asserted. Otherwise, it should suspend ACBS mode by performing a software and **82566** reset.

Note: The software device driver must first check that the Auto Connect Battery Saver Supported bit in the FEXTNVM is set to 1b. If this bit is not set, initiating the ACBS flow might damage the **82566**.

Note: If a software reset is issued during ACBS mode then the software driver must also issue a PHY reset in order to properly exit ACBS mode.

Note: Entering ACBS mode, when LANPHYPC is not used, requires setting *Enable VR Shutdown in Power Down* in the PHY.

6.5.1.2 Software Flow (During ACBS Mode)

While the **82566** is in ACBS mode, the software device driver should poll both the cable disconnect indication and the LANPHYPC indication in the FEXT register. If the software device driver detects that the **82566** has been powered up by the software device driver and a link has not been established, then the software device driver should repeat the Enter ACBS Flow.

6.5.1.3 D0 Hardware Flow (Entering ACBS Mode)

Following the assertion of the PHY_CTRL.DSPDA by the software device driver, the MAC sets the **82566** to a power down state by one of the following:

1. Initiate and power down the in-band message to the **82566**.
2. Assert the LANPHYPC signal if enabled by the strapping options defined in the *ICH8/ICH9 Design Guide* and the appropriate GPIO is set to native mode. If the LANPHYPC signal is enabled, the MAC turns off the **82566**'s power source(s) before the in-band power down message is sent out.

6.5.1.4 D0 Hardware Flow (During ACBS Mode)

After the MAC enters ACBS mode, the LCI link to the PHY is kept at a low level. LANRSTSYNC is driven to low while the LANRXD[2:0] and LANTXD[2:0] signals are kept low by internal pull down resistors.

6.5.1.5 D0 Hardware Flow (Exit ACBS Mode)

The MAC recognizes that a link partner has been re-connected by sensing a level transition on the ENERGY_DETECT_GP13 input. The MAC senses a level transition if the signal on the ENERGY_DETECT_GP13 input is high for at least two rising edges of a 15 MHz clock. It then clears the *PHY_CTRL.DSPD* bit. As a result, the LANPHYPC is de-asserted (if implemented) to resume power to the **82566**. The MAC then issues an **82566** reset. Following the **82566** reset, the **82566** and the MAC proceeds with nominal initialization flow.

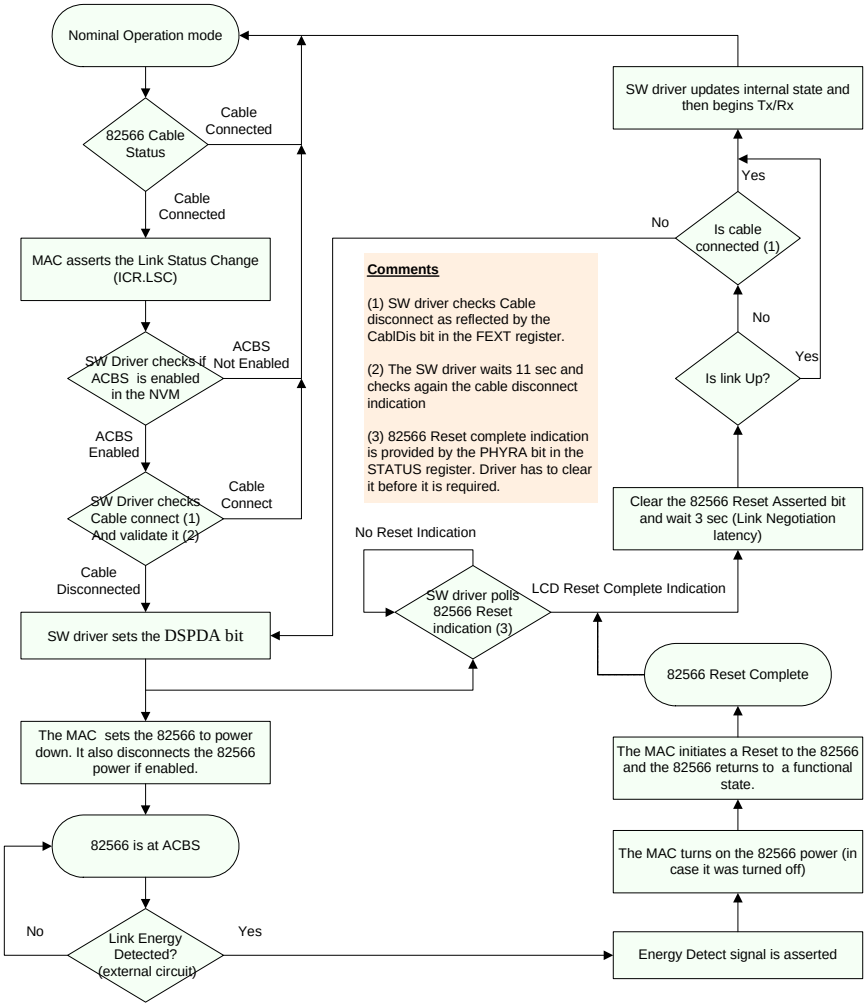


Figure 32. ACBS Flow

6.5.1.6 ACBS Event Flow at D3 State

During a D3 state, the software device driver has no control over the **82566** while the system might be in the Sx power state. The MAC can enter autonomously to ACBS mode in a D3 state if it is not gated by the *DisDSPD* field in the FEXTNVM register. Once the MAC is at the D3 state and the PHY reports no link energy, the MAC initiates the ACBS flow (both conditions should be met while the *DisDSPD* bit is cleared):

The MAC initiates a power down in-band message to the PHY and asserts the LANPHYPC signal if enabled by the strapping options defined in the *82566 Datasheet* and the appropriate GPIO is set to native mode. If the LANPHYPC signal is enabled, the MAC turns off the PHY power source(s) before the in-band power down message is sent out.

During ACBS mode, the LCI link to the PHY is kept at the low level: LANRSTSYNC is driven to low while the LANRXD[2:0] and LANTXD[2:0] are kept at low by internal pull-down resistors.

The MAC recognizes that a link partner has been re-connected by sensing a level transition on the ENERGY_DETECT_GP13 input. As a result, the LANPHYPC is de-asserted (if implemented) resuming power to the PHY. The MAC then issues a PHY reset. Following the PHY reset, the PHY and the MAC proceeds with the normal initialization flow

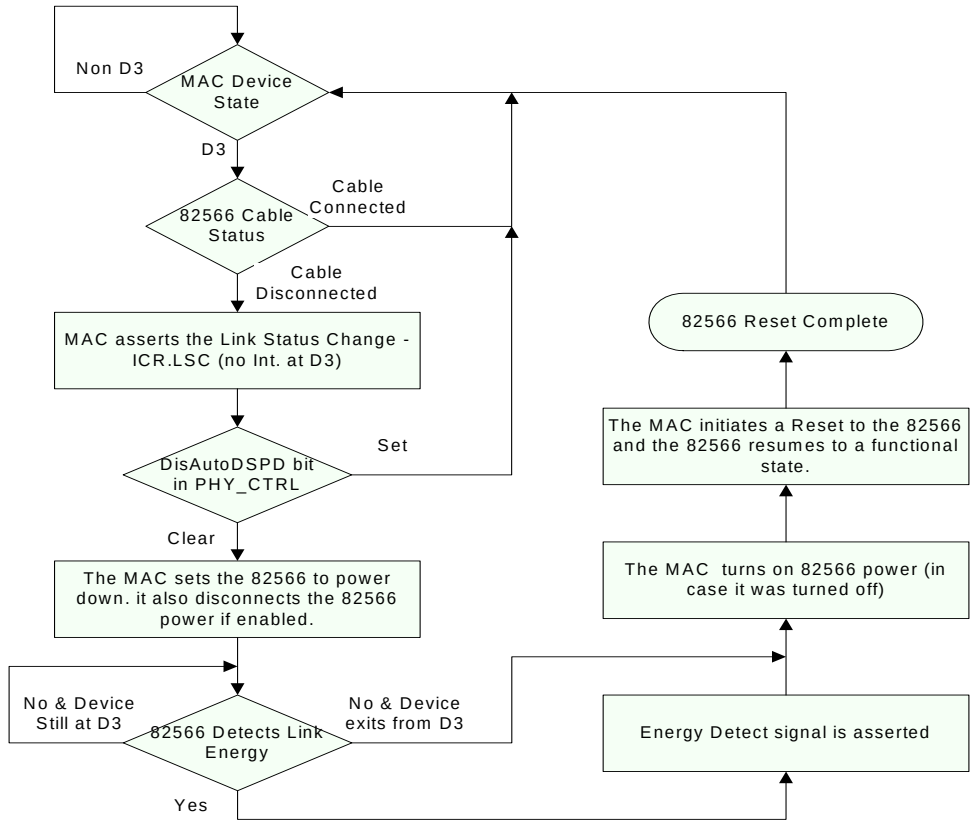


Figure 33. ACBS Flow in D3 State

6.6 LAN Disable

In some applications the MAC is not required and can be disabled. The LAN function can be disabled by the *LANDisable* bit in the Function Disable SUS Well (Backed Up Control RTC Well register for **ICH9**) that resides in the ICH core. This bit is read-only when the LAN Disable fuse is set and read/write when the LAN is enabled by the fuse. When the *LANDisable* bit is set, the MAC disappears from the system:

- It does not respond to any target host (configuration/memory/I/O accesses for **ICH9**) on the PCI space.
- It does not initiate any master accesses including NVM accesses (by the PPW unit). For the **ICH9**, the MAC must finish its NVM reading at least once following LAN_RST# asserted.
- If enabled on the **ICH9**, the PHY is powered down by an inband power down message and by LANPHYPC.

- For **ICH10**, the SPxB clock and MBB clock are gated as well as most of the internal clocks as follows: dma_clk, wake_dma_clk, gpt_clk, mac_clk, func_clk, ppw_clk.
- For **ICH8**, the LANRSTSYNC signal to the external PHY is kept at its static state.

Figure 34 shows the LAN disabling flow while operating with an 82566 or 82567 as a PHY.

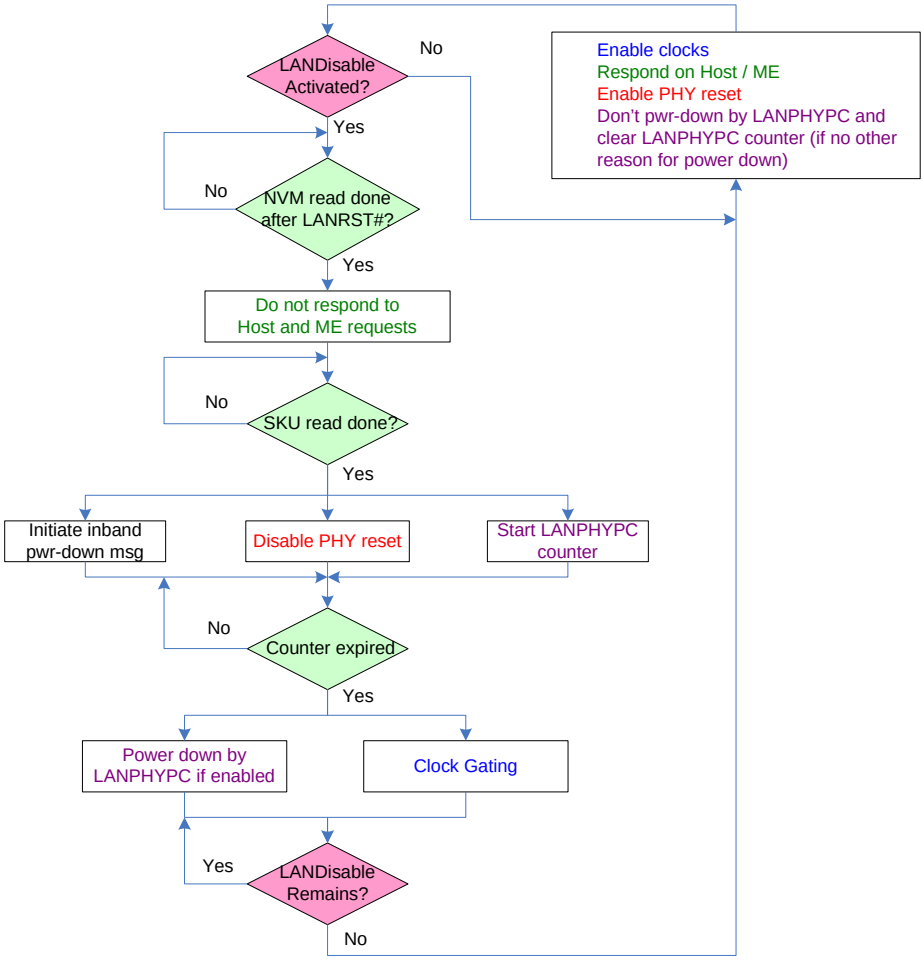


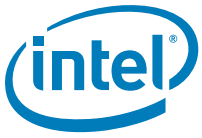
Figure 34. LAN Disabling Flow (Operating with an 82566/82567)

6.6.1 LAN Clock Enable

The LAN Clock enable is inactive when the *LANDisable* bit is set and the *LAN Static Clock Gating Enable* bit in the Power Reduction Control register is set. When inactive, the LAN Clock enable gates the following LAN clocks:

- bb_clk; mb_clk and mosc_clk.

For power saving, software (BIOS) should first set the PHY to power down. When operating with the **82562V** as the PHY, it is expected that the BIOS activates its LAN Disable pin. This can be done by using a GPIO in the ICH or any other device.



Note: This page intentionally left blank.

7.0 Ethernet Interface

7.1 Introduction

The MAC provides a complete CSMA/CD function supporting IEEE 802.3 (10 Mb/s), 802.3u (100 Mb/s), 802.3z and 802.3ab (1000 Mb/s) implementations. It performs all of the functions required for transmission, reception, and collision handling called out in the standards.

Note: The MAC is optimized for full-duplex operation in 1000 Mb/s mode. Half-duplex 1000 Mb/s operation is NOT supported and is not recommended.

7.1.1 MAC/PHY Connect Interface

The MAC and PHY communicate through an LAN Connect Interface (LCI) that can be configured for either 1000 Mb/s operation or 10/100 Mb/s mode of operation. All MAC configurations are performed using device control registers mapped into system memory or I/O space. The PHY is configured via the LCI interface.

The following interface modes are listed in Table 44 and are detailed in the following sections.

Table 44. MAC/PHY Interface Modes

Mode	Interfaces Active	PHY Connections
Legacy 10/100	LCI (1.1)	**82562V**
Normal 10/100/1000	LCI (2.0) and GLCI (2.0)	**82566**/**82567**
Low Pin Count 10/100/1000	GLCI (2.0)	**82566**/**82567**

Table 45 lists the LCI/GLCI states as a function of the PHY power state and the link speed.

Table 45. PHY Power State/Link Speed Functionality

Function	LCI Mode (82562V)		GLCI Mode (82566 & 82567)	
Mode	LCI	GLCI	LCI	GLCI
PHY Power Down	5 MHz JCLK	Power Down	Power Down	Power Down
10/100 Idle or Link Disc	Active Bus	Power Down	Active Bus	Power Down
10/100 Tx/Rx	Active Bus	Power Down	Active Bus	Power Down
GbE Idle	N/A	N/A	Active Bus, MNG Only	N/A
GbE Tx/Rx	N/A	N/A	Active Bus, MNG Only	Active Bus

7.1.2 LCI Interface

7.1.2.1 NJCLK Signal

LCI enables operation at 50 MHz for 100 Mb/s and 25 MHz or 5 MHz for 10 Mb/s operation. ICH8/ICH9 and its PHYs (**82566**/**82567** and **82562V**) operate at 50 MHz for 100 Mb/s and 5 MHz for 10 Mb/s operation.

7.1.2.2 LCI PHY-to-MAC In-Band Signals

Following are the LCI 2.0 PHY-to-MAC in-band signals and the MAC response.

Signals supported by the MAC:

- Rx DV - Indicates the Rx DV signal as defined in the IEEE 802.3 spec. This signal goes through the elasticity buffer synchronized to the data.
- JCRS - Carrier Sense Indication. The signal bypasses the elasticity buffer in both the PHY and the MAC to minimize half-duplex latencies.
- Rx / Tx cyc - Synchronization signals from the PHY to the MAC.
- Mdout - Part of the management frames.
- Rx Er - Any packet that is received with an error indication from the PHY is marked as a bad packet. In addition, the RXERRC register is counted if it passes the MAC address filtering.
- Speed 1 / 2 - Indicates the LAN link speed.
- Duplex - Indicates half-duplex/full-duplex modes.
- Link - Indicates that link is connected with energy detected and technology is established.
- Cable Disc - When asserted, indicates that the Ethernet link is unplugged, senses no energy for long enough duration (between 4-6 seconds), and the PHY turned off its transmitters for power saving. Cable disconnect indication is used by the **82566**/**82567**. This signal is ignored when operating with the **82562V**.

- SQL - When the squelch signal is de-asserted, it indicates that the Ethernet link is unplugged, senses no energy for long enough duration (between 4-6 seconds), and the PHY turned off its transmitters for power saving. Squelch indication is used by the **82562V**. This signal is ignored when operating with the **82566**/**82567**.
- RST-COMP - Reset complete indication is supported only by the **82566**/**82567**.
- Int Rqst - Rising edge indicates an interrupt request by the PHY. The Int Rqst is reflected in the *PHYINT* bit in the ICR. When operating with the **82566**/**82567**, software or firmware should access the Interrupt Cause register in the PHY following the Int Rqst.

Signals that are not supported by the MAC:

- Even bit - The PHY drives this bit to indicate if the last packet contained an even number of bytes. The **82566**/**82567** truncates the odd nibble while the **82562V** provides the even indication.
- RST REQ - This bit is never asserted by the PHY and is ignored by the MAC.
- LCI Er - The PHY never asserts the signal and the MAC ignores it.

7.1.2.3 LCI MAC-to-PHY In-Band Signals

Signals supported by the MAC:

- MDstart - Sync signal of management packet to the PHY.
- MDin - part of the MDIO.
- LED 0/1/2 - When set to 0b, the respective LED should be driven by the PHY or when set to 1b, turns if off. The active polarity of the LEDs can be changed in the LEDCTRL register as well as in the PHY. These signals are only used with the **82566**/**82567** and are driven low when connected to the **82562V**.
- Tx En - Active during frame transmission.
- PHYPD - ..When set to 1b, sets the PHY to the power down state. In this state, the CLK is stopped as well. The PHY might exit the power down state only by asserting the LANRSTSYNC pin.
- LED Select - Driven to the same level of the L*EDCTL.FLTACT* field. when set to 0b, the activity LED is driven completely by the PHY. When set to 1b, the PHY drives the activity LED as a response to Rx Addr match setting. The *FLTACT* bit can be set to 1b only for a 10/100 Mb/s PHY.
- Rx Addr Match - Indicates to the PHY that an incoming packet matched any of the MAC's MAC addresses since the last MAC-to-PHY in-band packet. When possible, an address match event triggers a transmission of a MAC-to-PHY in-band packet.

Signal not supported by the MAC:

- LPBK - Sets the PHY to loopback mode. This bit is always set to 0b. The PHY can also be set to loopback mode via access to the PHY Control register (bit 14).

7.1.3 GLCI Interface

All in-band signals to the PHY go through the LCI link.

7.1.3.1 NJCLK Signal

The GLCI clock shares the same I/O signal as the LCI clock. While in GLCI mode, the PHY provides a single ended 62.5 MHz clock input to the MAC. The clock is generated at all bus states as long as the PHY is not at power down. The MAC uses an internal PLL to generate the LAN_TX signals at a symbol rate of 1.25 G/s. The MAC extracts the clock of the input stream on LAN_RX from the data.

7.1.3.2 GLCI PHY-to-MAC In-Band Signals

Following are GLCI 2.0 PHY-to-MAC in-band signals and the MAC response.

Signals supported by the MAC:

- GLCI Link - Up / Down status indication of the receive Link of the PHY. The bit is the 2nd data byte (bit 5). When set to 1b, the link is up. When set to 0b, the link is down.
- Disconnect - Cable disconnect status indication at the 2nd data byte (bit 4). When set to 1b, the cable is disconnected. When asserted, indicates that the Ethernet link is unplugged, senses no energy for long enough duration (between 4-6 seconds), and the PHY turned off its transmitters for power saving.
- PHY Link - Link Up / Down status indication at the 2nd data byte (bit 3). When set to 1b, the link is up. When set to 0b, the link is down.
- Duplex Mode - Full-duplex (1b)/Half-duplex (0b) status indication at the 2nd data byte (bit 2).
- Link Speed - Indicates link speed at the 2nd data byte (bits 1:0); GbE (10b), 100 Mb/s (01b), and 10 Mb/s (00b).
- CRS - Implicitly indicated by GLCI symbols transmitted from the PHY
- Reset Complete - Reset complete indication at the 2nd data byte (bit 7). Only supported only by the **82566**.
- Interrupt Req - Interrupt request at the 2nd data byte (bit 6) asserts the *PHYINT* bit in the ICR. The Interrupt Req is reflected in the *PHYINT* bit in the ICR. When operating with the **82566**, software or firmware should access the Interrupt Cause register in the PHY following the Interrupt Req.

The **82566**/**82567** includes an Interrupt Cause register. Following a PHY interrupt indication, software or firmware should access this register to read and clear the interrupt cause.

7.1.3.3 GLCI MAC-to-PHY In-Band Signals

Signals supported by the MAC:

- Power down - When set to 1b, sets the PHY to the power down state (2nd data byte, bit 6). In the power down state, the ICH8/ICH9 drives the LANRSTSYNC signal to a low level.
- D-States - MAC power state indication at the 2nd data byte (bits 5:4): Dr (00b), D0u (01b), D0a (10b), and D3 (11b). The MAC always sends its power state to the PHY.
- MAC LED - LED 3:0 drive at the 2nd data byte (bits 3:). The MAC drives only LED 2:0 and drives LED 3 to 0b. When set to 0b, the respective LED should be driven by the PHY or when set to 1b, turns it off. The active polarity of the LEDs can be changed in the LEDCTRL register as well as in the PHY.

Signal not supported by the MAC:

- Port Reset - PHY reset signal. The MAC does not use the in-band reset. Instead, software uses the MDIO resets (PHY Control register, bit 15).

7.1.3.4 Electrical Idle

The **ICH8/ICH9/ICH10** supports an electrical Idle mode on the GLCI bus to the PHY called Legacy Static Electrical Idle. Software or firmware should enable this mode in the MAC as well as in the PHY.

7.1.3.4.1 Static GLCI Electrical Idle

The MAC and PHY support static electrical idle state of the GLCI link. In this state, the GLCI link is at electrical idle. In power down, the PHY might stop the NJCLK clock at low while the MAC's PLL is idle. Setting the GLCI bus to electrical idle is enabled or disabled by the GLCI Modes of Operation register.

7.1.3.4.2 Enter to Static Electrical Idle

The GLCI bus enters the static electrical idle state in the following cases:

- The MAC sets the PHY to the power down state.
- The PHY looses link up state.
- The PHY enters the link disconnect mode (no energy on the link for ~4-6 seconds).

7.1.3.4.3 Exit from Static Electrical Idle

The MAC exits the Static Electrical Idle in the following cases:

- Device state change from D3-Dr/DMoff to D0u/DMx state.
- A PHY status packet is received.
- The MAC wakes up the PHY from a power-down state.

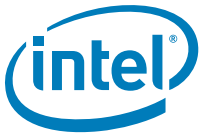
The MAC might initiate an exit from the static electrical idle by sending idle symbols on the LAN_TX signals. The MAC continues transmitting idle symbols until it receives a GLCI link up indication from the PHY. Once the MAC receives the link up indication, it resumes transmission to the PHY.

The PHY might initiate an exit from the static electrical idle state by activating the NJCLK clock (**ICH8**) and driving the link (**ICH8/ICH9/ICH10**). As a result, the MAC exits the electrical idle as well. The PHY wakes the GLCI link in case of LAN link establishment.

7.1.4 Differences Between PHYs

ICH8/ICH9 supports these PHYs: **82566**, **82567**, and **82562V**. This section describes the differences between the two.

- Interconnect - The **82562V** is connected to the MAC via the LCI 1x link. The **82566**/**82567** can be connected to the MAC via the LCI plus the GLCI link or by only using the GLCI link in low pin count mode.
- LED interface - The **82566**/**82567** might drive the LEDs autonomously or by in-band signaling from the MAC. When operating with the **82562V**, the MAC might only drive the activity LED.
- LPLU setting - Not applicable to the **82562V**. The **82566**/**82567** setting of the LPLU is set by the MAC through direct MDIO access to register 0.25 according to the NVM setting in word 17h and the MAC power state.



- GbE disable - Same mechanism as LPLU.
- Restart AN - When operating with the **82562V**, the MAC does not initiate autonomous restart AN instructions. Software or firmware might initiate a restart AN instruction by access to the PHY Control register. When operating with the **82566**/**82567**, the MAC initiates restart AN when it updates the LPLU and GbE disable. All bit fields: *LPLU*, *GbE Disable* and the *Restart AN* are located in the PHY Power Management register.
- LCI PD - In the **82566**/**82567**, the LANTXD signals are driven to 0b when the PHY is set to power down and pull-downs are activated on the LANCLK and the NJCLK. In the **82562V**, the LANTXD signals are driven to 0b as well when the PHY is set to power down but it does not activate the pull-downs on the LANCLK and the NJCLK.
- Interrupt - The PHY indicates a PHY interrupt event on the LCI interface. The **82566**/**82567** contains an Interrupt Cause register that software or firmware can query and acknowledge the PHY interrupt.
- Reset Complete - The **82566**/**82567** provides the reset complete indication on the LCI interface. The **82562V** does not provide this indication.
- Link Energy - The **82562V** reports SQL detection while the **82566**/**82567** reports a link disconnect status indication. Negation of the link-disconnect reports squelch detection as the legacy SQL indication. However, link disconnect is active only if SQL is not detected for a long enough time as defined in the PHY specifications.

Note: During DMIX diagnostics, PHY information is either 10/100 MB/s (IFE) information or 1000 Mb/s (IGP).

7.1.5 PHY Link Detection Protocol

LCI detection and initialization flow is indicated in the entire PHY initialization sequence as described in Section 11.0.

7.1.5.1 LCI Link Detection

During reset indication on the LANRSTSYNC signal, the PHY drives the LANRXD lines to a low level. The MAC includes weak pull-ups on these input signals. If the MAC samples the LANRXD lines at 0b just before de-asserting NJCLK, it concludes that the LCI link is active. Otherwise, it is assumed inactive.

7.1.5.2 GLCI Link Detection

The MAC receives an indication from the ICH8/ICH9 straps whether the GLCI link is available or used as a PCI link. The ICH8/ICH9 fetches this data from the ICH8/ICH9 strapping space in the NVM loaded at power up (same power source that the LAN is connected to).

Note: If the GLCI is unable to achieve link between the MAC and PHY, the lock loss function is used to establish the link. If the GLCI link cannot be established, then the speed drops to 100 Mb/s.

7.1.5.3 Legacy 10/100 Mb/s Mode

This mode uses the following LCI pins: NJCLK, LANRSTSYNC, LANTXD[2:0], LANRXD[2:0]. This is used for legacy 10/100 connections to the **82562V**s.

Required configuration:

- The *GLCI Enable* strapping bit in the NVM must be disabled.
- The *PHYT* field in the NVM should be set to the **82562V** setting.

With this configuration, the MAC expects to find a legacy 10/100 Mb/s device through the LCI ID phase. If not, the MAC aborts and disables the LAN (the LAN is not be visible in the configuration space).

7.1.5.4 Normal 10/100/1000 Mb/s Mode

This mode uses all of the LCI pins. It is used for full-featured GbE connections to **82566**/**82567** products.

Required configuration:

- The *GLCI Enable* strapping bit in the NVM must be enabled.
- The *PHYT* field in the NVM should be set to the **82566**/**82567**.
- The LCI pins should be connected between the MAc and PHY.

With this configuration, the MAC expects to find a GbE PHY through the LCI ID Phase. If not, it aborts and disables the LAN (the LAN is not visible in the configuration space.)

7.1.5.5 PHY Not Present

The MAC detects the PHY presence by counting 16 clocks on NJCLK during power up and PCI reset. If the PHY is not present, the MAC does not complete its initialization phase and leaves the *LAN Init Done* bits in the STATUS CSR in the inactive state.

7.1.6 Transmit and Receive Data Interface

The MAC provides a data transmit and receive interface for 10/100/1000 Mb/s operation along with the proper signaling for carrier sense and collision detection.

7.1.7 Management Register Access

The MAC provides a mechanism to read and write to registers in the PHY. These MII registers are detailed in the IEEE 802.3 specification. These registers are also described in Section 10.0.

7.1.8 Other MAC/PHY Control/Status

In addition to the data and register interfaces, the MAC provides additional information via the LCI interface:

- LCI reset
- Link status indication
- Link duplex indication
- Link speed indication
- MAC state indication
- LED indications
- PHY interrupt

7.2 Duplex Operation

Configuring duplex operation of the MAC/PHY can be forced or determined via the auto-negotiation process.

7.2.1 Full Duplex

All aspects of the IEEE 802.3, 802.3u, 802.3z, and 802.3ab specifications are supported in full duplex operation. Full duplex operation is enabled by several mechanisms depending on the speed configuration of the MAC and the specific capabilities of the PHY used in the application. During full duplex operation, the Ethernet LAN controllers might transmit and receive packets simultaneously across the link interface.

7.2.2 Half Duplex

In half duplex mode, the MAC/PHY attempts to avoid contention with other traffic on the wire by monitoring the carrier sense signal provided by the PHY and deferring to passing traffic. When the carrier sense signal is deasserted or after sufficient InterPacket Gap (IPG) has elapsed after a transmission, frame transmission can begin.

If a collision occurs, the PHY detects the collision and asserts the collision indication to the MAC. Transmission of the frame stops and the MAC sends a JAM sequence onto the link. After the end of a collided transmission, the MAC backs off and attempts to retransmit per the standard CSMA/CD method.

If a successful transmission occurs, the MAC is ready to transmit any other frame(s) queued in its transmit FIFO after the minimum Inter-Frame Spacing (IFS) of the link. has elapsed.

Note: The MAC does not support 1000 Mb/s half-duplex mode.

7.2.3 MAC Speed Resolution

For proper link operation, both the MAC and PHY must be configured to operate at the same link speed. The speed of the link can be determined and set by several methods, including:

- Software-forced configuration.
- Automatic detection of the link speed as provided by the PHY via the LCI.

7.2.3.1 Software-Forced Speed Configuration

To force MAC speed, the software device driver must set the *CTRL.FRCSPD* (force-speed) bit to 1b and then write the speed bits in the Device Control register (CTRL.SPEED) to the desired speed setting.

Note: Forcing the MAC speed using CTRL.FRCSPD overrides all other mechanisms for configuring the MAC speed and can yield non-functional links if the MAC and PHY are not operating at the same speed/configuration.

When forcing a specific speed configuration, the software device driver must also ensure that the PHY is configured to a speed setting consistent with MAC speed settings. This implies that software must access the PHY registers to either force the link speed or to read the PHY's status register bits that indicate the speed of the network link.

Note: Forcing the speed settings by CTRL.SPEED can also be accomplished by setting the *CTRL_EXT.SPD_BYPS* bit. This bit bypasses the internal clock switching logic and enables the software device driver complete control of when the speed setting takes place. The *CTRL.FRCSPD* bit uses the internal clock switching logic, which does delay the affect of the speed change.

7.2.3.2 Automatic Detection of Link Speed

The PHY provides a direct internal indication of its speed to the MAC (SPD_IND) via the LCI interface.

To set/determine speed from these direct internal indications from the PHY, the forced-speed override should be disabled (CTRL.FRCSPD = 0b). With the CTRL register configured, the speed is automatically reconfigured each time a new link-up event is detected.

7.2.3.3 MAC Full-/Half-Duplex Resolution

The duplex configuration of the link is also resolved by the PHY during the Auto-Negotiation process. The PHY provides an internal indication to the MAC of the resolved duplex configuration.

This internal duplex indication is normally sampled by the MAC each time the PHY indicates the establishment of a good link. The PHY's indicated duplex configuration is applied in the MAC and reflected in the MAC's Device Status register (STATUS.FD).

Software can override the duplex setting of the MAC via the *CTRL.FD* bit when the *CTRL.FRCDPLX* (force duplex) bit is set. If *CTRL.FRCDPLX* is 0b, the *CTRL.FD* bit is ignored and the PHY's duplex indication applied.

7.2.3.3.1 Using MII Registers

The software device driver might be required, under some circumstances, to read from or write to the MII management registers in the PHY. These accesses are performed via the MDIC. The MII registers enable the software device driver to have direct control over the PHY's operation, which can include:

- Resetting the PHY.
- Setting the preferred link configuration for advertisement during the Auto-Negotiation process.
- Restarting the auto-negotiation process.
- Reading auto-negotiation status from the PHY.
- Forcing the PHY to a specific link configuration.

The set of management registers required for all PHY's can be found in the IEEE P802.3ab draft standard.

7.2.4 Loss of Signal/Link Status Indication

The PHY provides an indication of physical link status to the MAC. This indicates whether the link is up or down; typically indicated after successful Auto-Negotiation. The MAC status bit *(STATUS.LU)*, when read, generally reflects whether the PHY has link (except under forced-link setup where even the PHY's link indication might have been forced).

When the link indication from the PHY is de-asserted, the MAC considers this to be a transition to a link-down situation (cable unplugged, loss of link partner, etc.). If the Link Status Change (LSC) interrupt is enabled, the MAC generates an interrupt to be serviced by the driver.

7.3 10/100 Mb/s Specific Performance Enhancements

7.3.1 Adaptive IFS

The MAC supports back-to-back transmit Inter-Frame-Spacing (IFS) of 960 ns in 100 Mb/s operation and 9.6 μs in 10 Mb/s operation. Although back-to-back transmission is normally desirable, sometimes it can actually hurt performance in half-duplex environments due to excessive collisions. Excessive collisions are likely to occur in environments where one station is attempting to send large frames back-to-back, while another station is attempting to send Acknowledge (ACK) packets.

The MAC contains an Adaptive IFS register that enables the implementation of a software device driver based adaptive IFS algorithm for collision reduction, which is similar to Intel's other Ethernet products (such as PRO/100 adapters). Adaptive IFS throttles back-to-back transmissions in the transmit MAC and delays their transfer to the CSMA/CD transmit function, and as a result can be used to delay the transmission of back-to-back packets on the wire. Normally, this register should be set to zero. However, if additional delay is desired between back-to-back transmits, then this register can be set with a value greater than zero. This can be helpful in high collision half-duplex environments.

The AIFS field provides a similar function to the IGPT field in the TIPG register (see Section 10.0). However, this Adaptive IFS throttle register counts in units of GTX/MTX_CLK clocks (which are 8 ns, 80 ns, 800 ns for 1000 Mb/s, 100 Mb/s, 10 Mb/s mode respectively), and is 16 bits wide, thus providing a greater maximum delay value.

Using values lower than a certain minimum (determined by the ratio of GTX/MTX_CLK clock to link speed), has no effect on back-to-back transmission. This is because the device does not start transmission until the minimum IEEE IFS (9.6 μs at 10 Mb/s, 960 ns at 100 Mb/s, and 96 ns at 1000 Mb/s) has been met regardless of the value of Adaptive IFS. For example, if a MAC is configured for 100 Mb/s operation, the minimum IEEE IFS at 100 Mb/s is 960 ns. Setting AIFS to a value of 10 (decimal) would not effect back-to-back transmission time on the wire, because the 800 ns delay introduced (10 * 80 ns = 800 ns) is less than the minimum IEEE IFS delay of 960 ns. However, setting this register with a value of 20, which corresponds to 1600 ns for the previous example, would delay back-to-back transmits because the ensuing 1600 ns delay is greater than the minimum IFS time of 960 ns.

Note that this register has no effect on transmissions that occur immediately after receives, or on transmissions that are not back-to-back (unlike the IPGR1 and IPGR2 values in the TIPG register). In addition, Adaptive IFS also has no effect on re-transmission timing (re-transmissions occur after collisions). Therefore, AIFS is only enabled in a back-to-back transmission.

Note: The AIFS value is not additive to the TIPG.IPGT value; instead, the actual IPG equals the larger of AIFS and TIPG.IPGT.

7.3.2 Flow Control

Flow control as defined in IEEE specification 802.3x, as well as the specific operation of asymmetrical flow control defined by 802.3z, are supported. The following registers are defined for the implementation of flow control:

Table 46. Flow Control Registers

Register Name	Description
Flow Control Address Low, High (FCAL/H)	6-byte flow control multicast address
Flow Control Receive Thresh Hi (FCRTH)	13-bit high water mark indicating receive buffer fullness
Flow Control Transmit Timer Value (FCTTV)	16 bit timer value to include in transmitted PAUSE frame
Flow Control Type (FCT)	16-bit field to indicate flow control type
Flow Control Receive Thresh Lo (FCRTL)	13-bit low water mark indicating receive buffer emptiness

Flow control is implemented as a means of reducing the possibility of receive buffer overflows which result in the dropping of received packets, and allows for local control 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.

7.3.3 MAC Control Frames and Reception of Flow Control Packets

Three comparisons are used to determine the validity of a flow control frame:

1. A match on the 6-byte multicast address for MAC Control Frames or to the station address of the device (Receive Address Register 0).
2. A match on the type field.
3. A comparison of the MAC Control Opcode field.

Standard 802.3x defines the MAC Control Frame multicast address as 01_80_C2_00_00_01h.

The Flow Control Type register (FCT) contains a 16-bit field that is compared against the flow control packet's type field to determine if it is a valid flow control packet: XON or XOFF. 802.3x reserves this value as 8808h. This number must be loaded into the Flow Control Type (FCT) register.

The final check for a valid PAUSE frame is the MAC Control Opcode. At this time only the PAUSE control frame opcode is defined. It has a value of 0001h.

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 slot time. A slot time is hard wired to a 64-byte times.

Note: An XON frame signals the cancellation of the pause from initiated by an XOFF frame (pause for zero slot times).

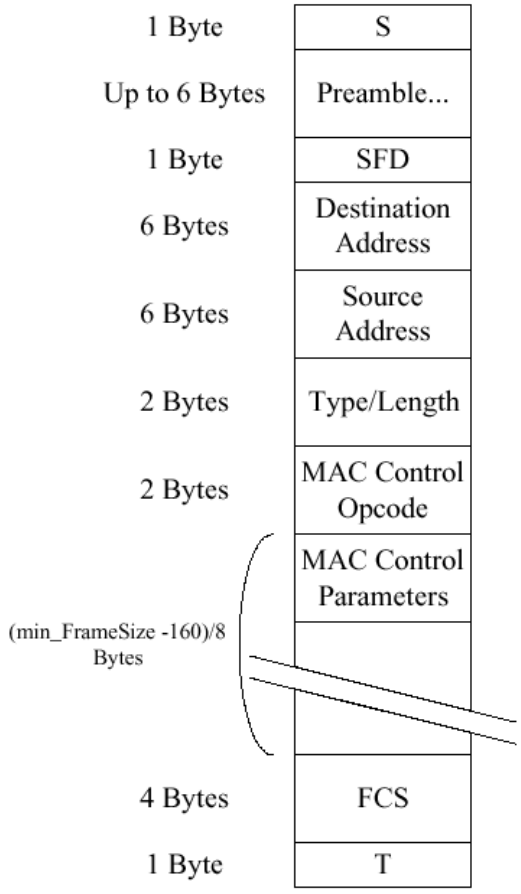


Figure 35. 802.3x MAC Control Frame Format

Note: "S" is the Start-of-Packet delimiter and "T" is the first part of the End-of-Packet delimiters for 802.3z encapsulation.

The receiver is enabled to receive flow control frames if flow control is enabled through the *RFCE* bit in the Device Control register (CTRL).

Note: Flow control capability must be negotiated between link partners via the Auto-Negotiation process. The Auto-Negotiation process can modify the value of these bits based on the resolved capability between the local device and the link partner.

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

- Increment the appropriate statistics register(s)
- Set the TXOFF bit in the Device Status Register (STATUS)
- Initialize the pause timer based on the packet's PAUSE timer field
- Disable packet transmission or schedule the disabling of transmission after the current packet completes.

Resumption of transmission can occur under the following conditions:

- Expiration of the PAUSE timer
- Reception of on XON frame (a frame with its PAUSE timer set to 0b)

Either condition clears the STATUS.TXOFF bit and transmission can resume. Hardware records the number of received XON frames in the XONRXC counter.

7.3.4 Discard PAUSE Frames and Pass MAC Control Frames

Two bits in the Receive Control register (RCTL) are implemented specifically for control over receipt of PAUSE and MAC control frames. These bits are Discard PAUSE Frames (DPF) and Pass MAC Control Frames (PMCF). See Section 10.0 for DPF and PMCF bit definitions.

The DPF bit is reserved. Software should not set this bit.

The PMCF bit enables the passing of any valid MAC control frames to the system that do not have a valid PAUSE opcode. The frame can have the correct MAC control frame multicast address (or the MAC station address) as well as the correct type field match with the FCT register, but does not have the defined PAUSE opcode of 0001h. Frames of this type are transferred to host memory when PMCF is logic high.

7.3.5 Transmission of PAUSE Frames

Transmitting PAUSE frames is enabled by software writing a 1b to the CTRL.TFCE bit in the Device Control register.

Note: Similar to the reception flow control packets described earlier, XOFF packets can be transmitted only if this configuration has been negotiated between the link partners via the Auto-Negotiation process. In other words, the setting of this bit indicates the desired configuration.

The contents of the Flow Control Receive Threshold High register (FCRTH) determine at what point hardware transmits a PAUSE frame. Hardware monitors the fullness of the receive FIFO and compares it with the contents of FCRTH. When the threshold is reached, hardware sends a PAUSE frame with its pause time field equal to FCTTV. Once the receive buffer fullness reaches the low water mark, hardware sends an XON message (a PAUSE frame with a timer value of 0b). Software enables this capability with the XONE field of the FCRTL.

Hardware sends one more PAUSE frames if it has previously sent one and the FIFO overflows (so the threshold must not be set greater than the FIFO size). This function is intended to minimize the number of packets dropped if the first PAUSE frame does not reach its target. Since the secure receive packets use the same data path, the behavior is identical when secure packets are received.

Note: Transmitting Flow Control frames should only be enabled in full duplex mode per the IEEE 802.3 standard. Software should ensure that the transmission of flow control packets is disabled when the MAC is operating in half-duplex mode.

7.3.6 Software Initiated PAUSE Frame Transmission

The MAC has the added capability to transmit an XOFF frame through software. This function is accomplished by software writing a 1b to the SWXOFF bit of the Transmit Control register (TCTL). Once this bit is set, hardware initiates the transmission of a PAUSE frame in a manner similar to that automatically generated by hardware.

The *SWXOFF* bit is self clearing after the PAUSE frame has been transmitted.

The state of the *CTRL.TFCE* bit or the negotiated flow control configuration does not affect software generated PAUSE frame transmission.

Note: Software sends an XON frame by programming a zero in the PAUSE timer field of the FCTTV register.

Note: XOFF transmission is not supported in 802.3x for half duplex links. Software should not initiate an XOFF or XON transmission if the MAC is configured for half duplex operation.

7.3.7 Loopback

This section details the various levels of host loopback support in the MAC.

7.3.7.1 MAC Loopback

Loopback is supported by the internal MAC interface. Loopback includes all data path logic except for the LCI interface. Note that this mode can only operate in full duplex mode. The following bits must be configured to force full duplex mode:

- CTRL.FRCDPLX = 1b - // Force duplex mode by the MAC
- CTRL.FD = 1b - // Set full-duplex mode
- GLCI Diagnostic.NELPBK = 1b - // Set MAC to loopback mode

7.3.7.2 MAC Loopback Using the 82567

The functionality, timing, and signal integrity of the MAC interface can be tested by placing the **82567** in MAC interface loopback mode (PHY Control register, bit 14 = 1b). In loopback mode, the data received from the MAC is not transmitted out on the media interface. Instead, the data is looped back and sent to the MAC. During loopback, link is lost and packets are not received. If auto-negotiation occurs while loopback is enabled, FLP auto-negotiation codes are transmitted. If in forced 10BASE-T mode and loopback is enabled, 10BASE-T idle link pulses are transmitted on the copper side. If in forced 100BASE-T mode and loopback is enabled, 100BASE-T idles are transmitted on the copper side.

The speed of the interface is determined by the MAC Specific Control register, bits 2:0 (Section 10.0). In addition to setting the default MAC speed during loopback and link down, the MAC Specific Control register also determines the speed of the transmit clock (TX_CLK) during link down and 100BASE-T operation (if the MAC requires a continuous TX_CLK).

7.4 Non Volatile Memory Interface

Several LAN clients can access the NVM. Following are the various clients and their access type to the NVM: Hardware, LAN Driver, and BIOS.

7.4.1 Hardware: LAN Configuration Auto-Read

The MAC requires non-volatile content for the device configuration. This configuration is loaded by the MAC. Communication is done by peer accesses from the MAC to the Flash within the chipset. The MAC accesses its space in the NVM at offset 0. The ICH8 NVM controller translates it to a physical address in the NVM by adding the LAN base address as defined in GFPREG.PRB.

The LAN configuration space can reside in one of two sectors as shown in Figure 36. The MAC goes through the following steps when access is required to its configuration image:

- Following PCI reset de-assertion, the MAC looks for the signature in the Shared Initialization Word in the LAN sector 0.
- If invalid, the MAC checks for a valid signature in sector 1.
- The MAC indicates the active sector in the EEC.SEC1VAL bit. The SEC1VAL bit is set to 1b only if sector 0 is found invalid and sector 1 was found valid.
- If the signature is valid, the MAC continues reading the whole LAN sector from the valid sector as indicated by EEC.SEC1VAL
- If the signature is not valid in both sectors, the MAC concludes that the LAN image is not valid and initializes itself by the hardware defaults. The LAN communication unit remains in the reset mode until software of firmware initializes it.

Following a software PHY reset that requires only a partial read from the NVM, the MAC reads the data from the current active LAN sector. The MAC does not check the validity of the NVM in these cases but instead relies on the EEC.NV_PRES indication.

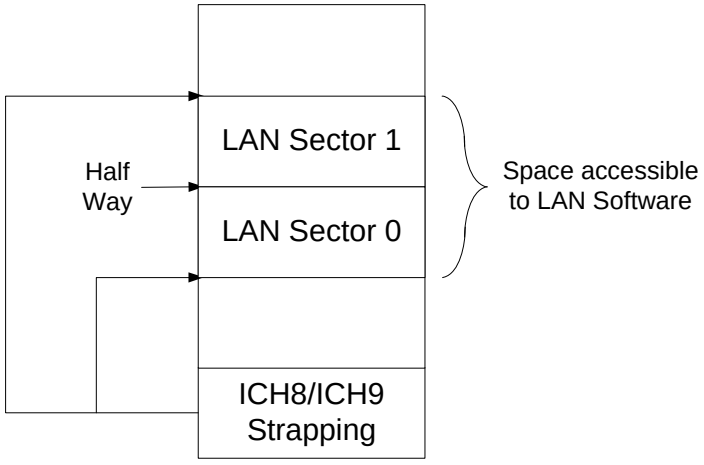


Figure 36. LAN Configuration NVM Image

7.4.2 Host Access to LAN Space in the NVM

Software can initiate a read or write cycle to the NVM through the Flash registers in the Flash Bar. LAN drivers that require this support include: PXE, software device driver, DOS tools, etc.

7.4.3 LAN Driver: LAN Configuration NVM Update

The LAN configuration space includes 2 sectors that the LAN can initiate itself. Only one sector must be valid enabling the other one to be used as a temporary storage of the previous valid configuration space (Figure 36).

During a LAN configuration image update, software should ensure that at any given time at least one of the two sectors (Sector 0 or Sector 1) is valid. Software should program first the updated configuration in the new sector while programming the Flash signature last (byte 13h in the Flash). Only then can software invalidate the previous configuration sector in the Flash. The complete software flow is as follows:

- Use the EEC.SEC1VAL information as an indication on the current valid sector.
- Erase the other sector. Software might need to query the block erase size in the HSFSTS.BERASE field.
- Create a soft image (shadow RAM) of the current configuration sector. Software might need to read the whole sector either at driver initialization or by the Flash programming utility.

Note: The size of the shadow RAM is 4096 bytes (2048 words). For additional information about shadow RAM, refer to the *I/O Control Hub 8 LAN NVM Map and Programming Information Guide*, the *I/O Control Hub 9 LAN NVM Map and Programming Information Guide*, or the *I/O Control Hub 10 LAN NVM Map and Programming Information Guide*.

- Update the soft image (shadow RAM) as required.
- Program the new sector in the Flash while the signature byte is programmed last.
- Invalidate the old sector by programming the signature to 0b to the upper byte of word offset 13h in the LAN configuration space in the Flash so the EEC register can have the current values loaded into it.
- Software should initialize the MAC from the NVM so the new setup takes place.

7.4.4 BIOS: Network Boot Expansion ROM (PXE)

The BIOS might access the boot expansion ROM during the boot sequence. The host CPU might also execute the code directly from the Flash or copy the content to the main memory for its execution. The MAC is not involved in the transactions between the CPU and the Flash.

7.4.5 Initial Programming

The MAC identifies invalid (or empty) LAN space by recognition a bad signature in the Shared Init Control Word in the NVM.

Software should make sure that one of the LAN configuration sectors is empty (all bytes equal FFh). If no such sector exists, then software should erase one of these sectors. Software needs to query the block erase size in the FCMDST register.

Software can then program the LAN configuration content while the signature byte should be programmed last. This sequence ensures that the MAC properly interprets one half of the programmed sector as invalid in case power is lost during the programming process.

7.5 Configurable LEDs

7.5.1 Operating with the 82566/82567

The MAC supports three controllable and configurable LEDs that are driven from the PHY. Each of the three 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. Also, the hardware-default configuration for all the LED outputs can be specified via NVM fields, thereby supporting LED displays configurable to a particular OEM preference.

Each of the three LED's can be configured to use one of a variety of sources for an output indication. The MODE bits control the LED source:

- LINK_100/1000 is asserted when link is established at either 100 or 1000 Mb/s.
- LINK_10/1000 is asserted when link is established at either 10 or 1000 Mb/s.
- LINK_UP is asserted when any speed link is established and maintained.
- ACTIVITY is asserted when link is established and packets are being transmitted or received.
- LINK/ACTIVITY is asserted when link is established AND there is NO transmit or receive activity.
- LINK_10 is asserted when a 10 Mb/s link is established and maintained.
- LINK_100 is asserted when a 100 Mb/s link is established and maintained.
- LINK_1000 is asserted when a 1000 Mb/s link is established and maintained.
- FULL_DUPLEX is asserted when the link is configured for full-duplex operation.
- COLLISION is asserted when a collision is observed.
- PAUSED is asserted when the device's transmitter is flow controlled.
- LED_ON is always asserted; LED_OFF is always de-asserted.

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 while the LED source is asserted, and the blinking frequency (either 200 ms on and 200 ms off or 83 ms on and 83 ms off). The blink control might be especially useful for ensuring that certain events, such as ACTIVITY indication, cause LED transitions, which are sufficiently visible to a human eye. The same blinking rate is shared by all LEDs.

If ¼ CLK is enabled (*CLK_CNT_1_4* bit in the STATUS register is set) and the link rate is 10 Mb/s or 100 Mb/s, the LEDs might blink at a rate four times faster than the previously mentioned setting depending on the activity factor.

Note: 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, and blinking if there is LINK and ACTIVITY.

7.5.2 Operating with the 82562V

The MAC might affect only one of the LEDs that are driven by the PHY. By hardware defaults, all LEDs are driven completely by the PHY. The MAC might be configured to control the activity LED as follows:

- LEDCTL.FLTACT = 0b - The MAC drives the LED select in the in-band LCI MAC-to-PHY packets to 0b. As a result, the PHY drives the activity LED autonomously.
- LEDCTL.FLTACT = 1b - The MAC drives the LED select in the in-band MAC-to-PHY packets to 1b. As a result, the PHY drives the activity LED according to the *Rx Addr Match* bit in the in-band MAC-to-PHY packets.

When the *FLTACT* is set to 1b, the MAC sets an Rx address match event. Address match event is set once per event in the *Rx Addr Match* field in the MAC-to-PHY in-band packet. The address match event is cleared after it is indicated in the in-band packet. Once the address match event is set, the MAC initiates an in-band packet. The in-band packet might be postponed if a nominal Tx packet is in progress or another in-band packet is in progress but the MAC missed the indication of the address match event.

7.6 Software/Hardware Firmware Semaphore

There are some portions of the MAC that might be accessed autonomously by the hardware, by software, or by firmware referred as clients. This section describes a semaphore mechanism required to avoid collisions between these three clients. Following are the relevant resources:

- Accesses to the PHY via the MDIO registers might be accessed by all three clients.
- All software CSR space can be accessed by the firmware as well. Following are some CSR registers that software and firmware should use the semaphore protocol before any write access: CTRL, CTRL_EXT, MDIC, GLCI Control registers.

Each client has a *Request/Grant* bit that it used to communicate with the other client. Before any client can write to the common registers, it must check first that the registers are not busy as described in the sections that follow.

7.6.1 Hardware Semaphore

Hardware uses the *MDIO HW Ownership* bit in the EXTCNF_CTRL register for the semaphore. Hardware writes a 1b to this bit and then waits until it is set before it configures the PHY using the MDIC register. Hardware clears this bit when PHY accesses complete.

After setting the *MDIO HW Ownership* bit the MAC waits until it is set. The *MDIO HW Ownership* bit is set only after the *SWFLAG* bit in the same register and the *FWFLAG* bit in the FWSM register are cleared. It is the responsibility of software and firmware to clear their ownership flags to enable the other clients to access the PHY.

For **ICH8**, in order to avoid deadlock, hardware includes a 100 ms watchdog mechanism enabled by the *MDIOWatchEna* bit in the EXTCNF_CTRL register. If hardware waits more then 100 ms then it ignores the *SWFLAG* and *FWFLAG* and accesses the PHY.

For **ICH9/ICH10**, in order to avoid deadlock, hardware includes a 100 ms watchdog mechanism enabled by bit 10 in the FEXTNVM register. If hardware waits more then 100 ms then it clears the *SWFLAG* and *FWFLAG* enabling the hardware to proceed.

For **ICH9/ICH10**, when the hardware completes its cycles to the PHY it again checks the request flags of the software and the firmware (see Section 7.6.2 and Section 7.6.3). If any of them are set the hardware sets the relevant bit in the CSR register. Only one bit can be active at a given time.

7.6.2 Software Semaphore

The software uses the *SWFLAG* bit in the EXTCNF_CTRL register for the semaphore. The software device driver writes a 1b to this bit each time it needs to write to any CSR register that might be occupied by the firmware or hardware. The hardware sets this bit only after it detects that all shared resources are available. For example, both the *MDIO HW Ownership* in the EXTCNF_CTRL register and *FWFLAG* in the FWSM register are cleared. The software device driver should wait until the *SWFLAG* is set by hardware before it can write to the shared CSR registers. Software should clear this bit when finishing its access.

Note: Software monitors the *MDIO HW Ownership* in the EXTCNF_CTRL register to conclude when the PHY initialization is in progress and the *FWFLAG* bit in the FWSM register to conclude when firmware accesses these resources.

7.6.3 Firmware Semaphore

Firmware uses the *FWFLAG* bit in the FWSM register for the semaphore. Firmware writes a 1b to this bit each time it needs to write to any CSR register that might be occupied by software or hardware. Hardware sets this bit only after it detects that all shared resources are available. For example, both the *MDIO HW Ownership* bit and the *SWFLAG* bit in the EXTCNF_CTRL register are cleared. Firmware should wait until the *FWFLASG* is set by hardware before it can write to the shared CSR registers. After the *FWFLASG* bit is set, firmware should clear the *RSPCIPHY* bit in the FWSM register to avoid a PHY reset by software. Firmware should clear this bit when finishing its access. It can also set the *RSPHY* bit to 1b if it enables software to initialize the PHY.

Note: Firmware monitors the *MDIO HW Ownership* bit and the *SWFLAG* bit in the EXTCNF_CTRL register to conclude when hardware and software accesses these resources.

7.6.4 Ownership Arbitration

Shared Resources Arbitration - Arbitration between software, hardware, and firmware clients is done only when none of them already have ownership by the arbiter. If multiple clients are requesting the ownership of the shared resources, hardware has always highest priority. Firmware and software have equal priority with round robin arbitration between them. Once a client has ownership of the shared resources, it should release the arbitration as fast as possible to avoid starvation of the other clients.

Hardware always obeys the arbitration rules. It takes ownership on the shared resources accessing the MDIO registers only after both the *SWFLAG* bit in the EXTCNF_CTRL register and *FWFLAG* bits in the FWSM register are cleared, which is indicated by setting the *MDIO HW Ownership* bit in the EXTCNF_CTRL register. If the hardware waits too long and the arbitration timer expires, hardware clears the software and firmware flags.

- SWFLAG does not gate software accesses to the shared resources; however, software should obey it.
- FWFLAG does not gate firmware accesses to the shared resources; however, firmware should obey it.

Note: Firmware should implement a watch dog mechanism that can detect software or hardware violation of the arbitration and overcome it. Firmware clears the hardware ownership by device initialization.

8.0 802.1q VLAN Support

The MAC provides several specific mechanisms to support 802.1q VLANs:

- Optional adding (for transmits) of IEEE 802.1q VLAN tags.
- Optional stripping (for receives) of IEEE 802.1q VLAN tags.

8.1 802.1q VLAN Packet Format

Table 47 compares the format of an untagged 802.3 Ethernet packet with an 802.1q VLAN tagged packet.

Table 47. VLAN Packet Format Comparison

802.3 Packet	#Octets
DA	6
SA	6
Type/Length	2
Data	46-1500
CRC	4

802.1q VLAN Packet	#Octets
DA	6
SA	6
8021.q Tag	4
Type/Length	2
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 802.3 Ethernet packet.

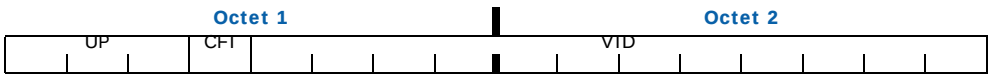
8.1.1 802.1q Tagged Frames

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 2 octets. The first 16 bits of the tag header make 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 (see Table 48 for details):

- User Priority (UP).
- Canonical Form Indicator (CFI). The CFI should be 0b for transmits. For receives, its a don't care.
- VLAN Identifier (VID)

Table 48. 802.1q Tagged Frames



8.2 Transmitting and Receiving 802.1q Packets

Since the 802.1q tag is only four bytes, adding and stripping of tags can done completely in software. (For transmits, software inserts the tag into packet data before it builds the transmit descriptor list, and for receives, software strips the four byte tag from the packet data before delivering the packet to upper layer software.)

However, adding and stripping of tags in software results in more overhead for the host,. The MAC has additional capabilities to add and strip tags in hardware, as discussed in the following two sections.

8.2.1 Adding 802.1q Tags on Transmits

Software can command the MAC to insert an 802.1q VLAN tag on a per packet basis. If *CTRL.VME* is set to 1b, and the VLE bit in the transmit descriptor is set to 1b, then the MAC inserts a VLAN tag into the packet that it transmits over the wire. The Tag Control Information (TCI) of the 802.1q tag comes from the special field of the transmit descriptor.

8.2.2 Stripping 802.1q Tags on Receives

Software can instruct the MAC to strip 802.1q VLAN tags from received packets. If the *CTRL.VME* bit is set to 1b, and the incoming packet is an 802.1q VLAN packet, then the MAC strips the 4-byte VLAN tag from the packet, and stores the TCI in the *Special* field of the receive descriptor. The MAC also sets the *VP* bit in the receive descriptor to indicate that the packet had a VLAN tag that was stripped.

If the *CTRL.VME* bit is not set, the 802.1Q packets can still be received if they pass the receive filter. In this case, the VLAN tag is not stripped and the *VP* bit is not set.

9.0 PHY Functionality and Features

This section describes the PHY functionality and features for the **82566** and **82567** PHYs.

Note: For **82562V** PHY functionality and features, refer to the *82562V 10/100 Mbps Platform LAN Connect (PLC) Datasheet*.

9.1 Initialization

The PHY MDIO registers are automatically initialized at power up or reset by the MAC through the LCI/GLCI interface.

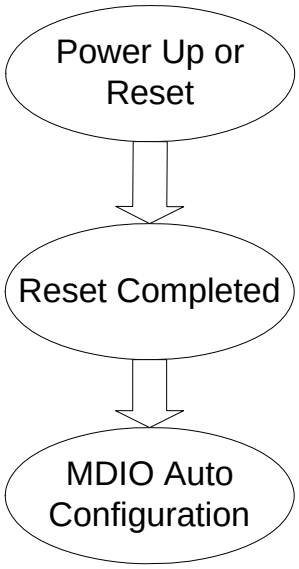


Figure 37. PHY Initialization

Note: Allow 15 ms for a complete PHY reset.

9.2 Determining Link State

The PHY and its link partner determine the type of link established through one of three methods:

- Auto-Negotiation
- Parallel Detection
- Forced Operation

Auto-Negotiation is the only method allowed by the 802.3ab standard for establishing a 1000BASE-T link, although forced operation could be used for test purposes. For 10/100 links, any of the three methods can be used. The sections that follow discuss each in greater detail.

Figure 38 provides an overview of link establishment. First the PHY checks if Auto-Negotiation is enabled. By default, the PHY supports Auto-Negotiation (PHY Control register, bit 12). If not, the PHY forces operation as directed. If Auto-Negotiation is enabled, the PHY begins transmitting Fast Link Pulses (FLPs) and receiving FLPs from its link partner. If FLPs are received by the PHY, Auto-Negotiation proceeds. It also can receive 100BASE-TX MLT3 and 10BASE-T Normal Link Pulses (NLPs). If either MLT3 or NLPs are received, it aborts FLP transmission and immediately brings up the corresponding half-duplex link.

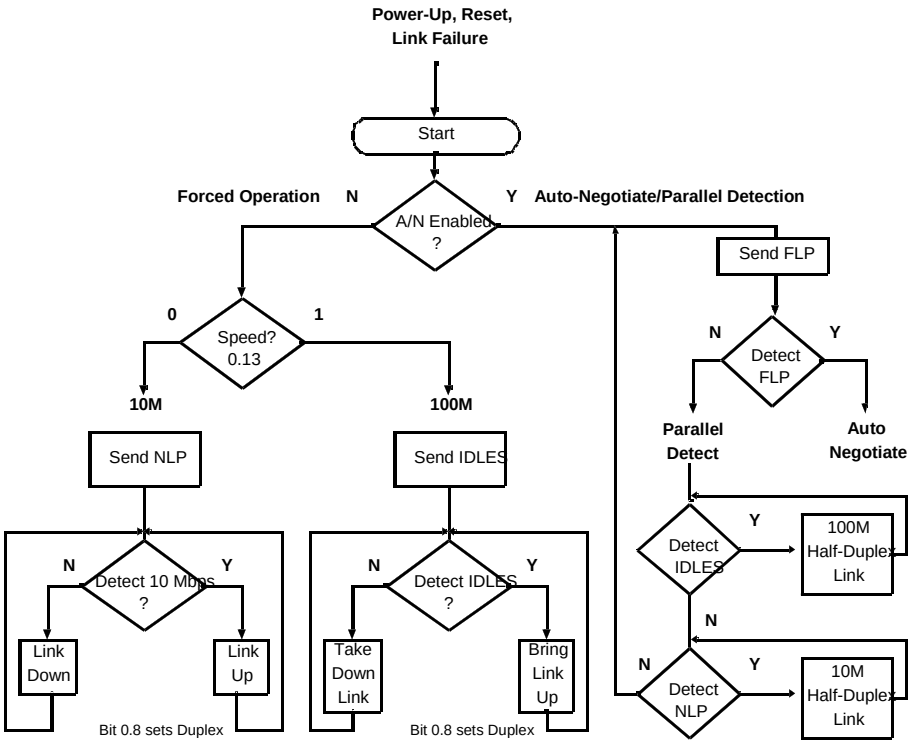


Figure 38. Overview of Link Establishment

9.2.1 False Link

When the PHY is first powered on, reset, or encounters a link down state, it must determine the line speed and operating conditions to use for the network link.

The PHY first checks the MDIO registers (initialized via the Hardware Control Interface or written by software) for operating instructions. Using these mechanisms, programmers can command the PHY to do one of the following:

- Force twisted-pair link operation to:
  - 1000T Full Duplex
  - 1000T Half Duplex
  - 100TX, Full Duplex
  - 100TX, Half Duplex
  - 10BASE-T, Full Duplex
  - 10BASET, Half Duplex
- Allow auto-negotiation/parallel-detection.

In the first six cases (forced operation), the PHY immediately begins operating the network interface as commanded. In the last case, the PHY begins the auto-negotiation/parallel-detection process.

9.2.2 Forced Operation

Forced operation can be used to establish 10 and 100 links, and 1000 links for test purposes. In this method, Auto-Negotiation is disabled completely and the link state of the PHY is determined by the PHY Control register.

Note: When speed is forced, the MDI/MDI-X crossover feature is not functional.

In forced operation, the programmer sets the link speed (10, 100, or 1000 Mb/s) and duplex state (full or half). For gigabit (1000 Mb/s) links, the programmer must explicitly designate one side as the master and the other as the slave.

Table 49 lists the link establishment procedures.

Table 49. Determining Duplex State Via Parallel Detection

Configuration	Result
Both sides set for Auto-Negotiate.	Link is established via Auto-Negotiation.
Both sides set for forced operation.	No problem as long as duplex settings match.
One side set for Auto-Negotiation and the other for forced, half-duplex.	Link is established via parallel detect.
One side set for Auto-Negotiation and the other for forced full-duplex.	Link is established; however, sides disagree, resulting in transmission problems. Forced side is full-duplex, Auto-Negotiation side is half-duplex.

Note that according to the 802.3 specification, if one side of the link is forced to full-duplex operation and the other side has auto-negotiation enabled, the auto-negotiating partner parallel-detects to a half-duplex link while the forced side operates as directed in full-duplex mode. The result is spurious, unexpected collisions on the side configured to auto-negotiate (see Section 9.2.2.1).

9.2.2.1 10/100 Mb/s Mismatch Resolution

It is a common occurrence where a link partner is configured for forced full-duplex 10/100 Mb/s operation. The normal auto-negotiation sequence results in the other end settling for half-duplex 10/100 Mb/s operation. The mechanism described in this section resolves the mismatch and automatically transitions the **82566**/**82567** into full-duplex mode, enabling it to operate with a partner configured for full-duplex operation.

The **82566**/**82567** enables the software device driver to detect the mismatch event previously described and sets its duplex mode into the appropriate value without going through another auto-negotiation sequence or breaking link. Once software detects a possible mismatch, it instructs the **82566**/**82567** to change its mode to either half-duplex or full-duplex. Software sets the *Duplexity_manual_set* bit in the Misc Cntrl register (bit 12) to indicate that the duplex mode should be changed to the value indicated by the *Duplex Mode* bit in the Port Status 1 register (bit 9). Any change in the value of the *Duplex Mode* bit in the Port Status 1 register while the *Duplexity_manual_set* bit is 1b also causes a change in the PHY duplex mode.

The *Duplexity_manual_set* bit is cleared on all PHY resets, following auto-negotiation, and when the link goes down. Software might track the change in duplex mode through the PHY *Duplex Mode* bit in Port Status 1 register or a MAC indication.

Note: The change in duplex mode is reflected in the status bits sent to the MAC over the GLCI or LCI interface. As a result, the MAC might issue a LSC interrupt to the software device driver (if enabled). The software device driver should avoid a link down indication to the upper layer system components, but should still indicate link speed/duplex change to operating system.

9.2.2.1.1 Usage Model and Flow

This section describes a recommended usage model and flow to resolve a mismatch in 10/100 Mb/s duplex mode.

Mismatch resolution is controlled by the **82566**/**82567** software device driver and is limited to the following:

- Link was established through auto-negotiation.
- The **82566**/**82567** is at half-duplex 10/100 Mb/s mode (acquired through parallel detection).

This situation is detected by software when the **82566**/**82567** is auto-negotiation enabled (PHY Control register, bit 12) and the link partner in not link partner auto-negotiation able (Auto-Negotiation Expansion register, bit 0).

Once software detects these conditions, it monitors the link collision behavior, especially that of late collision events. Software might try to improve performance by transitioning the **82566**/**82567** into full-duplex mode as previously described. Software then monitors the link collision behavior for improved behavior. If this does not happen, software sets the **82566**/**82567** back to half-duplex mode.

9.2.3 Auto-Negotiation

The PHY supports the IEEE 802.3u auto negotiation scheme with next page capability. Next Page exchange uses the Auto-Negotiation Next Page Transmit register to send information and the Auto-Negotiation Next Page Ability register to receive them. Next Page exchange can only occur if both ends of the link advertise their ability to exchange Next Pages.

9.2.4 Parallel Detection

Parallel detection can only be used to establish 10 and 100 links. It occurs when the PHY tries to negotiate (transmit FLPs to its link partner), but instead of sensing FLPs from the link partner, it senses 100BASE-TX MLT3 code or 10BASE-T Normal Link Pulses (NLPs) instead. In this case, the PHY immediately stops auto-negotiation (terminates transmission of FLPs) and immediately brings up whatever link corresponds to what it has sensed (MLT3 or NLPs). If the PHY senses both of the technologies together, a parallel detection fault is detected and the PHY continues sending FLPs.

With parallel detection, IEEE standard requires the PHY to assume a half-duplex link. Parallel detection also does not allow exchange of flow-control ability (PAUSE and ASM_DIR) or master/slave relationship required by 1000BASE-T. As a result, parallel detection cannot be used to establish GbE Ethernet links.

If the link partner is configured to full duplex forced operation, spurious unexpected collisions would be detected on the side configured to auto-negotiate.

The software device driver can detect this situation and configure the PHY to full duplex operation, using the Misc Cntrl register (bit12) and the PHY Control register (bit 8).

9.3 Auto Crossover

Twisted pair Ethernet PHY's must be correctly configured for MDI or MDI-X operation to interoperate. This has historically been accomplished using special patch cables, magnetics pin-outs or PCB wiring. The PHY supports the automatic MDI/MDI-X configuration originally developed for 1000Base-T and standardized in IEEE 802.3u section 40. Manual (i.e., non-automatic) configuration is still possible.

For 1000BASE-T links, pair identification is determined automatically in accordance with the standard.

For 10/100 links and during auto-negotiation, pair usage is determined by bits 12 and 13 in the Port Control register.

In addition, the PHY has an Automatic Crossover Detection function. If bit 12 of the Port Control register = 1b, the PHY automatically detects which application is being used and configures itself accordingly

The automatic MDI/MDI-X state machine facilitates switching the MDI_PLUS[0] and MDI_MINUS[0] signals with the MDI_PLUS[1] and MDI_MINUS[1] signals respectively prior to the auto-negotiation mode of operation so that FLPs can be transmitted and received in compliance with Clause 28 Auto-Negotiation specifications. An algorithm that controls the switching function determines the correct polarization of the crossover circuit. This algorithm uses an 11-Bit Linear Feedback Shift Register (LFSR) to create a pseudo-random sequence that each end of the link uses to determine its proposed configuration. Upon making the selection to either MDI or MDI-X, the node waits for a specified amount of time while evaluating its receive channel to determine whether the other end of the link is sending link pulses or PHY-dependent data. If link pulses or PHY-dependent data are detected, it remains in that configuration. If link pulses or PHY-dependent data are not detected, it increments its LFSR and makes a decision to switch based on the value of the next bit. The state machine does not move from one state to another while link pulses are being transmitted.

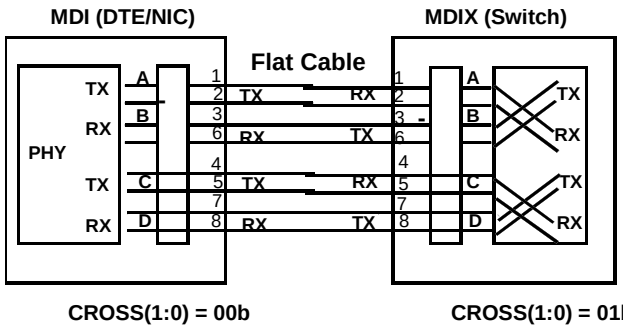


Figure 39. Crossover Function

9.4 Link Criteria

Once the link state is determined (via auto-negotiation, parallel detection or forced operation) the PHY and its link partner bring up the link.

9.4.1 1000BASE-T

For 1000BASE-T links, the PHY and its link partner enter a training phase. They exchange idle symbols and use the information gained to set their adaptive filter coefficients. These coefficients are used to equalize the incoming signal as well as eliminate signal impairments such as echo and cross talk.

Either side indicates completion of the training phase to its link partner by changing the encoding of the idle symbols it transmits. When both sides so indicate, the link is up. Each side continues sending idle symbols whenever it has no data to transmit. The link is maintained as long as valid idle or data symbols are received.

9.4.2 100BASE-TX

For 100BASE-TX links, the PHY and its link partner immediately begin transmitting idle symbols. Each side continues sending idle symbols each time it has no data to transmit. The link is maintained as long as valid idle symbols or data is received.

In 100 Mb/s mode, the PHY establishes a link each time the scrambler becomes locked and remains locked for approximately 50 ms. Link remains up unless the descrambler receives less than 12 consecutive idle symbols in any 2 ms period.

9.4.3 10BASE-T

For 10BASE-T links, the PHY and its link partner begin exchanging Normal Link Pulses (NLPs). The PHY transmits an NLP every 16 ms, and expects to receive one every 10 to 20 ms. The link is maintained as long as normal link pulses are received.

In 10 Mb/s mode, the PHY establishes link based on the link state machine found in 802.3, clause 14. 100 Mb/s idle patterns does not bring up a 10 Mb/s link.

9.5 Link Enhancements

The PHY offers two enhanced link functions, each of which are discussed in the sections that follow:

- SmartSpeed
- Flow Control

9.5.1 SmartSpeed

SmartSpeed is an enhancement to auto-negotiation that enables the PHY to react intelligently to network conditions that prohibit establishment of a 1000BASE-T link, such as cable problems. Such problems might enable auto-negotiation to complete, but then inhibit completion of the training phase. Normally, if a 1000BASE-T link fails, the PHY returns to the auto-negotiation state with the same speed settings indefinitely. With SmartSpeed enabled, after five failed attempts, the PHY automatically downgrades the highest ability it advertises to the next lower speed: from 1000 to 100 to 10 Mb/s. Once a link is established, and if it is later broken, the PHY automatically upgrades the capabilities advertised to the original setting.

9.5.1.1 Using SmartSpeed

SmartSpeed is enabled by setting bit 7 of the Port Configuration register to 1b. When SmartSpeed downgrades the PHY advertised capabilities, it sets bit 15 of the Link Health register. When link is established, its speed is indicated in Port Status 1 register, bits 15:14. SmartSpeed automatically resets the highest-level auto-negotiation abilities advertised, if link is established and then lost for more than two seconds. the number of failed attempts allowed is configured by the Misc Cntrl register, bits 8:6.

Note: When SmartSpeed is enabled, the M/S (Master-Slave) resolution is not given seven attempts to try to resolve M/S status (see IEEE 802.3 clause 40.5.2), this is due to the fact that SmartSpeed downgrades the link after at most five attempts.

9.5.2 Flow Control

Flow control is a function that is described in Clause 31 of the IEEE 802.3 standard. It enables congested nodes to pause traffic. Flow control is essentially a MAC-to-MAC function. MACs indicate their ability to implement flow control during auto-negotiation. This ability is communicated through two bits in the auto-negotiation registers (Auto-Negotiation Advertisement register, bits 10 and 11).

The PHY transparently supports MAC-to-MAC advertisement of flow control through its auto-negotiation process. Prior to auto-negotiation, the MAC indicates its flow control capabilities via the Auto-Negotiation Advertisement register, bits 10 and 11. After auto-negotiation, the link partner's flow control capabilities are indicated in the Auto-Negotiation Base Page Ability register, bits 11:10.

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

Table 50 lists the intended operation for the various settings of ASM_DIR and Pause. This information is provided for reference only; it is the responsibility of the MAC to implement the correct function. The PHY merely enables the two MACs to communicate their abilities to each other.

Table 50. Pause And Asymmetric Pause Settings

ASM_DIR Settings Local (PHY Register 4d, Bit 11) and Remote (PHY Register 5d, Bit 11)	Pause Setting - Local (PHY Register 4d, Bit 10)	Pause Setting - Remote (PHY Register 5d, Bit 10)	Result
Both ASM_DIR = 1b	1b	1b	Symmetric - Either side can flow control the other
	1b	0b	Asymmetric - Remote can flow control local only
	0b	1b	Asymmetric - Local can flow control remote
	0b	0b	No flow control
Either or both ASM_DIR = 0b	1b	1b	Symmetric - Either side can flow control the other
	Either or both = 0b		No flow control

9.6 Management Data Interface

The PHY supports the IEEE 802.3 MII Management Interface also known as the Management Data Input/Output (MDIO) Interface. The MDIO interface consists of a physical connection to the MAC, a specific protocol which runs across the connection, and a 16-bit MDIO register set.

PHY registers 0d through 10d and 15d are required and their functions are specified by the IEEE 802.3 specification. Additional registers are included for expanded functionality.

9.7 Low Power Operation and Power Management

The PHY enters a low-power state according to MAC control (Power Management controls) or via the PHY Control register. In either power down mode, the PHY is not capable of receiving or transmitting packets.

9.7.1 Power Down via the PHY Register

The PHY can be powered down using the control bit found in bit 11 of the PHY Control register. This bit powers down a significant portion of the port but clocks to the register section remain active. This enables the PHY management interface to remain active during power-down. The power-down bit is active high. When the PHY exits software power-down (bit 11 of the PHY Control = 0b), it re-initializes all analog functions, but retains its previous configuration settings.

9.7.2 Link Speed Control

Normal PHY speed negotiation drives to establish a link at the highest possible speed. This section describes how to deviate from this mode of operation.

9.7.2.1 Low Power Link Up (LPLU)

The **82566**/**82567** supports a mode of operation where the PHY drives to establish a link at a low speed. The link-up process enables a link to come up at the lowest possible speed in cases where power is more important than performance.

When speed negotiation starts, the PHY tries to negotiate for a 10 Mb/s link, independent of speed advertisement. If link establishment fails, the PHY tries to negotiate with different speeds; it enables all speeds up to the lowest speed supported by the partner. For example, the **82566**/**82567** advertises 10 Mb/s only and the partner supports 1000 Mb/s only. After the first try fails, the **82566**/**82567** enables 10/100/1000 Mb/s and tries again. The PHY continues to try and establish a link until it succeeds or until it is instructed otherwise. In the second step (adjusting to partner speed), the PHY also enables parallel detect (if needed). Note that automatic MDI/MDI-X resolution is done during the first auto-negotiation stage.

LPLU is controlled through the *LPLU* bit in the PHY Power Management register. The MAC sets and clears the bit according to hardware/software settings. the **82566**/**82567** auto-negotiates with the updated LPLU setting on the following auto-negotiation operation. The **82566**/**82567** does not automatically auto-negotiate after a change in the LPLU value.

9.7.2.2 GbE Disable

Table 51 lists power limitations introduced by different form factors.

Table 51. Power Limits by Form-Factor

	Form Factor	
	LOM[1]	PCI add-in card (x1 connector)
Main	3 A @ 3.3V	3 A @ 3.3V
Auxiliary (aux enabled)	375 mA @ 3.3V	375 mA @ 3.3V
Auxiliary (aux disabled)	20 mA @ 3.3V	

1. This is not a formal requirement but a policy that some OEMs take based on their choice.

Note: Auxiliary current limit only applies when the primary 3.3V voltage source is not available. For example, the PHY is in a low power D3 state.

The **82566**/**82567** exceeds the allowed power consumption in GbE speed. As a result, it cannot run from auxiliary power thus restricting PHY speed in Dr state.

GbE speed enabling is controlled through the *Disable1000* bit in PHY Power Management register. The MAC sets and clears this bit according to hardware/software settings. The **82566**/**82567** auto-negotiates with the updated setting on the following auto-negotiation operation. The **82566**/**82567** does not automatically auto-negotiate after a change in the Disable1000 value.

9.7.3 Link Disconnect; Smart Power-Down (82566 Only)

The link-disconnect state applies to all power management states (Dr, D0u, D0a, D3). Note that the link might resume and be lost while in any of these states. This section defines the minimal power management capabilities in the link-disconnect state. In some cases, the PHY enters a deeper power management state than defined here. For example, if there is no requirement to maintain the link, the PHY enters a deeper power management state in the non-D0 states.

PHY Smart Power Down (SPD) is a PHY mechanism to reduce PHY power when the link is lost. The PHY enters this state when is detects a Link lost state and the PHY *SPD* bit is set.

Note: The PHY does not enter the SPD state unless auto-negotiation is enabled.

While in the SPD state, the PHY powers down all circuits not required for detection of link activity. The PHY must still be able to detect link pulses (including parallel detect) and wake up to engage in link negotiation.

While in a link-disconnect state, the PHY must allow software access to its registers.

9.7.3.1 Back-to-Back Smart Power-Down

While in Link Disconnect, the **82566** monitors the link for link pulses to identify when a link is re-connected. The **82566** also periodically transmits pulses to resolve the case of two **82566** devices (or devices with **82566**-like behavior) connected to each other across the link. Otherwise, two such devices might be locked in Smart Power-Down mode and not capable of identifying that a link was re-connected.

The link pulses are transmitted every 100 ms and add <1% to the total **82566** power in Link Disconnect mode. Pulses might not conform to IEEE specification regarding link pulse template. A single pulse should be enough to bring a receiver out of SPD mode in a worst-case configuration (for example, maximum cable length, highest cable attenuation, etc.).

If the link partners are disconnected and then reconnected, it is possible that the two controllers transmit their pulses at the same time. Since the **82566** masks its receiver during pulse transmission, such synchronization causes pulses to be missed by both partners. A randomization factor is therefore applied to the timing of transmitted pulses, affecting the period between pulses. The randomization factor is specific per device and should reduce the probability of a lock by 10^{-4}. Note that if the two partners happen to transmit within the same slot, and if the randomization factor happens to be similar, it might take longer for the partners to get out of sync with each other.

Back-to-back SPD is enabled by the *SPD_B2B_EN* bit in the PHY registers. The default value is enabled. The enable bit applies in SPD mode.

Note: This bit should not be altered by software once the **82566** was set in Smart Power-Down mode. If software requires changing the back-to-back status, it first needs to transition the PHY out of Smart Power-Down mode, and only then change the back-to-back bit to the required state.

9.7.4 Link Energy Detect

The PHY de-asserts the *Link Energy Detect* Bit (PHY Power Management register, bit 4) each time energy is not detected on the link. This bit provides an indication of a cable being plugged in or unplugged.

This bit is valid only if auto-negotiation is enabled and SPD is enabled (PHY Power Management register, bit 0).

In order to correctly deduce that there is no energy, the bit must read 0b for three consecutive reads each second.

9.7.5 Energy Detect Power Down Modes (82567)

The **82567** can be placed in energy detect power down modes by selecting either of the two energy detect modes. Both modes enable the **82567** to wake up on its own by detecting activity on the CAT 5 cable. The status of the energy detect is reported in the Copper Specific Status Register 1 (bit 4) and the energy detect changes are reported in the Copper Specific Status Register 2 (bit 4).

9.7.6 Energy Detect; Mode 1 (82567)

Energy Detect (Mode 1) is entered by setting bits 9:8 in the Copper Specific Control Register 1 to 10b. In Mode 1, only the signal detection circuitry and serial management interface are active. If the **82567** detects energy on the line, it starts to auto-negotiate sending FLPs for five seconds. If at the end of five seconds auto-negotiation has not completed, then the **82567** stops sending FLPs and goes back to monitoring receive energy. If auto-negotiation completes, then the **82567** goes into normal 10/100/1000 Mb/s operation. If during normal operation the link is lost, the **82567** re-starts auto-negotiation. If no energy is detected after five seconds, the **82567** goes back to monitoring receive energy.

9.7.7 Energy Detect +*; Mode 2 (82567)

Energy Detect (Mode 2) is entered by setting bits 9:8 in the Copper Specific Control Register 1 to 11b. In Mode 2, the **82567** sends out a single 10 Mb/s Normal Link Pulse (NLP) every one second. This is the only difference between Mode 2 and Mode 1 operation. If the by is in Mode 1, it cannot wake up a connected device; therefore, the connected device must be transmitting NLPs, or either device must be woken up through register access. If the **82567** is in Mode 2, then it can wake a connected device.

9.7.8 Power State Upon Exiting Power Down (82567)

When the **82567** exits power down (bit 11 of the PHY Control register or bit 2 of the Copper Specific Control Register 1) the active state depends upon whether the energy detect function is enabled (Copper Specific Control Register 1, bits 9:8 = 1x). If the energy detect function is enabled, the **82567** transitions to the energy detect state first and wakes up only if there is a signal on the wire.

Table 52. Power State after Exiting Power Down (82567)

PHY Control Register (Bit 11)	Copper Specific Control Register 1 (Bit 2)	Copper Specific Control Register 1 (Bits 9:8)	Mode
1b	X	XX	Power down.
X	1b	XX	Power down.
1b to 0b	0b	00b	Transition to power up.
0b	1b to 0b	00b	Transition to power up.
1b to 0b	0b	1Xb	Transition to energy detect state.
0b	1b to 0b	1Xb	Transition to energy detect state.

9.7.9 Normal 10/100/1000 Mb/s Operation (82567)

Normal 10/100/1000 Mb/s operation can be entered by turning off energy detect mode (set bits 9:8 of the Copper Specific Control Register 1 to 00b).

9.8 1000 Mb/s Operation

9.8.1 Introduction

This section provides an overview of 1000BASE-T functions, followed by discussion and review of the internal functional blocks shown in Figure 40.

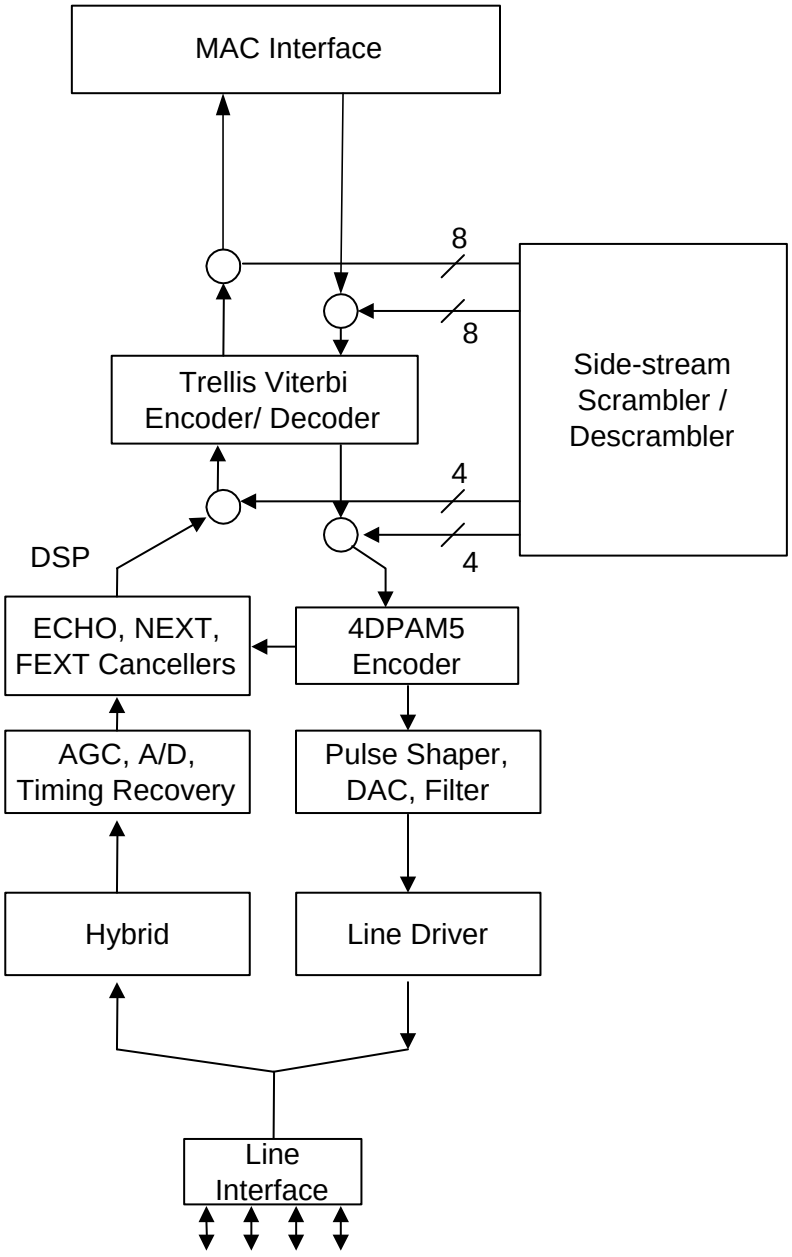


Figure 40. 1000 Base-T PHY Functions Overview

9.8.2 Transmit Functions

This section describes functions used when the MAC transmits data through the PHY and out onto the twisted-pair connection.

9.8.2.1 Scrambler

The scrambler randomizes the transmitted data. The purpose of scrambling is two fold:

1. Scrambling eliminates repeating data patterns from the 4DPAM5 waveform to reduce EMI.
2. Each channel (A, B, C, D) gets a unique signature that the receiver uses for identification.

The scrambler is driven by a Linear Feedback Shift Register (LFSR), which is randomly loaded at power-up. The LFSR function used by the Master differs from that used by the Slave, giving each direction its own unique signature. The LFSR, in turn, generates uncorrelated outputs. These outputs randomize the inputs to the 4DPAM5 and Trellis encoders and randomize the sign of the 4DPAM5 outputs.

9.8.3 Transmit FIFO

The transmit FIFO re-synchronizes data transmitted by the MAC to the transmit reference used by the PHY.

For the **82567**, The depth of the transmit FIFO can be programmed via the MAC Specific Control Register 1 (bits 15:14). The FIFO depths can be increased in length by programming this register to support longer frames. The **82567** can handle jumbo frame sizes up to 10 KB with up to ± 150 ppm clock jitter. The deeper the FIFO depth, the higher the latency is. The status of the FIFO can be interrogated. The MAC Specific Status register (bit 7) indicates the transmit FIFO overflow and underflow status. Bits 2 and 3 of the MAC Specific Status register are set depending on whether the FIFO inserted or deleted idle symbols. Idles inserted or deleted are flagged only if the inter-packet gap is 24 bytes or less at the input of the FIFO. Inserted or deleted idles are ignored if the inter-packet gap is greater than 24 bytes. The FIFO status bits can generate interrupts by setting the corresponding bits in the MAC Specific Interrupt Enable Register register.

9.8.3.1 Transmit Phase-Locked Loop PLL

This function generates the 125 MHz timing reference used by the PHY to transmit 4DPAM5 symbols. When the PHY is the Master side of the link, the crystal input is the reference for the transmit PLL. When the PHY is the Slave side of the link, the recovered receive clock is the reference for the transmit PLL.

9.8.3.2 Trellis Encoder

The Trellis Encoder uses the two high-order bits of data and its previous output to generate a ninth bit, which determines if the next 4DPAM5 pattern should be even or odd. This function provides forward error correction and enhances the signal-to-noise (SNR) ratio by a factor of 6 dB.

9.8.3.3 4DPAM5 Encoder

The 4DPAM5 encoder translates 8B codes transmitted by the MAC into 4DPAM5 symbols. The encoder operates at 125 Mhz, which is both the frequency of the MAC interface and the baud rate used by 1000BASE-T.

Each 8B code represents one of 256 data patterns. Each 4DPAM5 symbol consists of one of five signal levels (-2,-1,0,1,2) on each of the four twisted pair (A,B,C,D) representing 5^4 or 625 possible patterns per baud period. Of these, 113 patterns are reserved for control codes, leaving 512 patterns for data. These data patterns are divided into two groups of 256 even and 256 odd data patterns. As a result, each 8B octet has two possible 4DPAM5 representations—one even and one odd pattern.

9.8.3.4 Spectral Shaper

This function causes the 4DPAM5 waveform to have a spectral signature that is very close to that of the MLT3 waveform used by 100BASE-TX. This enables 1000BASE-T to take advantage of infrastructure (cables, magnetics) designed for 100BASE-TX.

The shaper works by transmitting 75% of a 4DPAM5 code in the current baud period, and adding the remaining 25% into the next baud period.

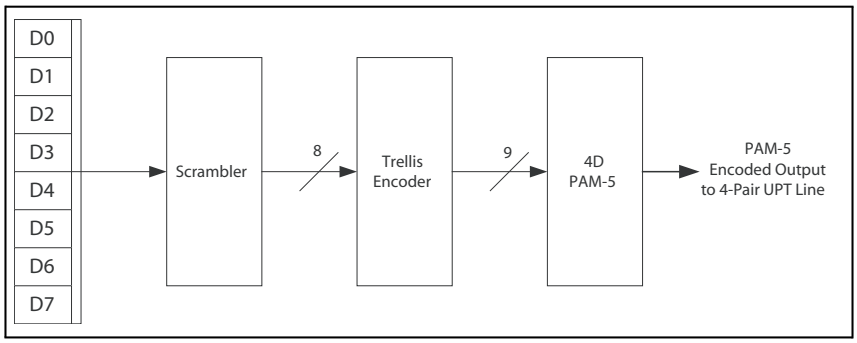
9.8.3.5 Low-Pass Filter

To aid with EMI, this filter attenuates signal components more than 180 Mhz. In 1000BASE-T, the fundamental symbol rate is 125 MHz.

9.8.3.6 Line Driver

The line driver drives the 4DPAM5 waveforms onto the four twisted-pair channels (A, B, C, D), adding them onto the waveforms that are simultaneously being received from the link partner.

9.8.3.7 Transmit/Receive Flow



Scrambler Polynomials:

- $1 + x^{13} + x^{33}$ (Master PHY Mode)
- $1 + x^{20} + x^{33}$ (Slave PHY Mode)

Figure 41. 1000BASE-T Transmit Flow And Line Coding Scheme

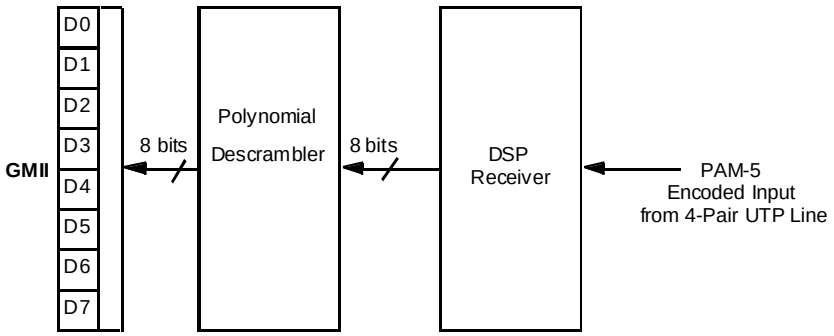


Figure 42. 1000BASE-T Receive Flow

9.8.4 Receive Functions

This section describes function blocks that are used when the PHY receives data from the twisted pair interface and passes it back to the MAC.

9.8.4.1 Hybrid

The hybrid subtracts the transmitted signal from the input signal, allowing the use of simple 100BASE-TX compatible magnetics.

9.8.4.2 Automatic Gain Control

The Automatic Gain Control (AGC) normalizes the amplitude of the received signal, adjusting for the attenuation produced by the cable.

9.8.4.3 Timing Recovery

This function re-generates a receive clock from the incoming data stream which is used to sample the data. On the slave side of the link, this clock is also used to drive the transmitter.

9.8.4.4 Analog-to-Digital Converter

The Analog-to-Digital (ADC) function converts the incoming data stream from an analog waveform to digitized samples for processing by the DSP core.

9.8.4.5 Digital Signal Processor

The Digital Signal Processor (DSP) provides per-channel adaptive filtering, which eliminates various signal impairments including:

- Inter-symbol interference (equalization).
- Echo caused by impedance mismatch of the cable.
- Near-end crosstalk (NEXT) between adjacent channels (A, B, C, D).
- Far-end crosstalk (FEXT)
- Propagation delay variations between channels of up to 120 ns.
- Extraneous tones that have been coupled into the receive path.

The adaptive filter coefficients are initially set during the training phase. They are continuously adjusted (adaptive equalization) during operation through the decision-feedback loop.

9.8.4.6 Descrambler

The descrambler identifies each channel by its characteristic signature, removing the signature and re-routing the channel internally. In this way, the receiver can correct for channel swaps and polarity reversals. The descrambler uses the same base LFSR used by the transmitter on the other side of the link.

The descrambler requires approximately 15 μs. to lock, normally accomplished during the training phase.

9.8.4.7 Viterbi Decoder/Decision Feedback Equalizer (DFE)

The Viterbi decoder generates clean 4DPAM5 symbols from the output of the DSP. The decoder includes a Trellis encoder identical to the one used by the transmitter. The Viterbi decoder simultaneously looks at the received data over several baud periods. For each baud period, it predicts whether the symbol received should be even or odd, and compares that to the actual symbol received. The 4DPAM5 code is organized in such a way that a single level error on any channel changes an even code to an odd one and vice versa. In this way, the Viterbi decoder can detect single-level coding errors, effectively improving the Signal-To-Noise (SNR). When an error occurs, this information is quickly fed back into the equalizer to prevent future errors.

9.8.4.8 4DPAM5 Decoder

The 4DPAM5 decoder generates 8B data from the output of the Viterbi decoder.

9.9 100 Mb/s Operation

The MAC passes data to the PHY over the MII. The PHY encodes and scrambles the data, then transmits it using MLT-3 for 100TX over copper. The PHY descrambles and decodes MLT-3 data received from the network. When the MAC is not actively transmitting data, the PHY sends out idle symbols on the line.

9.10 10 Mb/s Operation

The PHY operates as a standard 10 Mb/s transceiver. Data transmitted by the MAC as 4-bit nibbles is serialized, Manchester-encoded, and transmitted on the MDI[0]+/- outputs. Received data is decoded, de-serialized into 4-bit nibbles and passed to the MAC across the internal MII. The PHY supports all the standard 10 Mb/s functions.

9.10.1 Link Test

In 10 Mb/s mode, the PHY always transmits link pulses. If the Link Test Function is enabled, it monitors the connection for link pulses. Once it detects 2 to 7 link pulses, data transmission is enabled and remains enabled as long as the link pulses or data reception continues. If the link pulses stop, the data transmission is disabled.

If the Link Test function is disabled, the PHY might transmit packets regardless of detected link pulses. Setting the Port Configuration register (bit 14) can disable the Link Test function.

9.10.2 10Base-T Link Failure Criteria and Override

Link failure occurs if link test is enabled and link pulses stop being received. If this condition occurs, the PHY returns to the auto-negotiation phase if auto-negotiation is enabled. Setting the Port Configuration register (bit 14) disables the link integrity test function, then the PHY transmits packets, regardless of link status.

9.10.3 Jabber

If the MAC begins a transmission that exceeds the jabber timer, the PHY disables the transmit and loopback functions and asserts collision indication to the MAC. The PHY automatically exits jabber mode after 250 to 750 ms. This function can be disabled by setting the Port Configuration register (bit 10) to 1b.

9.10.4 Polarity Correction

The PHY automatically detects and corrects for the condition where the receive signal (MDI_PLUS[0]/MDI_MINUS[0]) is inverted. Reversed polarity is detected if eight inverted link pulses, or four inverted end-of-frame markers, are received consecutively. If link pulses or data are not received for 96 to 130 ms, the polarity state is reset to a non-inverted state. Automatic polarity correction can be disabled by setting the Misc Cntrl register, bit 5.

9.10.5 Dribble Bits

The PHY device handles dribble bits for all of its modes. If between one to four dribble bits are received, the nibble is passed across the interface. The data passed across is padded with 1b's if necessary. If between five to seven dribble bits are received, the second nibble is not sent onto the internal MII bus to the MAC. This ensures that dribble bits between 1-7 do not cause the MAC to discard the frame due to a CRC error.

9.11 Downshift Feature (82567)

Connecting between two GbE link partners requires a four-pair RJ-45 cable to establish 10, 100, or 1000 Mb/s link. However, there are existing cables that have only two-pairs, which are used to connect 10 Mb/s and 100 Mb/s Ethernet PHYs. With the availability of only pairs 1,2 and 3,6, GbE link partners can auto-negotiate to 1000 Mb/s, but fail to link. The **82567** repeatedly goes through the auto-negotiation but fails 1000 Mb/s link and never tries to link at 10 Mb/s or 100 Mb/s. With the downshift feature enabled, the **82567** is able to auto-negotiate with another GbE link partner using cable pairs 1,2 and 3,6 to downshift and link at 10 Mb/s or 100 Mb/s, whichever is the next highest advertised speed common between the two GbE PHYs. If a three pair cable (additional pair 4,5 or 7,8 - but not both) is used, the same downshift function for two-pair cables applies. By default, the downshift feature is turned off. To enable the downshift feature, the following bits must be set in the Copper Specific Control Register 1:

- When bit 11 = 1, downshift is enabled
- Bits 14:12 - Sets the number of link attempts before downshifting

9.12 PHY Loopback

Loopback is supported in the PHY. Software or firmware should set the PHY to the loopback mode via the MDIC register by writing to the PHY Control register. The MAC must be in forced link and in full duplex mode for PHY loopback to operation. The following configurations are needed to support PHY loopback:

- CTRL.FRCDPLX = 1b - // Force duplex mode by the MAC
- CTRL.FD = 1b - // Set full duplex mode
- GLCI Diagnostic.NELPBK = 0b // Nominal MAC operation

Note: This page intentionally left blank.

10.0 Register Descriptions

10.1 Introduction

This section details the state inside the MAC and PHY that are visible to the programmer. In some cases, it describes hardware structures invisible to software in order to clarify a concept.

Note: There are two separate areas of the MAC that are separated by Virtual Channels (VC). A virtual channel acts as a separate device and is identified by the virtual channel field in the PCI header. Note that these are similar, but different from functions. Each of these virtual channels are actually function #0. Also note that these two virtual channel devices are on the same physical and logical bus.

10.2 Host PCI Configuration Space on PCI Interface

PCI is the host interface that is seen by the operating system. The MAC's address space is mapped into regions with PCI Base Address registers as listed in Table 53.

Table 53. PCI Address Space

Addressable Content	How Mapped	Size of Region
Internal registers and memories	Direct memory mapped	128 KB
Internal registers and memories	I/O Window mapped	32 Bytes

The internal registers and memories can be accessed though I/O space by doing a level of indirection as explained in Section 10.2.1.2.

The internal register/memory space is described in the following sections and divided up into the following categories:

- General
- Interrupt
- MAC Receive
- MAC Transmit
- Statistics
- Management/Wake Up
- Time Sync (**ICH10** only)
- Diagnostics (including packet buffer memory access)
- LinkSec (**ICH10** only)
- PHY Receive, Transmit and Special Function

Note: The PHY registers are accessed indirectly through the MDI/O interface. For **82562V** PHY register descriptions, refer to the *82562V 10/100 Mbps Platform LAN Connect (PLC) Datasheet*.

10.2.1 Memory and I/O Address Decoding

10.2.1.1 Memory-Mapped Access to Internal Registers and Memories

The internal registers and memories can be accessed as direct memory-mapped offsets from the base address register (BAR0/BAR1). The CSR space consists of 32-bit registers and as a result, write accesses should be limited to whole 32-bit dword cycles.

10.2.1.2 I/O-Mapped Access to Internal Registers and Memories

To support pre-boot operation (prior to the allocation of physical memory base addresses), all internal registers, memories, and Flash 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 PCI 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 as a window to the register or memory address specified by IOADDR.Write access to both IOADDR and IODATA registers should be limited to whole 32-bit dword cycles.

Offset	Abbreviation	Name	RW	Size
00h	IOADDR	Internal Register, Internal or Memory Location Address 00000h - 1FFFFh — Internal Registers and Memories 20000h - 7FFFFh — Undefined	RW	4 Bytes
04h	IODATA	Data field for reads or writes to the Internal Register, Internal or Memory location as identified by the current value in IOADDR. All 32 bits of this register are read/write-able.	RW	4 Bytes
08h:1Fh	Reserved	Reserved.	RO	4 Bytes

10.2.1.3 IOADDR

The IOADDR register must always be written as a dword access. Writes that are less than 32 bits are ignored. Reads of any size always return 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 PCI bus. Since 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.

Since only a particular range is addressable, the upper bits of this register are hard coded to 0b. Bits 31 through 20 are not write-able and always read back as 0b.

At hardware reset (LAN_RST#) or PCI reset, this register value resets to 00h. Once written, the value is retained until the next write or reset.

10.2.1.4 IODATA

The IODATA register must always be written as a Dword access when the IOADDR register contains a value for the Internal Register and Memories (00000h - 1FFFCh).

Table 54. IODATA Register Configurations

Access Type	IOADDR Register Bits [1:0]	Target IODATA Access BE[3:0]# Bits in Data Phase
DWORD (32 bits)	00b	0000b

For software programmers, the IN and OUT instructions must be used to cause I/O cycles to be used. Where 32-bit quantities are required on writes, the source register of the OUT instruction must be EAX (the only 32-bit register supported by the OUT command).

Writes and reads to IODATA when the IOADDR register value is in an undefined range (20000h:7FFFCh) should not be performed. Results can be indeterminate.

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 Ethernet controller delays the results through normal bus methods. For example, a split transaction or transaction retry.

Note: Because a register/memory/flash read or write takes two IO cycles to complete, software must provide a guarantee that the two IO cycles occur as an atomic operation. Otherwise, results can be indeterminate.

10.2.1.5 Undefined I/O Offsets

I/O offsets 08h through 1Fh are considered to be reserved offsets with the I/O window. Dword reads from these addresses will return FFFFh; writes to these addresses are discarded.

10.3 PCI Registers

See Section 4.3 for detailed information about PCI registers.

Register Based Legend:

- RW - Read write register.
- RO - Read only register.
- RW/RO - Read write by firmware; read only by software.
- R/WC - Read write clear registers. Writing 0b has no affect. Writing 1b clears the appropriate fields (see detailed description of the specific registers).
- RC/WC - Read write clear registers. Writing 0b has no affect. Writing 1b clears the appropriate fields. Note that a read might also clear the register depending on enablement (see appropriate registers).
- WO - Write only registers. Reading from these registers does not reflect any meaningful data. Generally this would be all zero's (see detailed description of appropriate registers).

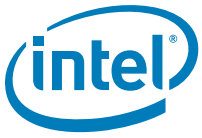
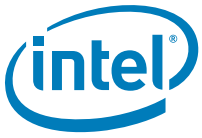


Table 55. PCI Register Summary

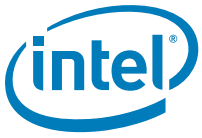
Category	Offset	Abbreviation	Name	R/W
General Register Descriptions				
General	00000h	CTRL	Device Control	R/W
General	00008h	STATUS	Device Status	RO
General	0000Ch	STRAP	Strapping Option (**ICH9 / ICH10**)	RO
General	00010h	EEC	NVM/Flash Control (Not applicable to **ICH10**)	R/W
General	00018h	CTRL_EXT	Extended Device Control	R/W
General	00020h	MDIC	MDI Control	R/W
General	00028h	FEXTNVM	Future Extended NVM	R/W
General	0002Ch	FEXT	Future Extended	R/W
General	00038h	BUSNUM	Device and Bus Number (**ICH9 / ICH10**)	RO
General	00170h	FCTTV	Flow Control Transmit Timer Value	R/W
General	05F40h	FCRTV	Flow Control Refresh Threshold Value (**ICH10**)	R/W
General	00E00h	LEDCTL	LED Control	R/W
General	00F00h	EXTCNF_CTRL	Extended Configuration Control	R/W
General	00F08h	EXTCNT_SIZE	Extended Configuration Size	R/W
General	00F10h	PHY_CTRL	PHY Control Register	R/W
General	00F18h	PCIANACFG	PCI Analog Configuration (**ICH10**)	R/W
General	01000h	PBA	Packet Buffer Allocation	R/W
General	01008h	PBS	Packet Buffer Size	R/W
Interrupt Register Descriptions				
Interrupt	000C0h	ICR	Interrupt Cause Read	RC/WC
Interrupt	000C4h	ITR	Interrupt Throttling Rate	R/W
Interrupt	000C8h	ICS	Interrupt Cause Set	WO
Interrupt	000D0h	IMS	Interrupt Mask Set/Read	R/W
Interrupt	000D8h	IMC	Interrupt Mask Clear	WO
Interrupt	000E0h	Mask - IAM	Interrupt Acknowledge Auto Mask	R/W
Receive Register Descriptions				
Receive	00100h	RCTL	Receive Control	R/W
Receive	00104h	RCTL1	Receive Control 1 (**ICH10**)	R/W
Receive	02008h	ERT	Early Receive Threshold	R/W
Receive	02160h	FCRTL	Flow Control Receive Threshold Low	R/W
Receive	02168h	FCRTH	Flow Control Receive Threshold High	R/W
Receive	02170h 02170h + n*4h [n+0..1) (**ICH10**)	PSRCTL	Packet Split Receive Control	R/W

Category	Offset	Abbreviation	Name	R/W
Receive	02800h + n*100h [n+0..1) (**ICH10**)	RDBAL0 RDBAL (**ICH10**)	Receive Descriptor Base Low Queue 0	R/W
Receive	02804h 02804h + n*100h [n+0..1) (**ICH10**)	RDBAH0 RDBAH (**ICH10**)	Receive Descriptor Base High Queue 0	R/W
Receive	02808h 02808h + n*100h [n+0..1) (**ICH10**)	RDLEN0 RDLEN (**ICH10**)	Receive Descriptor Length Queue 0	R/W
Receive	02810h 02810h + n*100h [n+0..1) (**ICH10**)	RDH0 RDH (**ICH10**)	Receive Descriptor Head Queue 0	R/W
Receive	02818h 02818h + n*100h [n+0..1) (**ICH10**)	RDT0 RDT (**ICH10**)	Receive Descriptor Tail Queue 0	R/W
Receive	02820h 02820h + n*100h [n+0..1) (**ICH10**)	RDTR RDTR (**ICH10**)	Receive Interrupt Packet Delay Timer	R/W
Receive	02828h 02828h + n*100h [n+0..1) (**ICH10**)	RXDCTL	Receive Descriptor Control	R/W
Receive	0282Ch	RADV	Receive Interrupt Absolute Delay Timer	R/W
Receive	02900h	RDBAL1	Receive Descriptor Base Low Queue 1 (not applicable to **ICH10**)	R/W
Receive	02904h	RDBAH1	Receive Descriptor Base High Queue 1 (not applicable to **ICH10**)	R/W
Receive	02908h	RDLEN1	Receive Descriptor Length Queue 1 (not applicable to **ICH10**)	R/W
Receive	02910h	RDH1	Receive Descriptor Head Queue 1 (not applicable to **ICH10**)	R/W
Receive	02918h	RDT1	Receive Descriptor Tail Queue 1 (not applicable to **ICH10**)	R/W
Receive	02C00h	RSRPD	Receive Small Packet Detect	R/W
Receive	02C08h	RAID	Receive ACK Interrupt Delay	R/W
Receive	02C10h	CPUVEC	CPU Vector	R/W
Receive	05000h	RXCSUM	Receive Checksum Control	R/W
Receive	05008h	RFCTL	Receive Filter Control	R/W
Receive	05200h- 0527Ch	MTA[31:0]	Multicast Table Array (n)	R/W
Receive	05400h 05400h + 8*n (n=0..6) (**ICH10**)	RAL0 RAL (**ICH10**)	Receive Address Low 0	R/W
Receive	05404h 05404h + 8*n (n=0..6) (**ICH10**)	RAH0 RAH (**ICH10**)	Receive Address High 0	R/W
Receive	05408h	RAL1	Receive Address Low 1 (not applicable to **ICH10**)	R/W
Receive	0540Ch	RAH1	Receive Address High 1 (not applicable to **ICH10**)	R/W
	...	...	...	...
Receive	05430h	RAL6	Receive Address Low 6 (not applicable to **ICH10**)	R/W



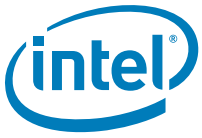
Category	Offset	Abbreviation	Name	R/W
Receive	05434h	RAH6	Receive Address High 6 (not applicable to **ICH10**)	R/W
Receive	05438h 05438h + 8*n (n=0..3) (**ICH10**)	SHRAL0 SHRALn (**ICH10**)	Shared Receive Address Low 0	R/W
Receive	05438C + 8*n (n=0..3) (**ICH10**)	SHRAH0 SHRAHn (**ICH10**)	Shared Receive Address High 0	R/W
Receive		SHRAL1	Shared Receive Address Low 1 (not applicable to **ICH10**)	R/W
Receive		SHRAH1	Shared Receive Address High 1 (not applicable to **ICH10**)	R/W
	...	...	...	...
Receive		SHRAL3	Shared Receive Address Low 3 (not applicable to **ICH10**)	R/W
Receive	05454h	SHRAH3	Shared Receive Address High 3 (not applicable to **ICH10**)	R/W
Receive	05818h	MRQC	Multiple Receive Queues Command	R/W
Receive	05864h	RSSIM	RSS Interrupt Mask	R/W
Receive	05868h	RSSIR	RSS Interrupt Request	R/W
Receive	05C00h + 4*n (n=0...9) 05C00h-05C7Ch (**ICH9**) 05C00h + 4*n (n=0...31) (**ICH10**)	RETA	Redirection Table	R/W
Receive	05C80h + 4*n (=0...9) 05C80h-05CA4h (**ICH9**) 05C80h + 4*n (n=0...9) (**ICH10**)	RSSRK	RSS Random Key	R/W
Transmit Register Descriptions				
Transmit	00400h	TCTL	Transmit Control	R/W
Transmit	00410h	TIPG	Transmit IPG	R/W
Transmit	00458h	AIT	Adaptive IFS Throttle	R/W
Transmit	03004h	KABGTXD	GLCI AFE Band Gap Transmit Reference Data (**ICH8**)	R/W
Transmit	03800h 03800h + n*100[n=0..1) (**ICH10**)	TDBAL0 TDBAL (**ICH10**)	Transmit Descriptor Base Low 0	R/W
Transmit	03804h 03804h + n*100[n=0..1) (**ICH10**)	TDBAH0 TDBAH (**ICH10**)	Transmit Descriptor Base High 0	R/W
Transmit	03808h 03808h + n*100[n=0..1) (**ICH10**)	TDLEN0 TDLEN (**ICH10**)	Transmit Descriptor Length 0	R/W
Transmit	03810h 03810h + n*100[n=0..1) (**ICH10**)	TDH0 TDH (**ICH10**)	Transmit Descriptor Head 0	R/W

Category	Offset	Abbreviation	Name	R/W
Transmit	03818h 03818h + n*100[n=0..1) (**ICH10**)	TDT0 TDT (**ICH10**)	Transmit Descriptor Tail 0	R/W
Transmit	03820h	TIDV	Transmit Interrupt Delay Value	R/W
Transmit	03828h 03828h + n*100[n=0..1) (**ICH10**)	TXDCTL0 TXDCTL (**ICH10**)	Transmit Descriptor Control 0	R/W
Transmit	0382Ch	TADV	Transmit Absolute Interrupt Delay Value	R/W
Transmit	03840h3840h + n*100[n=0..1) (**ICH10**)	TARC0 TARC (**ICH10**)	Transmit Arbitration Counter Queue 0	R/W
Transmit	03900h	TDBAL1	Transmit Descriptor Base Low Queue 1 (not applicable to **ICH10**)	R/W
Transmit	03904h	TDBAH1	Transmit Descriptor Base High Queue 1 (not applicable to **ICH10**)	R/W
Transmit	03908h	TDLEN1	Transmit Descriptor Length Queue 1 (not applicable to **ICH10**)	R/W
Transmit	03910h	TDH1	Transmit Descriptor Head Queue 1 (not applicable to **ICH10**)	R/W
Transmit	03918h	TDT1	Transmit Descriptor Tail Queue 1 (not applicable to **ICH10**)	R/W
Transmit	03928h	TXDCTL1	Transmit Descriptor Control 1 (not applicable to **ICH10**)	R/W
Transmit	03940h	TARC1	Transmit Arbitration Counter Queue 1 (not applicable to **ICH10**)	R/W
Statistics Register Descriptions				
Statistics	04000h	CRCERRS	CRC Error Count	RO
Statistics	04004h	ALGNERRC	Alignment Error Count (**ICH10**)	RO
Statistics	0400Ch	RXERRC	RX Error Count	RO
Statistics	04010h	MPC	Missed Packets Count	RO
Statistics	04014h	SCC	Single Collision Count	RO
Statistics	04018h	ECOL	Excessive Collisions Count	RO
Statistics	0401Ch	MCC	Multiple Collision Count	RO
Statistics	04020h	LATECOL	Late Collisions Count	RO
Statistics	04028h	COLC	Collision Count	RO
Statistics	04030h	DC	Defer Count	RO
Statistics	04034h	TNCRS	Transmit - No CRS	RO
Statistics	0403Ch	CEXTERR	Carrier Extension Error Count	RO
Statistics	04040h	RLEC	Receive Length Error Count	RO
Statistics	04048h	XONRXC	XON Received Count	RO
Statistics	0404Ch	XONTXC	XON Transmitted Count	RO
Statistics	04050h	XOFFRXC	XOFF Received Count	RO
Statistics	04054h	XOFFTXC	XOFF Transmitted Count	RO
Statistics	04058h	FCRUC	FC Received Unsupported Count	RO
Statistics	04074h	GPRC	Good Packets Received Count	RO



Category	Offset	Abbreviation	Name	R / W
Statistics	04078h	BPRC	Broadcast Packets Received Count	RO
Statistics	0407Ch	MPRC	Multicast Packets Received Count	RO
Statistics	04080h	GPTC	Good Packets Transmitted Count	RO
Statistics	04088h	GORCL	Good Octets Received Count (Low)	RO
Statistics	0408Ch	GORCH	Good Octets Received Count (Hi)	RO
Statistics	04090h	GOTCL	Good Octets Transmitted Count (Low)	RO
Statistics	04094h	GOTCH	Good Octets Transmitted Count (Hi)	RO
Statistics	040A0h	RNBC	Receive No Buffers Count	RO
Statistics	040A4h	RUC	Receive Undersize Count	RO
Statistics	040A8h	RFC	Receive Fragment Count	RO
Statistics	040ACh	ROC	Receive Oversize Count	RO
Statistics	040B0h	RJC	Receive Jabber Count	RO
Statistics	040B4h	MNGPRC	Management Packets Received Count	RO
Statistics	040B8h	MNGPDC	Management Packets Dropped Count	RO
Statistics	040BCh	MNGPTC	Management Pkts Transmitted Count	RO
Statistics	040C0h	TORL	Total Octets Received (Lo)	RO
Statistics	040C4h	TORH	Total Octets Received (Hi)	RO
Statistics	040C8h	TOTL	Total Octets Transmitted (Lo)	RO
Statistics	040CCh	TOTH	Total Octets Transmitted (Hi)	RO
Statistics	040D0h	TPR	Total Packets Received	RO
Statistics	040D4h	TPT	Total Packets Transmitted	RO
Statistics	040F0h	MPTC	Multicast Packets Transmitted Count	RO
Statistics	040F4h	BPTC	Broadcast Packets Transmitted Count	RO
Statistics	040F8h	TSCTC	TCP Segmentation Context Transmitted Count	RO
Statistics	040FCh	TSCTFC	TCP Segmentation Context Tx Fail Count (not applicable to **ICH10**)	RO
Statistics	04100h	IAC	Interrupt Assertion Count	RO
Management Register Descriptions				
MNG	05800h	WUC	Wakeup Control	R/W
MNG	05808h	WUFC	Wakeup Filter Control	R/W
MNG	05810h	WUS	Wakeup Status	R
MNG	05820h	MANC_S	Management Control Shadow	RO
MNG	05824h	MANC2_S	Management Control 2 Shadow	RO
MNG	05860h	MANC2H_S	Management Control to Host Shadow	RO
MNG	05838h	IPAV	IP Address Valid	R/W
MNG	05840h + 8*n (=1..3) 05840h- 05858h (**ICH9**)	IP4AT	IPv4 Address Table	R/W
MNG	05880h + 4*n (n=0..3) 05880h- 0588Fh (**ICH9**)	IP6AT	IPv6 Address Table	R/W

Category	Offset	Abbreviation	Name	R/W
MNG	05B50h	H2ME	Host to ME	R/W
MNG	05B54h	FWSM_S	Firmware Semaphore Shadow	RO
MNG	05F00h- 05F18h 05F00h + 8*n [n=0..3] (**ICH10**)	FFLT	Flexible Filter Length Table	R/W
MNG	09000h-093F8h 09000h + 8*n [n=0..3] (**ICH10**)	FFMT	Flexible Filter Mask Table	R/W
MNG	09800h- 09BF8h 09800h + 8*n [n=0..3] (**ICH10**)	FFVT	Flexible Filter Value Table	R/W
Time Sync Register Descriptions (ICH10)				
Time Sync	0B620	TSYNCRXCTL	RX Time Sync Control Register	RW
Time Sync	0B628	RXSTMPH	RX Timestamp High	RW
Time Sync	0B624	RXSTMPL	RX Timestamp Low	RW
Time Sync	0B62C	RXSATRL	RX Timestamp Attributes Low	RW
Time Sync	0B630	RXSATRH	RX Timestamp Attributes High	RW
Time Sync	0B634	RXMTRL	RX Ethertype and Message Type Register	RW
Time Sync	0B638	RXUDP	RX UDP Port	RW
Time Sync	0B614	TSYNCTXCTL	TX Time Sync Control Register	RW
Time Sync	0B618	TXSTMPL	TX Timestamp Value Low	RW
Time Sync	0B61C	TXSTMPH	TX Timestamp Value High	RW
Time Sync	0B600	SYSTIML	System Time Register Low	RW
Time Sync	0B604	SYSTIMH	System Time Register High	RW
Time Sync	0B608	TIMINCA	Increment Attributes Register	RW
Time Sync	0B60C	TIMADJL	Time Adjustment Offset Register Low	RW
Time Sync	0B610	TIMADJH	Time Adjustment Offset Register High	RW
LinkSec Register Descriptions (ICH10)				
LinkSec	0x0B000	LSECTXCAP	LinkSec TX Capabilities	RW
LinkSec	0x0B300	LSECRXCAP	LinkSec RX Capabilities	RW
LinkSec	0x0B004	LSECTXCTRL	LinkSec TX Control	RW
LinkSec	0x0B304	LSECRXCTRL	LinkSec RX Control	RW
LinkSec	0x0B008	LSECTXSCL	LinkSec TX SCI Low	RW
LinkSec	0x0B00C	LSECTXSCH	LinkSec TX SCI High	RW
LinkSec	0x0B010	LSECTXSA	LinkSec TX SA	RW
LinkSec	0x0B018	LSECTXPN0	LinkSec TX SA PN 0	RW
LinkSec	0x0B01C	LSECTXPN1	LinkSec TX SA PN 1	RW
LinkSec	0x0B01C	LSECTXKEY0 [n]	LinkSec TX Key 0 0 + 4*n (n=0...3) 0x0B02	WO
LinkSec	0x0B01C	LSECTXKEY1 [n]	LinkSec TX Key 1 0 + 4*n (n=0...3) 0x0B03	WO
LinkSec	0x0B3D0 + 4*n (n=0...3)	LSECRXSCL[n]	LinkSec RX SCI Low	RW



Category	Offset	Abbreviation	Name	R/W
LinkSec	0x0B3E0 + 4*n (n=0...3)	LSECRXSCH[n]	LinkSec RX SCI High	RW
LinkSec	0x0B310 + 4*n (n=0...7)	LSECRXSA[n]	LinkSec RX SA	RW
LinkSec	0x0B330 + 4*n (n=0...7)	LSECRXSAPN	LinkSec RX SA PN	RW
LinkSec	0x0B350 + 0x10*n (n=0...7) + 4*m (m=0..3)	LSECRXKEY[n,m]	LinkSec RX Key	WO
LinkSec	0x04300	LSECTXUT	Tx Untagged Packet Counter	RC
LinkSec	0x04304	LSECTXPKTE	Encrypted Tx Packets	RC
LinkSec	0x04308	LSECTXPKTP	Protected Tx Packets	RC
LinkSec	0x0430C	LSECTXOCTE	Encrypted Tx Octets	RC
LinkSec	0x04310	LSECTXOCTP	Protected Tx Octets	RC
LinkSec	0x04314	LSECRXUT	LinkSec Untagged RX Packet	RC
LinkSec	0x0431C	LSECRXOCTE	LinkSec RX Octets Decrypted	RC
LinkSec	0x04320	LSECRXOCTP	LinkSec RX Octets Validated	RC
LinkSec	0x04324	LSECRXBAD	LinkSec RX Packet with Bad Tag	RC
LinkSec	0x04328	LSECRXNOSCI	LinkSec RX Packet No SCI	RC
LinkSec	0x432C	LSECRXUNSCI	LinkSec RX Packet Unknown SCI count	RC
LinkSec	0x04330	LSECRXUNCH	LinkSec RX Unchecked Packets	RC
LinkSec	0x04340 + 4*n (n=0...3)	LSECRXDELAY[n]	LinkSec RX Delayed Packets	RC
LinkSec	0x04350 + 4*n (n=0...3)	LSECRXLATE[n]	LinkSec RX Late Packets	RC
LinkSec	0x04360 + 4*n (n=0...7)	LSECRXOK[n]	LinkSec RX Packet OK	RC
LinkSec	0x04380 + 4*n (n=0...7)	LSECRXINV[n]	LinkSec Check RX Invalid	RC
LinkSec	0x43A0 + 4*n [n=0...7]	LSECRXNV[n]	LinkSec RX Not valid count	RC
LinkSec	0x043C0 + 4*n (n=0...3)	LSECRXUNSA[n]	LinkSec RX Unused SA	RC
LinkSec	0x043D0 + 4*n (n=0...3)	LSECRXNUSA[n]	LinkSec RX Not Using SA	RC

10.4 PCI Register Descriptions

This section contains detailed descriptions for PCI general purpose, DMA, interrupt, receive, and transmit registers. These registers correspond to the main functions of the Ethernet LAN controllers.

10.4.1 Device Control Register - CTRL (00000h; R/W)

This register and the Extended Device Control register (CTRL_EXT) control the major operational modes for the MAC.

While software writes to this register to control device settings, several bits (such as FD and SPEED) can be overridden depending on other bit settings and the resultant link configuration determined by the PHY's auto-negotiation resolution.

The FD (duplex) and SPEED configuration of the MAC is normally determined from the link configuration process. Software can specifically override/set these MAC settings via these bits in a forced-link scenario; if so, the values used to configure the MAC must be consistent with the PHY settings.

Manual link configuration is controlled through the PHY's MII management interface.

Host software reset can be used to globally reset the entire host data path. This register is provided as a software mechanism to recover from an indeterminate or suspected hung hardware state. Most registers (receive, transmit, interrupt, statistics, etc.), and state machines are set to their power-on reset values, approximating the state following a power-on or PCI reset. One internal configuration register, the Packet Buffer Allocation (PBA) register, also retains its value through a software reset.

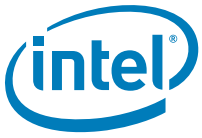
LCD_RST can be used to force an external reset to the PHY by asserting the LANRSTSYNC signal. Software must guarantee that this signal is asserted for at least 1 ms to ensure a proper reset to the PHY.

Note: To ensure that a global device reset has fully completed and that the MAC responds to subsequent accesses, a wait of approximately 1 μs is required after the setting before attempting to check if the bit has cleared or to access (read or write) any other device register.

Note: This register's address is reflected also at address 00004h. The software device driver nor firmware should not use it since it might be unsupported.

Table 56. CTRL Register Bit Description

Field	Bit(s)	Initial Value	Description
FD	0	1b	Full-Duplex Controls the MAC duplex setting when explicitly setting by software. Loaded from the NVM word 13h 0b = Half duplex. 1b = Full duplex.
Reserved	1	0b	Reserved, Write as 0b for future compatibility.
Master Disable	2	0b	When set, the MAC blocks new master requests on the PCI device. If no master requests are pending by this function, the *Master Enable Status* bit is set.
Reserved	5:3	000b	Reserved Write as 0b for future compatibility.
Reserved	6	1b	Reserved
Reserved	7	0b	Reserved Always set this bit to 0b.
SPEED	9:8	10b	Speed selection. These bits determine the speed configuration and are written by software after reading the PHY configuration through the MDIO interface. These signals are ignored when Auto-Speed Detection is enabled. 00b = 10 Mb/s 01b = 100 Mb/s 10b = 1000 Mb/s 11b = not used
Reserved	10	0b	Reserved Write as 0b for future compatibility.



Field	Bit(s)	Initial Value	Description
FRCSPD	11	0b	Force Speed This bit is set when software needs to manually configure the MAC speed settings according to the *Speed* bits. When using a PHY, note that the PHY must resolve to the same speed configuration or software must manually set it to the same speed as the MAC. Word 13h in the NVM loads the default value (asserted). Software must clear this bit to enable the PHY or ASD function to control the MAC speed setting. Note that this bit is superseded by the *CTRL_EXT.SPD_BYPS* bit, which has a similar function.
FRCDPLX	12	0b	Force Duplex When set to 1b, software might override the duplex indication from the PHY that is indicated in the FDX to the MAC. Otherwise, the duplex setting is sampled from the PHY FDX indication into the MAC on the asserting edge of the PHY LINK signal. When asserted, the *CTRL.FD* bit sets the duplex mode.
Reserved	13	0b	Reserved
Reserved	14	0b	Reserved. Must be set to 0b.
Reserved	15	0b	Reserved Reads as 0b.
LANPHYPC Override	16	0b	For **ICH10**, when set to 1b, this bit provides the software device driver the ability to control the LANPHYPC pin value.
LANPHYPC Value	17	0b	For **ICH10**, when bit 16 is set to 1b, this bit defines the LANPHYPC pin value.
Reserved	18	0b	Reserved. Must be set to 0b.
Reserved	19	0b	Reserved
Reserved	20	1b	Reserved
Reserved	23:21	000b	Reserved
LCDPD	24	0b	PHY Power Down When the bit is set to 0b, the PHY power down setting is controlled by the internal logic of the MAC. When set to 1b and the CTRL_EXT.PHYPDEN bit is set as well, the MAC sets the PHY to power down mode. Also, if LANPHYPC is implemented, the MAC disconnects the PHY's power supply.
H2MEINT	25	0b	Host to ME Interrupt Setting this bit asserts the host interrupt to ME function. Note that this bit is self clearing.
SWRST	26	0b	Host Software Reset This bit performs a reset to the PCI data path and the relevant shared logic. Writing a 1b initiates the reset. Note that this bit is self-clearing.
RFCE	27	0b	Receive Flow Control Enable When set, indicates that the MAC responds to the reception of flow control packets. If Auto-Negotiation is enabled, this bit is set to the negotiated duplex value.
TFCE	28	0b	Transmit Flow Control Enable When set, indicates that the MAC transmits flow control packets (XON and XOFF frames) based on the receiver fullness. If Auto-Negotiation is enabled, this bit is set to the negotiated duplex value.

Field	Bit(s)	Initial Value	Description
Reserved	29	0b	Reserved Reads as 0b.
VME	30	0b	VLAN Mode Enable When set to 1b, all packets transmitted from the MAC that have the *VLE* bit set in their descriptor is sent with an 802.1Q header added to the packet. The contents of the header come from the transmit descriptor and from the VLAN type register. On receive, VLAN information is stripped from 802.1Q packets and is loaded to the packet's descriptor.
LCD_RST	31	0b	LAN Connected Device (PHY) Reset Controls a hardware-level reset (LANRSTSYNC) to the PHY. 0b = Normal operation. 1b = Reset to PHY asserted. The LCD_RST functionality is gated by the *FWSM.RSPCIPHY* bit. If the last is not set to 1b, then setting the *LCD_RST* has no impact on LANRSTSYNC. Setting the *LCD_RST* bit initializes the GLCI bus in the PHY. For proper operation, software or firmware must also set the *SWRST* bit in the register at the same time. Note that this bit is self-clearing.

10.4.2 Device Status Register - STATUS (00008h; R)

This register provides software status indication about the MAC/PHY settings and modes of operation.

Table 57. Status Register Bit Description

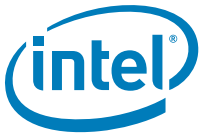
Field	Bit(s)	Initial Value	Description
FD	0	X	Full Duplex 0b = Half duplex. 1b = Full duplex. Reflects the duplex setting of the MAC and/or link.
LU	1	X	Link Up 0b = No link established. 1b = Link established. For this field to be valid, the *Set Link Up* bit of the Device Control register (CTRL.SU) must be set.
PHYTYPE	3:2	00b	PHY Type Indication Indicates the LAN Connected Device attached to MAC and resulted mode of operation of the MAC/LAN Connected Device Link buses. PHYTYPE - PHY-Device - GLCI Mode - LCI Mode 00b - **82566**/**82567** - GLCI 2.0 - LCI 2.0 01b - Reserved 10b - **82562V** - PCI - LCI 1.0 11b - Reserved This field is loaded from the Shared Init Control word in the NVM. In the **82566/82567**, the LCI link can be active or inactive according to its setting. The MAC identifies one of the two cases during initialization.
TXOFF	4	X	Transmission Paused Indicates the Pause state of the transmit function when symmetrical flow control is enabled.

Field	Bit(s)	Initial Value	Description
PHYPWR	5	1b	PHY Power Up not (PHYPWR) This read-only bit indicates the power state of the PHY. 0b (1b **ICH9/ICH10**) = The PHY is in the power down state. 1b (0b **ICH9/ICH10**) = The PHY is powered on in the active state. The *PHYPWR* bit is valid only after PHY reset is asserted. **Note:** The PHY Power Up indication does not reflect the status of the LANPHYPC signaling to the PHY.
SPEED	7:6	X	Link Speed Setting Reflects the speed setting of the MAC and/or link. Speed indication is mapped as follows: 00b = 10 Mb/s 01b = 100 Mb/s 10b = 1000 Mb/s 11b = 1000 Mb/s
MRCB	8	X	Master Read Completions Blocked This bit is set when the MAC receives a completion with error (EP = 1 or status != successful). this bit is cleared on a PCI reset.
LAN Init Done	9	0b	LAN Init Done Asserted following completion of the LAN initialization from the Flash when a PHY is attached. Software is expected to clear this field to make it usable for the next Init Done event.
PHYRA	10	1b	PHY Reset Asserted Hardware sets this bit following the assertion of PHY reset (either hardware or in-band). This bit is cleared when writing a 0b to it. This bit can also be used by software an indication that the ME initiated a PHY reset.
Reserved	11	0b	Reserved. Must be set to 0b.
Reserved	18:12	0	Reserved
Master Enable Status	19	1b	Master Enable Status This bit is cleared by the MAC when the *Master Disable* bit is set and no master requests are pending by this function. Otherwise, set this bit. Also indicates that no master requests are issued by this function as long as the *Master Disable* bit is set.
Reserved	30:20	0b	Reserved Reads as 0b.
CLK_CNT_1_4	31	1b 0b **ICH9**	Clock Control 1/4 This bit is loaded from NVM word 13h and indicates that the MAC supports lowering its DMA clock to ¼ of its value.

10.4.3 ICH9/ICH10 Strapping Option - STRAP (0000Ch, RO)

This register reflects the values of the soft strapping options fetched from the NVM descriptor in the ICH space. These signals are sampled by the MAC following a LAN_RST# or global reset (assertion of PCI reset).

Field	Bit(s)	Initial Value	Description
Reserved	0	1b	Reserved
NVMS	5:1	0b	LAN NVM Size LAN NVM space size indicated in multiples of 4 KB. LAN NVM size can very from 4K bytes to 128 KB; a zero value = 4 KB.
NVMV	6	0b	NVM Valid Indication Indicates to the MAC that ICH NVM descriptor is valid with good signature. The MAC can read its configuration from the NVM only if the NVM is valid. An invalid NVM has the same impact on the default settings of the MAC as the NVM signature in the LAN space.
NICM	7	0b	LAN NIC Mode Sets the ICH in a LAN NIC mode. The MAC device ID on the host interface is device zero instead of 19h. This mode enables pre-validation of the MAC as an external device to ICH9.
KUME	8	1b	GLCI Enable Selects between GLCI and PCI mode on PCI link six in the ICH. When LAN function is enabled and a required GLCI link to a PHY, the GLCI enable should be set.
PHYPCSEL	9	0b	LAN PHY Power Control Sel Selects the functionality of the GP12 IO signal: When set to 0b, the GP12 is used as general purpose I/O. When set to 1b, the GP12 is in native mode as a LANPHYPC signal.
Reserved	10	0b	Reserved. Must be set 0b.
ENDETSEL	11	0b	Energy Detect Select Selects the functionality of the GP13 IO signal: When set to 0b, the GP13 is used as general purpose I/O. When set to 1b the GP13 is in native mode as an energy detect signal. In the desktop SKU, GPIO13 cannot be set to native mode.
NJCLKSEL	12	0b	NJCLK Trip Point Select Selects the trip point level of the NJCLK input buffer: 0b = NJCLK is matched to a 3.3 V output driver in the PHY. 1b = NJCLK is matched to a 1.8 V output driver in the PHY. **Note:** This is a reserved bit for **ICH9**.
Reserved	31:15	00h	Reserved



10.4.4 NVM/Flash Control Register - EEC (00010h; RO)[1]

Note: Sometimes referred to as the EECD register.

Note: EEC can be accessed only via the CSR Memory BAR at PCI config offset 10h and can't be accessed via the IO BAR.

Table 58. EEC Register Bit Description

Field	Bit	Initial Value	Description
Reserved	7:0	00h	Reserved Reads as 0b.
NV_Pres	8	0b	NVM Present 1b = NVM present and has the correct signature field. 0b = NVM not present.
Auto_RD	9	0b	NVM Auto Read Done The Auto_RD bit is cleared to 0b by the MAC each time an auto-read request is presented to the Fleep. It is set to 1b by the MAC after the auto-read by hardware from the NVM is done. This bit is also set when the NVM is not present or when its signature is not valid. The bit is set after all fields that are required in the basic configuration space are fetched.
Reserved	21:10	00h	Reserved Reads as 0b.
SEC1VAL	22	0b	Sector 1 Valid If NVPRES is set, a 0b indicates that Sector 0 in the NVM contains valid signatures. 1b indicates that Sector 1 contains valid signatures. This bit is only valid if the *NV_PRES* and *Auto_RD* bits are both set.
Reserved	31:23	00h	Reserved Reads as 0b.

10.4.5 Extended Device Control Register - CTRL_EXT (00018h, R/W)

This register and the Device Control register (CTRL) controls the major operational modes for the MAC. CTRL_EXT provides extended control of the MAC functionality over the Device Control register (CTRL).

Table 59. CTRL_EXT Register Bit Description

Field	Bit(s)	Initial Value	Description
Reserved	11:0	0b	Reserved
LSecCK	12	1b	LinkSec Clock Gate (**ICH10** Only) When cleared, the LinkSec logic gets its clocks. When the LSecCK is set, the LinkSec logic (including all its CSR registers) do not get any clocks. This bit is loaded from NVM word 13h.
Reserved	13	0b	Reserved

1. Not applicable to **ICH10**.

Field	Bit(s)	Initial Value	Description
EXTPPOL	14	0b	External PHY Power Control Polarity Defines the polarity of the LAN PHY PC (power control). When cleared to 0b, defines an active Hi power enable signal. When set to 1b, defines an active low power enable signal. Loaded from the *ExtPwrPol* bit in NVM word 13h.
SPD_BYPS	15	0b	Speed Select Bypass When set to 1b, all speed detection mechanisms are bypassed, and the MAC is immediately set to the speed indicated by CTRL.SPEED. This provides a method for software to have full control of the speed settings of the MAC and when the change takes place by overriding the hardware clock switching circuitry.
Reserved	18:16	0b	Reserved
DynCK	19	0b	Dynamic Clock Gating When set, enables dynamic clock gating of the DMA and MAC units. The bit is loaded from NVM word 13h.
PHYPDEN	20	1b	PHY Power down Enable When set, enables the PHY to enter a low-power state when the MAC is at the DMoff/D3 or Dr and no WoL. This bit is loaded from word 13h in the NVM.
Reserved	21	0b	Reserved
Reserved	22	1b	Reserved
Reserved	23	0b	Reserved
Reserved	24	1b	Reserved
DMACKCTL	25	0b	DMA Clock Controls the DMA clock source in non-GbE mode (10/100 Mb/s and no link). In GbE mode, the DMA clock source is always GLCI PLL divided by two. In nominal operation, this bit should be in the default state on which the DMA clock source in non-GbE is mosc_clk.
PLLGateDis	26	0b	Disable Static GLCI PLL Gating By default, the PLL is functional only when the GLCI link is required and inactive when it is not required (at non-GbE mode if LCI is available). When set to 1b, the GLCI PLL is always active.
IAME	27	0b	Interrupt Acknowledge Auto-Mask Enable When set to 1b, a read or write to the ICR register has the side effect of writing the value in the IAM register to the IMC register. When set to 0b, this feature is disabled.
DRV_LOAD	28	0b	Driver Loaded This bit should be set by the software device driver after it loaded. This bit should be cleared when the driver unloads or after a PCI soft reset. The MNG controller loads this bit to indicate to the manageability controller that the driver has loaded.
INT_TIMERS_ CLEAR_ENA	29	0b	When set, this bit enables the clear of the interrupt timers following an IMS clear. In this state, successive interrupts happen only after the timers expire again. When clear, successive interrupts following IMS clear might happen immediately.
Reserved	30	0b	Reserved
Reserved	31	0b	Reserved. Reads as 0b.

Note: If software uses the EE_RST function and desires to retain the current configuration information, the contents of the control registers should be read and stored by software. Control register values are changed by a read of the NVM which occurs upon assertion of the EE_RST bit.

Note: The EEPROM reset function can read configuration information out of the NVM which affects the configuration of PCI configuration space BAR settings. The changes to the BARs are not visible unless the system is rebooted and the BIOS is allowed to re-map them.

Note: The *SPD_BYPS* bit performs a similar function to the CTRL.FRCSPD bit in that the MAC's speed settings are determined by the value software writes to the *CRTL.SPEED* bits. However, with the *SPD_BYPS* bit asserted, the settings in CTRL.SPEED take effect rather than waiting until after the MAC's clock switching circuitry performs the change.

10.4.6 MDI Control Register - MDIC (00020h; R/W)

Software uses this register to read or write Management Data Interface (MDI) registers in the PHY.

Note: Internal logic uses the MDIC to communicate with the PHY. All fields in this register is indicated as "/V" since the internal logic might use it to access the PHY. Since the hardware uses this register, all hardware, software, and firmware must use a semaphore logic (the *Ownership* flags) before accessing the MDIC.

For an MDI read cycle, the sequence of events is as follows:

- The CPU performs a PCI write cycle to the MII register with:
  - Ready = 0b
  - Interrupt Enable set to 1b or 0b
  - Opcode = 10b (read)
  - PHYADD = PHY address from the MDI register
  - REGADD = Register address of the specific register to be accessed (0 through 31)
- The MAC applies the following sequence on the MDIO signal to the PHY:

  <PREAMBLE><01><10><PHYADD><REGADD><Z> where Z stands for the MAC tri-stating the MDIO signal
- The PHY returns the following sequence on the MDIO signal:

  <0><DATA><IDLE>
- The MAC discards the leading bit and places the following 16 data bits in the MII register
- The MAC asserts an interrupt indicating MDI "Done" if the *Interrupt Enable* bit was set
- The MAC sets the *Ready* bit in the MII register indicating the Read completed.
- The CPU reads the data from the MII register and then issues a new MDI command

For a MDI write cycle, the sequence of events is as follows:

- Ready = 0b
- Interrupt Enable set to 1b or 0b
- Opcode = 01b (write)
- PHYADD = PHY address from the MDI register
- REGADD = Register address of the specific register to be accessed (0 through 31)
- Data = Specific data for desired control of the PHY

- The MAC applies the following sequence on the MDIO signal to the PHY:

  <PREAMBLE><01><01><PHYADD><REGADD><10><DATA><IDLE>
- The MAC asserts an interrupt indicating MDI "Done" if the *Interrupt Enable* bit was set
- The MAC sets the *Ready* bit in the MII register to indicate that the write operation completed
- The CPU might issue a new MDI command

Note: An MDI read or write might take as long as 64 μs from the CPU write to the *Ready* bit assertion.

If an invalid opcode is written by software, the MAC does not execute any accesses to the PHY registers.

If the PHY does not generate a 0b as the second bit of the turn-around cycle for reads, the MAC aborts the access, sets the *E* (error) bit, writes FFFFh to the data field to indicate an error condition, and sets the *Ready* bit.

10.4.7 Accessing Wake Up Registers Using MDIC

When software needs to configure the wake up state (either read or write to these registers) the MDIO page should be set to 800 (for host accesses) or 801 (for ME accesses) until the page is not changed to a different value wake up register access is enabled. After the page was set to the wake up page, the address field is no longer translated as *reg_addr* (register address) but as an instruction. If the given address is in the [0..15] range, meaning PHY registers, the functionality remains unchanged. There are three valid instructions:

1. Address Set – 11h – Wake up space address is set for either reading or writing.
2. Data cycle – 12h – Wake up space accesses read or write cycle.
3. Data and Address Increment – 14h – Wake up space accesses read or write cycle followed by address increment for the next cycle.

For the PHY, the wake area read cycle sequence of events is as follows:

1. Setting page 0800h The software device driver performs a write cycle to the MDI register with:
   a. Ready = 0b
   b. Op-Code = 01b (write)
   c. PHYADD = The PHY's address from the MDI register
   d. REGADD = Page setting
   e. DATA = 0800h (wake up page)
2. Address setting; the software device driver performs a write cycle to the MDI register with:
   a. Ready = 0b
   b. Op-Code = 01b (write)
   c. PHYADD = The PHY's address from the MDI register
   d. REGADD = 11h (address set)
   e. DATA = XXXX (address of the register to be read)

3. Reading a register; the software device driver performs a write cycle to the MDI register with:
   a. Ready = 0b
   b. Op-Code = 10b (read)
   c. PHYADD = The PHY's address from the MDI register
   d. REGADD = 12h (data cycle for read)
   e. DATA = YYYY (data is valid when the ready bit is set)
4. To read a register and prepare the address for the next register in-line; the software device driver performs a write cycle to the MDI register with:
   a. Ready = 0b
   b. Op-Code = 10b (read)
   c. PHYADD = The PHY's address from the MDI register
   d. REGADD = 14h (data cycle for read and address increment)
   e. DATA = YYYY (data is valid when the ready bit is set)

For the PHY, the wake area write cycle sequence of events is as follows:

1. Setting page 0x0800; the software device driver performs a write cycle to the MDI register with:
   a. Ready = 0b
   b. Op-Code = 01b (write)
   c. PHYADD = The PHY's address from the MDI register
   d. REGADD = Page setting
   e. DATA = 0800h (wake up page)

2. Address setting; The software device driver performs a write cycle to the MDI register with:
   a. Ready = 0b
   b. Op-Code = 01b (write)
   c. PHYADD = The PHY's address from the MDI register
   d. REGADD = 11h (address set)
   e. DATA = XXXX (address of the register to be read)
3. Writing a register; the software device driver performs a write cycle to the MDI register with:
   a. Ready = 0b
   b. Op-Code = 01b (write)
   c. PHYADD = The PHY's address from the MDI register
   d. REGADD = 12h (data cycle for write)
   e. DATA = YYYY (data to be written to the register)
4. To write a register and prepare the address for the next register in-line; the software device driver performs a write cycle to the MDI register with:
   a. Ready = 0b
   b. Op-Code = 01b (write)
   c. PHYADD = The PHY's address from the MDI register
   d. REGADD = 14h (data cycle for write and address increment)
   e. DATA = YYYY (data to be written to the register)

Note: The PHY register bit descriptions follow Table 60. For **82562V** PHY register descriptions, refer to the *82562V 10/100 Mbps Platform LAN Connect (PLC) Datasheet*.

Table 60. MDI Control Register Bit Description

Field	Bit(s)	Initial Value	Description
DATA	15:0	X	Data In a Write command, software places the data bits and the Ethernet controller shifts them out to the PHY. In a Read command, the MAC reads these bits serially from the PHY and software can read them from this location. With a page switch instruction, the page number should be written to the 11 MSB's of the *Data* field while bits 4:0 should be set to 0b.
REGADD	20:16	0b	PHY Register Address: Reg. 0, 1, 2, ...31. A value of 1Fh indicates that this is a command to switch to another MDIO page.
PHYADD	25:21	0b	PHY Address Software or firmware must set this field to 00001b for the **82566** and 00010b for the **82567**.
OP	27:26	0b	Opcode 01b = MDI Write 10b = MDI Read All other values are reserved.

Field	Bit(s)	Initial Value	Description
R	28	0b	Ready Bit Set to 1b by the MAC at the end of the MDI transaction (for example, indication of a Read or Write completion). It should be reset to 0b by software at the same time the command is written.
I	29	0b	Interrupt Enable When set to 1b by software, it causes an Interrupt to be asserted to indicate the end of an MDI cycle.
E	30	0b	Error This bit is set to 1b by hardware when it fails to complete an MDI read. Software should make sure this bit is clear (0b) before issuing an MDI read or write command.
Reserved	31	0b	Reserved Write as 0b for future compatibility.

10.4.8 PHY Register Addressing

PHY register addressing is based on IEEE 802.3 MII Management Interface specification defined in clause 22 of 802.3, particularly section 22.2.4.

The PHY address in the **82566/82567** is hard coded to 01h. Accesses to other PHY addresses are rejected.

The IEEE specification allows five bits for register access. Registers 0d to 15d are defined by the specification, while registers 16 to 31 are left available for OEMs.

The **82566/82567** implements many registers for diagnostic purposes. In addition, the **82566/82567** contains registers controlling the GLCI interface as well as other **82566/82567** functions. The total number of registers implemented exceeds the 16 registers available to OEMs. When this occurs, a common technique is to use paging. The **82566/82567** registers are divided into pages. Each page has 32 registers. PHY registers 0d to 15d are identical in all the pages and are the IEEE defined registers. PHY register 31d is the page register in all pages. Register 22d is also identical in all pages to keep compatibility with the **82563EB/82564EB**, where this register is the page register. PHY registers 29d to 30d are also identical in all pages. All remaining PHY registers are page specific.

In order to read or write a PHY register other than registers 0d to 15d, 22d, or 29d to 31d, software should first set the page register to map to the appropriate page. Software can then read or write any register in that page.

This document uses a special nomenclature to define the read/write mode of individual bits in each register. See Table 61.

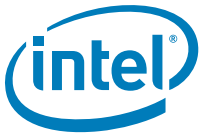
For all binary equations appearing in the register map, the symbol “|” is equivalent to a binary OR operation.

Table 61. Terminology

Register Mode	Description
R/W	Read/Write. A register with this attribute can be read and written. If written since reset, the value read reflects the value written.
R/W S	Read/Write Status. A register with this attribute can be read and written. This bit represents status, so the value read might not reflect the value written.
RO	Read Only. If a register is read only, writes to this register have no effect.
WO	Write Only. Reading this register might not return a meaningful value.
R/WC	Read/Write Clear. A register bit with this attribute can be read and written. However, a write of 1b clears (sets to 0b) the corresponding bit and a write of a 0b has no effect.
R/W SC	Read/Write Self Clearing. When written to 1b, the bit causes an action to be initiated. Once the action completes the bit returns to 0b.
RO/LH	Read Only, Latch High. The bit records an event or the occurrence of a condition to be recorded. When the event occurs the bit is set to 1b. After the bit is read, it returns to 0b unless the event is still occurring.
RO/LL	Read Only, Latch Low. The bit records an event. When the event occurs, the bit is set to 0b. After the bit is read, it reflects the current status.
RO/SC	Read Only, Self Clear. Writes to this register have no effect. Reading the register clears (set to 0b) the corresponding bits.
RW0	Ignore Read, Write Zero. The bit is a reserved bit. Any values read should be ignored. When writing to this bit always write to 0b.

10.4.8.1 PHY Control Register - PCTRL (00d; R/W)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Reserved	5:0	Reserved. Always read as 0b. Write to 0b for normal operation	RW	Always 000000b	
Speed Selection (MSB)	6	Speed Selection is determined by bit 6 (Speed = 1b) and bit 13 (Speed 0b) as follows. 11b = Reserved 10b = 1000 Mb/s 01b = 100 Mb/s 00b = 10 Mb/s A write to these bits do not take effect until a software reset is asserted, Restart Auto-Negotiation is asserted, or Power Down transitions from power down to normal operation. **Note:** If auto-negotiation is enabled (PHY register 0d, bit 12 set), this bit is ignored unless duplex is manually set.	R/W	0b	Update
Collision Test	7	1b = Enable COL signal test. 0b = Disable COL signal test. **Note:** This bit is ignored unless loopback is enabled (PHY register 0d, bit 14 = 1b).	R/W	0b	0b



Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Duplex Mode	8	1b = Full Duplex. 0b = Half Duplex. **Note:** If auto-negotiation is enabled (PHY register 0d, bit 12 set), this bit is ignored unless duplex is manually set.	R/W	1b	Update
Restart Auto-Negotiation	9	1b = Restart Auto-Negotiation Process. 0b = Normal operation. Auto-Negotiation automatically restarts after hardware or software reset regardless of whether or not the restart bit is set.	WO, SC	N/A	Self Clear
Isolate	10	This bit has no effect on PHY functionality. Program to 0b for future compatibility.	R/W	0b	0b
Power Down	11	1b = Power down. 0b = Normal operation. Power down shuts down the PHY except for the MAC interface if the MAC interface power down bit is set to 1b. If it equals 0b, then the MAC interface also shuts down.	R/W	0b	0b
Auto-Negotiation Enable	12	1b = Enable Auto-Negotiation Process. 0b = Disable Auto-Negotiation Process. This bit must be enabled for 1000BASE-T operation.	R/W	1b	Update
Speed Selection (LSB)	13	See Speed Selection (MSB), bit 6. **Note:** If auto-negotiation is enabled (PHY register 0d, bit 12 set), this bit is ignored unless duplex is manually set.	R/W	1b	Update
Loopback	14	1b = Enable loopback (DSP). 0b = Disable loopback.	R/W	0b	0b
Reset	15	1b = PHY reset. 0b = Normal operation.	WO, SC	N/A	Self Clear

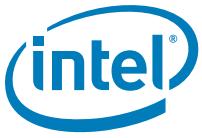
10.4.8.2 PHY Status Register - PSTATUS (01d; R)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Extended Capability	0	1b = Extended register capabilities.	RO	Always 1b	
Jabber Detect	1	1b = Jabber condition detected. 0b = Jabber condition not detected.	RO LH	0b	0b
Link Status	2	1b = Link is up. 0b = Link is down. This register indicates whether link was lost after the last read. For the current link status, either read this register back-to-back or read the Link Real Time bit 17 in the PHY Specific Status Register.	RO, LL	0b	0b
Auto-Negotiation Ability	3	1b = PHY able to perform Auto-Negotiation.	RO	Always 1b	
Remote Fault	4	1b = Remote fault condition detected. 0b = Remote fault condition not detected.	RO LH	0b	0b

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Auto-Negotiation Complete	5	1b = Auto-Negotiation process complete. 0b = Auto-Negotiation process not complete.	RO	0b	0b
MF Preamble Suppression	6	0b = PHY does not accept management frames with preamble suppressed.	RO	Always 0b	
Reserved	7	Reserved. Should be set to 0b	RO	Always 0b	
Extended Status	8	1b = Extended status information in the Extended PHY Status Register (15d).	RO	Always 1b	
100BASE-T2 Half Duplex	9	0b = PHY not able to perform half duplex 100BASE-T2.	RO	Always 0b	
100BASE-T2 Full Duplex	10	0b = PHY not able to perform full duplex 100BASE-T2.	RO	Always 0b	
10 Mb/s Half Duplex	11	1b = PHY able to perform half duplex 10BASE-T. 0b = PHY not able to perform half duplex 10BASE-T.	RO	1b	
10 Mb/s Full Duplex	12	1b = PHY able to perform full duplex 10BASE-T. 0b = PHY not able to perform full duplex 10BASE-T.	RO	1b	
100BASE-X Half Duplex	13	1b = PHY able to perform half duplex 100BASE-X. 0b = PHY able to perform half duplex 100BASE-X.	RO	1b	
100BASE-X Full Duplex	14	1b = PHY able to perform full duplex 100BASE-X. 0b = PHY not able to perform full duplex 100BASE-X.	RO	1b	
100BASE-T4	15	0b = PHY not able to perform 100BASE-T4. 1b = PHY able to perform 100BASE-T4.	RO	Always 0b	

10.4.8.3 PHY Identification Register 1 (LSB) - PHY ID 1 (02d; R)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
PHY ID Number	15:0	The PHY identifier composed of bits 3 through 18 of the Organizationally Unique Identifier (OUI)	RO	Always 02A8h (0141h for the **82567**)	



10.4.8.4 PHY Identification Register 2 (MSB) - PHY ID 2 (03d; R)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Manufacturer's Revision Number	3:0	4 bits containing the manufacturer's revision number.	RO	0h	0h
Manufacturer's Model Number	9:4	6 bits containing the manufacturer's part number.	RO	111001b (001011b for the **82567**)	00h
PHY ID Number	15:10	The PHY identifier composed of bits 19 through 24 of the OUI	RO	00h	00h

10.4.8.5 Auto-Negotiation Advertisement Register - ANA (04d; R/W)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Selector Field	4:0	00001b = 802.3 Other combinations are reserved. Unspecified or reserved combinations should not be transmitted. **Note:** Setting this field to a value other than 00001b can cause auto negotiation to fail.	R/W	Always 00001b	
10Base-T	5	1b = DTE is 10BASE-T capable. 0b = DTE is not 10BASE-T capable.	R/W	1b	Retain
10Base-T Full Duplex	6	1b = DTE is 10BASE-T full duplex capable. 0b = DTE is not 10BASE-T full duplex capable.	R/W	1b	Retain
100Base-TX	7	1b = DTE is 100BASE-TX capable. 0b = DTE is not 100BASE-TX capable.	R/W	1b	Retain
100BASE-TX Full Duplex	8	1b = DTE is 100BASE-TX full duplex capable. 0b = DTE is not 100BASE-TX full duplex capable.	R/W	1b	Retain
100BASE-T4	9	0b = Not capable of 100BASE-T4.	R/W	Always 0b	
PAUSE	10	Advertise to Partner that Pause operation (as defined in 802.3x) is desired.	R/W	1b	Retain
ASM_DIR	11	Advertise Asymmetric Pause direction bit. This bit is used in conjunction with PAUSE.	R/W	1b	Retain
Reserved	12	Ignore on read.	R/W	0b	Retain
Remote Fault	13	1b = Set Remote Fault bit. 0b = Do not set Remote Fault bit.	R/W	0b	Retain
Reserved	14	Always read as 0b. Write to 0b for normal operation.	R/W	Always 0b	
Next Page	15	1b = Manual control of Next Page (Software). 0b = Ethernet controller control of Next Page (Auto).	R/W	0b	Retain

10.4.8.6 Auto-Negotiation Base Page Ability Register - (05d; R)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Selector Fields[4:0]	4:0	<00001> = IEEE 802.3 Other combinations are reserved. Unspecified or reserved combinations shall not be transmitted. If field does not match PHY Register 04d, bits 4:0, the AN process does not complete and no HCD is selected.	RO	N/A	
10BASE-T	5	1b = Link Partner is 10BASE-T capable. 0b = Link Partner is not 10BASE-T capable.	RO	N/A	
10BASE-T Full Duplex	6	1b = Link Partner is 10BASE-T full duplex capable. 0b = Link Partner is not 10BASE-T full duplex capable.	RO	N/A	
100BASE-TX	7	1b = Link Partner is 100BASE-TX capable. 0b = Link Partner is not 100BASE-TX capable.	RO	N/A	
100BASE-TX Full Duplex	8	1b = Link Partner is 100BASE-TX full duplex capable. 0b = Link Partner is not 100BASE-TX full duplex capable.	RO	N/A	
100BASE-T4	9	1b = Link Partner is 100BASE-T4 capable. 0b = Link Partner is not 100BASE-T4 capable.	RO	N/A	
LP Pause	10	Link Partner uses Pause Operation as defined in 802.3x.	RO	N/A	
LP ASM_DIR	11	Asymmetric Pause Direction Bit 1b = Link Partner is capable of asymmetric pause. 0b = Link Partner is not capable of asymmetric pause.	RO	N/A	
Reserved	12	Reserved. Write as 0b. Always read as 0b.	RO	Always read as 0b	
Remote Fault	13	1b = Remote fault. 0b = No remote fault.	RO	N/A	
Acknowledge	14	1b = Link Partner has received Link Code Word from the PHY. 0b = Link Partner has not received Link Code Word from the PHY.	RO	N/A	
Next Page	15	1b = Link Partner has ability to send multiple pages. 0b = Link Partner has no ability to send multiple pages.	RO	N/A	

10.4.8.7 Auto-Negotiation Expansion Register - ANE (06d; R)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Link Partner Auto-Negotiation Able	0	1b = Link Partner is Auto-Negotiation able. 0b = Link Partner is not Auto-Negotiation able.	RO	0b	0b
Page Received	1	Indicates that a new page has been received and the received code word has been loaded into PHY register 05d (base pages) or PHY register 08d (next pages) as specified in clause 28 of 802.3. This bit clears on read. If PHY register 21d bit 13 (Alternate NP Feature) is set, the *Page Received* bit also clears when mr_page_rx = false or transmit_disable = true.	RO/ LH	0b	0b
Next Page Able	2	1b = Local device is next page able. 0b = Local device is not next page able.	RO	1b	
Link Partner Next Page Able	3	1b = Link Partner is next page able. 0b = Link Partner is not next page able.	RO	0b	0b
Parallel Detection Fault	4	1b = A fault has been detected via the Parallel Detection function. 0b = A fault has not been detected via the Parallel Detection function.	RO/ LH	0b	0b
Base Page	5	This bit indicates the status of the auto-negotiation variable, base page. If flags synchronization with the auto-negotiation state diagram enabling detection of interrupted links. This bit is only used if PHY register 21d, bit 13 (Alternate NP Feature) is set. 1b = base_page = true. 0b = base_page = false. **Note:** This is a reserved bit for the **82567**.	RO/ LH	0b	0b
Reserved	15:6	Always read as 0b.	RO	Always 0b	

10.4.8.8 Auto-Negotiation Next Page Transmit Register - NPT (07d; R/W)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Message/ Unformatted Field	10:0	11-bit message code field.	R/W	1b	1b
Toggle	11	1b = Previous value of the transmitted Link Code Word = 0b. 0b = Previous value of the transmitted Link Code Word = 1b.	RO	0b	0b
Acknowledge 2	12	1b = Complies with message. 0b = Cannot comply with message.	R/W	0b	0b
Message Page	13	1b = Message page. 0b = Unformatted page.	R/W	0b	0b
Reserved	14	Always read as 0b. Write to 0b for normal operation.	RO	0b	0b

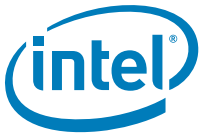
Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Next Page	15	1b = Additional next pages follow. 0b = Last page.	R/W	0b	0b

10.4.8.9 Auto-Negotiation Next Page Ability Register - LPN (08d; R)

Bit(s)	Field	Description	Mode	HW Rst	SW Rst
10:0	Message/ Unformatted Field	11-bit message code field.	RO	0b	0b
11	Toggle	1b = Previous value of the transmitted Link Code Word = 0b. 0b = Previous value of the transmitted Link Code Word = 1b.	RO	0b	0b
12	Acknowledge 2	1b = Link Partner complies with the message. 0b = Link Partner cannot comply with the message.	RO	0b	0b
13	Message Page	1b = Page sent by the Link Partner is a Message Page. 0b = Page sent by the Link Partner is an Unformatted Page.	RO	0b	0b
14	Acknowledge	1b = Link Partner has received Link Code Word from the PHY. 0b = Link Partner has not received Link Code Word from the PHY.	RO	0b	0b
15	Next Page	1b = Link Partner has additional next pages to send. 0b = Link Partner has no additional next pages to send.	RO	0b	0b

10.4.8.10 1000BASE-T/100BASE-T2 Control Register - GCON (09d; R/W)

Bit(s)	Field	Description	Mode	HW Rst	SW Rst
7:0	Reserved	Always read as 0b. Write to 0b for normal operation.	R/W	0b	0b
8	1000BASE-T Half Duplex	1b = DTE is 1000BASE-T capable. 0b = DTE is not 1000BASE-T capable. This bit is used by Smart Negotiation.	R/W	0b	Retain
9	1000BASE-T Full Duplex	1b = DTE is 1000BASE-T full duplex capable. 0b = DTE is not 1000BASE-T full duplex capable. This bit is used by Smart Negotiation.	R/W	1b	Retain
10	Port Type	1b = Prefer multi-port device (Master)[1]. 0b = Prefer single port device (Slave)[1].	R/W	0b	Retain
11	Master/Slave Configuration Value	1b = Manual configure as Master[2]. 0b = Manual configure as Slave[2].	R/W	0b	Retain
12	Master/Slave Manual Configuration Enable	1b = Manual Master/Slave configuration. 0b = Automatic Master/Slave configuration.	R/W	0b	Retain



Bit(s)	Field	Description	Mode	HW Rst	SW Rst
15:13	Test mode	000b = Normal Mode. 001b = Pulse and droop template. 010b = Jitter template. 011b = Jitter template. 100b = Distortion packet. 101b, 110b, 111b = Reserved.	R/W	000b	000b

1. Only when PHY register 09d, bit 12 is set to 0b.
2. Only when PHY register 09d, bit 12 is set to 1b.

10.4.8.11 1000BASE-T/100BASE-T2 Status Register - GSTATUS (10d; R)

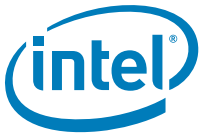
Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Idle Error Count	7:0	Idle Error Counter. This register counts the number of invalid idle codes when the link is high and the PHY is in either 1000BASE-T or 100BASE-T modes. If overflow, these bits are held at all on1bs. They are cleared on read or after a hard or soft reset.	RO, LH	0000b 0000b	0000b 0000b
Reserved	9:8	Reserved. Should be set to 00b	RO	00b	00b
Link Partner 1000BASE-T Half Duplex Capability	10	1b = Link Partner is capable of 1000BASE-T half duplex. 0b = Link Partner is not capable of 1000BASE-T half duplex. Valid after auto-negotiation completes (PHY register 1, bit 5 = 1b).	RO	0b	0b
Link Partner 1000BASE-T Full Duplex Capability	11	1b = Link Partner is capable of 1000BASE-T full duplex. 0b = Link Partner is not capable of 1000BASE-T full duplex. Valid after auto-negotiation completes (PHY register 1, bit 5 = 1b).	RO	0b	0b
Remote Receiver Status	12	1b = Remote Receiver OK. 0 b = Remote Receiver Not OK.	RO	0b	0b
Local Receiver Status	13	1b = Local Receiver OK. 0b = Local Receiver Not OK.	RO	0b	0b
Master/Slave Resolution	14	1b = Local PHY configuration resolved to Master. 0b = Local PHY configuration resolved to Slave. Valid after auto-negotiation completes (PHY register 1, bit 5 = 1b).	RO	0b	0b
Master/Slave Configuration Fault	15	1b = Master/Slave configuration fault detected. 0b = No Master/Slave configuration fault detected.	RO, LH	0b	0b

10.4.8.12 Extended Status Register - ESTATUS (15d; R)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Reserved	11:0	Reserved. Always read as 0b.	RO	0b	0b
1000BASE-T Half Duplex	12	1b = 1000BASE-T half duplex capable. 0b = not 1000BASE-T half duplex capable.	RO	1b	1b
1000BASE-T Full Duplex	13	1b = 1000BASE-T full duplex capable. 0b = Not 1000BASE-T full duplex capable.	RO	1b	1b
1000BASE-X Half Duplex	14	1b =1000BASE-X half duplex capable. 0b = Not 1000BASE-X half duplex capable.	RO	0b	0b
1000BASE-X Full Duplex	15	1b =1000BASE-X full duplex capable. 0b = Not 1000BASE-X full duplex capable.	RO	0b	0b

10.4.8.13 Port Configuration Register - PCONF (16d; R/W)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Reserved	0	Write to 0b for normal operation.	R/W	0b	0b
Alternate NP Feature	1	1b = Enable alternate Auto-Negotiate next page feature. 0b = Disable alternate Auto-Negotiate next page feature.	R/W	0b	0b
Reserved	3:2	Write to 0b for normal operation.	R/W	0b	0b
Auto MDIX Parallel Detect Bypass	4	Auto_MDIX Parallel Detect Bypass. Bypasses the fix to IEEE auto-MDIX algorithm for the case where the PHY is in forced-speed mode and the link partner is auto-negotiating. 1b = Strict 802.3 Auto-MDIX algorithm. 0b = Auto-MDIX algorithm handles Auto-Negotiation disabled modes. This is accomplished by lengthening the auto-MDIX switch timer before attempting to swap pairs on the first time out.	R/W	0b	0b
PRE_EN	5	Preamble Enable 0b = Set RX_DV high coincident with SFD. 1b = Set RX_DV high and RXD = preamble (after CRS is asserted).	R/W	1b	1b
Reserved	6	Write to 0b for normal operation.	R/W	0b	0b
Smart Speed	7	1b = Smart Speed selection enabled. 0b = Smart Speed selection disabled.	R/W	1b	1b
TP Loopback (10BASE-T)	8	1b = Disable TP loopback during half-duplex operation. 0b = Normal operation.	R/W	1b	1b
Reserved	9	Write to 0b for normal operation.	R/W	0b	0b
Jabber (10BASE-T)	10	1b = Disable jabber. 0b = Enable jabber.	R/W	0b	0b



Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Bypass 4B5B (100BASE-TX)	11	1b = Bypass4B5B encoder and decoder. 0b = Normal operation.	R/W	0b	0b
Bypass Scramble (100BASE-TX)	12	1b = Bypass scrambler and descrambler. 0b = Normal operation.	R/W	0b	0b
Transmit Disable	13	1b = Disable twisted-pair transmitter. 0b = Normal operation.	R/W	0b	0b
Link Disable	14	1b = Force link pass 0b = Normal operation For 10BASE-T, this bit forces the link signals to be active. In 100BASE-T mode, setting this bit should force the Link Monitor into it's LINKGOOD state. For 1000BASE-T operation, this bypasses Auto-Negotiation—the link signals still correctly indicate the appropriate status.	R/W	0b	0b
Reserved	15	Always read as 0b. Write 0b for normal operation.	R/W	0b	0b

10.4.8.14 82567 Copper Specific Control Register - (16d; R/W)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Disable Jabber	0	1b = Disable jabber function. 0b = Enable jabber function. **Note:** Jabber is only applicable while in 10BASE-T half-duplex mode.	R/W	0b	Retain
Polarity Reversal Disable	1	If polarity is disabled, then the polarity is forced to normal in 10BASE-T. 1b = Polarity reversal disabled. 0b = Polarity reversal enabled. The detected polarity status is shown in the Copper Specific Status register (bit 1) or while in 1000BASE-T mode, bits 3:0 of the 1000BASE-T Pair Swap and Polarity register.	R/W	0b	Retain
Power Down	2	Power down is controlled by the PHY Control register (bit 11) and the Copper Specific Control register (bit 2). Both bits must be set to 0b before the **82567** transitions from power down to normal operation. When the port is switched from power down to normal operation, software reset and restart auto-negotiation are performed even when bits 9 and 15 of the PHY Control register are not set. IEEE power down shuts down the **82567**, except for the GMII interface, if bit 3 of the MAC Specific Control Register 1 is set to 1b. If set to 0b, then the GMII interface also shuts down. 1b = Power down. 0b = Normal operation.	R/W	0b	Retain
Copper Transmitter Disable	3	1b = Transmitter disable. 0b = Transmitter enable.	R/W	0b	Retain

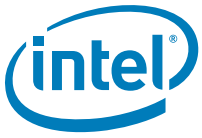
Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Reserved	4	Reserved. Write to 0b for normal operation.	R/W	0b	Retain
MDI Crossover Mode	6:5	Changes to these bits can be disruptive to the normal operation. As a result, any changes to these registers must be followed by a software reset to take effect. 00b = Manual MDI configuration. 01b = Manual MDI-X configuration. 10b = Reserved. 11b = Enable automatic crossover for all modes.	R/W	11b	Update
Enable Extended Distance	7	When using cable exceeding 100 m, the 10BASE-T receive threshold must be lowered in order to detect incoming signals. 1b = Lower 10BASE-T receive threshold. 0b = Normal 10BASE-T receive threshold.	R/W	0b	Retain
Energy Detect	9:8	After a hardware reset, both bit takes on the value of pd_config_edet_a. 0xb = Off. 10b = Sense only on receive (energy detect). 11b = Sense and periodically transmit NLP (energy detect and TM).	R/W		Update
Force Copper Link Good	10	If link is forced good, the link state machine is bypassed and the link is always up. In 1000BASE-T mode, this field has no effect. 1b = Force link good. 0b = Normal operation.	R/W	0b	Retain
Downshift Enable	11	Changes to these bits can be disruptive to normal operation. As a result, any changes to these registers must be followed by a software reset to take effect. 1b = Enable downshift. 0b = Disable downshift.	R/W	0b	Update
Downshift Counter	14:12	Changes to these bits can be disruptive to normal operation. As a result, any changes to these registers must be followed by a software reset to take effect. 1x, 2x,...8x is the number of times the **82567** attempts to establish GbE link before it downshifts to the next highest speed. 000b = 1x. 100b = 5x. 001b = 2x. 101b = 6x. 010b = 3x. 110b = 7x. 011b = 4x. 111b = 8x.	R/W	011b	Update
Disable Link Pulses	15	1b = Disable link pulse. 0b = Enable link pulse.	R/W	0b	0b

10.4.8.15 82567 MAC Control Register 1 - (Page 2, 16d; R/W)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Reserved	1:0	Reserved. Write as 00b	R/W	0b	Retain
Reserved	2	Reserved. Write as 0b	R/W	0b	Update
GMII Interface Power Down	3	Changes to this bit can be disruptive to normal operation. As a result, any changes to these registers must be followed by a software reset to take effect. This bit determines whether the GMII RX_CLK powers down when bit 11 of the PHY Control register and bit 2 of the Copper Specific Control register are used to power down the **82567** or when the **82567** enters the energy detect state. 1b = Always power up. 0b = Can power down.	R/W	1b	Update
Reserved	6:4	Reserved. Write as 000b.	R/W	000b	Retain
Reserved	7	Reserved. Write as 1b.	R/W	1b	Update
Disable fi_50_clk	8	On a hardware reset, this bit takes on the value of pd_pwrdn_clk50_a. When pd_pwrdn_clk50_a transitions from 1b to 0b, this bit is set to 0b. When pd_pwrdn_clk50_a transitions from 0b to 1b, this bit is set to 1b. 1b = fi_50_clk low. 0b = fi_50_clk toggle.	R/W		Retain
Disable fi_125_clk	9	Changes to these bits can be disruptive to the normal operation. As a result, any changes to these registers must be followed by a software reset to take effect. On a hardware reset, this bit takes on the value of pd_pwrdn_clk125_a. When pd_pwrdn_clk125_a transitions from 1b to 0b, this bit is set to 0b. When pd_pwrdn_clk50_a transitions from 0b to 1b, this bit is set to 1b. 1b = fi_125_clk low. 0b = fi_125_clk toggle.	R/W		Retain
Reserved	13:10	Reserved	R/W	00h	Retain
Transmit FIFO Depth	15:14	1000BASE-T: 00b = ± 16 bits. 01b = ± 24 bits. 10b = ± 32 bits. 11b = ± 40 bits.	R/W	00h	Retain

10.4.8.16 Port Status 1 Register - PSTAT (17d; RO)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
LFIT Indicator	0	Status bit indicating the Auto-Negotiation Link Fail Inhibit Timer has expired. This indicates that the Auto-Negotiation process completed page exchanges but was unable to bring up the selected MAU's link. 1b = Auto-Negotiation has aborted Link establishment following normal page exchange. 0b = Auto-Negotiation has either completed normally, or is still in progress. This bit is cleared when read or when one of the following occurs: Link comes up (PHY register 17d, bit 10 = 1b). Auto-Negotiation is disabled (PHY register 0d, bit 12 = 0b). Auto-Negotiation is restarted (PHY register 0d, bit 9 = 1b).	RO/ LH/SC	0b	0b
Polarity Status	1	1b = 10BASE-T polarity is reversed. 0b = 10BASE-T polarity is normal.	RO	0b	0b
Reserved	8:2	Write to 0b for normal operation.	RO	0b	0b
Duplex Mode	9	1b = Full duplex. 0b = Half duplex.	RO	0b	0b
Link	10	Indicates the current status of the link. Differs from PHY register 01, bit 2 in that this bit changes anytime the link status changes. PHY register 01, bit 2 latches low and stays low until read regardless of link status. 1b = Link is currently up. 0b = Link is currently down.	RO	0b	0b
MDI-X Status	11	Status indicator of the current MDI/ MDI-X state of the twisted pair interface. This status bit is valid regardless of the MAU selected. 1b = PHY has selected MDI-X (crossed over). 0b = PHY has selected MDI (NOT crossed over).	RO	0b	0b
Receive Status	12	1b = PHY currently receiving a packet. 0b = PHY receiver is IDLE. When in internal loopback, this bit reads as 0b.	RO	0b	0b
Transmit Status	13	1b = PHY currently transmitting a packet. 0b = PHY transmitter is IDLE. When in internal loopback, this bit reads as 0b.	RO	00b	00b



Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Data Rate	15:14	00b = Reserved. 01b = PHY operating in 10BASE-T mode. 10b = PHY operating in 100BASE-TX mode. 11b = PHY operating in 1000BASE-T mode.	RO	00b	00b

10.4.8.17 82567 Copper Specific Status Register 1 - (17d; RO)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Jabber (Real Time)	0	1b = Jabber enabled. 0b = Jabber disabled.	RO	0b	0b
Polarity (Real Time)	1	1b = Reversed. 0b = Normal polarity reversal can be disabled by writing to the Copper Specific Control Register 1 (bit 1). If in 1000BASE-T mode, polarity of all pairs are shown in bits 3:0 of the 1000BASE-T Pair Swap and Polarity register.	RO	0b	0b
Reserved	2	Reserved	RO	0b	0b
Global Link Status	3	1b = Copper link is up. 0b = Copper link is down.	RO	0b	0b
Copper Energy Detect Status	4	1b = Sleep. 0b = Active.	RO	0b	0b
Downshift Status	5	1b = Downshift. 0b = No Downshift.	RO	0b	0b
MDI Crossover Status	6	This status bit is valid only after resolved bit 11 = 1b. The resolved bit is set when auto-negotiation completes or when auto-negotiation is disabled. This bit is 0b or 1b depending on what is written to bits 6:5 of the Copper Specific Control Register 1 (while in manual configuration mode). **Note:** Copper Specific Control Register 1 bits 6:5 are updated by a software reset. 1b = MDIX. 0b = MDI.	RO	0b	0b
Reserved	7	Reserved	RO	0b	0b
Receive Pulse Enabled	8	This bit reflects the MAC pause resolution and is used for information purposes. It is not used by the **82567**. This status bit is valid only after resolved bit 11 = 1b. The resolved bit is set when auto- negotiation completes or when auto-negotiation is disabled. 1b = Receive pause enabled. 0b = Receive pause disabled.	RO	0b	0b

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Transmit Pause Enable	9	This bit reflects the MAC pause resolution and is used for information purposes. It is not used by the **82567**. This status bit is valid only after resolved bit 11 = 1b. The resolved bit is set when auto- negotiation completes or when auto-negotiation is disabled. 1b = Transmit pause enabled. 0b = Transmit pause disabled.	RO	0b	0b
Copper Link (Real Time)	10	1b = Link is currently up. 0b = Link is currently down.	RO	0b	0b
Speed and Duplex Resolved	11	When auto-negotiation is not enabled, this bit equals 1b. 1b = Resolved. 0b = Not resolved.	RO	0b	0b
Page Received	12	1b = Page received. 0b = Page not received.	RO	0b	0b
Duplex	13	This status bit is valid only after resolved bit 11 equals 1b. The resolved bit is set when auto-negotiation completes or when auto-negotiation is disabled. 1b = Full duplex. 0b = Half duplex.	RO	0b	Retain
Speed	15:14	These status bits are valid only after resolved bit 11 equals 1b. The resolved bit is set when auto-negotiation completes or when auto-negotiation is disabled. 11b = Reserved. 10b = 1000 Mb/s. 01b = 100 Mb/s. 00b = 10 Mb/s.	RO	10b	Retain

10.4.8.18 82567 CRC Counters Register - (Page 6, 17d; R/W)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
CRC Error Count	7:0	00h = No packets received. FFh = 256 packets received (max count). Bit 4 of the Packet Generation register must be set to 1b in order for this register to be valid.	R/W	00h	Retain
CRC Packet Count	15:8	00h = No CRC errors detected in the packets received. FFh = 256 CRC errors detected in the packets received (max count). Bit 4 of the Packet Generation register must be set to 1b in order for this register to be valid.	R/W	00h	Retain

10.4.8.19 Port Control Register - PCONT (18d; R/W)

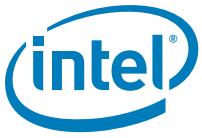
Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Reserved	3:0	Always read as 0b. Write to 0b for normal operation.	R/W	0b	0b
TP Loopback	4	Allow GbE loopback on twisted pairs.	R/W	0b	0b
Extend SPD Delay[1]	5	When set, extends the delay of a power down if the Ethernet cable is disconnected. 0b = Wait four seconds before beginning power down. 1b = Wait 6.3 seconds before beginning power down.	R/W	0b	0b
Reserved	8:6	Always read as 0b. Write to 0b for normal operation.	R/W	0b	0b
Non-Compliant Scrambler Compensation	9	1b = Detect and correct for non-compliant scrambler. 0b = Detect and report non-compliant scrambler.	R/W	1b	1b
TEN_CRS_Select	10	1b = Extend CRS to cover 1000Base-T latency and RX_DV. 0b = Do not extend CRS (RX_DV can continue past CRS).	R/W	1b	1b
Flip_Chip	11	Used for applications where the core or application is mirror-imaged. Channel D acts like channel A with t10pol_inv set and vice-versa. Channel C acts like channel B with t10pol_inv set and vice-versa. This forces the correctness of all MDI/MDIX and polarity issues.	R/W	0b	0b
Auto-MDI-X	12	Auto-MDI-X algorithm enable. 1b = Enable Auto-MDI-X mode. 0b = Disable Auto-MDI-X mode (manual mode). **Note:** When forcing speed to 10BASE-T or 100BASE-T, use manual mode. Clear the bit and set PHY register 18d, bit 13 according to the required MDI-X mode.	R/W	1b	1b
MDI-X Mode	13	Force MDI-X mode. Valid only when operating in manual mode. (PHY register 18d, bit 12 = 0b. 1b = MDI-X (cross over). 0b = MDI-X (no cross over).	R/W	0b	0b
Reserved	14	Always read as 0b. Write to 0b for normal operation.	R/W	0b	0b

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Jitter Test Clock	15	This configuration bit is used to enable the PHY to drive its differential transmit clock out through the appropriate Analog Test (ATEST+/-) output pads. This feature is required in order to demonstrate conformance to the IEEE Clause 40 jitter specification. When high, it sends Jitter Test Clock out. This bit works in conjunction with internal/external PHY register 4011h, bit 15. In order to have the clock probed out, it is required to perform the following write sequence: PHY register 18d, bit 5 = 1b PHY register 31d = 4010h (page select) PHY register 17d = 0080h PHY register 31d = 0000h (page select)	R/W	0b	0b

1. Not applicable to the 82567.

10.4.8.20 82567 Copper Specific Interrupt Enable Register - (18d; R/W)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Jabber Interrupt Enable	0	1b = Interrupt enable. 0b = Interrupt disable.	R/W	0b	Retain
Polarity Changed Interrupt Enable	1	1b = Interrupt enable. 0b = Interrupt disable.	R/W	0b	Retain
Reserved	2	Reserved. Set to 0b.	R/W	0b	Retain
FLP Exchange Complete But No Link Interrupt Enable	3	1b = Interrupt enable. 0b = Interrupt disable.	R/W	0b	Retain
Energy Detect Interrupt Enable	4	1b = Interrupt enable. 0b = Interrupt disable.	R/W	0b	Retain
Downshift Interrupt Enable	5	1b = Interrupt enable. 0b = Interrupt disable.	R/W	0b	Retain
MDI Crossover Changed Interrupt Enable	6	1b = Interrupt enable. 0b = Interrupt disable.	R/W	0b	Retain
Reserved	7	Reserved. Set to 0b.	R/W	0b	Retain
False Carries Interrupt Enable	8	1b = Interrupt enable. 0b = Interrupt disable.	R/W	0b	Retain
Symbol Error Interrupt Enable	9	1b = Interrupt enable. 0b = Interrupt disable.	R/W	0b	Retain
Link Status Changed Interrupt Enable	10	1b = Interrupt enable. 0b = Interrupt disable.	R/W	0b	Retain
Auto-Negotiation Completed Interrupt Enable	11	1b = Interrupt enable. 0b = Interrupt disable.	R/W	0b	Retain
Page Received Interrupt Enable	12	1b = Interrupt enable. 0b = Interrupt disable.	R/W	0b	Retain



Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Duplex Changed Interrupt Enable	13	1b = Interrupt enable. 0b = Interrupt disable.	R/W	0b	Retain
Speed Changed Interrupt Enable	14	1b = Interrupt enable. 0b = Interrupt disable.	R/W	0b	Retain
Auto-Negotiation Error Interrupt Enable	15	1b = Interrupt enable. 0b = Interrupt disable.	R/W	0b	Retain

10.4.8.21 82567 MAC Specific Interrupt Enable Register - (Page 2, 18d; R/W)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Reserved	1:0	Reserved.	R/W	00h	Retain
FIFO Idle Detected Interrupt Enable	2	1b = Interrupt enable. 0b = Interrupt disable.	R/W	0b	Retain
FIFO Idle Inserted Interrupt Enable	3	1b = Interrupt enable. 0b = Interrupt disable.	R/W	0b	Retain
Reserved	6:4	Reserved.	R/W	00h	Retain
FIFO Over/Underflow Interrupt Enable	7	1b = Interrupt enable. 0b = Interrupt disable.	R/W	0b	Retain
Reserved	15:8	Reserved.	R/W	00h	Retain

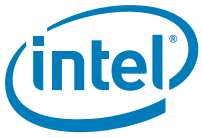
10.4.8.22 Link Health Register - LINK (19d; RO)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Valid Channel A	0	The channel A DSP had converged to incoming data.	RO	0b	0b
Valid Channel B	1	The channel B DSP had converged to incoming data.	RO	0b	0b
Valid Channel C	2	The channel C DSP had converged to incoming data.	RO	0b	0b
Valid Channel D	3	The channel D DSP had converged to incoming data. If An_Enable is true, valid_chan_A = dsplockA latched on the rising edge of link_fail_inhibit_timer_done and link = 0b. If An_enable is false, valid_chan_A = dsplockA.	RO	0b	0b
Auto-Negotiation Active	4	Auto-Negotiate is actively deciding HCD.	RO	0b	0b
Reserved	5	Always read as 0b.	RO	0b	0b
Auto-Negotiation Fault	6	Auto-Negotiate Fault: This is the logical OR of PHY register 01d, bit 4, PHY register 06d, bit 4, and PHY register 10d, bit 15.	RO	0b	0b
Reserved	7	Always read as 0b.	RO	0b	0b

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Data Err[0]	8	Mode: 10: 10 Mb/s polarity error. 100: Symbol error. 1000: GbE idle error.	LH	0b	0b
Data Err[1]	9	Mode: 10: N/A. 100: Scrambler unlocked. 1000: Local receiver not OK.	RO/ LH	0b	0b
Count Overflow	10	32 idle error events were counted in less than 1 ms.	RO/ LH	0b	0b
Gigabit Rem Rcvr NOK	11	Gig has detected a remote receiver status error. This is a latched high version of PHY register 10d, bit 12.	RO/ LH	0b	0b
Gigabit Master Resolution	12	Gig has resolved to master. This is a duplicate of PHY register 10d, bit 14. Programmers must read PHY register 10d, bit 14 to clear this bit.	RO	0b	0b
Gigabit Master Fault	13	A fault has occurred with the gig master/slave resolution process. This is a copy of PHY register 10, bit 15. Programmers must read PHY register 10, bit 15 to clear this bit.	RO	0b	0b
Gigabit Scrambler Error	14	1b indicates that the PHY has detected gigabit connection errors that are most likely due to a non-IEEE compliant scrambler in the link partner. 0b = Normal scrambled data. Definition is: If an_enable is true and in GbE mode, on the rising edge of internal signal link_fail_inibit timer_done, the dsp_lock is true but loc_rcvr_OK is false.	RO	0b	0b
SS Downgrade	15	Smart Speed has downgraded the link speed from the maximum advertised.	RO/ LH	0b	0b

10.4.8.23 82567 Copper Specific Status Register 2 - (19d; RO)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Jabber	0	1b = Jabber. 0b = No jabber.	RO/ LH	0b	0b
Polarity Changed	1	1b = Polarity changed. 0b = Polarity not changed.	RO/ LH	0b	0b
Reserved	2	Reserved.	RO	0b	0b
FLP Exchange Complete But No Link	3	1b = FLP exchange completed but link not established. 0b = No event detected.	RO	0b	0b
Energy Detect Changed	4	1b = Energy detect state changed. 0b = No energy detect state changed detected.	RO/ LH	0b	0b
Downshift Interrupt	5	1b = Downshift detected. 0b = No downshift.	RO/ LH	0b	0b
MDI Crossover Changed	6	1b = Crossover changed. 0b = Crossover not changed.	RO/ LH	0b	0b



Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Reserved	7	Always 0b.	RO	0b	0b
Copper False Carrier	8	1b = False carrier. 0b = No false carrier.	RO/ LH	0b	0b
Copper Symbol Error	9	1b = Symbol error. 0b = No symbol error.	RO/ LH	0b	0b
Copper Link Status Changed	10	1b = Link status changed. 0b = Link status not changed.	RO/ LH	0b	0b
Copper Auto-Negotiation Completed	11	1b = Auto-negotiation completed. 0b = Auto-negotiation not completed.	RO/ LH	0b	0b
Copper Page Received	12	1b = Page received. 0b = Page not received.	RO/ LH	0b	0b
Copper Duplex Changed	13	1b = Duplex changed. 0b = Duplex not changed.	RO/ LH	0b	0b
Copper Speed Changed	14	1b = Speed changed. 0b = Speed not changed.	RO/ LH	0b	0b
Copper Auto-Negotiation Error	15	An error can occur if the master/slave does not resolve parallel detect fault, no common HCD, or link does not come up after negotiation completes. 1b = Auto-negotiation error. 0b = No auto-negotiation error.	RO/ LH	0b	0b

10.4.8.24 82567 MAC Specific Status Register - (Page 2, 19d; RO)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Reserved	1:0	Reserved. Always 00b.	RO	00b	00b
FIFO Idle Deleted	2	1b = Idle deleted. 0b = Idle not deleted.	RO/ LH	0b	0b
FIFO Idle Inserted	3	1b = Idle inserted. 0b = No idle inserted.	RO/ LH	0b	0b
Reserved	6:4	Reserved. Always 000b	RO/ LH	000b	000b
FIFO Over/Underflow	7	1b = Over/underflow error. 0b = No FIFO error.	RO/ LH	0b	0b
Reserved	15:8	Reserved. Always 00h	RO	00h	00h

10.4.8.25 GMII FIFO Register and Smart Power Down[1] - (20d; RO)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Buffer Size	3:0	An unsigned integer that stipulates the number of write clocks to delay the read controller after the internal GMII's tx_en is first asserted. This buffer protects from underflow at the expense of latency. The maximum value that can be set is 13d or Dh.	R/W	0101b	0101b

1. Smart power down not applicable to the **82567**.

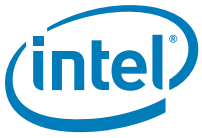
Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Reserved	7:4	Reserved. Always read as 0b. Write to 0b for normal operation.	R/W	0000b	0000b
FIFO Out Steering	9:8	00b, 01b - Enable the output data bus from GMII FIFO to transmitters, drives 0bs on the output loopback bus from the GMII FIFO to an external application and to the DSP RX-FIFOs in test mode. 10b - Drive 0bs on the output bus from the GMII FIFO to transmitters, Enable data on the output loopback bus from GMII FIFO to an external application and to the DSP RX-FIFOs in test mode. 11b - Enable the output data bus from the GMII FIFO to both transmitters and loopback bus.			
Disable Error Out	10	When set, disables the addition of under/overflow errors to the output data stream on internal GMII's tx_error.	R/W	0b	0b
Reserved	13:11	Reserved. Always read as 0b. Write to 0b for normal operation.	R/W	000b	000b
FIFO Overflow'	14	Status bit set when read clock that is slower than internal GMII's gtx_clk has enabled the FIFO to fill to capacity mid packet. Decrease buffer size.	RO/ LH	0b	0b
FIFo Underflow	15	Status bit set when read clock that is faster than internal GMII's gtx_clk empties the FIFO mid packet. Increase the buffer size.	RO/ LH	0b	0b

10.4.8.26 82567 Copper Specific Register 3 - (20d; R/W)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Reverse MDI_PLUS/ MDI_MINUS [3] Transmit Polarity	0	0b = Normal Transmit Polarity 1b = Reverse Transmit Polarity	R/W	0b	Retain
Reverse MDI_PLUS/ MDI_MINUS [2] Transmit Polarity	1	0b = Normal Transmit Polarity 1b = Reverse Transmit Polarity	R/W	0b	Retain
Reverse MDI_PLUS/ MDI_MINUS [1] Transmit Polarity	2	0b = Normal Transmit Polarity 1b = Reverse Transmit Polarity	R/W	0b	Retain
Reverse MDI_PLUS/ MDI_MINUS [0] Transmit Polarity	3	0b = Normal Transmit Polarity 1b = Reverse Transmit Polarity	R/W	0b	Retain
Reserved	15:4	Reserved. Write all as 0b.	R/W	00h	Retain

10.4.8.27 82567 1000BASE-T Pair Skew Register - (Page 5, 20d; RO)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Pair 1, 2 (MDI[0]±)	0	Skew = Bit value x 8 ns. Value is correct to within ± 8 ns. The contents of bits 15:0 are valid only if bit 6 of the 1000BASE-T Pair Swap and Polarity register equals 1b.	RO	0b	0b
Pair 3, 6 (MDI[1]±)	1	Skew = Bit value x 8 ns. Value is correct to within ± 8 ns.	RO	0b	0b



Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Pair 4, 5 (MDI[2]±)	2	Skew = Bit value x 8 ns. Value is correct to within ± 8 ns.	RO	0b	0b
Pair 7, 8 (MDI[3]±)	3	Skew = Bit value x 8 ns. Value is correct to within ± 8 ns.	RO	0b	0b

10.4.8.28 Channel Quality Register - CHAN (21d; RO)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
MSE_A	3:0	The converged mean square error for Channel A.	RO	0b	0b
MSE_B	7:4	The converged mean square error for Channel B.	RO	0b	0b
MSE_C	11:8	The converged mean square error for Channel C.	RO	0b	0b
MSE D	15:12	The converged mean square error for Channel D. This field is only meaningful in gigabit, or in 100BASE-TX if this is the receive pair. Use of this field is complex and needs interpretation based on the chosen threshold value.	RO	0b	0b

10.4.8.29 82567 Receive Error Counter Register - (21d; RO)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Receive Error Count	15:0	Counter pegs at FFFFh and does not roll over. Both false carrier and symbol errors are reported.	RO/ LH	00h	Retain

10.4.8.30 82567 MAC Specific Control Register 2 - (Page 2, 21d; RO)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Default MAC interface speed	2:0	Changes to these bits can be disruptive to the normal operation. As a result, any changes to these registers must be followed by a software reset to take effect. MAC interface speed during link down while auto-negotiation is enabled. TX_CLK speed bit speed link down 1000BASE-T. 000b = 10 Mb/s 2.5 MHz. 001b = 100 Mb/s 25 MHz. 01Xb = 1000 Mb/s 0 MHz. 100b = 10 Mb/s 2.5 MHz. 101b = 100 Mb/s 25 MHz. 110b = 1000 Mb/s 2.5 MHz. 111b = 1000 Mb/s 25 MHz.	R/W	110b	Update

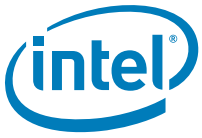
Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Block Carrier Extension Bit	3	1b = Enable block carrier extension. 0b = Disable block carrier extension.	R/W	0b	Retain
Reserved	5:4	Reserved.	R/W	00b	Retain
Reserved	6	Reserved.	R/W	1b	Update
Reserved	11:7	Reserved.	R/W	00h	00h
Reserved	13:12	Reserved.	R/W	11b	Update
Line Loopback	14	1b = Enable line loopback. 0b = Normal operation.	R/W	0b	0b
Reserved	15	Reserved.	R/W	0b	0b

10.4.8.31 82567 1000BASE-T Pair Swap and Polarity Register - (Page 5, 21d; RO)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Pair 1, 2 (MDI[0]±) Polarity	0	1b = Negative. 0b = Positive.	RO	0b	0b
Pair 3, 6 (MDI[1]±) Polarity	1	1b = Negative. 0b = Positive.	RO	0b	0b
Pair 4, 5 (MDI[2]±) Polarity	2	1b = Negative. 0b = Positive.	RO	0b	0b
Pair 7, 8 (MDI[3]±) Polarity	3	1b = Negative. 0b = Positive.	RO	0b	0b
A, B Crossover	4	1b = Channel A received on MDI[0]±; Channel B received on MDI[1]±. 0b = Channel B received on MDI[0]±; Channel A received on MDI[1]±.	RO	0b	0b
C, D Crossover	5	1b = Channel A received on MDI[2]±; Channel B received on MDI[3]±. 0b = Channel B received on MDI[2]±; Channel A received on MDI[3]±.	RO	0b	0b
Register 20_5 and 21_5 Valid	6	The contents of these register bits are valid only if bit 6 equals 1b. 1b = Valid. 0b = Invalid.	RO	0b	0b
Reserved	15:7	Reserved	RO	00h	00h

10.4.8.32 82567 Page Address Register (Any Page) - (22d; RO)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Page Select for Registers 0d Through 28d	7:0	Page number.	RO	Always 00h	Always 00h
Reserved	15:8	Reserved	RO	Always 00h	Always 00h



10.4.8.33 PHY Power Management - (25d; R/W)

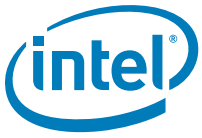
Note: OEM Bits register for the **82567**.

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Reserved	15:11	Always read as 0b. Write to 0b for normal operation.	R/W	00h	00h
Restart AN Aneg_now (**82567**)	10	Shadow bit of PHY register 0d, bit 9. For the **82567**, Restart auto-negotiation. Note that this bit is self-cleared.	R/W, SC	0b	0b
cdc_rst	9	Internal CDC reset indication. **Note:** This is a reserved bit for the **82567**.	RO/ R/W[1]	0b	0b
rst_compl	8	Indicates PHY internal reset cleared. **Note:** This is a reserved bit for the **82567**.	LH R/W[1]	0b	0b
SPD_B2B_EN	7	Smart Power Down Back-to-Back Enable. **Note:** This is a reserved bit for the **82567**.	R/W	1b 0b[1]	1b 0b[1]
Disable 1000	6	When set, disables 1000 Mb/s operation on the next auto-negotiation (either by link lose or by initiated restart auto-negotiation command by the programmer). Note that bit can be loaded from the NVM.	R/W	0b	0b Retain[1]
Reserved	5	Reserved.	R/W	0b	0b
Link Energy Detect	4	This bit is set when the PHY detects energy on the link. Note that this bit is valid only if AN enabled (PHY register 00d, bit 12) and SPD_EN is enabled (PHY register 25d, bit 0). **Note:** This is a reserved bit for the **82567**.	RO	0b	0b
Reserved	3	Reserved. Must be set to 0b.	R/W	0b	0b
LPLU rev_aneg (**82567**)	2	Low Power on Link Up When set, enables the decrease in link speed while in non-D0a states when the power policy and power management state specify it. Note that bit can be loaded from the NVM.	R/W	1b	1b Retain[1]
Reserved	1	Reserved. Must be set to 0b.	R/W	0b	0b
SPD_EN	0	Smart Power Down When set, enables PHY Smart Power Down mode. Note that bit can be loaded from the NVM. **Note:** This is a reserved bit for the **82567**.	R/W	1b 0b[1]	1b 0b[1]

1. **82567** only.

10.4.8.34 82567 Copper Specific Control Register 2 - (26d; R/W)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
1000BASE-T Transmitter Type	15	0b = Class B. 1b = Class A.	R/W	0b	0b
Disable 1000BASE-T	14	When set to disabled, 1000BASE-T is not advertised even if bits 9 and 8 of the 1000BASE-T/100BASE-T2 Control register are set to 1b. A write to this register bit does not take effect until any one of the following occurs: • Software reset is asserted (PHY Control register, bit 15) • Restart auto-negotiation is asserted (PHY Control register, bit 9) • Power down: PHY Control register (bit 11) and Copper Specific Control register 1 (bit 2) transitions from power down to normal operation. Copper link goes down. After a hardware reset, this bit defaults as follows: • ps_a1000_dis_s (bit 14). When ps_a1000_dis_s transitions from 1b to 0b, this bit is set to 0b. • When ps_a1000_dis_s transitions from 0b to 1b, this bit is set to 1b. 1b = Disable 1000BASE-T advertisement. 0b = Enable 1000BASE-T advertisement.	R/W		Retain
Reverse Autoneg	13	A write to this register bit does not take effect until any one of the following occurs: • Software reset is asserted (PHY Control register, bit 15). • Restart auto-negotiation is asserted (PHY Control register, bit 9) • Power down (PHY Control register, bit 11 and bit 2 of the, Copper Specific Control register 1) transitions from power down to normal operation. Copper link goes down. After a hardware reset, this bit defaults as follows: • pd_rev_aneg_a (bit 13). When pd_rev_aneg_a transitions from 1b to 0b, this bit is set to 0b. • When pd_rev_aneg_a transitions from 0b to 1b, this bit is set to 1b. 1b = Reverse auto-negotiation. 0b = Normal auto-negotiation.	R/W		Retain
100BASE-T Transmitter Type	12	0b = Class B. 1b = Class A.	R/W	0b	Retain
Reserved	11:4	Reserved. Write all as 0b.	R/W	00h	Retain



Field	Bit(s)	Description	Mode	HW Rst	SW Rst
100 MB Test Select	3:2	0x = Normal operation. 10b = Select 112 ns sequence. 11b = Select 16 ns sequence.	R/W	00b	Retain
10 BT Polarity Force	1	1b = Force negative polarity for receive only. 0b = Normal operation.	R/W	0b	Retain
Reserved	0	Reserved	R/W	0b	Retain

10.4.8.35 Misc Cntrl Register - (27d; R/W)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Reserved	15:13	Reserved.	R/W	000b	000b
Duplexity_manual_set	12	When set, the **82566** sets the duplex mode according to the *Duplex Mode* bit in register 0.	R/W	0b	0b
Reserved	11:9	Reserved.	R/W	0h	0h
ss_cfg_cntr	8:6	Smart speed counter configuration: 1-5 (001b:101b).	R/W	010b	010b
T10_auto_pol_dis	5	When set, disables the auto-polarity mechanism in the 10 block.	R/W	0b	0b
Reserved	4:0	Reserved.	R/W	0h	0h

10.4.8.36 Misc Cntrl Register 2 - (28d; RO)[1]

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Reserved	15:14	Reserved.	R/W	00b	00b
actual_an_adv	13:8	actual_an_adv, which might differ from PHY registers 04d and 09d due to LPLU, an1000_dis, and ssd.	RO	N/A	N/A
Reserved	7:1	Reserved.	R/W	0h	0h
Phy_cdc_reset_mask	0	Mask the phy_cdc_reset signal, 0b = phy_cdc_reset is not masked. 1b = phy_cdc_reset signal is masked, always = 0b	R/W	0b	0b

Note: Always write to this register before reading or writing PHY registers 30 or 31.

Note: Before reading or writing this register, PHY register 29 must first be set to 0b.

1. Not applicable to the **82567**.

10.4.8.37 Page Select Core Register - (31d; WO)[1]

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
PAGE_SEL	15:0	This register is used to swap out the Base Page containing the IEEE registers for Intel reserved test and debug pages residing within the Extended Address space.	WO	0h	0h

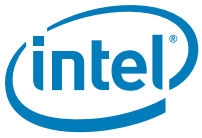
10.4.8.38 GLCI Diagnostic - (Base Address = 34h, Offset = 3h)

Field	Bit(s)	Description	Mode	HW Rst	SW Rst
Reserved	15:14	Reserved	RO	00b	00b
Reserved	13	Reserved	RW	0b	0b
NELPBK	12	Near End Loopback Digital near-end loopback mode. Connects the 8B/10B encoder (Tx) to the 8B/10B decoder (Rx).	RW	0b	0b
Reserved	11:0	Reserved	RO	0h	0h

10.4.8.39 Port General Configuration - (Page 193, 17d; R/W) (Page 769 for the 82567)

Field	Bit(s)	Initial Value	Description	Mode
Reserved	15:2	00h	Reserved for the **82566**.	R/W
TX Gate Wait IFS	15:10	Ah	For the **82567**, determines the size (in nibbles) of non-deferring window from CRS de-assertion.	R/W
TX gate 100 Enable	9	0b	For the **82567**, enables Tx gating for 100 Mb/s half duplex.	R/W
TX gate 10 Enable	8	0b	For the **82567**, enables Tx gating for 10 Mb/s half duplex.	R/W
Reserved	7:6	00b	Reserved for the **82567**.	R/WO
ME_WU_Active	5	0b	Enables ME Wakeup from the **82567**.	
Host_WU_Active	4	0b	Enables host wakeup from the **82567**. This bit is cleared by a PHY reset.	R/W
Wakeup Clocks Stop	3	1b	For the **82567**, wakeup clocks are stopped while wakeup is disabled.	R/W
MACPD_Enable	2	1b	For the **82567**, this bit is written as 1b when the ICH9 needs to globally enable the MAC power down feature while it supports WoL. When set to 1b, pages 800 and 801 are enabled for configuration and *Host_WU_Active*, *ME_WU_Active* are not blocked for writes.	R/W

1. Not applicable to the **82567**.



Field	Bit(s)	Initial Value	Description	Mode
LCI Gigabit Speed	1	0b	Determines the frequency of the LCI data signals when the **82566/82567** is at gigabit speed. 0b = Data signals run at 1/16 of the 62.5 MHz frequency. 1 = Data signals run 62.5 MHz frequency.	R/W
LCI Enable	0	1b	Indicates the value of the LCI_EN strapping options. 0b= LCI-only mode. 1b = GLCI + LCI mode.	RO

10.4.8.40 82567 PHY LEDs Control Register (Page 769, 18d; R/W)

Name	Bits	Initial Value	Description	Mode
Reserved	15	1b	Reserved	RW
Reserved	14:12	000b	Reserved	R/W
no_phy_leds_in_wol	11	0b	When set to 1b, the 82567 LEDs are not used while the 82567 wake-up feature is enabled.	R/W
Led2_sel	10:8	1b	Selects which of the 82567's six LEDs is driven on the LED2 pin when the 82567 LEDs are used.	R/W
Led1_sel	7:5	0b	Selects which of the 82567's six LEDs is driven on the LED1 pin when the 82567 LEDs are used.	R/W
Led0_sel	4:2	010b	Selects which of the 82567's six LEDs is driven on the LED0 pin when the 82567 LEDs are used.	R/W
phy_led_cfg	1:0	10b	LEDs configuration to the 82567.	R/W

10.4.8.41 Rate Adaptation Control - (Page 193, 25d; R/W) (Page 769 for the 82567)

Name	Bits	Initial Value	Description	Mode
Reserved	15:9	0100010b	Reserved, write as read.	RWP
rx_en_rxdv_preamble	8	1b	Enable generation of early preamble based on RX_DV in the receive path.	R/W
rx_en_crs_preamble	7	0b	Enable generation of early preamble based on CRS in the receive path.	R/W
reserved	6	0b	Reserved, write as read.	RWP
rx_flip_bad_sfd	5	1b	Align the packet's start of frame delimiter to a byte boundary in the receive path.	R/W
read_delay_fd	4:0	10011b	Reserved, write as read.	RWP

10.4.8.42 GLCI FIFO's Control and Status - (Page 194, 16d; RO) (Page 770 for the 82567)

Name	Bits	Initial Value	Description	Mode
Reserved	15:10	00h	Reserved.	RO
RX FIFO Underflow	9	0b	Rx FIFO underflow occurred.	RO/SC
RX FIFO Overflow	8	0b	Rx FIFO overflow occurred.	RO/SC
Reserved	7:2	00h	Reserved.	RO
TX FIFO Underflow	1	0b	Tx FIFO underflow occurred.	RO/SC
TX FIFO Overflow	0	0b	Tx FIFO overflow occurred.	RO/SC

10.4.8.43 Power Management Control - (Page 194, 17d; R/W) (Page 770 for the 82567)

Name	Bits	Initial Value	Description	Mode
10/100 Burst Mode Enabled	15	X	10/100 Burst enabled 1b = Burst in 10/100 enabled. 0b = Legacy mode (FIFOs bypassed). Bit reflects status of burst strapping mode. **Note:** This is a reserved bit for the **82566**.	RO
K1 Mode Enable in 10/100	14	0b	K1 enable in 10/100 operating mode. 1b = Enable K1 power down mode. 0b = Disable K1 power down mode. **Note:** This is a reserved bit for the **82566**.	R/W
JCLK Stop in K1	13	0b	Stops JCLK when in K1 power down mode. 0b = JCLK running. 1b = stop JCLK. **Note:** This is a reserved bit for the **82566**.	R/W
Reserved	12:7	00h	Reserved, write as 0b.	RW0
Reserved	6	1b	Reserved	R/W
Reserved	5	1b	Reserved	R/W
Reserved	4:1	00h	Reserved	R/W
Enable Electrical Idle in Link Disconnect	0	1b	Enables GLCI electrical idle in link disconnect.	R/W

10.4.8.44 Inband Control - (Page 194, 18d; R/W) (Page 770 for the 82567)

Name	Bits	Initial Value	Description	Mode
Link Status Transmit Timeout	15:8	5h	Link status retransmission period in tens of µs.	R/W
Reserved	7	0b	Reserved, write as 0b.	RW0
Max Retries	6:0	7h	Maximum retries when not receiving an acknowledge to an inband message.	R/W

10.4.8.45 Acknowledge Timeouts - (Page 194, 20d; R/W) (Page 770 for the 82567)

Name	Bits	Initial Value	Description	Mode
Inband MDIO Acknowledge Timeout	15:8	55h	Timeout in µs for receiving acknowledge for an inband MDIO message (test mode only (**82566/ 82567** operating as a MAC).	R/W
Inband Status Acknowledge Timeout	7:0	2h	Timeout in µs for receiving acknowledge for an inband status message.	R/W

10.4.8.46 Voltage Regulator Control - (Page 200, 18d; R/W)[1]

Name	Bits	Initial Value[1]	Description	Mode
Reserved	15:10	00h	Reserved.	R/W
Device Power-Down Mode	9:8	10b	Defines the power-down mode that the **82566** enters as a result of a power-down command from the MAC. 00b = No action taken. 01b = Enter power-down; keep VRs on. 10b = Enter power-down; shut down VRs. 11b = Reserved.	
Reserved	7:6	00b	Reserved.	R/W
1.0v LVR Output Setting	5:3	011b	**82566**: 000b = 0.85v 001b = 0.90v 010b = 0.95v 011b = 1.00v 100b = 1.05v 101b = 1.10v 110b = 1.15v 111b = 1.20v	R/W

1. Not applicable to the **82567**.

Name	Bits	Initial Value[1]	Description	Mode
1.8v LVR Output Setting	2:0	011b	**82566**: 000b = 1.67v 001b = 1.71v 010b = 1.76v 011b = 1.80v 100b = 1.84v 101b = 1.89v 110b = 1.94v 111b = 1.98v	R/W

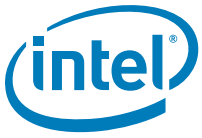
1. Can be set in the NVM.

10.4.8.47 Capability - (Page 200, 19d; RO) (Page 776 for the 82567)

The **82566/82567** Capability register is loaded with the set of capabilities that correspond to the selected **82566/82567** SKU. A change in SKU is reflected in a change in this register. SKU capability is enabled when its corresponding bit is set to 1b.

Name	Bits	Initial Value	Description	Mode
Reserved	15:10	00h	Reserved for future capabilities.	RO
Intel® AMT	9	0b	Intel® AMT Feature. Enables Intel AMT capability.	RO
802.1Q & 802.1p	8	0b	802.1Q & 802.1p. Enables support for VLAN per 802.1Q & 802.1p.	RO
Receive Side Scaling	7	0b	Receive Side Scaling. Enables Receive Side Scaling.	RO
2 Tx & 2 Rx Queues	6	0b	2 Tx & 2 Rx Queues. When set, enables dual transmit and dual receive queues. When cleared, a single receive and a single transmit queue are enabled.	RO
Smart Power Down[1]	5	0b	Smart Power Down. Enables Smart Power-Down capability.	RO
AC/DC Auto Link Speed Connect	4	0b	AC/DC Auto Link Speed Connect. Enables different power management policy in AC and battery modes.	RO
Low Power Linkup (LPU)	3	0b	Low Power Link Up (LPLU). Enables the LPLU capability. **Note:** This is a reserved bit for the **82567**.	RO
ASF	2	0b	ASF. Enables Alert Standard Format (ASF) support.	RO
WfM	1	0b	WfM. Enables Wired-for-Manageability, including ACPI, WoL, and PXE.	RO
Ability to Initiate a Team	0	0b	Ability to initiate a team. Enables teaming capability.	RO

1. Not applicable to the **82567**.

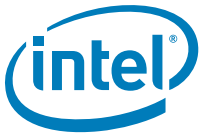


10.4.9 Future Extended NVM - FEXTNVM (00028h; R/W)

This register is initialized to the hardware default only at LAN_RST#. Software should not modify these fields to a different value than their recommended value. Bits 15:0 of this register are loaded from the NVM word 19h and bits 31:16 are loaded from the NVM word 1Ah.

Field	Bit(s)	Initial Value	Description
LANPHYPC Delay	0	0b	**Note:** For **ICH8M** B-1 stepping only: When this bit is set to 1b, it creates a delay between a power down message sent to the PHY and turning off the power to the PHY by setting *LANPHYPC* to an inactive state. When set to 0b, there is no delay between the two actions. This is a reserved bit for **ICH9/ICH10**.
SW PHY Config Enable	0	0b	**Note:** For **ICH8** B-0 Stepping only: This bit has no impact on the hardware but instead influences software flow. Software should initialize the PHY using the Extended Configuration image in the NVM only when both the *SW PHY Config Enable* bit is set and *PHY Write Enable* bit in the EXTCNF_CTRL register is cleared. This is a reserved bit for **ICH9/ICH10**.
dma_clk_ enable_d	1	0b	Enable Dynamic Clock Stop When this bit is set to 1b, the clock is always ticking. The default value is 0b (hardware and NVM).
wake_dma_ clk_enable_d	2	0b	Enable Dynamic Clock Stop When this bit is set to 1b, the clock is always ticking. The default value is 0b (hardware and NVM).
gpt_clk_ enable_d	3	0b	Enable Dynamic Clock Stop When this bit is set to 1b, the clock is always ticking,. The default value is 0b (hardware and NVM).
mac_clk_ enable_d	4	0b	Enable Dynamic Clock Stop When this bit is set to 1b, the clock is always ticking. The default value is 0b (hardware and NVM).
m2k_clk_ enable_d	5	0b	Enable Dynamic Clock Stop When this bit is set to 1b, the clock is always ticking. The default value is 0b (hardware and NVM).
CSUM	6	0b	Invalid image CSUM When cleared, this bit indicates to EEUpdate that the Image CSUM needs correcting. When set, the CSUM is assumed to be correct.
ACBS	7	0b	Auto Connect Battery Saver Supported. This bit indicates to the software device driver that the OEM has enabled the required circuitry for Energy Detection. If this bit is set to 1b, the software device driver is permitted to assert PHY_CTRL.DSPDA. If cleared, PHY power down due to energy detect is unsupported and the software device driver must not assert PHY_CTRL.DSPDA. **Note:** This is a reserved bit for **ICH9/ICH10** and for the **82567**.
Reserved	8	0b	Reserved

Field	Bit(s)	Initial Value	Description
LCI PD_EN	9	0b	LCI Pull Down Enable Enables the pull-down of the LCI I/F pins for the **82566** External Power Down Mode using LANPHYPC. 1b = LCI I/F pins are pulled up on ICH8 side. 0b = LCI I/F pins are pulled down on ICH8 side. This bit can be cleared to 0b on the **82566** if LANPHYPC is not used to control the **82566**'s power supply. **Note:** This is a reserved bit for the **82567**.
MDIOWatchEna	10	0b	Enable MDIO Watchdog Timer (**ICH9/ICH10**) When set to 0b, the 100 ms MDIO watchdog timer is enabled. **Note:** This is a reserved bit for **ICH8** The default NVM setting is 0b.
Reserved	19:11	00h	Reserved
	20	0b	Disable CLK gate Enable Due to D3hot (**ICH9/ICH10**) When set it disables assertion of bb_clkgaten due to D3hot. The default NVM setting is 0b. **Note:** This is a reserved bit for **ICH8**.
	21	0b	LAN Disable Mode (**ICH9/ICH10**) When set to 1b, legacy flow managed by BIOS routine should be performed to disable the LAN. Otherwise, the entire flow is managed by hardware when *Lan-Disable RTC well* bit is set to 1b. The default NVM setting is 0b. **Note:** This is a reserved bit for **ICH8**.
Reserved	26:22	00h	Reserved
SW PHY Config Enable	27	0b	For **ICH9/ICH10** and **ICH8** B-1 stepping only: This bit has no impact on the hardware but instead influences software flow. Software should initialize the PHY using the Extended Configuration image in the NVM only when both the *SW PHY Config Enable* bit is set and *PHY Write Enable* bit in the EXTCNF_CTRL register is cleared. **Note:** This is a reserved bit for **ICH8** B-0 stepping.
Reserved	31:28	00h	Reserved



10.4.10 Future Extended - FEXT (0002Ch; R/W)

This register is initialized to the hardware default only at LAN_RST#. Software should not modify these fields to a different value then their recommended value.

Field	Bit(s)	Initial Value	Description
Reserved	1:0	0b	Reserved
CablDis	2	0b	**82566/82567** Cable Disconnected When set to 1b, indicates that cable has been disconnected. Valid only when the **82566/82567** has a power supply.
PHYPC	3	0b	LAN PHY Power Control 1b = Indicates external power to the **82566** is on. 0b = External power is off. **Note:** This is a reserved bit for the **82567**.
Reserved	6:4	00h	Reserved
	7	0b	XOFF on Pause When set to 1b, enables the transmission of XOFF frames while the transmitter is paused. **Note:** This is a reserved bit for **ICH8/ICH9**.
	8	0b	Hardware/Software CRC Mismatch Trigger When set to 1b, the MAC generates a trigger signal each time there is a mismatch between the software calculated CRC and the hardware calculated CRC. This feature is ignored when the CRC calculation is offloaded to hardware. **Note:** This is a reserved bit for **ICH8/ICH9**.
	9	0b	Write Disable Ghost and DMA RAMs on CRC Mismatch When set to 1b, disables any writes to the following RAMs in the event of a CRC mismatch until reset: • Ghost read PCI descriptor • Ghost read PCI data The four RAMs in the descriptor engine **Note:** This is a reserved bit for **ICH8/ICH9**.
Reserved	31:10	00h	Reserved

10.4.11 ICH9/ICH10 Device and Bus Number – BUSNUM (00038h, RO)

Field	Bit(s)	Initial Value	Description
Reserved	7:0	00h	Reserved
	10:8	00h	Function Number The MAC is a single PCI function being function 0.
	15:11	19h	Device Number During nominal operation, the MAC has a pre-defined device number that equals 25 (19h).
	23:16	00h	Bus Number The MAC captures its bus number during host configuration write cycles type 0 aimed to it. This field is initialized by LAN_RST#, PCI Reset, and D3-to-D0 transition.
	31:24	00h	Reserved

10.4.12 Flow Control Transmit Timer Value - FCTTV (00170h; R/W)

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). If software needs to send an XON frame, it must set the TTV bit to 0b prior to initiating the pause frame.

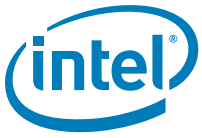
Note: The slot time value that is used is a fixed slot of 64-byte times.

Table 62. FCTTV Register Bit Description

Field	Bit(s)	Initial Value	Description
TTV	15:0	X	Transmit Timer Value This field is included in the XOFF frame.
Reserved	31:16	0b	Reserved Reads as 0b. Should be written to 0b for future compatibility.

10.4.13 ICH10 Flow Control Refresh Threshold Value - FCRTV (0x05F40; RW)

Field	Bit(s)	Initial Value	Description
FCRT	15:0	X	Flow Control Refresh Threshold (FCRT) This value indicates the threshold value of the flow control shadow counter. When the counter reaches this value, and the conditions for a pause state are still valid (buffer fullness above low threshold value), a pause (XOFF) frame is sent to the link partner. The FCRTV timer count interval is the same as other flow control timers and counts at slot times of 64 byte times. If this field contains a zero value, flow control refresh is disabled.
Reserved	31:16	00h	Reserved. Read as 00h. Should be written to 00h for future compatibility.



10.4.14 LED Control - LEDCTL (00E00h; RW)

Table 63. LED Control Bit Description

Field	Bit	Initial Value	Description
LED0_MODE	3:0	0100b	This field specifies the control source for the LED0 output. An initial value of 0010b selects the LINK_UP# indication.
Reserved	4	0b	Reserved. Read-only as 0b. Write as 0b for future compatibility.
GLOBAL_ BLINK_MODE	5	0b	Global Blink Mode This field specifies the blink mode of all the LEDs. 0b = Blink at 200 ms on and 200 ms off. 1b = Blink at 83 ms on and 83 ms off. For **ICH9**, see Note 3 after Table 64.
LED0_IVRT	6	0b	LED0 Invert This field specifies the polarity/inversion of the LED source prior to output or blink control. 0b = Do not invert LED source. The MAC drives an active high level on the in-band message to the PHY and the PHY drives an active low level to the LED I/O pin. 1b = Invert LED source. The MAC drives an active low level on the in-band message to the PHY and the PHY drives an active high level to the LED I/O pin. Active high polarity is not recommended unless the LED is connected between two LED outputs.
LED0_BLINK	7	1b	LED0 Blink This field specifies whether to apply blink logic to the (inverted) LED control source prior to the LED output. 0b = Do not blink. 1b = Blink.
LED1_MODE	11:8	0111b	This field specifies the control source for the LED1 output. An initial value of 0011b selects the activity indication.
FLTACT	12	0b	Enable Filtered Activity LED When set to 0b the activity LED is driven completely by the PHY. When set to 1b, the PHY blinks the activity LED according to the address match event indication by the MAC. This bit can be set to 1b only for a 10/100 PHY.
LED1_ BLINK_MODE	13	0b	LED1 Blink Mode. This field needs to be configured with the same value as GLOBAL_BLINK_MODE as it specifies the blink mode of the LED. 0b = Blink at 200 ms on and 200ms off. 1b = Blink at 83 ms on and 83 ms off. For **ICH9**, see Note 3 after Table 64.
LED1_IVRT	14	0b	LED1 Invert.
LED1_BLINK	15	1b	LED1 Blink.
LED2_MODE	19:16	0110b	This field specifies the control source for the LED2 output. An initial value of 0011b selects the LINK_100 indication.
Reserved	20	00b	Reserved Read-only as 0b. Write as 0b for future compatibility.
LED2_ BLINK_MODE	21	0b	LED2 Blink Mode. This field needs to be configured with the same value as GLOBAL_BLINK_MODE as it specifies the blink mode of the LED. 0b = Blink at 200 ms on and 200 ms off. 1b = Blink at 83 ms on and 83 ms off. For **ICH9**, see Note 3 after Table 64.

Field	Bit	Initial Value	Description
LED2_IVRT	22	0b	LED2 Invert.
LED2_BLINK	23	0b	LED2 Blink.
Reserved	31:24	00h	Reserved

10.4.14.1 MODE Encodings for LED Outputs

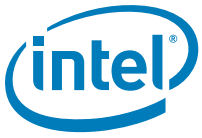
Table 64 lists the MODE encodings used to select the desired LED signal source for each LED output.

Notes:
1. When LED Blink mode is enabled, the appropriate LED Invert bit should be set to 0b.
2. The dynamic LED's modes (FILTER_ACTIVITY, LINK/ACTIVITY, COLLISION and ACTIVITY, PAUSED) should be used with LED Blink mode enabled.
3. For **ICH9**, if the ¼ CLK is enabled (CLK_CNT_1_4 bit in the STATUS register is set) and the link rate is 10 Mb/s or 100 Mb/s, the LEDs might blink at a rate four times faster then the previously mentioned setting depending on the activity factor.
4. For **ICH8**, When LED blink mode is enabled and CCM PLL is shut, the blinking frequencies are 1/5 of the rates listed in Table 64.

Note: All modes listed are functional.

Table 64. Mode Encodings for LED Outputs

Mode	Mnemonic	State / Event Indicated
0000b	LINK_10/1000	Asserted when either 10 or 1000 Mb/s link is established and maintained.
0001b	LINK_100/1000	Asserted when either 100 or 1000 Mb/s link is established and maintained.
0010b	LINK_UP	Asserted when any speed link is established and maintained.
0011b	FILTER_ACTIVITY	Asserted when link is established and packets are being transmitted or received that passed MAC filtering.
0100b	LINK/ACTIVITY	Asserted when link is established and when there is no transmit or receive activity.
0101b	LINK_10	Asserted when a 10 Mb/s link is established and maintained.
0110b	LINK_100	Asserted when a 100 Mb/s link is established and maintained.
0111b	LINK_1000	Asserted when a 1000 Mb/s link is established and maintained.
1000b	Reserved	Reserved
1001b	FULL_DUPLEX	Asserted when the link is configured for full duplex operation (deasserted in half-duplex).
1010b	COLLISION	Asserted when a collision is observed.
1011b	ACTIVITY	Asserted when link is established and packets are being transmitted or received.
1100b	BUS_SIZE	Asserted when the MAC detects a 1 Lane PCI connection.



Mode	Mnemonic	State / Event Indicated
1101b	PAUSED	Asserted when the MAC's transmitter is flow controlled.
1110b	LED_ON	Always asserted.
1111b	LED_OFF	Always de-asserted.

10.4.15 Extended Configuration Control - EXTCNF_CTRL (00F00h; R/W)

Table 65. Extended Configuration Control Bit Description

Field	Bit(s)	Initial Value	Description
PHY Write Enable	0	0b	When set, enables loading the Extended PHY Configuration area in the PHY. When disabled, the Extended PHY Configuration area is ignored. **Note:** Loaded from NVM word 14h.
MDIOWatchEna	1	1b	Enable MDIO Watchdog Timer When set to 1b, the 100 µs MDIO watchdog timer is enabled. This bit is loaded from the *MDIO Watch Dog Enable* bit in NVM word 14h. **Note:** This is a reserved bit for **ICH9/ICH10**.
Reserved	2	0b	Reserved. Must be set to 0b.
OEM Write Enable	3	0b	When set, enables auto load of the OEM bits from the PHY_CTRL register to the PHY. **Note:** Loaded from NVM word 14h.
Reserved	4	0b	Reserved
SWFLAG	5	0b	SW Semaphore Flag This bit is set by the software device driver to gain access permission to shared CSR registers with the firmware and hardware. This bit is initialized on power-up PCI reset and software reset.
MDIO HW Ownership	6	0b	Hardware request for access to MDIO. This represents part of the arbitration scheme for MDIO access.
Reserved	15:7	00h	Reserved.
Extended Configuration Pointer	27:16	0b	Defines the base address (in dwords) of the Extended Configuration area in the NVM. A value of 000h is not allowed when operating with the **82566/82567**.
Reserved	31:28	0b	Reserved.

10.4.16 Extended Configuration Size - EXTCNF_SIZE (00F08h; R/W)

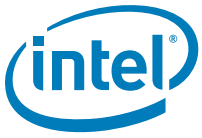
Table 66. Extended Configuration Size Bit Description

Field	Bit(s)	Initial Value	Description
Reserved	7:0	00h	Reserved
Reserved	15:8	00h	Reserved. Must be set to 00h.
Extended PHY Length	23:16	00h	Size (in dwords) of the Extended LAN Connected Device Configuration area loaded from Extended Configuration word 2 in the NVM. If an extended configuration area is disabled by the *PHY Write Enable* field in word 14h in the NVM, its length must be set to 0b.
Reserved	31:24	00h	Reserved

10.4.17 PHY Control - PHY_CTRL (00F10h; R/W)

This register is initialized to the hardware default at LAN_RST#.

Field	Bit(s)	Initial Value	Description
Reserved	31:29	00h	Reserved
	28:25	00h	**82566** SKU Read Data These four bits contain the SKU value read from the **82566** SKU register. Using these bits, the SKU mechanism determines the Device ID. **Note:** These are reserved bits for the **82567**.
Reserved	24	0b	Reserved
	23	0b	SKU Done (**ICH9/ICH10**) This bit indicates the termination of an SKU read. **Note:** This is a reserved bit for **ICH8**.
Reserved	22	0b	Reserved
DisAutoDSPD	21	0b	Disable ACBS in D3 When this bit is cleared, the MAC activates the ACBS flow autonomously when it is in the D3 state (either D3 hot or D3 cold) and the PHY reports no link indication. When the DisAutoDSPD bit is set, this mode it not enabled. **Note:** This is a reserved bit for **ICH9/ICH10** and the **82567**.
Reserved	20:17	02h	Reserved
DSPDA	16	0b	Auto Connect Battery Saver Activation When set and using the **82566**, it is either powered down by an in-band message or the external power supply is shut down using the LANPHYPC output pin (implementation specific). **Note:** This is a reserved bit for **ICH9/ICH10** and the **82567**.
Reserved	15:8	00h	Reserved
B2B Ena[1]	7	0b	Enables SPD in Back To Back link setup. This bit is initialized by word 17h bit 15 in the NVM.



Field	Bit(s)	Initial Value	Description
Global GbE Disable	6	0b	Prevents the PHY auto negotiating 1000 Mb/s link in all power states (including D0a). This bit is initialized by word 17h bit 14 in the NVM.
Reserved	5:4	00b	Reserved
GbE Disable at non D0a	3	0b	Prevents the PHY from auto negotiating 1000 Mb/s link in all power states except D0a (DR, D0u and D3). This bit is initialized by word 17h bit 11 in the NVM. This bit must also be set since GbE is not supported in Sx by the platform.
LPLU in non D0a	2	0b	Enables the PHY to negotiate for the slowest possible link (Reverse AN) in all power states except D0a (DR, D0u and D3). This bit is initialized by word 17h bit 10 in the NVM.
LPLU in D0a	1	0b	Enables the PHY to negotiate for the slowest possible link (Reverse AN) in all power states (including D0a). This bit overrides the *LPLU* bit. This bit is initialized by word 17h bit 9 in the NVM.
SPD Ena[1]	0	0b	Enables PHY Smart Power Down mode. This bit is initialized by word 17h bit 8 in the NVM.

1. Not applicable to the **82567**.

10.4.18 ICH10 PCI Analog Configuration - PCIANACFG (00F18h; RW)

Field	Bit(s)	Initial Value	Description
	0	0b	Invert Polarity Indicates to the GP unit to invert the bit polarity (receiver only) that is set from the NVM.
	6:1	20h	Command Mode Voltage Select
	31:7	00h	Reserved. Reads as 00h and ignored on writes.

10.4.19 Packet Buffer Allocation - PBA (01000h; R/W)

This register sets the on-chip receive and transmit storage allocation ratio.

Note: Programming this register does not automatically re-load or initialize internal packet-buffer RAM pointers. Software must reset both transmit and receive operation (using the global device reset *CTRL.RST* bit) after changing this register in order for it to take effect. The PBA register itself is not reset by assertion of the global reset, but is only reset upon initial hardware power-on.

Note: The receive packet buffer should be larger than the maximum expected received packet plus 32 bytes.

Note: For best performance, the transmit buffer allocation should be set to accept two full-sized packets. Transmit packet buffer size should be configured to be more than 4 KB. The recommended size should be 8 KB.

Table 67. PBA Register Bit Description

Field	Bit(s)	Initial Value	Description
RXA	4:0	0008h	Receive Packet Buffer Allocation Defines the size of the Rx buffer size in KB. The default is 8 KB. Note that the RXA can be set in the range of 2h...0Fh. For the **ICH10**, defines the size of the Rx buffer size in KB. The default is 10 KB. Note that the RXA can be set in the range of 2h...13h.
Reserved	15:5	X	Reserved
TXA	20:16	000Ch 000Eh (**ICH10**)	Transmit Packet Buffer Allocation Defines the size of the Tx buffer size in KB. This field is read only and equals to 16 minus RXA (16 KB is the total size of the Tx and Rx packet buffer). For the **ICH10**, defines the size of the Tx buffer size in KB. This field is read only and equals to the Packet Buffer Size (PBS) minus RXA (the default value of PBS is 24 KB).
Reserved	31:21	X	Reserved

10.4.20 Packet Buffer Size - PBS (01008h; R/W)

Note: The default setting of this register is 20 KB and is incorrect. This register must be programmed to 16 KB.

Field	Bit(s)	Initial Value	Description
Reserved	4:0	0014h 0018h (**ICH9/ICH10**)	Reserved
Reserved	31:5	00h	Reserved

This register sets the on-chip receive and transmit storage allocation size, The allocation value is read/write for the lower six bits. The division between transmit and receive is done according to the PBA register.

Note: Programming this register does not automatically re-load or initialize internal packet-buffer RAM pointers. Software must reset both transmit and receive operation (using the global device reset CTRL.SWRST bit) after changing this register in order for it to take effect. The PBS register itself is not reset by asserting a software reset, but is only reset at initial hardware power-on.

Note: Programming this register should be aligned with programming the PBA register hardware operation, if PBA and PBS are not coordinated is not determined.

10.4.21 Interrupt Cause Read Register - ICR (000C0H; R)

This register contains all interrupt conditions for the MAC. Each time an interrupt causing event occurs, the corresponding interrupt bit is set in this register. An interrupt is generated each time one of the bits in this register is set and the corresponding interrupt is enabled via the Interrupt Mask Set/Read Register (see Section 10.4.24).

Each time an interrupt causing event occurs, all timers of delayed interrupts are cleared and their cause event is set in the ICR register.

How the ICR register is read can differ as explained in the cases that follow:

- Case 1: Interrupt Mask Register = 0000h (mask all) - ICR content is cleared.
- Case 2: Interrupt asserted (ICR.INT_ASSERTED = 1b) - ICR content is cleared and Auto Mask is active (the IAM register is written to the IMC register).
- Case 3: Interrupt not asserted (ICR.INT_ASSERTED = 0b) - ICR content is **NOT CLEARED**.

Writing a 1b to any bit in this register also clears that bit. Writing a 0b to any bit has no effect on that bit. The INT_ASSERTED bit is a special case. Writing a 1b or 0b to this bit has no effect. It is cleared only when all interrupt sources are cleared. Bits 7, 16, and 17 are also cleared when the CPU vector is cleared.

Note: Following a Link Status Change interrupt, software might need to check the Cable Disconnected indication in FEXT register as a potential cause for the interrupt.

Field	Bit(s)	Initial Value	Description
TXDW	0	0b	Transmit Descriptor Written Back Set when hardware processes a transmit descriptor with the RS bit set (and possibly IDE set). If using delayed interrupts (IDE set), the interrupt occurs after one of the delayed-timers (TIDV or TADV) expires.
TXQE	1	0b	Transmit Queue Empty Set when the last descriptor block for a transmit queue has been used. When configured to use more than one transmit queue, this interrupt indication is issued if one of the queues is empty and is not cleared until all the queues have valid descriptors.
LSC	2	0b	Link Status Change This bit is set each time the link status changes (either from up to down, or from down to up). This bit is affected by the link indication from the PHY.
Reserved	3	0b	Reserved
RXDMT0	4	0b	Receive Descriptor Minimum Threshold Hit Indicates that the minimum number of receive descriptors are available and software should load more receive descriptors.
DSW	5	0b	Disable Software Write Access This bit indicates that the firmware changed the status of the DISSW bit in the FWSM register.
RXO	6	0b	Receiver Overrun Set on receive data FIFO overrun. Could be a result caused by no available receive buffers or because PCI receive bandwidth is inadequate.
RXT0	7	0b	Receiver Timer Interrupt Set when the receiver timer expires. In addition to the normal clearing rules of the ICR register, this bit is also cleared when the CPU Vector is cleared (either by reading or writing).
Reserved	8	0b	Reserved
MDAC	9	0b	MDI/O Access Complete This bit is set when the MDI/O access is completed.
Reserved	11:10	00b	Reserved
PHYINT	12	0b	PHY Interrupt Set with the PHY generates an interrupt.
Reserved	13	0b	Reserved

Field	Bit(s)	Initial Value	Description
LSECPN	14	0b	LinkSec Packet Number (**ICH10**) The Tx packet number hit the packet number exhaustion threshold as defined in the LSECTXCTRL register and the host is the KaY. **Note:** This is a reserved bit for **ICH8/ICH9**.
TXD_LOW	15	0b	Transmit Descriptor Low Threshold hit. Indicates that the descriptor ring has reached the threshold specified in the Transmit Descriptor Control register.
SRPD	16	0b	Small Receive Packet Detected. Indicates that a packet of size < RSRPD.SIZE register has been detected and transferred to host memory. The interrupt is only asserted if RSRPD.SIZE register has a non-zero value. In addition to the normal clearing rules of the ICR register, this bit is also cleared when CPU Vector is cleared (either by reading or writing).
ACK	17	0b	Receive ACK Frame Detected Indicates that an ACK frame has been received and the timer in RAID.ACK_DELAY has expired. In addition to the normal clearing rules of the ICR register, this bit is also cleared when CPU Vector is cleared (either by reading or writing).
MNG	18	0b	Manageability Event Detected Indicates a manageability read to the H2ME_S register. Can also be set by setting the *MNG* bit in the ICS register.
Reserved	19	0b	Reserved. Must be set to 0b.
Reserved	30:20	00h	Reserved
INT_ASSERTED	31	0b	Interrupt Asserted This bit is set when the LAN port has a pending interrupt. If the Interrupt is enabled in the PCI configuration space, an Interrupt is asserted.

10.4.22 Interrupt Throttling Rate - ITR (000C4h; R/W)

Field	Bit(s)	Initial Value	Description
Interval	15:0	00h	Minimum inter-interrupt interval. The interval is specified in 256 ns increments. Setting this bit to 0b disables interrupt throttling logic.
Reserved	31:16	00h	Reserved. Should be written with 0b to ensure future compatibility.

Software can use this register to pace (or even out) the delivery of interrupts to the host CPU. This register provides a guaranteed inter-interrupt delay between interrupts asserted by the network controller, regardless of network traffic conditions. To independently validate configuration settings, software can use the following algorithm to convert the inter-interrupt interval value to the common interrupts/sec performance metric:

$$\text{interrupts/second} = (256 \times 10^{-9} \times \text{interval})^{-1}$$

For example, if the interval is programmed to 500d, the Ethernet controller guarantees the CPU is not interrupted by the Ethernet controller for 128 μsec from the last interrupt. The maximum observable interrupt rate from the Ethernet controller must never exceed 7813 interrupts/sec.

Inversely, inter-interrupt interval value can be calculated as:

$$\text{inter-interrupt interval} = (256 \times 10^{-9} \times \text{interrupts/sec})^{-1}$$

The optimal performance setting for this register is very system and configuration specific. A initial suggested range is 651-5580 (28Bh - 15CCh).

Note: When working at 10/100 Mb/s and running at ¼ clock, the interval time is doubled by four.

10.4.23 Interrupt Cause Set Register - ICS (000C8h; W)

Software uses this register to set an interrupt condition. Any bit written with a 1b sets the corresponding interrupt. This results in the corresponding bit being set in the Interrupt Cause Read Register (see Section 10.4.21). A PCI interrupt is generated if one of the bits in this register is set and the corresponding interrupt is enabled through the Interrupt Mask Set/Read Register (see Section 10.4.24).

Bits written with 0b are unchanged.

Table 68. ICS Register Bit Description

Field	Bit(s)	Initial Value	Description
TXDW	0	X	Sets the Transmit Descriptor Written Back Interrupt.
TXQE	1	X	Sets the Transmit Queue Empty Interrupt.
LSC	2	X	Sets the Link Status Change Interrupt.
Reserved	3	X	Reserved
RXDMT0	4	X	Sets the Receive Descriptor Minimum Threshold Hit Interrupt.
DSW	5	X	Sets block software write accesses.
RXO	6	X	Sets the Receiver Overrun Interrupt. Sets on Receive Data FIFO Overrun.
RXT0	7	X	Sets the Receiver Timer Interrupt.
Reserved	8	X	Reserved
MDAC	9	X	Sets the MDI/O Access Complete Interrupt.
Reserved	11:10	X	Reserved
PHYINT	12	X	Sets the PHY interrupt.
Reserved	13	X	Reserved
LSECPN	14	X	For **ICH10**, sets the LinkSec packet number interrupt
TXD_LOW	15	X	Transmit Descriptor Low Threshold Hit.
SRPD	16	X	Clears the mask for the Small Receive Packet Detect Interrupt.
ACK	17	X	Sets the Receive ACK Frame Detect Interrupt.
MNG	18	X	Sets the Manageability Event interrupt.
Reserved	19	X	Reserved
EPRST	20	X	Sets an ME reset event.
Reserved	21	0b	Reserved
Reserved	31:22	0b	Reserved Should be written with 0b to ensure future compatibility.

10.4.24 Interrupt Mask Set/Read Register - IMS (000D0h; R/W)

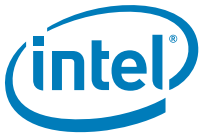
Reading this register returns the bits that have an interrupt mask set. An interrupt is enabled if its corresponding mask bit is set to 1b, and disabled if its corresponding mask bit is set to 0b. A PCI interrupt is generated each time one of the bits in this register is set and the corresponding interrupt condition occurs. The occurrence of an interrupt condition is reflected by having a bit set in the Interrupt Cause Read Register (see Section 10.4.21).

Note: If software needs to disable a particular interrupt condition that had been previously enabled, it must write to the Interrupt Mask Clear Register (see Section 10.4.25), rather than writing a 0b to a bit in this register.

When the *CTRL_EXT.INT_TIMERS_CLEAR_ENA* bit is set, then the following writing of all 1's to the IMS register (enable all interrupts) all interrupt timers are cleared to their initial value. This auto-clear provides the required latency before the next INT event.

Table 69. IMS Register Bit Description

Field	Bit(s)	Initial Value	Description
TXDW	0	0b	Sets the mask for the Transmit Descriptor Written Back.
TXQE	1	0b	Sets the mask for the Transmit Queue Empty.
LSC	2	0b	Sets the mask for the Link Status Change.
Reserved	3	0b	Reserved
RXDMT0	4	0b	Sets the mask for the Receive Descriptor Minimum Threshold hit.
DSW	5	0b	Sets the mask for block software write accesses.
RXO	6	0b	Sets the mask for Receiver Overrun. Sets on Receive Data FIFO Overrun.
RXT0	7	0b	Sets the mask for the Receiver Timer Interrupt.
Reserved	8	0b	Reserved
MDAC	9	0b	Sets the mask for the MDI/O Access Complete Interrupt.
Reserved	11:10	00b	Reserved
PHYINT	12	0b	Sets the mask for a PHY interrupt.
Reserved	13	0b	Reserved
LSECPN	14	X	For **ICH10**, sets the LinkSec packet number interrupt
TXD_LOW	15	0b	Sets the mask for the Transmit Descriptor Low Threshold hit.
SRPD	16	0b	Sets the mask for the Small Receive Packet Detection.
ACK	17	0b	Sets the mask for the Receive ACK Frame Detection.
MNG	18	0b	Sets the mask for a Manageability Event interrupt.
Reserved	19	0b	Reserved
EPRST	20	0b	Sets the mask for an ME Reset event.
Reserved	21	0b	Reserved
Reserved	31:22	00h	Reserved Should be written with 0b to ensure future compatibility.



10.4.25 Interrupt Mask Clear Register - IMC (000D8h; W)

Software uses this register to disable an interrupt. Interrupts are presented to the bus interface only when the mask bit is set to 1b and the cause bit set to 1b. The status of the mask bit is reflected in the Interrupt Mask Set/Read Register (see Section 10.4.24), and the status of the cause bit is reflected in the Interrupt Cause Read Register (see Section 10.4.21).

Software blocks interrupts by clearing the corresponding mask bit. This is accomplished by writing a 1b to the corresponding bit in this register. Bits written with 0b are unchanged (their mask status does not change).

The main purpose for this register is to enable software a way to disable certain or all interrupts. Software disables a given interrupt by writing a 1b to the corresponding bit in this register.

Table 70. IMC Register Bit Description

Field	Bit(s)	Initial Value	Description
TXDW	0	0b	Clears the mask for the Transmit Descriptor Written Back.
TXQE	1	0b	Clears the mask for the Transmit Queue Empty.
LSC	2	0b	Clears the mask for the Link Status Change.
Reserved	3	0b	Reserved
RXDMT0	4	0b	Sets mask for Receive Descriptor Minimum Threshold hit.
DSW	5	0b	Clears the mask for Receive Descriptor Minimum Threshold hit.
RXO	6	0b	Clears the mask for Receiver Overrun. Sets on Receive Data FIFO Overrun.
RXT0	7	0b	Clears the mask for the Receiver Timer Interrupt.
Reserved	8	0b	Reserved
MDAC	9	0b	Clears the mask for the MDI/O Access Complete Interrupt.
Reserved	11:10	0b	Reserved
PHYINT	12	0b	Clears a PHY Interrupt.
Reserved	13	0b	Reserved
LSECPN	14	X	For **ICH10**, sets the LinkSec packet number interrupt
TXD_LOW	15	0b	Clears the mask for the Transmit Descriptor Low Threshold hit.
SRPD	16	0b	Clears the mask for the Small Receive Packet Detect Interrupt.
ACK	17	0b	Clears the mask for the Receive ACK Frame Detect Interrupt.
MNG	18	0b	Clears the mask for the Manageability Event Interrupt.
D/UD_C	19	0b	Clears the mask for the ME Reset event.
EPRST	20	0b	Clears the mask for the ME Reset event.
Reserved	21	0b	Reserved
Reserved	31:22	00h	Reserved Should be written with 0b to ensure future compatibility.

10.4.26 Interrupt Acknowledge Auto Mask Register - IAM (000E0h; R/W)

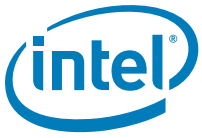
Field	Bit(s)	Initial Value	Description
IAM_VALUE	31:0	0b	Each time the *CTRL_EXT.IAME* bit is set and the *ICR.INT_ASSERT* = 1b, an ICR read or write has the side effect of writing the contents of this register to the IMC register.

10.4.27 Receive Control Register - RCTL (00100h; R/W)

This register controls all MAC receiver functions.

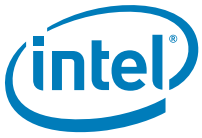
Table 71. RCTL Register Bit Description

Field	Bit(s)	Initial Value	Description
Reserved	0	0b	Reserved Write to 0b for future compatibility.
EN	1	0b	Receiver Enable The receiver is enabled when this bit is set to 1b. Writing this bit to 0b stops reception after receipt of any in-progress packet. All subsequent packets are then immediately dropped until this bit is set to 1b. Note that this bit only controls the DMA functionality to the host. Packets are counted by the statistics even when this bit is cleared.
SBP	2	0b	Store Bad Packets 0b = Do not store. 1b = Store bad packets. Note that CRC errors before the SFD are ignored. Any packet must have a valid SFD in order to be recognized by the MAC (even bad packets). **Note:** Erred packets are not routed to manageability even if this bit is set.
UPE	3	0b	Unicast Promiscuous Enabled 0b = Disabled. 1b = Enabled.
MPE	4	0b	Multicast Promiscuous Enabled 0b = Disabled. 1b = Enabled.
LPE	5	0b	Long Packet Reception Enable 0b = Disabled. 1b = Enabled. LPE controls whether long packet reception is permitted. Hardware discards long packets if LPE is 0b. A long packet is one longer than 1522 bytes. If LPE is 1b, the maximum packet size that the MAC can receive is 16384 bytes.
Reserved	7:6	00b	Reserved



Field	Bit(s)	Initial Value	Description
RDMTS	9:8	00b	Receive Descriptor Minimum Threshold Size The corresponding interrupt ICR.RXDMT0 is set each time the fractional number of free descriptors becomes equal to RDMTS. The following table lists which fractional values correspond to RDMTS values. The size of the total receiver circular descriptor buffer is set by RDLEN. See Section 10.4.35 for details regarding RDLEN. 00b = Free buffer threshold is set to 1/2 of RDLEN. 01b = Free buffer threshold is set to 1/4 of RDLEN. 10b = Free buffer threshold is set to 1/8 of RDLEN. 11b = Reserved.
DTYP	11:10	0b	Descriptor Type 00b = Legacy description type. 01b = Packet split description type. 10b = Reserved. 11b = Reserved.
MO	13:12	0b	Multicast Offset The MAC is capable of filtering multicast packets based on 4096-bit vector multicast filtering table. The MO determines which bits of the incoming multicast address are used in looking up the 4096-bit vector. 00b = bits [47:36] of received destination multicast address. 01b = bits [46:35] of received destination multicast address. 10b = bits [45:34] of received destination multicast address. 11b = bits [43:32] of received destination multicast address.
Reserved	14	0b	Reserved
BAM	15	0b	Broadcast Accept Mode. 0b = Ignore broadcast. 1b = Accept broadcast packets. When set, passes and does not filter out all received broadcast packets. Otherwise, the MAC accepts, or rejects a broadcast packet only if it matches through perfect or imperfect filters.
BSIZE	17:16	0b	Receive Buffer Size Controls the size of the receive buffers, allowing the software to trade off between system performance and storage space. Small buffers maximize memory efficiency at the cost of multiple descriptors for bigger packets. RCTL.BSEX = 0b: 00b = 2048 Bytes. 01b = 1024 Bytes. 10b = 512 Bytes. 11b = 256 Bytes. RCTL.BSEX = 1b: 00b = Reserved; software should not program this value. 01b = 16384 Bytes. 10b = 8192 Bytes. 11b = 4096 Bytes. **Note:** BSIZE is only used when DTYP = 00b. When DTYP = 01b, the buffer sizes for the descriptor are controlled by fields in the Packet Split Receive Control (PSRCTL) register. BSIZE is not relevant when FLXBUF is something other than 0b. in that case, FLXBUF determines the buffer size.
Reserved	21:18	0b	Reserved Should be written with 0b.
Reserved	22	0b	Reserved

Field	Bit(s)	Initial Value	Description
PMCF	23	0b	Pass MAC Control Frames 0b = Do not (specially) pass MAC control frames. 1b = Pass any MAC control frame (type field value of 8808h) that does not contain the pause opcode of 0001h. PMCF controls the DMA function of MAC control frames (other than flow control). A MAC control frame in this context must be addressed to either the MAC control frame multicast address or the station address, match the type field and NOT match the PAUSE opcode of 0001h. If PMCF = 1b then frames meeting this criteria are transferred to host memory. Otherwise, they are filtered out.
Reserved	24	0b	Reserved Should be written with 0b to ensure future compatibility. Reads as 0b.
BSEX	25	0b	Buffer Size Extension When set to 1b, the original BSIZE values are multiplied by 16. Refer to the RCTL.BSIZE bit description.
SECRC	26	0b	Strip Ethernet CRC from incoming packet 0b = Do not strip CRC field. 1b = Strip CRC field. Controls whether the hardware strips the Ethernet CRC from the received packet. This stripping occurs prior to any checksum calculations. The stripped CRC is not transferred to host memory and is not included in the length reported in the descriptor.
FLXBUF	30:27	0b	Determines a flexible buffer size. When this field = 0000b, the buffer size is determined by BSIZE. If this field is something other than 0000b, the receive buffer size is the number represented in KB: For example, 0001b = 1 KB (1024 Bytes).
Reserved	31	0b	Reserved Should be written with 0b to ensure future compatibility.



10.4.28 ICH10 Receive Control Register 1 - RCTL1 (0x00104; RW)

This register is used to configure queue1 registers when operating in VMDq mode.

Field	Bit(s)	Initial Value	Description
Reserved	7:0	00h	Reserved. Write as 00h for future compatibility.
RDMTS	9:8	00h	Receive Descriptor Minimum Threshold Size The corresponding interrupt is set each time the fractional number of free descriptors becomes equal to RDMTS (bits 9:8 of the RCTL register).
DTYP	11:10	00b	Descriptor Type 00b = Legacy or Extended descriptor type. 01b = Packet Split descriptor type. 10b and 11b = Reserved. The value of RCTL1.DTYP should be the same as RCTL.DTYP (same descriptor types used in both descriptor queues).
Reserved	15:12	00h	Reserved
BSIZE	17:16	00b	Receive Buffer Size RCTL.BSEX = 0b: 00b = 2048 bytes. 01b = 1024 bytes. 10b = 512 bytes. 11b = 256 bytes. RCTL.BSEX = 1b: 00b – Reserved. 01b = 16384 bytes. 10b = 8192 bytes. 11b = 4096 bytes. BSIZE is only used when DTYP = 00b. When DTYP = 01b, the buffer sizes for the descriptor are controlled by fields in the PSRCTL register. BSIZE is not relevant when the FLXBUF is other than zero, in that case, FLXBUF determines the buffer size.
Reserved	24:18	00h	Reserved. Should be written with 00h.
BSEX	25	0b	Buffer Size Extension Modifies the buffer size indication (BSIZE). 0b = Buffer size, as defined in BSIZE. 1b = Original BSIZE values are multiplied by 16.
Reserved	26	0b	Reserved. Should be written with 0b.
FLXBUF	30:27	00h	Flexible Buffer Determines a flexible buffer size. When this field = 0000b, the buffer size is determined by BSIZE. If this field is different from 0000b, the receive buffer size is the number represented in KB: For example: 0001b = 1KB (1024 bytes).
Reserved	31	0b	Reserved. Should be written with 0b to ensure future compatibility.

10.4.29 Early Receive Threshold - ERT (02008h; R/W)

This register contains the RxThreshold value. This threshold determines how many bytes of a given packet should be in the MAC's on-chip receive packet buffer before it attempts to begin transmission of the frame on the PCI bus. This register enables software to configure the MAC to use early receives.

This field has a granularity of 8 bytes. So, writing this field to 20h corresponds to a threshold of 256 (decimal) bytes. If the size of a given packet is smaller than the threshold value, or if this register is set to 0b, then MAC starts the PCI transfer only after the entire packet is contained in MAC's receive packet buffer. The MAC examines this register on a cycle by cycle basis to determine if there is enough data to start a transfer for the given frame over the PCI bus.

Once the Ethernet controller acquires the bus, it attempts to transfer all of the current data collected in the internal receive packet buffer.

The only negative affect of setting this value too low is that it causes additional PCI bursts for the packet. In other words, this register enables software to trade-off latency versus bus utilization. Too high a value effectively eliminates the early receive benefits (at least for short packets) and too low a value deteriorates PCI bus performance due to a large number of small bursts for each packet. The RUTEC statistic counts certain cases where the ERT has been set too low, and thus provides software a feedback mechanism to better tune the value of the ERT.

It should also be noted that this register has an effect only when the receive packet buffer is nearly empty (the only data in the packet buffer is from the packet that is currently on the wire).

Note: When Early Receive is used in parallel to the Packet Split Receive feature, the minimum value of the ERT register should be larger than the header size to enable the actual packet split.

Table 72. ERT Register Bit Description

Field	Bit(s)	Initial Value	Description
RxThreshold	12:0	00h	Receive Threshold Value This threshold is in units of eight bytes.
Reserved	31:13	0b	Reads as 0b. Should be written to 0b for future compatibility.

10.4.30 Flow Control Receive Threshold Low - FCRTL (02160h; R/W)

This register contains the receive threshold used to determine when to send an XON packet. It counts in units of bytes. Each time the receive FIFO crosses the receive high threshold FCRTH.RTH (filling up), and then crosses the receive low threshold FCRTL.RTL, with FCRTL.XONE enabled (1b), hardware transmits an XON frame.

Flow control reception/transmission are negotiated capabilities by the Auto-Negotiation process. When the MAC is manually configured, flow control operation is determined by the CTRL.RFCE and CTRL.TFCE bits.

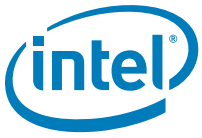


Table 73. FCRTL Register Bit Description

Field	Bit(s)	Initial Value	Description
Reserved	2:0	0b	Reserved Should be written with 0b to ensure future compatibility.
RTL	15:3	0b	Receive Threshold Low. FIFO low water mark for flow control transmission.
Reserved	30:16	0b	Reserved Should be written with 0b for future compatibility. Reads as 0b.
XONE	31	0b	XON Enable 0b = Disabled. 1b = Enabled.

10.4.31 Flow Control Receive Threshold High - FCRTH (02168h; R/W)

This register contains the receive threshold used to determine when to send an XOFF packet. It counts in units of bytes. This value must be at least 8 bytes less than the maximum number of bytes allocated to the Receive Packet Buffer (PBA, RXA), and the lower 3 bits must be programmed to 0b (8 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.

Flow control reception/transmission are negotiated capabilities by the Auto-Negotiation process. When the MAC is manually configured, flow control operation is determined by the *CTRL.RFCE* and *CTRL.TFCE* bits.

Table 74. FCRTH Register Bit Description

Field	Bit(s)	Initial Value	Description
Reserved	2:0	0b	Reserved Must be written with 0b.
RTH	15:3	0b	Receive Threshold High. FIFO high water mark for flow control transmission. Each time the receive FIFO reaches the fullness indicated by RTH, the MAC transmits a Pause packet if enabled to do so.
Reserved	31:16	0b	Reserved Should be written with 0b for future compatibility. Reads as 0b.

10.4.32 Packet Split Receive Control Register - PSRCTL (02170h; R/W) (02170h + n*4h [n=0..1] for ICH10)

This register determines the receive buffer size.

Note: If software sets a buffer size to zero, all buffers following that one must be set to zero as well. Pointers in the receive descriptors to buffers with a zero size should be set to anything but NULL pointers.

Table 75. PSRCTL Register Bit Description

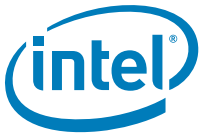
Field	Bit(s)	Initial Value	Description
BSIZE0	6:0	256 bytes	Receive Buffer size for Buffer 0 The value is in 128-byte resolution. The value can be from 128 bytes to 16256 bytes (15.875 KB). The default buffer size is 256 bytes. Software should not program this field to a zero value.
Reserved	7	0b	Reserved. Should be written with 0b to ensure future compatibility.
BSIZE1	13:8	4 KB	Receive Buffer Size for Buffer 1 The value is in 1 KB resolution. The value can be from 1 KB to 63 KB. The default buffer size is 4 KB. Software should not program this field to a zero value.
Reserved	15:14	00b	Reserved. Should be written with 0b to ensure future compatibility.
BSIZE2	21:16	4 KB	Receive Buffer Size for Buffer 2 The value is in 1 KB resolution. The value can be from 1 KB to 63 KB. The default buffer size is 4 KB. Software should not program this field to a zero value.
Reserved	23:22	00b	Reserved. Should be written with 0b to ensure future compatibility.
BSIZE3	29:24	0 KB	Receive Buffer Size for Buffer 3 The value is in 1 KB resolution. The value can be from 1 KB to 63 KB. The default buffer size is 0 KB. Software should not program this field to a zero value.
Reserved	31:30	00b	Reserved. Should be written with 0b to ensure future compatibility.

10.4.33 Receive Descriptor Base Address Low Queue 0 - RDBAL0 (02800h;R/W); (02800h + n*100h [n=0..1] for ICH10)

This register contains the lower bits of the 64-bit descriptor base address. The four low-order register bits are always ignored. The Receive Descriptor Base Address must point to a 16-byte aligned block of data.

Table 76. RDBAL0 Register Bit Description

Field	Bit(s)	Initial Value	Description
0	3:0	0b	Ignored on writes. Returns 0b on reads.
RDBAL0	31:4	X	Receive Descriptor Base Address Low Queue 0.



10.4.34 Receive Descriptor Base Address High Queue 0 - RDBAH0 (02804h; R/W); (02804h + n*100h [n=0..1] for ICH10)

This register contains the upper 32 bits of the 64-bit Descriptor Base Address.

Table 77. RDBAH0 Register Bit Description

Field	Bit(s)	Initial Value	Description
RDBAH0	31:0	X	Receive Descriptor Base Address Queue 0[63:32]

10.4.35 Receive Descriptor Length Queue 0 - RDLEN0 (02808h; R/W); (02808h + n*100h [n=0..1] for ICH10)

This register determines the number of bytes allocated to the circular receive descriptor buffer. This value must be 128-byte aligned (the maximum cache line size). Since each descriptor is 16 bytes in length, the total number of receive descriptors is always a multiple of eight.

Table 78. RDLEN0 Register Bit Description

Field	Bit(s)	Initial Value	Description
Zero	6:0	0b	Zero value Ignore on write. Reads back as 0b.
LEN0	19:7	0b	Receive Descriptor Length Queue 0 Provides the number of receive descriptors (in multiples of eight).
Reserved	31:20	0b	Reserved Should be written with 0b to ensure future compatibility. Reads as 0b.

10.4.36 Receive Descriptor Head Queue 0 - RDH0 (02810h; R/W); (02810h + n*100h [n=0..1] for ICH10)

This register contains the head pointer for the receive descriptor buffer. The register points to a 16-byte datum. Hardware controls the pointer. The only time that software should write to this register is after a reset (CTRL.RST) and before enabling the receiver function (RCTL.EN). If software were to write to this register while the receive function was enabled, the on-chip descriptor buffers can be invalidated and other indeterminate operations might result. Reading the descriptor head to determine which buffers are finished is not reliable.

Table 79. RDH0 Register Bit Description

Field	Bit(s)	Initial Value	Description
RDH0	15:0	0b	Receive Descriptor Head Queue 0.
Reserved	31:16	0b	Reserved. Should be written with 0b.

10.4.37 Receive Descriptor Tail Queue 0 - RDT0 (02818h; R/W); (02818h + n*100h [n=0..1] for ICH10)

This register contains the tail pointers 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. Software should also write an even number to the tail register if the MAC uses the Packet Split feature.

Table 80. RDT0 Register Bit Description

Field	Bit(s)	Initial Value	Description
RDT0	15:0	0b	Receive Descriptor Tail Queue 0.
Reserved	31:16	0b	Reserved Reads as 0b. Should be written with 0b for future compatibility.

10.4.38 Receive Interrupt Delay Timer (Packet Timer) Register - RDTR (02820h; R/W); (02820h + n*100h [n=0..1] for ICH10)

This register is used to delay interrupt notification for the receive descriptor ring. Delaying interrupt notification helps maximize the number of receive packets serviced by a single interrupt.

Table 81. RDTR Register Bit Description

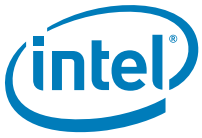
Field	Bit(s)	Initial Value	Description
Delay	15:0	0b	Receive delay timer measured in increments of 1.024 µs.
Reserved	30:16	0b	Reserved. Reads as 0b.
FPD	31	0b	Flush Partial Descriptor Block when set to 1b; ignore otherwise. Reads 0b (self-clearing).

This feature operates by initiating a countdown timer upon successfully receiving each packet to system memory. If a subsequent packet is received BEFORE the timer expires, the timer is re-initialized to the programmed value and re-starts its countdown. If the timer expires due to NOT having received a subsequent packet within the programmed interval, pending receive descriptor writebacks are flushed and a receive timer interrupt is generated.

Setting the value to 0b represents no delay from a receive packet to the interrupt notification, and results in immediate interrupt notification for each received packet.

Writing this register with FPD set initiates an immediate expiration of the timer, causing a writeback of any consumed receive descriptors pending writeback, and results in a receive timer interrupt in the ICR.

Receive interrupts due to a Receive Absolute Timer (RADV) expiration cancels a pending RDTR interrupt. The RDTR countdown timer is reloaded but halted, so as to avoid generation of a spurious second interrupt after the RADV has been noted, but might be restarted by a subsequent received packet.



10.4.39 Receive Descriptor Control - RXDCTL (02828h; R/W); (02828h + n*100h [n=0..1] for ICH10)

This register controls the fetching and write-back of receive descriptors. The three threshold values are used to determine when descriptors are read from and written to host memory. The values can be in units of cache lines or descriptors (each descriptor is 16 bytes) based on the GRAN flag. If GRAN = 0b (specifications are in cache-line granularity), the thresholds specified (based on the cacheline size specified in the PCI header CLS field) must not represent greater than 31 descriptors.

Note: When GRAN = 1b, all descriptors are written back (even if not requested).

Note: For **ICH10**, RXDCTL1 is only accessible when VMDq is enabled (MRQC.MRxQueue = 10).

Table 82. RXDCTL Register Bit Description

Field	Bit(s)	Initial Value	Description
PTHRESH	5:0	00h	Prefetch Threshold Used to control when a prefetch of descriptors is considered. This threshold refers to the number of valid, unprocessed receive descriptors the MAC has in its on-chip buffer. If this number drops below PTHRESH, the algorithm considers prefetching descriptors from host memory. This fetch does not happen unless there are at least RXDCTL.HTHRESH valid descriptors in host memory to fetch. Value of PTHRESH can be in either cache line units, or based on number of descriptors based on RXDCTL.GRAN.
RSV	7:6	00b	Reserved
HTHRESH	13:8	00h	Host Threshold Provides the threshold of the valid descriptors in host memory. A descriptors prefetch is performed each time enough valid descriptors (TXDCTL.HTHRESH) are available in host memory, no other DMA activity of greater priority is pending (descriptor fetches and write backs or packet data transfers) and the number of receive descriptors the MAC has on its on-chip buffers drops below RXDCTL.PTHRESH. Value of HTHRESH can be in either cache line units, or based on number of descriptors based on RXDCTL.GRAN.
Reserved	15:14	00b	Reserved
WTHRESH	21:16	01h	Write Back Threshold WTHRESH controls the write back of processed receive descriptors. This threshold refers to the number of receive descriptors in the MAC's on-chip buffer which are ready to be written back to host memory. In the absence of external events (explicit flushes), the write back occurs only after more than WTHRESH descriptors are available for write back. WTHRESH must contain a non-zero value to take advantage of the write back bursting capabilities of the MAC. A value of 1b causes the descriptors to be written back as soon as one cache line is available. A value of WTHRESH can be in either cache line units, or based on number of descriptors based on RXDCTL.GRAN.

Field	Bit(s)	Initial Value	Description
Reserved	23:22	00b	Reserved
GRAN	24	1b	Granularity Set the values of PTHRESH, HTHRESH and WTHRESH in units of cache lines or descriptors (each descriptor is 16 bytes) 0b = Cache line granularity. 1b = Descriptor granularity.
Reserved	31:25	00h	Reserved

10.4.40 ICH10 Receive Descriptor Control 1 - RXDCTL1 (xxxxxh + n*100h [n=0..1]; R/W)

TBD

10.4.41 Receive Interrupt Absolute Delay Timer - RADV (0282Ch; RW)

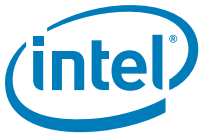
Field	Bit(s)	Initial Value	Description
Delay	15:0	0b	Receive Absolute Delay Timer Measured in increments of 1.024 µs (0b = disabled)
Reserved	31:16	0b	Reserved Reads as 0b.

If the packet delay timer is used to coalesce receive interrupts, the MAC ensures that when receive traffic abates, an interrupt is generated within a specified interval of no receives. During times when receive traffic is continuous, it may be necessary to ensure that no receive remains unnoticed for too long an interval. This register can be used to ENSURE that a receive interrupt occurs at some predefined interval after the first packet is received.

When this timer is enabled, a separate absolute countdown timer is initiated upon successfully receiving each packet to system memory. When this absolute timer expires, pending receive descriptor writebacks are flushed and a receive timer interrupt is generated.

Setting this register to 0b disables the absolute timer mechanism (the RDTR register should be used with a value of 0b to cause immediate interrupts for all receive packets).

Receive interrupts due to a Receive Packet Timer (RDTR) expiration cancels a pending RADV interrupt. If enabled, the RADV countdown timer is reloaded but halted, so as to avoid generation of a spurious second interrupt after the RDTR has been noted.



10.4.42 Receive Descriptor Base Address Low Queue 1 - RDBAL1 (02900h; R/W)

This register contains the lower bits of the 64-bit descriptor base address. The four low-order register bits are always ignored. The Receive Descriptor Base Address must point to a 16-byte aligned block of data.

Table 83. RDBAL1 Register Bit Description

Field	Bit(s)	Initial Value	Description
0	3:0	0b	Ignored on writes. Returns 0b on reads.
RDBAL1	31:4	X	Receive Descriptor Base Address Low Queue 1.

10.4.43 Receive Descriptor Base Address High Queue 1 - RDBAH1 (02904h; R/W)

This register contains the upper 32 bits of the 64-bit Descriptor Base Address.

Table 84. RDBAH1 Register Bit Description

Field	Bit(s)	Initial Value	Description
RDBAH1	31:0	X	Receive Descriptor Base Address Queue 1 [63:32]

10.4.44 Receive Descriptor Length Queue 1 - RDLEN1 (02908h; R/W)

This register determines the number of bytes allocated to the circular receive descriptor buffer. This value must be 128-byte aligned (the maximum cache line size). Since each descriptor is 16 bytes in length, the total number of receive descriptors is always a multiple of eight.

Table 85. RDLEN1 Register Bit Description

Field	Bit(s)	Initial Value	Description
Zero	6:0	0b	Zero value Ignore on write. Reads back as 0b.
LEN1	19:7	0b	Receive Descriptor Length Queue 1 Provides the number of receive descriptors (in multiples of eight).
Reserved	31:20	0b	Reserved Should be written with 0b to ensure future compatibility. Reads as 0b.

10.4.45 Receive Descriptor Head Queue 1 - RDH1 (02910h; R/W)

This register contains the head pointer for the receive descriptor buffer. The register points to a 16-byte datum. Hardware controls the pointer. The only time that software should write to this register is after a reset (CTRL.RST) and before enabling the receiver function (RCTL.EN). If software were to write to this register while the receive function was enabled, the on-chip descriptor buffers can be invalidated and other indeterminate operations might result. Reading the descriptor head to determine which buffers are finished is not reliable.

Table 86. RDH1 Register Bit Description

Field	Bit(s)	Initial Value	Description
RDH1	15:0	0b	Receive Descriptor Head Queue 1.
Reserved	31:16	0b	Reserved. Should be written with 0b.

10.4.46 Receive Descriptor Tail Queue 1 - RDT1 (02918h; R/W)

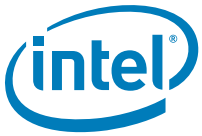
This register contains the tail pointers for the receive descriptor buffer. The register points to a 16-byte datum. Software writes the tail register to add receive descriptors to the hardware free list for the ring. Software should also write an even number to the tail register if the MAC uses the Packet Split feature.

Table 87. RDT1 Register Bit Description

Field	Bit(s)	Initial Value	Description
RDT1	15:0	0b	Receive Descriptor Tail Queue 1.
Reserved	31:16	0b	Reserved Reads as 0b. Should be written with 0b for future compatibility.

10.4.47 Receive Small Packet Detect Interrupt - RSRPD (02C00h; R/W)

Field	Bit(s)	Initial Value	Description
SIZE	11:0	0b	If the interrupt is enabled, any receive packet of size ≤ SIZE asserts an Interrupt. SIZE is specified in bytes and includes the headers and the CRC. It does not include the VLAN header in size calculation if it is stripped.
Reserved	31:12	X	Reserved



10.4.48 Receive ACK Interrupt Delay Register - RAID (02C08h; R/W)

Field	Bit(s)	Initial Value	Description
ACK_DELAY	15:0	0b	The ACK delay timer measured in increments of 1.024 µs. When the Receive ACK Frame Detect interrupt is enabled in the IMS register, ACK packets being received use a unique delay timer to generate an interrupt. When an ACK is received, an absolute timer loads to the value of ACK_DELAY. The interrupt signal is set only when the timer expires. If another ACK packet is received while the timer is counting down, the timer is not reloaded to ACK_DELAY.
Reserved	31:16	0b	Reserved

Note: If an immediate (non-scheduled) interrupt is desired for any received ACK frame, ACK_DELAY should be set to 0b.

10.4.49 CPU Vector Register - CPUVEC (02C10h; R/W)

Field	Bit(s)	Initial Value	Description
CPU Vector	31:0	0h	Each bit is associated with one of 32 possible host processors/ queues. It denotes that the processor has work to do. A bit is set in the CPU vector if the respective mask bit is set in the RSS Interrupt Mask register and the respective *RSS Interrupt Request* bit is set. Reading the CPU Vector clears the CPU Vector, any bit in the RSS Interrupt Mask register for whom the corresponding bit in the CPU Vector was set, and the RXT0, ACK & SPRD bits in the ICR. Writing to a bit in the CPU Vector clears the corresponding bit in the RSS Interrupt Mask register (RSSIM).

10.4.50 Receive Checksum Control - RXCSUM (05000h; R/W)

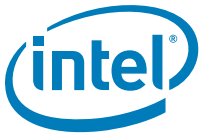
The Receive Checksum Control register controls the receive checksum offloading features of the MAC. The MAC supports the offloading of three receive checksum calculations: the Packet Checksum, the IP Header Checksum, and the TCP/UDP Checksum.

The frame types that are supported:

- Ethernet II
- Ethernet SNAP

Table 88. RXCSUM Register Bit Description

Field	Bit(s)	Initial Value	Description
PCSS	7:0	0b	Packet Checksum Start Controls the starting byte for the Packet Checksum calculation. The Packet Checksum is the one's complement over the receive packet, starting from the byte indicated by RXCSUM.PCSS (0b corresponds to the first byte of the packet), after stripping. For example, for an Ethernet II frame encapsulated as an 802.3ac VLAN packet and with RXCSUM.PCSS set to 14, the packet checksum would include the entire encapsulated frame, excluding the 14-byte Ethernet header (DA,SA,Type/Length) and the 4-byte VLAN tag. The Packet Checksum does not include the Ethernet CRC if the RCTL.SECRC bit is set. Software must make the required offsetting computation (to back out the bytes that should not have been included and to include the pseudo-header) prior to comparing the Packet Checksum against the TCP checksum stored in the packet.
IPOFLD	8	0b	IP Checksum Off-load Enable RXCSUM.IPOFLD is used to enable the IP Checksum offloading feature. If RXCSUM.IPOFLD is set to 1b, the MAC calculates the IP checksum and indicates a pass/fail indication to software through the Checksum Error bit (CSE) in the ERROR field to the receive descriptor. If both RXCSUM.IPOFLD and RXCSUM.TUOFLD are set, the Checksum Error bit (CSE) is set if either checksum was incorrect. If neither RSCSUM.IPOFLD nor RXCSUM.TUOFLD is set, the Checksum Error bit (CSE) is be 0b for all packets.
TUOFLD	9	0b	TCP/UDP Checksum Off-load Enable RXCSUM.TUOFL is used to enable the TCP/UDP Checksum off-loading feature. When set to 1b, the MAC calculates the TCP or UDP checksum and indicate a pass/fail indication to software through the Checksum Error bit (CSE). If both RXCSUM.TUOFLD and RXCSUM.TUOFLD are set, the Checksum Error bit (CSE) is set if either checksum was incorrect. If neither RXCSUM.IPOFLD nor RXCSUM.TUOFLD is set, the Checksum Error bit (CSE) is 0b for all packets.
Reserved	11:10	00b	Reserved



Field	Bit(s)	Initial Value	Description
IPPCSE	12	0b	IP Payload Checksum Enable PCSS and the IPPCSE control the packet checksum calculation. As previously stated, the packet checksum shares the same location as the RSS field. The packet 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 that is reported in the Rx packet checksum field is the unadjusted 16-bit ones complement of the packet. The packet checksum starts from the byte indicated by RXCSUM.PCSS (0b corresponds to the first byte of the packet), after VLAN stripping if enabled (by CTRL.VME). For example, for an Ethernet II frame encapsulated as an 802.3ac VLAN packet and with RXCSUM.PCSS set to 14, the packet checksum would include the entire encapsulated frame, excluding the 14-byte Ethernet header (DA, SA, Type/Length) and the 4-byte VLAN tag. The Packet Checksum does not include the Ethernet CRC if the RCTL.SECRC bit is set. Software must make the required offsetting computation (to back out the bytes that should not have been included and to include the pseudo-header) prior to comparing the packet checksum against the TCP checksum stored in the packet. If the RXCSUM.IPPCSE is set, the packet checksum is aimed to accelerate checksum calculation of fragmented UDP packets. **Note:** the PCSS value should not exceed a pointer to the IP header start or else it erroneously calculates the IP header checksum or TCP/UDP checksum.
PCSD	13	0b	Packet Checksum Disable The packet checksum and IP Identification fields are mutually exclusive with the RSS hash. Only one of the two options is reported in the Rx descriptor. Legacy Rx Descriptor (RCTL.DTYP = 00b): 0b = Checksum enable - Packet checksum is reported in the Rx descriptor. 1b = Checksum disable - Not supported. Extended or Header Split Rx Descriptor (RCTL.DTYP = 00b): 0b = Checksum enable - Packet checksum and IP identification is reported in the Rx descriptor. 1b = Checksum disable - RSS hash value is reported in the Rx descriptor.
Reserved	31:14	0b	Reserved

10.4.51 Receive Filter Control Register - RFCTL (05008h; R/W)

Field	Bit(s)	Initial Value	Description
ISCSI_DIS	0	0b	iSCSI Disable Disables the iSCSI filtering for header split functionality.
ISCSI_DWC	5:1	0b	iSCSI Dword Count This field indicates the Dword count of the iSCSI header that is used for the packet split mechanism.
NFSW_DIS	6	0b	NFS Write Disable Disables filtering of NFS write request headers for header split functionality.
NFSR_DIS	7	0b	NFS Read Disable Disables filtering of NFS read reply headers for header split functionality.
NFS_VER	9:8	00b	NFS Version 00b = NFS version 2. 01b = NFS version 3. 10b = NFS version 4. 11b = Reserved for future use.
Reserved	11:10	00b	Reserved
ACKDIS	12	0b	ACK Accelerate Disable When set, the MAC does not accelerate interrupt on TCP ACK packets.
ACKD_DIS	13	0b	ACK Data Disable 0b = MAC recognizes ACK packets according to the ACK bit in the TCP header plus no CP data. 1b = MAC recognizes ACK packets according to the ACK bit only. This bit is relevant only if the ACKDIS bit is not set.
IPFRSP_DIS	14	0b	IP Fragment Split Disable When this bit is set the header of IP fragmented packets are not set.
EXSTEN	15	0b	Extended Status Enable When the EXSTEN bit is set or when the packet split receive descriptor is used, the MAC writes the extended status to the Rx descriptor.
Reserved	31:16	0b	Reserved Should be written with 0b to ensure future compatibility.

10.4.52 Transmit Control Register - TCTL (00400h; R/W)

This register controls all transmit functions for the Ethernet controller.

10.4.52.0.1 TCTL Register Bit Description

Field	Bit(s)	Initial Value	Description
Reserved	0	0b	Reserved Write as 0b for future compatibility.
EN	1	0b	Transmit Enable The transmitter is enabled when this bit is set to 1b. Writing 0b to this bit stops transmission after any in progress packets are sent. Data remains in the transmit FIFO until the MAC is re-enabled. Software should combine this operation with reset if the packets in the FIFO should be flushed.
Reserved	2	0b	Reserved Reads as 0b. Should be written to 0b for future compatibility.
PSP	3	1b	Pad Short Packets 0b = Do not pad. 1b = Pad short packets. Padding makes the packet 64 bytes long. This is not the same as the minimum collision distance. If padding of short packet is enabled, the value in TX descriptor length field should be not less than 17 bytes.
CT	11:4	0Fh	Collision Threshold This determines the number of attempts at retransmission prior to giving up on the packet (not including the first transmission attempt). While this can be varied, it should be set to a value of 15 in order to comply with the IEEE specification requiring a total of 16 attempts. The Ethernet back-off algorithm is implemented and clamps to the maximum number of slot-times after 10 retries. This field only has meaning while in half-duplex operation.
COLD	21:12	3Fh	Collision Distance Specifies the minimum number of byte times that must elapse for proper CSMA/CD operation. Packets are padded with special symbols, not valid data bytes. Hardware checks and pads to this value plus one byte even in full-duplex operation. The default value is 64 bytes = 512-bit time.
SWXOFF	22	0b	Software XOFF Transmission When set to a 1b, the MAC schedules the transmission of an XOFF (PAUSE) frame using the current value of the PAUSE timer. This bit self clears upon transmission of the XOFF frame.
Reserved	23	0b	Reserved
RTLC	24	0b	Re-transmit on Late Collision When set, enables the MAC to re-transmit on a late collision event.
Reserved	27:25	00h	Reserved

Field	Bit(s)	Initial Value	Description
Reserved	28	1b	Reserved
RRTHRESH	30:29	01b	Read Request Threshold These bits define the threshold size for the intermediate buffer to determine when to send the Read command to the packet buffer. Threshold is defined as follows: RRTHRESH = 00b - Threshold = 2 lines of 16 bytes. RRTHRESH = 01b - Threshold = 4 lines of 16 bytes. RRTHRESH = 10b - Threshold = 8 lines of 16 bytes. RRTHRESH = 11b - Threshold = No threshold (transfer data when after all the request is in the RFIFO).
Reserved	31	0b	Reserved Reads as 0b. Write to 0b for future compatibility.

Two fields deserve special mention: CT and COLD. Software can choose to abort packet transmission in less than the Ethernet mandated 16 collisions. For this reason, hardware provides CT.

Wire speeds of 1000 Mb/s result in a very short collision radius with traditional minimum packet sizes. COLD specifies the minimum number of bytes in the packet to satisfy the desired collision distance. It is important to note that the resulting packet has special characters appended to the end. These are NOT regular data characters. Hardware strips special characters for packets that go from 1000 Mb/s environments to 100 Mb/s environments. Note that the hardware evaluates this field against the packet size in full duplex as well.

Note: While 802.3x flow control is only defined during full duplex operation, the sending of PAUSE frames via the SWXOFF bit is not gated by the duplex settings within the MAC. Software should not write a 1b to this bit while the MAC is configured for half-duplex operation.

RTLC configures the MAC to perform retransmission of packets when a late collision is detected. Note that the collision window is speed dependent: 64 bytes for 10/100 Mb/s and 512 bytes for 1000 Mb/s operation. If a late collision is detected when this bit is disabled, the transmit function assumes the packet has successfully transmitted. This bit is ignored in full-duplex mode.

10.4.53 Transmit IPG Register - TIPG (00410;R/W)

This register controls the IPG (Inter Packet Gap) timer. IPGT specifies the IPG length for back-to-back transmissions in both full and half duplex. Note that an offset of 4 byte times is added to the programmed value to determine the total IPG. As a result, a value of 8 is recommended to achieve a 12 byte time IPG.

IPGR1 specifies the portion of the IPG in which the transmitter defers to receive events. This should be set to 2/3 of the total effective IPG, or 8.

IPGR specifies the total IPG time for non back-to-back transmissions (transmission following deferral) in half duplex.

An offset of 5 byte times is added to the programmed value to determine the total IPG after a defer event. Therefore, a value of 7 is recommended to achieve a 12 byte effective IPG. Note the IPGR should never be set to a value greater than IPGT. If IPGR is set to a value equal to or larger than IPGT, it overrides the IPGT IPG setting in half duplex, resulting in inter packet gaps that are larger than intended by IPGT. Full duplex is unaffected by this and only relies on IPGT.

The recommended TIPG value to achieve 802.3 compliant minimum transmit IPG values in full and half duplex is 00702008h.

Table 89. TIPG Register Bit Description

Field	Bit(s)	Initial Value	Description
IPGT	9:0	08h	IPG Back to Back Specifies the IPG length for back-to-back transmissions equal to (IPGT+4) x 8 bit time.
IPGR1	19:10	08h	IPG Part 1 Specifies the defer IPG part 1 (during which carrier sense is monitored). Equals to (IPGR1 x 8) when DJHDX = 0b and equals to (IPGR1+2) x 8 when DJHDX = 1b.
IPGR	29:20	07h	IPG After Deferral Specifies the defer IPG. Equals to (IPGR2+3) x 8 when DJHDX = 0b and equals to (IPGR2+5) x 8 when DJHDX = 1b.
Reserved	31:30	0	Reserved Read as 0b. Should be written with 0b for future compatibility.

10.4.54 Adaptive IFS Throttle - AIT (00458h; R/W)

Adaptive IFS throttles back-to-back transmissions in the transmit packet buffer and delays their transfer to the CSMA/CD transmit function and can be used to delay the transmission of back-to-back packets on the wire. Normally, this register should be set to 0b; however, if additional delay is desired between back-to-back transmits then this register can be set with a value greater than zero.

The *AdaptiveIFS* field provides a similar function to the IPGT field in the TIPG register; however, it only affects the initial transmission timing not re-transmission timing.

Note: If the value of the *AdaptiveIFS* field is less than the *IPGTransmitTime* field in the Transmit IPG registers then it has have no effect as the MAC selects the maximum of the two values.

Table 90. AIT Register Bit Description

Field	Bit(s)	Initial Value	Description
AIFS	15:0	00h	Adaptive IFS Value This value is in units of 8 ns.
Reserved	31:16	00h	Reserved

10.4.55 ICH8 KABGTXD (03004h; RW)

The MAC requires this register to be programmed to 00050000h. There are no user-configurable fields in this register so no further explanation is required. Programming this register to any other value will cause indeterminate behavior.

10.4.56 Transmit Descriptor Base Address Low Queue 0 - TDBAL0 (03800h; R/W); (03800h + n*100h [n=0..1] ICH 10)

This register contains the lower bits of the 64-bit transmit descriptor base address. The lower four bits are ignored and the transmit descriptor base address must point to a 16 byte-aligned block of data.

Table 91. TDBAL0 Register Bit Description

Field	Bit(s)	Initial Value	Description
Reserved	3:0	0b	Reserved Ignored on writes. Returns 0b on reads.
TDBAL0	31:4	X	Transmit Descriptor Base Address Low

10.4.57 Transmit Descriptor Base Address High Queue 0 - TDBAH0 (03804h; R/W); (03804h + n*100h [n=0..1] ICH 10)

This register contains the upper 32 bits of the 64-bit transmit descriptor base address.

Table 92. TDBAH0 Register Bit Description

Field	Bit(s)	Initial Value	Description
TBDAH0	31:0	X	Transmit Descriptor Base Address [63:32]

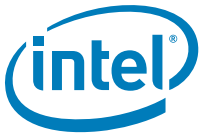
10.4.58 Transmit Descriptor Length Queue 0 - TDLEN0 (03808h; R/W); (03808h + n*100h [n=0..1] ICH 10)

This register contains the descriptor length and must be 128 byte-aligned.

Note: The descriptor ring must be equal to or larger than 16 descriptors.

Table 93. TDLEN Register Bit Description

Field	Bit(s)	Initial Value	Description
Reserved	6:0	00h	Reserved
LEN	19:7	00h	Descriptor Length
Reserved	31:20	0b	Reserved Reads as 0b. Should be written with 0b.



10.4.59 Transmit Descriptor Head Queue 0 - TDH0 (03810h; R/W); (03810h + n*100h [n=0..1] ICH 10)

This register contains the head pointer for the transmit descriptor ring. It points to a 16-byte datum. Hardware controls this pointer. The only time that software should write to this register is after a reset (CTRL.RST) and before enabling the transmit function (TCTL.EN). If software were to write to this register while the transmit function was enabled, the on-chip descriptor buffers can be invalidated and indeterminate operation can result. Reading the transmit descriptor head to determine which buffers have been used (and can be returned to the memory pool) is not reliable.

Table 94. TDH0 Register Bit Description

Field	Bit(s)	Initial Value	Description
TDH0	15:0	0b	Transmit Descriptor Head
Reserved	31:16	0b	Reserved Should be written with 0b.

10.4.60 Transmit Descriptor Tail Queue 0 - TDT0 (03818h; R/W); (03818h + n*100h [n=0..1] ICH 10)

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.

Table 95. TDT Register Bit Description

Field	Bit(s)	Initial Value	Description
TDT0	15:0	0b	Transmit Descriptor Tail
Reserved	31:16	0b	Reserved Reads as 0b. Should be written to 0b for future compatibility.

10.4.61 Transmit Arbitration Count - TARC0 (03840h; RW); (0x03840 + n*100h [n=0..1] ICH10)

The default hardware value for TARC0.COUNT is **3** (this value is also reflected after reset).

The counter is being subtracted as a part of the transmit arbitration.

It is being reloaded for its high (last written) value when it decreased below zero.

- Upon read, the hardware returns the current counter value.
- Upon write, the counter updates the high value in the **next** counter-reload.
- The counter can be decreased in chunks (when transmitting TCP segmentation packets). It should never roll because of that. The size of chunks is determined according to the TCP segmentation (number of packets sent).

When the counter reaches zero, other TX queues should be selected for transmission as soon as possible (usually after current transmission).

COMP is the enable bit to compensate between the two queues. When enabled (set to 1b), hardware compensates between the two queues if one of the queues is transmitting TCP segmentation packets and its counter went below zero. Hardware compensates the other queue according to the ratio in the TARC1.RATIO.

For example, if the TARC0.COUNT reached (-5) after sending TCP segmentation packets and both TARC0.COMP and TARC1.COMP are enabled (set to 1b) and TARC1.RATIO is 01b (1/2 compensation) TARC1.COUNT is adjusted by adding 5/2=2 to the current count.

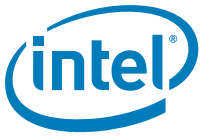
RATIO is the multiplier to compensate the between the two queues the compensation method as previously described.

For DHG 802.3p using qWAVE API the following configuration is used:

TARC0: COUNT = 1, COMP = 0, RATIO = 00.

TARC1: COUNT = 4, COMP = 1, RATIO = 00.

Field	Bit(s)	Initial Value	Description
Count	6:0	03h	Transmit Arbitration Count Number of packets that can be sent from queue 0 to make the N over M arbitration between the queues. Writing 00h to this register is not allowed.
Comp	7	0b	Compensation Mode When set to 1b, hardware compensates this queue according to the compensation ratio if the number of packets in a TCP segmentation in queue 1 caused the counter in queue 1 to drop below zero.
Ratio	9:8	00b	Compensation Ratio This value determines the ratio between the number of packets transmitted on queue1 in a TCP segmentation offload to the number of compensated packets transmitted from queue 0. 00b = 1/1 compensation. 01b = 1/2 compensation. 10b = 1/4 compensation. 11b = 1/8 compensation.
Enable	10	1b	Descriptor Enable The transmit queue 0 *Enable* bit should always be set.



Field	Bit(s)	Initial Value	Description
Reserved	26:11	00h	Reserved Reads as 00h. Should be written to 00h for future compatibility.
Reserved	27	0b	Reserved Multiple Tx request disable. This bit should not be modified by software.
Reserved	31:28	00h	Reserved Reads as 00h. Should be written to 00h for future compatibility.

10.4.62 Transmit Interrupt Delay Value - TIDV (03820h; R/W)

This register is used to delay interrupt notification for transmit operations by coalescing interrupts for multiple transmitted buffers. Delaying interrupt notification helps maximize the amount of transmit buffers reclaimed by a single interrupt. This feature only applies to transmit descriptor operations where:

1. Interrupt-based reporting is requested (RS set).
2. The use of the timer function is requested (IDE is set).

This feature operates by initiating a countdown timer upon successfully transmitting the buffer. If a subsequent transmit delayed-interrupt is scheduled before the timer expires, the timer is re-initialized to the programmed value and re-starts its countdown. When the timer expires, a transmit-complete interrupt (ICR.TXDW) is generated.

Setting the value to 0b is not recommended. If an immediate (non-scheduled) interrupt is desired for any transmit descriptor, the descriptor IDE should be set to 0b.

The occurrence of either an immediate (non-scheduled) or absolute transmit timer interrupt halts the TIDV timer and eliminates any spurious second interrupts.

Transmit interrupts due to a Transmit Absolute Timer (TADV) expiration or an immediate interrupt (RS/RSP = 1b, IDE = 0b) cancels a pending TIDV interrupt. The TIDV countdown timer is reloaded but halted, though it might be restarted by processing a subsequent transmit descriptor.

Writing this register with FPD set initiates an immediate expiration of the timer, causing a write back of any consumed transmit descriptors pending write back, and results in a transmit timer interrupt in the ICR.

Note: FPD is self-clearing.

Table 96. TIDV Register Bit Description

Field	Bit(s)	Initial Value	Description
IDV	15:0	00h	Interrupt Delay Value Counts in units of 1.024 μs. A value of 0b is not recommended.
Reserved	30:16	0b	Reserved Reads as 0b. Should be written to 0b for future compatibility.
FDP	31	0b	Flush Partial Description Block When set to 1b, ignored otherwise. Reads as 0b and is self clearing.

Since write back of transmit descriptors is optional (under the control of RS bit in the descriptor), not all processed descriptors are counted with respect to WTHRESH. Descriptors start accumulating after a descriptor with RS is set. Furthermore, with transmit descriptor bursting enabled, some descriptors are written back that did not have RS set in their respective descriptors.

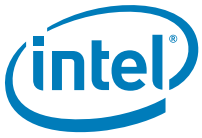
LLWTHRESH controls the number of pre-fetched transmit descriptors at which a transmit descriptor-low interrupt (ICR.TXD_LOW) is reported. This might enable software to operate more efficiently by maintaining a continuous addition of transmit work, interrupting only when the hardware nears completion of all submitted work. LWTHRESH specifies a multiple of 8 descriptors. An interrupt is asserted when the number of descriptors available transitions from (threshold level=8*LWTHRESH)+1 to (threshold level=8*LWTHRESH). Setting this value to 0b disables this feature.

10.4.63 Transmit Descriptor Control Queue 0 - TXDCTL0 (03828h; RW); (03828h + n*100h [n=0..1] ICH10)

Software should set this register to 0000h for normal operation.

Table 97. TXDCTL0 Register Bit Description

Field	Bit(s)	Initial Value	Description
PTHRESH	5:0	00h	Prefetch Threshold Used to control when a prefetch of descriptors are considered. This threshold refers to the number of valid, unprocessed transmit descriptors the MAC has in its on-chip buffer. If this number drops below PTHRESH, the algorithm considers pre-fetching descriptors from host memory. This fetch does not happen; however, unless there are at least HTHRESH valid descriptors in host memory to fetch.
Reserved	7:6	00h	Reserved
HTHRESH	13:8	00h	Host Threshold Should be given a non zero value each time PTHRESH is used.
Reserved	15:14	00h	Reserved
WTHRESH	21:16	00h	Write Back Threshold Controls the write-back of processed transmit descriptors. This threshold refers to the number of transmit descriptors in the on-chip buffer which 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.
Reserved	23:22	00b	Reserved
GRAN	24	0b	Granularity Units for the thresholds in this register. 0b = Cache line granularity. 1b = Descriptor granularity.
LWTHRESH	31:25	00h	Transmit descriptor Low Threshold Interrupt asserted when the number of descriptors pending service in the transmit descriptor queue (processing distance from the TDT) drops below this threshold.



10.4.64 Transmit Absolute Interrupt Delay Value - TADV (0382Ch; RW)

Field	Bit(s)	Initial Value	Description
IDV	15:0	0b	Interrupt Delay Value Counts in units of 1.024 μs. (0b = disabled)
Reserved	31:16	0b	Reads as 0b. Should be written to 0b for future compatibility.

The transmit interrupt delay timer (TIDV) can be used to coalesce transmit interrupts. However, it might be necessary to ensure that no completed transmit remains unnoticed for too long an interval in order ensure timely release of transmit buffers. This register can be used to ENSURE that a transmit interrupt occurs at some predefined interval after a transmit is completed. Like the delayed-transmit timer, the absolute transmit timer ONLY applies to transmit descriptor operations where (a) interrupt-based reporting is requested (RS set) and (b) the use of the timer function is requested (IDE is set).

This feature operates by initiating a countdown timer upon successfully transmitting the buffer. When the timer expires, a transmit-complete interrupt (ICR.TXDW) is generated. The occurrence of either an immediate (non-scheduled) or delayed transmit timer (TIDV) expiration interrupt halts the TADV timer and eliminates any spurious second interrupts.

Setting the value to 0b disables the transmit absolute delay function. If an immediate (non-scheduled) interrupt is desired for any transmit descriptor, the descriptor IDE should be set to 0b.

10.4.65 Transmit Descriptor Base Address Low Queue 1 - TDBAL1 (03900h; R/W)

This register contains the lower bits of the 64-bit descriptor base address. The lower four bits are ignored and the transmit descriptor base address must point to a 16 byte-aligned block of data.

Table 98. TDBAL1 Register Bit Description

Field	Bit(s)	Initial Value	Description
Reserved	3:0	0b	Reserved
TDBAL1	31:4	X	Transmit Descriptor Base Address Low

10.4.66 Transmit Descriptor Base Address High Queue 1 - TDBAH1 (03904h; R/W)

This register contains the upper 32 bits of the 64-bit transmit descriptor base address.

Table 99. TDBAH1 Register Bit Description

Field	Bit(s)	Initial Value	Description
TBDAH	31:0	X	Transmit Descriptor Base Address [63:32]

10.4.67 Transmit Descriptor Length Queue 1 - TDLEN1 (03908h; R/W)

This register contains the descriptor length and must be 128 byte-aligned.

Note: The descriptor ring must be equal or larger than 16 descriptors.

Table 100. TDLEN1 Register Bit Description

Field	Bit(s)	Initial Value	Description
Reserved	6:0	0b	Ignore on write. Reads back as 0b.
LEN	19:7	0b	Descriptor Length
Reserved	31:20	0b	Reserved Reads as 0b. Should be written with 0b.

10.4.68 Transmit Descriptor Head Queue 1 - TDH1 (03910h; R/W)

This register contains the head pointer for the transmit descriptor ring and points to a 16-byte datum. Hardware controls this pointer. The only time that software should write to this register is after a reset (hardware reset or CTRL.SWRST) and before enabling the transmit function (TCTL.EN). If software writes to this register while the transmit function was enabled, the on-chip descriptor buffers might be invalidated and the hardware could become unreliable.

Table 101. TDH1 Register Bit Description

Field	Bit(s)	Initial Value	Description
TDH1	15:0	0b	Transmit Descriptor Head
Reserved	31:16	0b	Reserved Should be written with 0b.

10.4.69 Transmit Descriptor Tail Queue 1 - TDT1 (03918h; R/W)

This register contains the tail pointer for the transmit descriptor ring and 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.

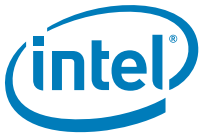


Table 102. TDT1 Register Bit Description

Field	Bit(s)	Initial Value	Description
TDT1	15:0	0b	Transmit Descriptor Tail
Reserved	31:16	0b	Reserved Reads as 0b. Should be written to 0b for future compatibility.

10.4.70 Transmit Descriptor Control 1 - TXDCTL1 (03928h; R/W)

This register controls the fetching and write back of transmit descriptors for queue 1. The three threshold values provided are used to determine when descriptors are read from and written to host memory. The values can be in units of cache lines or descriptors (each descriptor is 16 bytes) based on the GRAN flag.

Table 103. TXDCTL1 Register Bit Description

Field	Bit(s)	Initial Value	Description
PTHRESH	5:0	00h	Prefetch Threshold Used to control when a pre-fetch of descriptors is considered. This threshold refers to the number of valid, unprocessed transmit descriptors the MAC has in its on-chip buffer (queue 1). If this number drops below PTHRESH, the algorithm considers prefetching descriptors from host memory. This fetch does not happen unless there are at least TXDCTL.HTHRESH valid descriptors in host memory to fetch. Value of PTHRESH can be in either cache line units, or based on number of descriptors based on TXDCTL.GRAN.
Reserved	7:6	00h	Reserved
HTHRESH	13:8	00h	Host Threshold Provides the threshold of the valid descriptors in host memory A descriptor prefetch is performed each time enough valid descriptors (TXDCTL.HTHRESH) are available in host memory, no other DMA activity of greater priority is pending (descriptor fetches and write backs or packet data transfers) and the number of transmit descriptors the MAC has on its on-chip buffers drops below TXDCTL.PTHRESH. The value of HTHRESH can be in either cache line units, or based on number of descriptors based on TXDCTL.GRAN.
Reserved	15:14	00h	Reserved
WTHRESH	21:16	00h	Write Back Threshold WTHRESH controls the write back of processed transmit descriptors in queue 1. This threshold refers to the number of transmit descriptors in the MAC's on-chip buffer (queue 1) that are ready to be written back to host memory. In the absence of external events (explicit flushes), the write back occurs only after more than WTHRESH descriptors are available for write back. WTHRESH must contain a non-zero value to take advantage of the write back bursting capabilities of the MAC. A value of 0b causes the descriptors to be written back as soon as they are processed. The value of WTHRESH can be in either cache line units, or based on number of descriptors based on RXDCTL.GRAN.

Field	Bit(s)	Initial Value	Description
Reserved	23:22	00b	Reserved
GRAN	24	0h	Granularity Units for the thresholds in this register. 0b = Cache line granularity. 1b = Descriptor granularity.
LWTHRESH	31:25	0h	Transmit descriptor Low Threshold Interrupt asserted when the number of descriptors pending service in the transmit descriptor queue (processing distance from the TDT) drops below this threshold.

Since write back of transmit descriptors is optional (under the control of RS bit in the descriptor), not all processed descriptors are counted with respect to WTHRESH. Descriptors start accumulating after a descriptor with RS is set. Furthermore, with transmit descriptor bursting enabled, some descriptors are written back that did not have RS set in their respective descriptors.

LWTHRESH controls the number of pre-fetched transmit descriptors at which a transmit descriptor-low interrupt (ICR.TXD_LOW) is reported. This can enable software to operate more efficiently by maintaining a continuous addition of transmit work, interrupting only when the hardware nears completion of all submitted work. LWTHRESH specifies a multiple of eight descriptors. An interrupt is asserted when the number of descriptors available transitions from (threshold level=8*LWTHRESH)+1 to (threshold level=8*LWTHRESH). Setting this value to 0b causes this interrupt to be generated only when the transmit descriptor cache becomes completely empty.

10.4.71 Transmit Arbitration Counter Queue 1 - TARC1 (03940h; RW)

TARC0 and TARC1 registers allow tuning the arbitration parameters, which control the transmission of both transmission queues.

COUNT is the transmit arbitration counter value for queue 0.

The default hardware value for TARC1.COUNT is 1 (this value is also reflected after reset).

The counter is being subtracted as a part of the transmit arbitration.

It is being reloaded for its high (last written) value when it decreased below zero.

- Upon read, the hardware returns the current counter value.
- Upon write, the counter updates the high value in the next counter-reload.
- The counter might be decreased in chunks (when transmitting TCP segmentation packets). It should never roll because of that.

The size of chunks is determined according to the TCP segmentation (number of packets sent).

When the counter reaches zero, other TX queues should be selected for transmission as soon as possible (usually after current transmission).

COMP is the enable bit to compensate between the two queues. When enabled (set to 1b), hardware compensates between the two queues if one of the queues is transmitting TCP segmentation packets and its counter went below zero. Hardware also compensates the other queue according to the ratio in the TARC0.RATIO.

For example, if the TARC1.COUNT reached (-5) after sending TCP segmentation packets and both TARC1.COMP and TARC0.COMP are enabled (set to 1b) and TARC0.RATIO is 01b (1/2 compensation), TARC0.COUNT is adjusted by adding 5/2=2 to the current count.

RATIO is the multiplier to compensate between the two queues when using the compensation method.

Field	Bit(s)	Initial Value	Description
Count	6:0	3	Transmit Arbitration Count The number of packets that can be sent from queue 1 to make the N over M arbitration between the queues.
Comp	7	0b	Compensation Mode When set to 1b, hardware compensates this queues according to the compensation ratio if the number of packets in a TCP segmentation in queue 0 caused the counter in queue 0 to go below zero.
Ratio	9:8	00b	Compensation Ratio This value determines the ratio between the number of packets transmitted on queue 1 in a TCP segmentation offload to the number of compensated packets transmitted from queue 0. 00b = 1/1 compensation. 01b = 1/2 compensation. 10b = 1/4 compensation. 11b = 1/8 compensation.
Enable	10	1b	Descriptor Enable Always set to 1b.
Reserved	31:11	00h	Reserved Reads as 0h. Should be written to 0h for future compatibility.

10.5 Filter Registers

This section contains detailed descriptions for those registers associated with the Ethernet controller's address filter capabilities.

10.5.1 Multicast Table Array - MTA[31:0] (05200h-0527Ch; R/W)

There is one register per 32 bits of the Multicast Address Table for a total of 32 registers (MTA[31:0] designation). The size of the word array depends on the number of bits implemented in the multicast address table. Software must mask to the desired bit on reads and supply a 32-bit word on writes

Note: All access to this table must be 32-bit. Refer to Section 12.0 for example code detailing multicast hash functionality.

Field	Bit(s)	Initial Value	Description
Bit Vector	31:0	X	Word-wide bit vector specifying 32 bits in the multicast address filter table.

Figure 43 shows the multicast lookup algorithm. The destination address shown represents the internally stored ordering of the received DA. Note that Byte 1 (bit 0) indicated in this diagram is the first on the wire. The bits that are directed to the multicast table array in this diagram match a Multicast offset if the CTRL equals 00b. The complete multicast offset options are:

Multicast Offset	Bits Directed to the MTA
00b	DA[47:38] = Byte 6 bits 7:0, Byte 5 bits 1:0
01b	DA[46:37] = Byte 6 bits 6:0, Byte 5 bits 2:0
10b	DA[45:36] = Byte 6 bits 5:0, Byte 5 bits 3:0
11b	DA[43:34] = Byte 6 bits 3:0, Byte 5 bits 5:0

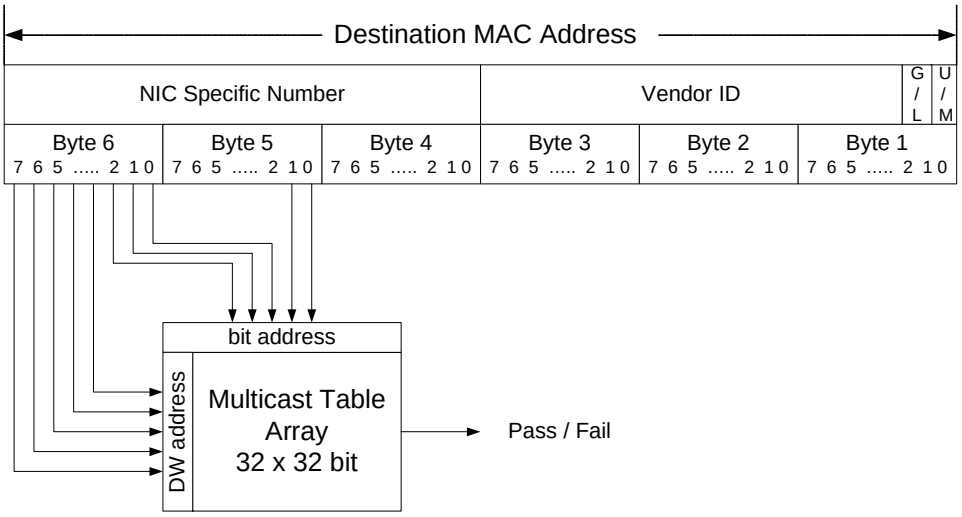


Figure 43. Multicast Table Array

10.5.2 Receive Address Low - RAL (05400h + 8*n; R/W); (05400h + 8*n [n=0..6] ICH10)

Note: "n" is the exact unicast/multicast address entry and is equal to 0, 1, . . .6.

Table 104. RAL Register Bit Description

Field	Bit(s)	Initial Value	Description
RAL	31:0	X	Receive Address Low The lower 32 bits of the 48-bit Ethernet address n (n=0, 1…6). RAL 0 is loaded from words 0 and 1 in the NVM.

10.5.3 Receive Address High - RAH (05404h + 8*n; R/W); (05404h + 8*n [n=0..6] ICH10)

Note: "n" is the exact unicast/multicast address entry and is equal to 0, 1, . . .6.

The first receive address register (RAR0) is also used for exact match pause frame checking (DA matches the first register). Therefore, RAR0 should always be used to store the individual Ethernet MAC address of the MAC.

After reset, if the NVM is present, the first register (Receive Address register 0) is loaded from the IA field in the NVM, its *Address Select* field is 00b, and its *Address Valid* field is 1b. If no NVM is present, the *Address Valid* field is 0b. The *Address Valid* field for all other registers is 0b.

Table 105. RAH Register Bit Description

Field	Bit(s)	Initial Value	Description
RAH	15:0	X	Receive Address High The upper 16 bits of the 48-bit Ethernet address n (n=0, 1…6). RAH 0 is loaded from word 2 in the NVM.
ASEL	17:16	X	Address Select Selects how the address is to be used in the address filtering. 00b = Destination address (required for normal mode). 01b = Source address. 10b = Reserved. 11b = Reserved.
VIND	18	0b	VMDq Output Index (**ICH10**) Defines the VMDq output index associated with a receive packet that matches the MAC address (RAH and RAL). **Note:** This is a reserved bit for **ICH8/ICH9**.
Reserved	30:19	0b	Reserved Ignored on writes. Reads as 0b.
AV	31		Address Valid Cleared after master reset. If the NVM is present, the *Address Valid* field of Receive Address register 0 is set to 1b after a software or PCI reset or NVM read. This bit is cleared by a master (software) reset.

10.5.4 Multiple Receive Queues Command Register - MRQC (05818h; R/W)

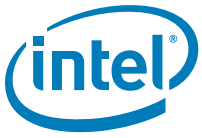
Table 106. MRQC Bit Description

Field	Bit(s)	Initial Value	Description
Multiple Receive Queues Enable	1:0	00h	Multiple Receive Queues Enable Enables support for Multiple Receive Queues and defines the mechanism that controls queue allocation. Note that the *RXCSUM.PCSD* bit must also be set to enable Multiple Receive Queues. This field can be modified only when receive-to-host is not enabled (RCTL.EN = 0b). 00b = Multiple Receive Queues are disabled. 01b = Multiple Receive Queues as defined by MSFT RSS. The RSS Field Enable bits define the header fields used by the hash function. 10b = Reserved 11b = Reserved
Reserved	15:2	00h	Reserved.
RSS Field Enable	21:16	00h	Each bit, when set, enables a specific field selection to be used by the hash function. Several bits can be set at the same time. Bit[16] = Enable TcpIPv4 hash function Bit[17] = Enable IPv4 hash function Bit[18] = Enable TcpIPv6Ex hash function Bit[19] = Enable IPv6Ex hash function Bit[20] = Enable IPv6 hash function Bit[21] = Reserved
Reserved	31:22	00h	Reserved

10.5.5 RSS Interrupt Mask Register - RSSIM (05864h; R/W)

Table 107. RSSIM Register Bit Description

Field	Bit(s)	Initial Value	Description
RSS Interrupt Mask	31:0	00h	Each bit in the RSS Interrupt Mask Register masks a corresponding bit in the CPU Vector register (a value of 0b masks the CPU Vector, while a value of 1b enables the CPU vector bit to be set). A bit in the RSS Interrupt Mask Register is cleared when the CPU Vector is read and the corresponding bit in the CPU Vector was set. It is also cleared when the corresponding bit in the CPU Vector is written with a 1b. A bit in the RSS Interrupt Mask Register is set by writing a 1b into it. Writing a 0b has not effect.



10.5.6 RSS Interrupt Request Register - RSSIR (05868h; R/W)

Table 108. RSSIR Register Bit Description

Field	Bit(s)	Initial Value	Description
RSS Interrupt Request	31:0	00h	Each bit in the RSS Interrupt Request Register indicates that packets are pending processing for the respective CPU. A bit is set when a packet and its descriptor have been posted to memory and the appropriate timer expired. Writing a 1b to a bit clears it. Writing a 0b has no effect.

10.5.7 Redirection Table - RETA (05C00h + 4*n (n=0..31); R/W)

The redirection table is a 32-entry table with each entry composed of four tags, eight bits wide. Only the six bits of each tag are used (five bits for the CPU index and one bit for the queue index).

Offset	31:24	23:16	15:8	7:0
0x05C00 + n*4	Tag 4*n+3	Tag 4*n+2	Tag 4*n+1	Tag 4*n

Field	Bit(s)	Initial Value	Description
CPU INDX 0	4:0	X	CPU index for Tag 4*n (n=0,1,...31)
Reserved	6:5	X	Reserved
QUE INDX 0	7	X	Queue Index for Tag 4*n (n=0,1,...31)
CPU INDX 1	12:8	X	CPU index for Tag 4*n+1 (n=0,1,...31)
Reserved	14:13	X	Reserved
QUE INDX 1	15	X	Queue Index for Tag 4*n+1 (n=0,1,...31)
CPU INDX 2	20:16	X	CPU index for Tag 4*n+2 (n=0,1,...31)
Reserved	22:21	X	Reserved
QUE INDX 2	23	X	Queue Index for Tag 4*n+2 (n=0,1,...31)
CPU INDX 3	28:24	X	CPU index for Tag 4*n+3 (n=0,1,...31)
Reserved	30:29	X	Reserved
QUE INDX 3	31	X	Queue Index for Tag 4*n+3 (n=0,1,...31)

10.5.8 RSS Random Key Register - RSSRK (05C80h + 4*n (n=0..9); R/W)

The RSS Random Key register stores a 40 byte key (10 dword entry table) used by the RSS hash function.

Field	Bit(s)	Initial Value	Description
K0	7:0	00h	Byte n*4 of the RSS random key (n=0,1,...9)
K1	15:8	00h	Byte n*4+1 of the RSS random key (n=0,1,...9)
K2	23:16	00h	Byte n*4+2 of the RSS random key (n=0,1,...9)
K3	31:24	00h	Byte n*4+3 of the RSS random key (n=0,1,...9)

10.6 MNG Registers

10.6.1 Wakeup Control Register - WUC (05800h; R/W)

The PME_Status bits are cleared under the following conditions:

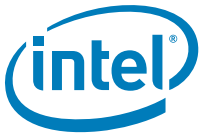
If VAUX, then the PME Status bits should be cleared by:

1. LAN_RST# [or PCI Reset (MAC only)]
2. Explicit software clear.

If no VAUX, then the PME Status bits should be cleared by:

1. LAN_RST# (or PCI Reset).
2. PCI reset de-assertion.
3. Explicit software clear.

Field	Bit(s)	Initial Value	Description
APME	0	0b	Advance Power Management Enable If set to 1b, APM Wakeup is enabled. This bit is loaded from the NVM word 0Ah.
PME_En[1]	1	0b	PME_En[1] This read/write bit is used by the driver to access the PME_En bit of the Power Management Control / Status Register (PMCSR) without writing to the PCI configuration space.
PME_Status	2	0b	PME_Status This bit is set when the MAC receives a wakeup event. It is the same as the PME_Status bit in the Power Management Control Status Register (PMCSR). Writing a 1b to this bit clears the PME_Status bit in the PMCSR.
APMPME	3	0b	Assert PME On APM Wakeup If set to 1b, the MAC sets the PME_Status bit in the Power Management Control / Status Register (PMCSR) and asserts Host_Wake when APM Wakeup is enabled and the MAC receives a matching Magic Packet.



Field	Bit(s)	Initial Value	Description
LSCWE	4	0b	Link Status Change Wake Enable enables wake on link status change as part of APM wake capabilities.
LSCWO	5	0b	Link Status Change Wake Override If set to 1b, wake on Link Status Change does not depend on the LNKC bit in the Wake Up Filter Control Register (WUFC). Instead, it is determined by the APM settings in the WUC register.
Reserved	31:6	00h	Reserved **Note:** Bit 8 is loaded from NVM word 13h (bit 8).

1. ACPI_En for the **82567**.

10.6.2 Wakeup Filter Control Register - WUFC (05808h; R/W)

This register is used to enable each of the pre-defined and flexible filters for wakeup support. A value of 1b means the filter is turned on, and a value of 0b means the filter is turned off.

Field	Bit(s)	Initial Value	Description
LNKC	0	0b	Link Status Change Wakeup Enable
MAG	1	0b	Magic Packet Wakeup Enable
EX	2	0b	Directed Exact Wakeup Enable
MC	3	0b	Directed Multicast Wakeup Enable
BC	4	0b	Broadcast Wakeup Enable
ARP	5	0b	ARP Request Packet Wakeup Enable
IPv4	6	0b	Directed IPv4 Packet Wakeup Enable
IPv6	7	0b	Directed IPv6 Packet Wakeup Enable
Reserved	14:8	0b	Reserved. Set these bits to 0b
NoTCO	15	0	Ignore TCO Packets for TCO 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.
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
Reserved	31:20	00h	Reserved

10.6.3 Wakeup Status Register - WUS (05810h; R)

This register is used to record statistics about all wakeup packets received. If a packet matches multiple criteria then multiple bits could be set. Writing a 1b to any bit clears that bit.

This register is not cleared when PCI_RST_N is asserted. It is only cleared when LAN_RST# is de-asserted or when cleared by the driver software.

Field	Bit(s)	Initial Value	Description
LNKC	0	0b	Link Status Change
MAG	1	0b	Magic Packet Received
EX	2	0b	Directed Exact Packet Received The packet's address matched one of the 16 pre-programmed exact values in the Receive Address registers.
MC	3	0b	Directed Multicast Packet Received The packet was a multicast packet whose hashed to a value that corresponded to a 1 bit in the Multicast Table Array.
BC	4	0b	Broadcast Packet Received
ARP	5	0b	ARP Request Packet Received
IPv4	6	0b	Directed IPv4 Packet Received
IPv6	7	0b	Directed IPv6 Packet Received
Reserved	15:8	0b	Reserved.
FLX0	16	0b	Flexible Filter 0 Match
FLX1	17	0b	Flexible Filter 1 Match
FLX2	18	0b	Flexible Filter 2 Match
FLX3	19	0b	Flexible Filter 3 Match
Reserved	31:20	0b	Reserved

10.6.4 IP Address Valid - IPAV (5838h; R/W)

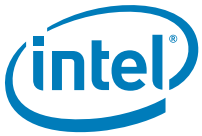
The IP Address Valid indicates whether the IP addresses in the IP Address Table are valid.

Field	Bit(s)	Initial Value	Description
Reserved	0	0b	Reserved
V41	1	0b	IPv4 Address 1 Valid
V42	2	0b	IPv4 Address 2 Valid
V43	3	0b	IPv4 Address 3 Valid
Reserved	15:4	00h	Reserved
V60	16	0b	IPv6 Address Valid
Reserved	31:17	00h	Reserved

10.6.5 IPv4 Address Table - IP4AT (05840h + 8*n (n=1..3); R/W)

The IPv4 address table is used to store the three IPv4 addresses for ARP/IPv4 request packet and directed IPv4 packet wake up. It is a 4 entry table using the following format:

Field	Bit(s)	Initial Value	Description
IPADD	31:0	X	IP address n (n=1, 2, 3)



10.6.6 IPv6 Address Table - IP6AT (05880h + 4*n (n=0..3); R/W)

The IPv6 address table is used to store the IPv6 addresses for directed IPv6 packet wake up and manageability traffic filtering. The IP6AT has the following format:

Field	Bit(s)	Initial Value	Description
IPV6 Address	31:0	X	IPv6 Address bytes n*4...n*4+3 (n=0, 1, 2, 3) while byte 0 is first on the wire and byte 15 is last.

Note: IP6AT can be used by both the host and manageability engine. An interrupt mechanism is added to signal the manageability engine on any change of these registers by the host software.

10.6.7 ICH10 Host to ME Register - H2ME (0x05B50; RW)

This register is used for message passing from the host to the ME.

Field	Bit(s)	Initial Value	Description
LSECREQ	0	0b	LinkSec Request The host driver might assert this bit to request control from the ME over LinkSec logic.
LSECA	1	0b	Host LinkSec Connection Active The host sets this bit to 1b when it owns LinkSec logic and the host software established a LinkSec channel. The host clears this bit when the LinkSec channel is inactive.
LSECSF	2	0b	LinkSec Setup Failed The host driver can assert this bit to notify the ME that it failed to setup a secured link.
LSECD	3	0b	LinkSec Disabled The host driver sets this bit to 1b to indicate the ME that the LinkSec functionality is disabled.
Reserved	4	0b	Reserved
H2MED	31:5	00h	Host to ME Data Message passing content from the host to ME.

10.6.8 Flexible Filter Length Table - FFLT (0x05F00 + 8*n (n=0…3); RW)

There are four flexible filters Lengths. The FFLT stores the minimum packet lengths required to pass each of the flexible filters. Any packets that are shorter than the programmed length won't pass that filter. Each flexible filter considers a packet that doesn't have any mismatches up to that point to have passed the flexible filter when it reaches the required length. It does not check any bytes past that point.

Field	Bit(s)	Initial Value	Description
LEN	10:0	X	Minimum length for flexible filter n.
Reserved	31:11	X	Reserved

All reserved fields read as 0b's and ignore writes.

Note: Before writing to the FFLT, the software device driver must first disable the flexible filters by writing 0b's to the *Flexible Filter Enable* bits in the Wake Up Filter Control register (WUFC.FLXn).

10.6.9 Flexible Filter Mask Table - FFMT (0x09000 + 8*n (n=0…127); RW)

There are 128 mask entries. The FFMT is used to store the four 1-bit masks for each of the first 128 data bytes in a packet, one for each flexible filter. If the mask bit is 1b, the corresponding flexible filter compares the incoming data byte at the index of the mask bit to the data byte stored in the FFVT.

Field	Bit(s)	Initial Value	Description
Mask 0	0	X	Mask for filter 0 byte n (n=0, 1… 127).
Mask 1	1	X	Mask for filter 1 byte n (n=0, 1… 127).
Mask 2	2	X	Mask for filter 2 byte n (n=0, 1… 127).
Mask 3	3	X	Mask for filter 3 byte n (n=0, 1… 127).
Reserved	31:4	X	Reserved

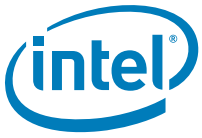
Note: The table is organized to permit an expansion of up to eight (or more) filters and 256 bytes in a future product without changing the address map.

Note: Before writing to the FFMT, the software device driver must first disable the flexible filters by writing 0b's to the *Flexible Filter Enable* bits of the Wake Up Filter Control Register (WUFC.FLXn).

10.6.9.0.1 Flexible Filter Value Table - FFVT (0x09800 + 8*n (n=0…127); RW)

There are 128 filter values. The FFVT is used to store the one value for each byte location in a packet for each flexible filter. If the corresponding mask bit is 1b, the flexible filter compares the incoming data byte to the values stored in this table.

Field	Bit(s)	Initial Value	Description
Value 0	7:0	X	Value of filter 0 byte n (n=0, 1… 127).
Value 1	15:8	X	Value of filter 1 byte n (n=0, 1… 127).



Field	Bit(s)	Initial Value	Description
Value 2	23:16	X	Value of filter 2 byte n (n=0, 1… 127).
Value 3	31:24	X	Value of filter 3 byte n (n=0, 1… 127).

Before writing to the FFVT, the software device driver must first disable the flexible filters by writing 0b's to the *Flexible Filter Enable* bits of the Wake Up Filter Control Register (WUFC.FLXn).

10.6.10 Firmware Semaphore - FWSM (00010h; R/W)

This register is read/write to the ME. It has a shadow read-only image in the host CSR space at offset 05B54h. The FWSM register is initialized only by LAN_RST#.

Field	Bit(s)	Initial Value	Description
FWFLAG	0	0b	Firmware Semaphore FLAG This bit is set by firmware (read-only to software). The bit is set by hardware only after the SWFLAG in the EXTCNF_CTRL register is cleared and hardware does not access the MDIO. Firmware should set this bit and than wait until it is set. When it is set, firmware can read/write from/to the shared resources to firmware and LAN on the PCI. Firmware should clear this bit when finishing its access.
FW_Mode	3:1	00h	Firmware Mode Indicates that the firmware mode is as follows: 000b = None (manageability off). 010b = Intel® AMT mode. 001b = ASF mode. Else reserved. FW_mode is meaningful only when the *FW_Val_bit* is set.
IDE On	4	0b	Set to 1b by firmware when IDE redirection is on.
SOL On	5	0b	Set to 1b by firmware when SOL is on.
RSPCIPHY	6	1b	Reset PHY on PCI Reset When this bit is set The PHY is initialized by PCI Reset and Software LCD_RST. This bit should be cleared if manageability functionality is required over the link across host resets. Remote IDE is just one example for such requirement.
Reserved	8:7	11b	Reserved.
RO	14:9	00h	Reserved.
Reserved	15	0b	Reserved.
Reset_Cnt	18:16	00h	Reset counter Firmware increments it at every reset.

Field	Bit(s)	Initial Value	Description
Ext_Err_Ind	24:19	00h	External error indication Firmware writes the reason that the firmware has reset / clock gated (i.e. EEPROM, flash, patch corruption). Possible values: 00h = No Error 01 = Invalid EEPROM checksum 02h = Unlocked secured EEPROM 03h = Clock Off host command 04h = Invalid Flash checksum 05h = C0 checksum failed 06h = C1 checksum failed 07h = C2 checksum failed 08h = C3 checksum failed 09h = TLB table exceeded 0Ah = DMA load failed 0Bh = Bad hardware version in patch load 0Ch = FLASH device not supported 0Dh = Unspecified Error 3Fh = Reserved - max error value
Reserved	27:25	00h	Reserved
DISSw	28	0b	Disable SW Write Access When set, software has no write access rights to the MAC. Software can read any of the CSR registers and NVM. Reading ICR clears it the same as it functions during nominal operation enabling the driver to clear the Interrupt source. A change in the state of this bit asserts the DSW interrupt bit in the PCI-ICR. Setting DISSW also clears the IAM register in the PCI space or the CTRL_EXT.IAME bit. It is the software device driver's responsibility to restore the IAM register once it gets back control of the MAC.
Reserved	30:29	11b	Reserved
Reserved	31	0b	Reserved

10.7 ICH10 Time Sync Registers

Note: The following registers are for logical needs only. When implementing, their offset or structure can be changed according to specific needs.

10.7.1 RX Time Sync Control Register - TSYNCRXCTL (Offset 0B620; RW)

Field	Bit(s)	Initial Value	Description
RXTT	0	0b	RXTT Rx time stamp valid. 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 register RXSTMPH.
Type	3:1	00h	Type Type of packets to timestamp: 000b = Time stamp L2 (V2) packets only (Sync or Delay_req depends on message type in Section 10.7.6 and packets with message ID 2 and 3). 001b = Time stamp L4 (V1) packets only (Sync or Delay_req depends on message type in Section 10.7.6). 010b = Time stamp V2 (L2 and L4) packets (Sync or Delay_req depends on message type in Section 10.7.6 and packets with message ID 2 and 3). 100b = Time stamp all packets (in this mode no locking is done to the value in the time stamp registers and no indications in receive descriptors are transferred). 101b = Time stamp all packets whose message id bit 3 is zero, which means time stamp all event packets. This is applicable for V2 packets only. 011b, 110b and 111b = Reserved.
En	4	0b	En Enable Rx time stamp 0x0 = Time stamping disabled. 0x1 = Time stamping enabled.
Reserved	31:4	00h	Reserved

10.7.2 Rx Time Stamp Low - RXSTMPL (Offset 0B624; RW)

Field	Bit(s)	Initial Value	Description
RXSTMPL	31:0	00h	RXSTMPL Rx time stamp LSB value.

10.7.3 Rx Time Stamp High - RXSTMPH (Offset 0B628; RW)

Field	Bit(s)	Initial Value	Description
RXSTMPH	31:0	00h	RXSTMPH Rx time stamp MSB value.

10.7.4 Rx Time Stamp Attributes Low - RXSATRL (Offset 0B62C; RW)

Field	Bit(s)	Initial Value	Description
SourceIDL	31:0	00h	SourceIDL Sourceuuid low The value of this register is in host order.

10.7.5 RX Time Stamp Attributes High- RXSATRH (Offset 0x0B630; RW)

Field	Bit(s)	Initial Value	Description
SourceIDH	15:0	00h	SourceIDH Sourceuuid high The value of this register is in host order.
SequenceID	31:16	00h	SequenceID The value of this register is in host order.

10.7.6 RX Ethertype and Message Type Register - RXCFGL (Offset 0B634; RW)

Field	Bit(s)	Initial Value	Description
PTP L2 EtherType	15:0	88F7h	PTP L2 EtherType to time stamp. The value of this register is programmed/read in network order.
V1	23:16	00h	V1 control to time stamp.
V2	31:24	00h	V2 messageId to time stamp.

10.7.7 RX UDP Port - RXUDP (Offset 0x0B638; RW)

Field	Bit(s)	Initial Value	Description
UPORT	15:0	319h	UPORT UDP port number to time stamp. The value of this register is programmed/read in network order.
Reserved	31:16	00h	Reserved

10.7.8 TX Time Sync Control Register - TSYNCTXCTL (Offset 0B614; RW)

Field	Bit(s)	Initial Value	Description
TXTT	0	0b	TXTT Tx time stamp valid. Equals 1b when a valid value for Tx timestamp is captured in the Tx time stamp register. Cleared by read of Tx time stamp register TXSTMPH.
Reserved	3:1	00h	Reserved
En	4	0b	EN Enable TX timestamp 0x0 = Time stamping disabled. 0x1 = Time stamping enabled.
Reserved	31:5	00h	Reserved

10.7.9 TX Time Stamp Value Low - TXSTMPL (Offset 0B618; RW)

Field	Bit(s)	Initial Value	Description
TXSTMPL	31:0	00h	TXSTMPL Tx timestamp LSB value

10.7.10 TX Time Stamp Value High - TXSTMPH (Offset 0B61C; RW)

Field	Bit(s)	Initial Value	Description
TXSTMPH	31:0	00h	TXSTMPH Tx timestamp MSB value

10.7.11 System Time Register Low - SYSTIML (Offset 0B600; RW)

Field	Bit(s)	Initial Value	Description
STL	31:0	00h	STL System time LSB register.

10.7.12 System Time Register High - SYSTIMH (Offset 0B604; RW)

Field	Bit(s)	Initial Value	Description
STH	31:0	00h	STH System time MSB register.

10.7.13 Increment Attributes Register - TIMINCA (Offset 0B608; RW)

Field	Bit(s)	Initial Value	Description
IV	23:0	00h	IV Increment value – incvalue.
IP	31:24	00h	IP Increment period – incperiod.

10.7.14 Time Adjustment Offset Register Low - TIMADJL (Offset 0B60C; RW)

Field	Bit(s)	Initial Value	Description
TADJL	31:0	00h	TADJL Time adjustment value – low.

10.7.15 Time Adjustment Offset Register High - TIMADJH (Offset 0B610; RW)

Field	Bit(s)	Initial Value	Description
TADJH	30:0	00h	TADJH Time adjustment value - high.
Sign	31	0h	Sign Sign (“0”=”+”, “1”=”-“).

10.8 ICH10 LinkSec Register Descriptions

All the fields in the LinkSec registers reflecting parts of the packet header are represented in host ordering.

10.8.1 LinkSec TX Capabilities Register - LSECTXCAP (0B000h; R/W)

Field	Bit(s)	Initial Value	Description
NCA	2:0	1b	TX CA-Supported Number of CA’s supported by the device.
NSC	6:3	1b	TX SC Capable. Number of SC’s supported by the device on the transmit data path. The ICH10 supports twice as many SA’s as the TX SC for seamless re-keying (2 SA’s).
Reserved	15:7	00h	Reserved
LSECTXSUM	23:16	00h	Tx LSEC Key SUM 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	00h	Reserved

10.8.2 LinkSec RX Capabilities Register - LSECRXCAP (0B300h; R/W)

Field	Bit(s)	Initial Value	Description
NCA	2:0	1b	RX CA-Supported Number of CA's supported by the device.
NSC	6:3	00h	RX SC Capable Number of SC's supported by the device on the receive data path. The ICH10 supports twice as many SA's as the RX SC for seamless re-keying (2 SA's).
Reserved	15:7	1h	Reserved
RXLKM	23:16	00h	Rx LSEC Key SUM A bit wise XOR of the Rx LinkSec keys 0…7 as defined in registers LSECRXKEY [n, m]. Each byte is XORed with the respective byte of the other keys. This register can be used by KaY (the programming entity) to validate key programming.
Reserved	31:24	00h	Reserved

10.8.3 LinkSec TX Control Register - LSECTXCTRL (0B004h; R/W)

Field	Bit(s)	Initial Value	Description
LSTXEN	1:0	00b	Enable Tx LinkSec Enable Tx LinkSec off loading. 00b = Disable Tx LinkSec (Tx all packets w/o LinkSec offload). 01b = Add integrity signature. 10b = Encrypt and add integrity signature. 11b = Reserved. When this field equals 00b (LinkSec offload is disabled). The Tx Untagged Packet register is not incremented for transmitted packets when the Enable Tx LinkSec equals 00b.
PNID	2	0b	PN Increase Disabled 0b = Normal operation 1b = PN is not incremented - used only for testability mode.
Reserved	4:3	00b	Reserved
AISCI	5	1b	Always Include SCI This field controls whether SCI is explicitly included in the transmitted SecTag. 0b = False. 1b = True.
Reserved	7:6	00b	Reserved
PNTRH	31:8	11..1b	PN Exhaustion Threshold 24 MSbits of the threshold over which hardware needs to interrupt KaY to warn TX SA PN exhaustion and triggers a new SA renegotiation. Bits 7:0 of the threshold are all 1b's. **Note:** Unlike the LSECTXPN0/1 registers, this field is stored in Network ordering.

10.8.4 LinkSec RX Control Register - LSECRXCTRL (0B304h; R/W)

Field	Bit(s)	Initial Value	Description
Reserved	1:0	00b	Reserved
LSRXEN	3:2	00b	Enable Rx LinkSec Controls the level of LinkSec packet filtering. 00b = Disable Rx LinkSec (passes all packets to host without LinkSec processing and no LinkSec header strip). 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). 10b = Strict (execute LinkSec offload and post frame to host and ME only if it does not fail LinkSec operation). 11b = Rx LinkSec Drop (drops all packets that include LinkSec header).
Reserved	5:4	11b	Reserved
PLSH	6	0b	Post LinkSec header When set, the ICH10 posts the LinkSec header and signature (ICV) to host memory. During normal operation this bit should be cleared. **Note:** When this bit is set VLAN offload and other filter capabilities are disabled.
RP	7	1b	Replay Protect Enable replay protection.
Reserved	31:8	00h	Reserved

10.8.5 LinkSec TX SCI Low - LSECTXSCL (0B008h; R/W)

Field	Bit(s)	Initial Value	Description
SecYL	31:0	0b	MAC Address SecY Low The 4 LS bytes of the MAC address copied to the SCI field in the LinkSec header.

10.8.6 LinkSec TX SCI High - LSECTXSCH (0B00Ch; R/W)

Field	Bit(s)	Initial Value	Description
SecYH	15:0	0b	MAC Address SecY High The 2 MS bytes of the MAC address copied to the SCI field in the LinkSec header.
PI	31:16	0b	Port Identifier Always zero for transmitted packets.

10.8.7 LinkSec TX SA - LSECTXSA (0B010h; R/W)

Field	Bit(s)	Initial Value	Description
AN0	1:0	0b	AN0 – Association Number 0 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 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) This bit selects between SA 0 or SA 1 smoothly (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) 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	00h	Reserved

10.8.8 LinkSec TX SA PN 0 - LSECTXPN0 (0B018h; R/W)

Field	Bit(s)	Initial Value	Description
PN	31:0	0x0	PN – Packet number 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. Packets should never be transmitted if the PN repeats itself. In order to protect against such event the 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. The hardware never transmits packets after the PN reach a value of FF...FFh. In order to guarantee it, the hardware clears the *Enable Tx LinkSec* field in the LSECTXCTRL register if the PN is greater or equals to FF...F0h.

10.8.9 LinkSec TX SA PN 1 - LSECTXPN1 (0B01Ch; R/W)

Field	Bit(s)	Initial Value	Description
PN	31:0	00h	PN – Packet number 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. Packets should never be transmitted if the PN repeats itself. In order to protect against such event the 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. The hardware never transmits packets after the PN reach a value of FF...FFh. In order to guarantee it, the hardware clears the *Enable Tx LinkSec* field in the LSECTXCTRL register to 00b once a packet is transmitted with a PN that equals to FF...F0h.

10.8.10 LinkSec TX Key 0 - LSECTXKEY0 (0B020h + 4*n [n=0...3]; WO)

Field	Bit(s)	Initial Value	Description
LSECK0	31:0	00h	LSEC Key 0 Transmit LinkSec Key of SA 0. n=0 - LSEC Key defines bits 31:0 of the Tx LinkSec Key n=1 - LSEC Key defines bits 63:32 of the Tx LinkSec Key n=2 - LSEC Key defines bits 95:64 of the Tx LinkSec Key n=3 - LSEC Key defines bits 127:96 of the Tx LinkSec Key This field is WO for confidentiality protection. For data integrity check, hash value can read the LSECTXSUM field in the LSECCAP register. If for some reason a read request is aimed to this register a value of all zeros is returned.

10.8.11 LinkSec TX Key 1 - LSECTXKEY1 (0B030h + 4*n [n=0...3]; WO)

Field	Bit(s)	Initial Value	Description
LSECK1	31:0	00h	LSEC Key 1 Transmit LinkSec Key of SA 1. n=0 - LSEC Key defines bits 31:0 of the Tx LinkSec Key n=1 - LSEC Key defines bits 63:32 of the Tx LinkSec Key n=2 - LSEC Key defines bits 95:64 of the Tx LinkSec Key n=3 - LSEC Key defines bits 127:96 of the Tx LinkSec Key This field is WO for confidentiality protection. For data integrity check, hash value can read the LSECTXSUM field in the LSECCAP register. If for some reason a read request is aimed to this register a value of all zeros is returned.

10.8.12 LinkSec RX SCI Low - LSECRXSCL (0B3D0h + 4*n [n=0...3]; R/W)

Field	Bit(s)	Initial Value	Description
MAL	31:0	00h	MAC Address SecY Low The 4 LS bytes of the MAC address in the SCI field of 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.

10.8.13 LinkSec RX SCI High - LSECRXSCH (0B3E0h + 4*n [n=0...3]; R/W)

Field	Bit(s)	Initial Value	Description
MAH	15:0	00h	MAC Address SecY High The 2 MS bytes of the MAC address in the SCI field of 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.
PI	31:16	00h	Port Identifier 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.

10.8.14 LinkSec RX SA - LSECRXSA[n] (0B310h + 4*n [n=0...7]; R/W)

The following registers relate to LinkSec Receive SA context. There are 2 SA(s) in the receive data path defined as SA0 and SA1. The registers below with index n relates to the SA index.

Field	Bit(s)	Initial Value	Description
AN	1:0	00b	AN Association Number. This field is compared with the AN field in the TCI field of the incoming packet for match.
SAV	2	0b	SA Valid This bit is set or cleared by the KaY to validate or invalidate the SA.
FRR (RO)	3	0b	Frame Received This bit is cleared when the SA Valid (bit 2) transitions from 0b→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. Note that 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 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 transitions to 1b). It is set to 1b when a packet is received with the other SA of the same SC. Note that 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	00h	Reserved.

10.8.15 LinkSec RX SA PN - LSECRXSAPN (0B330h + 4*n [n=0...7]; R/W)

Field	Bit(s)	Initial Value	Description
PN	31:0	00h	PN Packet number. This register holds the PN field of the next incoming packet that uses this SA. 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 the hardware for each received packet using this SA to be Received PN + 1.

10.8.16 LinkSec RX Key - LSECRXKEY (0B350h + 10*n [n=0...1] + 4*m (m=0...3); WO)

Field	Bit(s)	Initial Value	Description
LSECK	31:0	00h	LSEC Key Receive LinkSec key of SA n, while n=0, 1. m=0 - LSEC Key defines bits 31:0 of the Rx LinkSec Key m=1 - LSEC Key defines bits 63:32 of the Rx LinkSec Key m=2 - LSEC Key defines bits 95:64 of the Rx LinkSec Key m=3 - LSEC Key defines bits 127:96 of the Rx LinkSec Key This field is write only for confidentiality protection. For data integrity check, KaY hash value can read the LSECRXSUM field in the LSECCAP registers. If for some reason a read request is aimed to this register a value of all zeros is returned.

10.8.17 LinkSec Tx Port Statistics

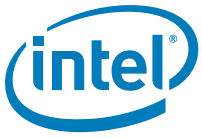
These counters are defined by the specification as 64 bits while implementing only 32 bits in the hardware. The KaY must implement the 64-bit counter in software by polling regularly the hardware statistic counters. The hardware section of the statistics counter is cleared upon read action.

10.8.17.1 Tx Untagged Packet Counter - LSECTXUT (04300h; RC)

Field	Bit(s)	Initial Value	Description
UPC	31:0	00h	Untagged Packet CNT Increments for each transmitted packet that is transmitted with the *ILSec* bit cleared in the packet descriptor while the *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.

10.8.17.2 Encrypted Tx Packets Count - LSECTXPKTE (04304h; RC)

Field	Bit(s)	Initial Value	Description
EPC	31:0	00h	Encrypted Packet CNT Increments for each transmitted packet through the controlled port with the *E* bit set (confidentiality was prescribed for this packet by software/firmware).



10.8.17.3 Protected Tx Packets Count - LSECTXPKTP (04308h; RC)

Field	Bit(s)	Initial Value	Description
PPC	31:0	00h	Protected Packet CNT Increments for each transmitted packet through the controlled port with the *E* bit cleared (integrity only was prescribed for this packet by software/firmware).

10.8.17.4 Encrypted Tx Octets Count - LSECTXOCTE (0430Ch; RC)

Field	Bit(s)	Initial Value	Description
EOC	31:0	00h	Encrypted Octet CNT Increments for each byte of user data through the controlled port with the *E* bit set (confidentiality was prescribed for this packet by software/firmware).

10.8.17.5 Protected Tx Octets Count - LSECTXOCTP (04310h; RC)

Field	Bit(s)	Initial Value	Description
POC	31:0	00h	Protected Octet CNT Increments for each byte of user data through the controlled port with the *E* bit reset (integrity only was prescribed for this packet by software/firmware).

10.8.18 LinkSec Rx Port Statistics

These counters are defined by the specification as 64 bits while implementing only 32 bits in the hardware. The KaY must implement the 64-bit counter in software by regularly polling the hardware statistic counters.

10.8.18.1 LinkSec Untagged RX Packet Count - LSECRXUT (04314h; RC)

Field	Bit(s)	Initial Value	Description
UPC	31:0	00h	Untagged Packet CNT Increments for each packet received having no tag. Increments only when *Enable Rx LinkSec* field in the LSECRXCTRL register is either 01b or 10b.

10.8.18.2 LinkSec RX Octets Decrypted Count - LSECRXOCTE (0431Ch; RC)

Field	Bit(s)	Initial Value	Description
DROC	31:0	00h	Decrypted Rx Octet CNT 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.

10.8.18.3 LinkSec RX Octets Validated Count - LSECRXOCTP (04320h; RC)

Field	Bit(s)	Initial Value	Description
RC	31:0	00h	Validated Rx Octet CNT 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.

10.8.18.4 LinkSec RX Packet with Bad Tag Count - LSECRXBAD (04324h; RC)

Field	Bit(s)	Initial Value	Description
BRPC	31:0	00h	Bad Rx Packet CNT Number of packets received having invalid tag.

10.8.18.5 LinkSec RX Packet No SCI Count - LSECRXNOSCI (04328h; RC)

Field	Bit(s)	Initial Value	Description
USRPC	31:0	00h	No SCI Rx Packet CNT Number of packets received having unrecognized SCI and dropped due to that condition.

10.8.18.6 LinkSec RX Packet Unknown SCI Count - LSECRXUNSCI (0432Ch; RC)

Field	Bit(s)	Initial Value	Description
USRPC	31:0	00h	Unknown SCI Rx Packet CNT Number of packets received with an unrecognized SCI but still forwarded to the host.

10.8.19 LinkSec Rx SC Statistic Register Descriptions

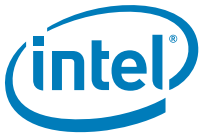
10.8.19.1 LinkSec RX Unchecked Packets Count - LSECRXUNCH (04330h + 4*n [n=0...3]; RC)

Software/firmware needs to maintain the full sized register.

Field	Bit(s)	Initial Value	Description
URPC	31:0	00h	Unchecked Rx Packet CNT Rx Packet CNT. Number of packets received with LinkSec encapsulation (SecTag) while ValidateFrames is disabled (LSECRXCTRL bits 3:2 equal 00b).

10.8.19.2 LinkSec RX Delayed Packets Count - LSECRXDELAY (04340h + 4*n [n=0...3]; RC)

Software/firmware needs to maintain the full sized register.



Field	Bit(s)	Initial Value	Description
DRPC	31:0	00h	Delayed Rx Packet CNT Number of packets received and accepted for validation having failed replay-protection and ReplayProtect is false (LSECRXCTRL bit 1 is 0b).

10.8.19.3 LinkSec RX Late Packets Count - LSECRXLATE (04350h + 4*n [n=0...3]; RC)

Software/firmware needs to maintain the full sized register.

Field	Bit(s)	Initial Value	Description
LRPC	31:0	00h	Late Rx Packet CNT Number of packets received and accepted for validation having failed replay-protection and ReplayProtect is true (LSECRXCTRL bit 1 is 1b). In strict mode, these packets are dropped.

10.8.20 LinkSec Rx SA Statistic Register Descriptions

10.8.20.1 LinkSec RX Packet OK Count - LSECRXOK[n] (0x4360+ 4*n [n=0...7]; RC)

Field	Bit(s)	Initial Value	Description
ORPC	31:0	00h	OK Rx Packet CNT Number of packets received that were valid (authenticated) and passed replay protection.

10.8.20.2 LinkSec RX Invalid count - LSECRXINV[n] (0x4380+ 4*n [n=0...7]; RC)

Field	Bit(s)	Initial Value	Description
ICRPC	31:0	00h	Invalid Rx Packet CNT Number of packets received that were not valid (authentication failed) and were forwarded to host.

10.8.20.3 LinkSec RX Not valid count - LSECRXNV[n] (043A0h + 4*n [n=0...7]; RC)

Field	Bit(s)	Initial Value	Description
ICRPC	31:0	00h	Invalid Rx Packet CNT Number of packets received that were not valid (authentication failed) and were dropped.

10.8.20.4 LinkSec RX Unused SA Count - LSECRXUNSA (043C0h + 4*n [n=0...3]; RC)

Field	Bit(s)	Initial Value	Description
ISSRPC	31:0	00h	Invalid SA Rx Packet CNT Number of packets received that were associated with an SA that is not in use (no match on AN or not valid or retired) and were dropped.

10.8.20.5 LinkSec RX Not Using SA Count - LSECRXNUSA (043D0h + 4*n [n=0...3]; RC)

Field	Bit(s)	Initial Value	Description
ISSRPC	31:0	00h	Invalid SA Rx Packet CNT Number of packets received that were associated with an SA that is not in use (no match on AN or not valid or retired) and where forwarded to host.

10.9 Statistics Registers

All statistics registers are implemented as 32-bit registers. 64-bit accesses to these registers must have the upper byte enables de-asserted. 32-bit registers with addresses not on a qword boundary cannot be accessed through a 64-bit access.

Registers that count octets make up 64-bit registers.

All Statistics registers reset when read. 64-bit registers reset whenever the upper 32 bits are read. In addition, they stick at FFFFh_FFFFh when the maximum value is reached.

The Statistics registers are not hardware initialized. Their default value is unknown. Software should read the contents of all registers in order to clear them prior to enabling the receive and transmit channels.

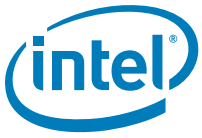
Note: For the receive statistics, it should be noted that a packet is indicated as “received” if it passes the device filters, and it is placed in the packet buffer memory. A packet does not have to be transferred to host memory in order to be counted as “received.”

10.9.1 CRC Error Count - CRCERRS (04000h; RC)

Counts the number of receive packets with CRC errors. In order for a packet to be counted in this register, it must pass address filtering and must be 64 bytes or greater (from <Destination Address> through <CRC>, inclusively) in length. If receives are not enabled, then this register does not increment.

Table 109. CRCERRS Register Bit Description

Field	Bit(s)	Initial Value	Description
CEC	31:0	0b	CRC error count



10.9.2 RX Error Count - RXERRC (0400Ch; RC)

Counts the number of packets received in which RX_ER was asserted by the PHY. In order for a packet to be counted in this register, it must pass address filtering and must be 64 bytes or greater (from <Destination Address> through <CRC>, inclusively) in length. If receives are not enabled, then this register does not increment. In internal SerDes mode, this register increments on the reception of /V/ codes.

Table 110. RXERRC Register Bit Description

Field	Bit(s)	Initial Value	Description
RXEC	31:0	0b	RX error count

10.9.3 Missed Packets Count - MPC (04010h; RC)

Counts the number of missed packets. Packets are missed when the receive FIFO has insufficient space to store the incoming packet. This can be caused because of too few buffers allocated, or because there is insufficient bandwidth on the PCI bus. Events setting this counter cause RXO, the Receiver Overrun Interrupt, to be set. This register does not increment if receives are not enabled.

These packets are also counted in the Total Packets Received register as well as in Total Octets Received.

Table 111. MPC Register Bit Description

Field	Bit(s)	Initial Value	Description
MPC	31:0	0b	Missed Packets Count

10.9.4 Single Collision Count - SCC (04014h; RC)

This register counts the number of times that a successfully transmitted packet encountered a single collision. This register only increments if transmits are enabled and the MAC is in half-duplex mode.

Table 112. SCC Register Bit Description

Field	Bit(s)	Initial Value	Description
SCC	31:0	0b	Number of times a transmit encountered a single collision.

10.9.5 Excessive Collisions Count - ECOL (04018h; RC)

When 16 or more collisions have occurred on a packet, this register increments, regardless of the value of collision threshold. If collision threshold is set below 16, this counter won't increment. This register only increments if transmits are enabled and the MAC is in half-duplex mode.

Table 113. ECOL Register Bit Description

Field	Bit(s)	Initial Value	Description
ECC	31:0	0b	Number of packets with more than 16 collisions.

10.9.6 Multiple Collision Count - MCC (0401Ch; RC)

This register counts the number of times that a transmit encountered more than one collision but less than 16. This register only increments if transmits are enabled and the MAC is in half-duplex mode.

Table 114. MCC Register Bit Description

Field	Bit(s)	Initial Value	Description
MCC	31:0	0b	Number of times a successful transmit encountered multiple collisions.

10.9.7 Late Collisions Count - LATECOL (04020h; RC)

Late collisions are collisions that occur after one slot time. This register only increments if transmits are enabled and the MAC is in half-duplex mode.

Table 115. LATECOL Register Bit Description

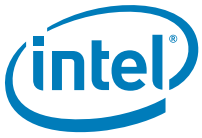
Field	Bit(s)	Initial Value	Description
LCC	31:0	0b	Number of packets with late collisions.

10.9.8 Collision Count - COLC (04028h; RC)

This register counts the total number of collisions seen by the transmitter. This register only increments if transmits are enabled and the MAC is in half-duplex mode. This register applies to clear as well as secure traffic.

Table 116. COLC Register Bit Description

Field	Bit(s)	Initial Value	Description
COLC	31:0	0b	Total number of collisions experienced by the transmitter.



10.9.9 Defer Count - DC (04030h; RC)

This register counts defer events. A defer event occurs when the transmitter cannot immediately send a packet due to the medium being busy either because another device is transmitting, the IPG timer has not expired, half-duplex deferral events, reception of XOFF frames, or the link is not up. This register only increments if transmits are enabled. This counter does not increment for streaming transmits that are deferred due to TX IPG.

Table 117. DC Register Bit Description

Field	Bit(s)	Initial Value	Description
CDC	31:0	0b	Number of defer events.

10.9.10 Transmit with No CRS - TNCRS (04034h; RC)

This register counts the number of successful packet transmissions in which the CRS input from the PHY was not asserted within one slot time of start of transmission from the MAC. Start of transmission is defined as the assertion of TX_EN to the PHY.

The PHY should assert CRS during every transmission. Failure to do so might indicate that the link has failed, or the PHY has an incorrect link configuration. This register only increments if transmits are enabled. This register is only valid when the MAC is operating at half duplex

Table 118. TNCRS Register Bit Description

Field	Bit(s)	Initial Value	Description
TNCRS	31:0	0b	Number of transmissions without a CRS assertion from the PHY.

10.9.11 Carrier Extension Error Count - CEXTERR (0403Ch; RC)

This register counts the number of packets received in which the carrier extension error was signaled across the GMII interface. The PHY propagates carrier extension errors to the MAC when an error is detected during the carrier extended time of a packet reception. An extension error is signaled by the PHY by the encoding of 1Fh on the receive data inputs while RX_ER is asserted to the MAC. This register only increments if receives are enabled and the MAC is operating at
1000 Mb/s.

Table 119. CEXTERR Register Bit Description

Field	Bit(s)	Initial Value	Description
CEXTERR	31:0	0b	Number of packets with a carrier extension error.

10.9.12 Receive Length Error Count - RLEC (04040h; RC)

This register counts receive length error events. A length error occurs if an incoming packet passes the filter criteria but is undersized or oversized. Packets less than 64 bytes are undersized. Packets over 1522 bytes are oversized if *LongPacketEnable* is set to 0b. If *LongPacketEnable* (LPE) is set to 1b, then an incoming, packet is considered oversized if it exceeds 16384 bytes.

If receives are not enabled, this register does not increment. These lengths are based on bytes in the received packet from <Destination Address> through <CRC>, inclusively.

Table 120. RLEC Register Bit Description

Field	Bit(s)	Initial Value	Description
RLEC	31:0	0b	Number of packets with receive length errors.

10.9.13 XON Received Count - XONRXC (04048h; RC)

This register counts the number of valid XON packets received. XON packets can use the global address, or the station address. This register only increments if receives are enabled.

Table 121. XONRXC Register Bit Description

Field	Bit(s)	Initial Value	Description
XONRXC	31:0	0b	Number of XON packets received.

10.9.14 XON Transmitted Count - XONTXC (0404Ch; RC)

This register counts the number of XON packets transmitted. These can be either due to a full queue or due to software initiated action (using TCTL.SWXOFF). This register only increments if transmits are enabled.

Table 122. XONTXC Register Bit Description

Field	Bit(s)	Initial Value	Description
XONTXC	31:0	0b	Number of XON packets transmitted.

10.9.15 XOFF Received Count - XOFFRXC (04050h; RC)

This register counts the number of valid XOFF packets received. XOFF packets can use the global address or the station address. This register only increments if receives are enabled.

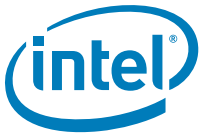


Table 123. XOFFRXC Register Bit Description

Field	Bit(s)	Initial Value	Description
XOFFRXC	31:0	0b	Number of XOFF packets received.

10.9.16 XOFF Transmitted Count - XOFFTXC (04054h; RC)

This register counts the number of XOFF packets transmitted. These can be either due to a full queue or due to software initiated action (using TCTL.SWXOFF). This register only increments if transmits are enabled.

Table 124. XOFFTXC Register Bit Description

Field	Bit(s)	Initial Value	Description
XOFFTXC	31:0	0b	Number of XOFF packets transmitted.

10.9.17 FC Received Unsupported Count - FCRUC (04058h; RC)

This register counts the number of unsupported flow control frames that are received.

The FCRUC counter increments when a flow control packet is received that matches either the reserved flow control multicast address (in FCAH/L) or the MAC station address, and has a matching flow control type field match (to the value in FCT), but has an incorrect opcode field. This register only increments if receives are enabled.

Table 125. FCRUC Register Bit Description

Field	Bit(s)	Initial Value	Description
FCRUC	31:0	0b	Number of unsupported flow control frames received.

10.9.18 Good Packets Received Count - GPRC (04074h; RC)

This register counts the number of good packets received of any legal length. The legal length for the received packet is defined by the value of *LongPacketEnable* (CTRL.LPE) (see Section 10.9.27). This register does not include received flow control packets and only counts packets that pass filtering. This register only increments if receives are enabled. This register does not count packets counted by the Missed Packet Count (MPC) register.

Table 126. GPRC Register Bit Description

Field	Bit(s)	Initial Value	Description
GPRC	31:0	0b	Number of good packets received (of any length).

10.9.19 Broadcast Packets Received Count - BPRC (04078h; RC)

This register counts the number of good (no errors) broadcast packets received. This register does not count broadcast packets received when the broadcast address filter is disabled. This register only increments if receives are enabled. This register does not count packets counted by the Missed Packet Count (MPC) register.

Table 127. BPRC Register Bit Description

Field	Bit(s)	Initial Value	Description
BPRC	31:0	0b	Number of broadcast packets received.

10.9.20 Multicast Packets Received Count - MPRC (0407Ch; RC)

This register counts the number of good (no errors) multicast packets received. This register does not count multicast packets received that fail to pass address filtering nor does it count received flow control packets. This register only increments if receives are enabled. This register does not count packets counted by the Missed Packet Count (MPC) register.

Table 128. MPRC Register Bit Description

Field	Bit(s)	Initial Value	Description
MPRC	31:0	0b	Number of multicast packets received.

10.9.21 Good Packets Transmitted Count - GPTC (04080h; RC)

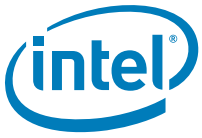
This register counts the number of good (no errors) packets transmitted. A good transmit packet is considered one that is 64 or more bytes in length (from <Destination Address> through <CRC>, inclusively) in length. This does not include transmitted flow control packets. This register only increments if transmits are enabled. This register does not count packets counted by the Missed Packet Count (MPC) register and counts clear as well as secure packets.

Table 129. GPTC Register Bit Description

Field	Bit(s)	Initial Value	Description
GPTC	31:0	0b	Number of good packets transmitted.

10.9.22 Good Octets Received Count - GORCL (04088h; RC)/GORCH (0408Ch; RC)

These registers make up a 64-bit register that counts the number of good (no errors) octets received. This register includes bytes received in a packet from the <Destination Address> field through the <CRC> field, inclusively. This register resets each time the upper 32 bits are read (GORCH).



In addition, it sticks at FFFFh_FFFFh_FFFFh_FFFFh when the maximum value is reached. Only octets of packets that pass address filtering are counted in this register. This register only increments if receives are enabled.

These octets do not include octets of received flow control packets.

Table 130. GORCL and GORCH Register Bit Description

Field	Bit(s)	Initial Value	Description
GORCL	31:0	0b	Number of good octets received – lower 4 bytes.
GORCH	31:0	0b	Number of good octets received – upper 4 bytes.

10.9.23 Good Octets Transmitted Count - GOTCL (04090h; RC)/ GOTCH (04094; RC)

These registers make up a 64-bit register that counts the number of good (no errors) octets transmitted. This register must be accessed using two independent 32-bit accesses. This register also resets each time the upper 32 bits are read (GOTCH).

In addition, it sticks at FFFF_FFFF_FFFF_FFFFh when the maximum value is reached. This register includes bytes transmitted in a packet from the <Destination Address> field through the <CRC> field, inclusively. This register counts octets in successfully transmitted packets that are 64 or more bytes in length. This register only increments if transmits are enabled. The register counts clear as well as secure octets.

These octets do not include octets in transmitted flow control packets.

Table 131. GOTCL and GOTCH Register Bit Description

Field	Bit(s)	Initial Value	Description
GOTCL	31:0	0b	Number of good octets transmitted – lower 4 bytes.
GOTCH	31:0	0b	Number of good octets transmitted – upper 4 bytes.

10.9.24 Receive No Buffers Count - RNBC (040A0h; RC)

This register counts the number of times that frames were received when there were no available buffers in host memory to store those frames (receive descriptor head and tail pointers were equal). The packet is still received if there is space in the FIFO. This register only increments if receives are enabled.

This register does not increment when flow control packets are received.

Table 132. RNBC Register Bit Description

Field	Bit(s)	Initial Value	Description
RNBC	31:0	0b	Number of receive no buffer conditions.

10.9.25 Receive Undersize Count - RUC (040A4h; RC)

This register counts the number of received frames that passed address filtering, and were less than minimum size (64 bytes from <Destination Address> through <CRC>, inclusively), and had a valid CRC. This register only increments if receives are enabled.

Table 133. RUC Register Bit Description

Field	Bit(s)	Initial Value	Description
RUC	31:0	0b	Number of receive undersize errors.

10.9.26 Receive Fragment Count - RFC (040A8h; RC)

This register counts the number of received frames that passed address filtering, and were less than minimum size (64 bytes from <Destination Address> through <CRC>, inclusively), but had a bad CRC (this is slightly different from the Receive Undersize Count register). This register only increments if receives are enabled.

Table 134. RFC Register Bit Description

Field	Bit(s)	Initial Value	Description
RFC	31:0	0b	Number of receive fragment errors.

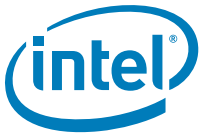
10.9.27 Receive Oversize Count - ROC (040ACh; RC)

This register counts the number of received frames with valid CRC field that passed address filtering, and were greater than maximum size. Packets over 1522 bytes are oversized if *LongPacketEnable* (RCTL.LPE) is 0b. If *LongPacketEnable* is 1b, then an incoming packet is considered oversized if it exceeds 16384 bytes.

If receives are not enabled, this register does not increment. These lengths are based on bytes in the received packet from <Destination Address> through <CRC>, inclusively.

Table 135. ROC Register Bit Description

Field	Bit(s)	Initial Value	Description
ROC	31:0	0b	Number of receive oversize errors.



10.9.28 Receive Jabber Count - RJC (040B0h; RC)

This register counts the number of received frames that passed address filtering, and were greater than maximum size and had a bad CRC (this is slightly different from the Receive Oversize Count register).

Packets over 1522 bytes are oversized if *LongPacketEnable* (RCTL.LPE) is 0b. If *LongPacketEnable* is 1b, then an incoming packet is considered oversized if it exceeds 16384 bytes.

If receives are not enabled, this register does not increment. These lengths are based on bytes in the received packet from <Destination Address> through <CRC>, inclusively.

Table 136. RJC Register Bit Description

Field	Bit(s)	Initial Value	Description
RJC	31:0	0b	Number of receive jabber errors.

10.9.29 Management Packets Received Count - MPRC (040B4h; RC)

This register counts the total number of packets received that pass the management filters as described in the appropriate Total Cost of Ownership (TCO) System Management Bus Interface Application Notes. Management packets include RMCP and ARP packets. Any packets with errors are not counted, except that packets dropped because the management receive FIFO is full or the packet is longer than 200 bytes is counted.

Field	Bit(s)	Initial Value	Description
MPRC	15:0	00h	Number of management packets received.
Reserved	31:16	00h	Reserved

10.9.30 Management Packets Dropped Count - MPDC (040B8h; RO)

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.

Field	Bit(s)	Initial Value	Description
MPDC	15:0	0b	Number of management packets dropped.
Reserved	31:16	00h	Reserved

10.9.31 Total Octets Received - TORL (040C0h; RC)/TORH (040C4h; RC)

These registers make up a 64-bit register that counts the total number of octets received. This register resets each time the upper 32 bits are read (TORH). In addition, it sticks at FFFF_FFFF_FFFF_FFFFh when the maximum value is reached.

All packets received (that pass address filtering) have their octets summed into this register, regardless of their length, whether they are erred, or whether they are flow control packets. This register includes bytes received in a packet from the <Destination Address> field through the <CRC> field, inclusively. This register only increments if receives are enabled.

Note: Broadcast rejected packets are counted in this counter (in contradiction to all other rejected packets that are not counted).

Table 137. TORL and TORH Register Bit Descriptions

Field	Bit(s)	Initial Value	Description
TORL	31:0	0b	Number of total octets received – lower 4 bytes.
TORH	31:0	0b	Number of total octets received – upper 4 bytes.

10.9.32 Total Octets Transmitted - TOTL (040C8h; R/W / TOTH (040CCh; RC)

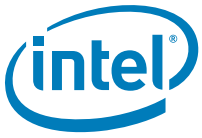
These registers make up a 64-bit register that counts the total number of octets transmitted. This register resets each time the upper 32 bits are read (TOTH). In addition, it sticks at FFFF_FFFF_FFFF_FFFFh when the maximum value is reached.

All transmitted packets have their octets summed into this register, regardless of their length or whether they are flow control packets. This register includes bytes transmitted in a packet from the <Destination Address> field through the <CRC> field, inclusively.

Octets transmitted as part of partial packet transmissions (collisions in half-duplex mode) are not included in this register. This register only increments if transmits are enabled.

Table 138. TOTL and TOTH Register Bit Descriptions

Field	Bit(s)	Initial Value	Description
TOTL	31:0	0b	Number of total octets transmitted – lower 4 bytes.
TOTH	31:0	0b	Number of total octets transmitted – upper 4 bytes.



10.9.33 Total Packets Received - TPR (040D0h; RC)

This register counts the total number of all packets received. All packets received are counted in this register, regardless of their length, whether they have errors, or whether they are flow control packets. This register only increments if receives are enabled.

Note: Broadcast rejected packets are counted in this counter (in contradiction to all other rejected packets that are not counted).

Table 139. TPR Register Bit Description

Field	Bit(s)	Initial Value	Description
TPR	31:0	0b	Number of all packets received.

10.9.34 Total Packets Transmitted - TPT (040D4h; RC)

This register counts the total number of all packets transmitted. All packets transmitted are counted in this register, regardless of their length, or whether they are flow control packets.

Partial packet transmissions (collisions in half-duplex mode) are not included in this register. This register only increments if transmits are enabled. This register counts all packets, including standard packets, secure packets, and packets generated by the ASF function.

Table 140. TPT Register Bit Description

Field	Bit(s)	Initial Value	Description
TPT	31:0	0b	Number of all packets transmitted.

10.9.35 Multicast Packets Transmitted Count - MPTC (040F0h; RC)

This register counts the number of multicast packets transmitted. This register does not include flow control packets and increments only if transmits are enabled. Counts clear as well as secure traffic.

Table 141. MPTC Register Bit Description

Field	Bit(s)	Initial Value	Description
MPTC	31:0	0b	Number of multicast packets transmitted.

10.9.36 Broadcast Packets Transmitted Count - BPTC (040F4h; RC)

This register counts the number of broadcast packets transmitted. This register only increments if transmits are enabled. Counts clear as well as secure traffic (Management packets are never more than 200 bytes).

Table 142. BPTC Register Bit Description

Field	Bit(s)	Initial Value	Description
BPTC	31:0	0b	Number of broadcast packets transmitted count.

10.9.37 TCP Segmentation Context Transmitted Count - TSCTC (040F8h; RC)

This register counts the number of TCP segmentation offload transmissions and increments once the last portion of the TCP segmentation context payload is segmented and loaded as a packet into the MAC's on-chip transmit buffer. Note that this is not a measurement of the number of packets sent out (covered by other registers). This register only increments if transmits and TCP Segmentation offload are enabled.

Field	Bit(s)	Initial Value	Description
TSCTC	31:0	0b	Number of TCP Segmentation contexts transmitted count.

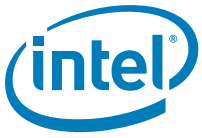
10.9.38 TCP Segmentation Context Tx Fail Count - TSCTFC (040FCh; RC)

This register counts the number of TCP segmentation offload requests to the hardware that failed to transmit all data in the TCP segmentation context payload. There is no indication by hardware of how much data was successfully transmitted. Only one failure event is logged per TCP segmentation context. Failures could be due to PAYLEN errors. This register only increments if transmits are enabled.

Field	Bit(s)	Initial Value	Description
TSCTFC	31:0	0b	Number of TCP Segmentation contexts where the MAC failed to transmit the entire data payload.

10.9.39 Interrupt Assertion Count - IAC (04100h; RC)

Field	Bit(s)	Initial Value	Description
IAC	31:0	00h	This is a count of the interrupt assertions that have occurred. It counts the total number of interrupts generated in the system.



Note: This page intentionally left blank.

11.0 Initialization and Reset Operation

11.1 Introduction

This section discusses initialization steps. Topics include:

- General hardware power-up state
- Basic device configuration
- Initialization of transmit and receive operation
- Link configuration
- Software reset capability
- Statistics initialization.

The entire functionality of the MAC and PHY is enabled only when the *LanDisable* bit in the Function Disable SUS register is not set.

11.2 Reset Operation

11.2.1 MAC Reset

The MAC reset sources are as follows:

LAN_RST#:

The MAC uses this reset as a master reset of the entire cluster. It is level sensitive, and while it is inactive at zero the entire device is initialized. LAN_RST# is interpreted to be an indication that MAC power supplies are all stable. LAN_RST# changes state during system power-up.

PCI Reset:

The MAC uses this to reset the PCI host logic, including CSRs and PCI configuration space. The PHY can also be initialized if the *RSPCIPHY* bit in the FWSM register is set.

ICH9/ICH10 Function Level Reset (FLR)

FLR has the same impact as the PCI reset except for the Interrupt Line register in the PCI configuration space. It enables a platform with multiple virtual machines to initialize the LAN function without resetting the rest of the platform. For this document, any reference to PCI reset relates to FLR as well unless specified differently.

Device Disable:

The MAC enters a device disable mode when the *LanDisable* bit in the Function Disable SUS register is set. In this mode, the LANRSTSYNC signal is held. It is expected that BIOS software sets the PHY to power down before asserting device disable.

D3hot to D0u Transition:

This is also known as ACPI Reset and has the same impact as PCI Reset. The MAC generates an internal reset on the host/PCI transition from D3hot power state to D0u.

PCI Software Reset:

Software can reset the MAC by setting the CTRL.SWRST. The MAC reads the NVM after a software reset. PCI configuration space (configuration and mapping) of the MAC is unaffected.

PHY Software Reset (PHY SW Reset):

Software or firmware can write a 1b to the LCD_RST bit in the CTRL register. The LCD_RST bit is propagated to the LCDRSTSYNC signal to the PHY only if the RSPCIPHY bit in the FWSM register is set (ME veto bit). PHY Reset also wakes up the PHY from a PHY power down state. Following a reset complete indication from the PHY, the MAC initializes it from the NVM.

The resets affect the following registers and logic:

Reset Activation	LAN_RST#	PCI Reset (1), D3hot to D0u	PCI SW Reset	PHY SW Reset	Notes
Read Basic NVM Image (PCI and Shared)		X	X		
PCI Wake context + PHY Type + LEDs	X				3
PCI Config Space	X	X			
PCI and Shared Data Path	X	X	X		4
PHY Reset	X	X		X	5

Notes:

1. PCI reset initiates an internal reset pulse on their de-assertion
2. NVM is loaded and the PHY is reset following LAN_RST#.

3. PCI Wake + PHY type + LEDs include the following registers:
   a. PME context and control CSRs: PME_En bit and PME_Status bit of the Power Management Control/Status Register (PMCSR); Wake Up Control Register; Wake Up Status Register.
   b. The packet buffer size and pointers that enable reading of wake up packets.
   c. LED settings are initialized to the hardware default only on power up. The registers are reloaded from the NVM according to the basic NVM image rules.
   d. The PHY Type Indication in the Status Register is initialized to the hardware default only at power up. The field is re-loaded from the NVM according to the basic NVM image rules.
4. PCI and Shared Data Path include the CSMA block and the following registers:
   a. General Registers; Interrupt Registers; Receive Registers; Transmit Registers; Statistics Registers; Diagnostic Registers; GLCI AFE.

Note: Of the previously mentioned registers, those that have no default value are preserved through all resets as long as power is applied to the MAC.

5. The **82566** might have an image in the NVM that is loaded to the PHY following a reset complete indication to the MAC. It includes the PHY Extended Space. These images should not be enabled for non-**82566** devices.
6. PHY_CTRL register bits (F10h) that are not loaded from the NVM are not reset at a software reset.
7. In the WUC register (5800h), bit APME is reset at a software reset since it's loaded from NVM.
8. For **ICH9/ICH10**, FLR has the same affect as PCI reset except for the Interrupt Line register in the PCI configuration space. The PCI reset as well as the FLR do not initialize PCI wake context including the *PME_Status* and *PME_En* bit fields in the PMCSR register.

11.2.2 PHY Reset

PHY reset occurs by the assertion of the LANRSTSYNC signal for a duration of at least 1 µs. This reset occurs in one of the following cases:

- Inactive LAN_RST# forces an active reset to the PHY.
- PCI hardware resets or setting of the *CTRL.LCD_RST* bit by software or firmware if enabled by the FWSM.RSPCIPHY flag.
- Exiting PHY power down mode; also for the **82566**, Auto Connect Battery Saver (ACBS).

The **82566/82567** completes its internal reset with a reset complete indication to the MAC. Following the reset completion, the MAC configures the PHY from the NVM as explained in Section 11.2.2.1 and Section 11.2.2.2.

Note: If the OEM bits are not enabled then the NVM image must include setting the *Restart AN* bit in the PHY Control register (bit 9) or the PHY Power Management register (bit 10). This ensures that the link initialization takes place.

11.2.2.1 MDIO PHY Reset

PHY reset is a reset to the portion of the PHY that uses the MDIO reset cycle. PHY reset occurs following an access to the MDIC CSR register writing an MDIO reset instruction (set the *Reset* bit in the PHY Control register. If the PHY is the **82566/82567**, it indicates a reset complete that is followed by the MAC with the PHY configuration. Afterwards, the *LAN Init Done* bit is set (in the STATUS and MICR).

Before the MAC accesses the PHY via MDIC register (following the reset completion indication) it first sets the *MDIO HW Ownership* bit and waits until both software and firmware completes their accesses to the shared resources by clearing the SWFLASG and HWFLAG bits. Note that both flags are initialized to an inactive state at LAN power up.

Note: If the OEM bits are not enabled then the NVM image must include setting the *Restart AN* bit in the PHY Control register (bit 9) or the PHY Power Management register (bit 10). This ensures that the link initialization takes place.

Note: Following a PHY reset instruction, software should release the SWFLAG as soon as possible enabling hardware to complete the reset sequence with minimal delays.

11.2.2.2 LAN Init Done Event

The MAC reports to the host CPU and ME that it completed its initialization by setting the *LAN Init Done* bit in the following registers:

- ME Interrupt register (MICR)
- For **ICH9/ICH10**, ME General Control register (MGCR)
- Host Status register (STATUS)

The *LAN Init Done* bit is set in one of the following cases when a PHY is attached:

- Following the assertion of the Auto_RD bit in the EEC register if there is no NVM or no Valid image in the NVM or no Extended NVM space.
- Or, following the completion of PHY configuration from the Extended NVM space.
- Or, following PHY Reset.
  - If there is no valid extended configuration in the NVM the *LAN Init Done* bit is set after the reset completion indication from the PHY.
- Or, Valid NVM image with extended NVM space, the *LAN Init Done* bit is set after configuration from the Extended NVM space.

11.3 Hardware Initialization Flow

11.3.1 Power-Up Sequence

The power-up sequence of the MAC is as follows:

1. Power rails ramp with LAN_RST# inactive. The entire MAC is set to its hardware default (including the reset control flags in the FWSM register and hardware/ firmware/software semaphore flags).
2. Power rails stabilize and LAN_RST# asserts (goes high).
3. MBB clock is active (both mosc_clk and mb_clk).
4. ICH reads the NVM descriptor including LAN soft straps that are latched by the MAC.
5. The MAC reads its configuration from the NVM (basic image). If PCI reset is de-asserted, then it is ready on the PCI bus as well (no retries).
6. Wait 1 μs after a stable NJCLK is sampled to guarantee reset width to the PHY.
7. Sample RxD signals (PHY auto-detect) and then de-assert the LANRSTSYNC signal.
8. Initialize link to the PHY. If PCI reset is de-asserted, then it is ready on the PCI bus as well (no retries).

9. If the PHY is connected, set the *LAN Init Done* bits in the PCI STATUS indicating that the MAC is now fully operational.
10. If the PCI device is at D3 or Dr state with no WoL to host functionality, then the MAC sets the PHY to power down.

Note: The latency of ~1 µs is measured as 3800h clocks of 12.5 MHz (mb_clk) after the clock is active on the NJCLK input. The MAC counts 16 clocks on NJCLK to ensure valid clock sensing. This width is sufficient for all supported PHYs.

Note: GLCI and LCI modes are enabled by the NVM and hardware settings as follows:

The ICH decides between GLCI or PCI according to the GLCI Enable strapping bit in the NVM. The *PHYT* field in the LAN space in the NVM enables GLCI and identifies between GLCI 1.x or 2.0. The PHYT field also enables LCI 1.x or 2.0 operation that requires further detection while LANRSTSYNC is asserted.

11.3.2 PCI Reset Sequence

The functionality of the PCI reset depends on the setting of the RSPCIPHY flag in the FWSM register. This section describes the hardware flow by de-asserting PCI reset if the RSPCIPHY flag is set.

1. The MAC might maintain its functionality to the ME over the MBB interface according to the MAC DM power state and its programming.
2. Following PCI reset de-assertion, the MAC is at the Dr state and is not accessible on the PCI interface. Initiated cycles are responded with retry status indication.
3. If the ME does not veto a PHY reset, then set the *MDIO HW Ownership* bit and waits until it is ready.
4. If the ME does not veto a PHY reset, then assert a reset on the LANRSTSYNC signal and wait for NJCLK.
5. Read the basic NVM image.

Note: At this point the MAC might respond to host configuration cycles on the PCI space.

6. Wait 1 µs after a stable NJCLK is sampled to guarantee reset width to the PHY.
7. Sample RxD signals (LCI auto-Detect) and then de-assert the LANRSTSYNC signal.
8. Initialize link to the PHY and initialize the PHY including fetching from the NVM the required extended configuration and OEM bits.
9. If the PHY is connected, set the *LAN Init Done* bits in the PCI STATUS indicating that the MAC is now fully operational.
10. If the PCI device is at D3 or Dr state with no WoL to host functionality, then the MAC sets the PHY to power down.

11.4 Software Initialization Sequence

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

- Disable Interrupts - see Interrupts during initialization.
- Issue software reset and perform general configuration.
- Setup the PHY and the link - see Link Setup Mechanisms and Control/Status Bit Summary.
- Initialize all statistical counters - see Initialization of Statistics.

- Initialize Receive - see Receive Initialization.
- Initialize Transmit - see Transmit Initialization.
- Enable Interrupts - see Interrupts During Initialization.

11.4.1 Interrupts During Initialization

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

- Disable interrupts
- Issue a software reset
- Disable interrupts (again)
- ...

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

11.4.2 Software Reset and General Configuration

Device initialization typically starts with a software reset that puts it into a known state and enables the software device driver to continue the initialization sequence.

Several values in the Device Control Register (CTRL) need to be set at power up or after a device reset for normal operation.

- FD should be set per interface negotiation (if done in software), or is set by the hardware if the interface is Auto-Negotiating. This is reflected in the Device Status register in reference to Auto-Negotiating. A default value is loaded from the NVM.
- Speed is determined via Auto-Negotiation by the PHY or forced by software if the link is forced. Status information for speed is also readable in STATUS.

11.4.2.1 Nominal Operation

When software is required to initialize the MAC and PHY for nominal operation, it should use the *SWRST* bit and the LCD_RST bit in the CTRL register. If the *RSPCIPHY* bit in the FWSM register is cleared, firmware cannot tolerate a link disconnect by the host. As a result, it gates the host resets from propagation to the PHY. It also implies to the host software that the ME is using the PHY and it should be in a functional state.

Note: Make sure that the PBS register is programmed correctly (see Section 10.4.20).

11.4.3 Link Setup Mechanisms and Control/Status Bit Summary

11.4.3.1 PHY Initialization

Refer to the PHY documentation for the initialization and link setup steps. The software device driver uses the MDIC register to initialize the PHY and setup the link. A delay of 15 ms is required for the PHY to complete its reset.

11.4.3.2 MAC/PHY Link Setup

This section summarizes the various means of establishing proper MAC/PHY link setups, differences in MAC CTRL register settings for each mechanism, and the relevant MAC status bits. The methods are ordered in terms of preference (the first mechanism being the most preferred).

- MAC settings automatically based on duplex and speed resolved by PHY (CTRL.FRCDPLX = 0b, CTRL.FRCSPD = 0b

CTRL.FD - Don't care; duplex setting is established from PHY's internal indication to the MAC (FDX) after PHY has auto-negotiated a successful link-up

CTRL.RFCE - Must be set by software after reading flow control resolution from PHY registers

CTRL.TFCE - Must be set by software after reading flow control resolution from PHY registers

CTRL.SPEED - Don't care; speed setting is established from PHY's internal indication to the MAC (SPD_IND) after PHY has auto-negotiated a successful link-up

STATUS.FD - Reflects the actual duplex setting (FDX) negotiated by the PHY and indicated to MAC

STATUS.LU - Reflects link indication (LINK) from PHY

STATUS.SPEED - Reflects actual speed setting negotiated by the PHY and indicated to the MAC (SPD_IND)

- MAC duplex setting automatically based on resolution of PHY, software-forced MAC/PHY speed (CTRL.FRCDPLX = 0b, CTRL.FRCSPD = 1b

CTRL.FD - Don't care; duplex setting is established from PHY's internal indication to the MAC (FDX) after PHY has auto-negotiated a successful link-up

CTRL.RFCE - Must be set by software after reading flow control resolution from PHY registers

CTRL.TFCE - Must be set by software after reading flow control resolution from PHY registers

CTRL.SPEED - Set by software to desired link speed (must match speed setting of PHY)

STATUS.FD - Reflects the actual duplex setting (FDX) negotiated by the PHY and indicated to MAC

STATUS.LU - Reflects link indication (LINK) from PHY

STATUS.SPEED - Reflects MAC forced speed setting written in CTRL.SPEED

- MAC duplex and speed settings forced by software based on resolution of PHY (CTRL.FRCDPLX = 1b, CTRL.FRCSPD = 1b

CTRL.FD - Set by software based on reading PHY status register after PHY has auto-negotiated a successful link-up

CTRL.RFCE - Must be set by software after reading flow control resolution from PHY registers

CTRL.TFCE - Must be set by software after reading flow control resolution from PHY registers

CTRL.SPEED - Set by software based on reading PHY status register after PHY has auto-negotiated a successful link-up.

STATUS.FD - Reflects the MAC forced duplex setting written to CTRL.FD

STATUS.LU - Reflects link indication (LINK) from PHY

STATUS.SPEED - Reflects MAC forced speed setting written in CTRL.SPEED

- MAC/PHY duplex and speed settings both forced by software (fully-forced link setup) (CTRL.FRCDPLX = 1b, CTRL.FRCSPD = 1b

CTRL.FD - Set by software to desired full/half duplex operation (must match duplex setting of PHY)

CTRL.RFCE - Must be set by software to desired flow-control operation (must match flow-control settings of PHY)

CTRL.TFCE - Must be set by software to desired flow-control operation (must match flow-control settings of PHY)

CTRL.SPEED - Set by software to desired link speed (must match speed setting of PHY)

STATUS.FD - Reflects the MAC duplex setting written by software to CTRL.FD

STATUS.LU - Reflects 1b. (positive link indication LINK from PHY). **Note:** since the PHY link indication LINK is forced, this bit set does not guarantee that operation of the link has been truly established.

STATUS.SPEED - Reflects MAC forced speed setting written in CTRL.SPEED

11.4.4 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 D0active power state (when internal registers become accessible, as enabled by setting the Memory Access Enable of the PCI Command register), and is guaranteed to be completed within 1 µs of this transition. Access to statistics registers prior to this interval may return indeterminate values.

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

11.4.5 Receive Initialization

Program the Receive address register(s) per the station address. This can come from the NVM or from any other means (for example, on some systems, this comes from the system PROM not the NVM on the NIC) Set up the MTA (Multicast Table Array) per software. Zeroing all entries initially and adding in entries as requested is generally required.

Program the Interrupt Mask Set/Read (IMS) register to enable any interrupt the software driver wants to be notified of when the event occurs. Suggested bits include RXT, RXO, RXDMT, RXSEQ, and LSC. There is no immediate reason to enable the transmit interrupts.

Program RCTL with appropriate values. If initializing it at this stage, it is best to leave the receive logic disabled (EN = 0b) until after the receive descriptor ring has been initialized. Select the receive descriptor type. Note that if using the header split RX descriptors, tail and head registers should be incremented by 2 per descriptor.

11.4.5.1 Initialize the Receive Control Register

To properly receive packets requires that the receiver is enabled. This should be done only after all other setup is accomplished. If software uses the Receive Descriptor Minimum Threshold Interrupt, that value should be set.

The following should be done once per receive queue:

- Allocate a region of memory for the receive descriptor list.
- Receive buffers of appropriate size should be allocated and pointers to these buffers should be stored in the descriptor ring.
- Program the descriptor base address with the address of the region (RDBALn, RDBAHn, n = 0b or 1b).
- Set the length register to the size of the descriptor ring (RDLENn, n = 0b or 1b).
- If needed, program the head and tail registers (RDHn, RDTn, n = 0b or 1b). Note that the head and tail pointers are initialized (by hardware) to 0b after a power-on or a software-initiated device reset.
- The tail pointer should be set to point one descriptor beyond the end.

11.4.6 Transmit Initialization

Program the TXDCTL register with the desired TX descriptor write back policy. Suggested values are:

- GRAN = 1b (descriptors)
- WTHRESH = 1b
- All other fields 0b

Program the TXDCTL register. Suggested configuration:

- CT = 0Fh (16d collision)
- COLD: HDX = 511 (1FFh); FDX = 63 (03Fh)
- PSP = 1b
- EN=1b
- All other fields 0b

Program the TIPG register.

Note: IPGR1 and IPGR2 are not needed in full-duplex, but it is easier to always program them to the values shown.

The following should be done once per transmit queue:

- Allocate a region of memory for the transmit descriptor list.
- Program the descriptor base address with the address of the region.
- Set the length register (TDLEN) to the size of the descriptor ring.
- If needed, program the head and tail registers (TDH, TDT). **Note:** the head and tail pointers are initialized (by hardware) to 0b after a power-on or a software-initiated device reset.

If working with multiple transmit queues, setup the queues priority by programming TARC0 and TARC1 registers.

Program the TIPG register (see the register description for the required value in Section 10.0).

Note: IPGR1 and IPGR2 are not needed in full-duplex, but it is easier to always program them to the values shown.

If operating with multiple transmit queues, the multiple request support in the TCTL register must be set and the queues priority must be set up by programming the TARC0 and TARC1 registers.

Initialize the transmit descriptor registers (TDBAL, TDBAH, TDH, TDT).

11.5 NVM Extended Configuration and OEM Bits

This section defines the procedure of the PHY setup if using the **82566/82567**. Accesses to the PHY are executed by MDIO cycles using PHY address of 00001b. Note that the MAC uses the MDIC for internal accesses to the PHY. As a result, software and firmware must use the *MDIO ownership* (semaphore) bit in the EXTCNF_CTRL and FWSM registers, respectively, before accessing the MDIC register.

Note: Refer to Section 12.0 for example code detailing NVM/EEPROM initialization.

Table 143 lists the Extended Configuration registers loaded in each case. Note that loading any of the configuration areas depends on the respective enable bit to be set.

Table 143. Loading the Extended Configuration

	PHY Reset (1) MDIO PHY Reset (2) State change to/from D0a (3) Write to the OEM bits (3)
PHY Extended Image Setting (5)	Yes
OEM Bits (7)	Yes (Go Disconnect = 0b, Restart AN = 1b)

Notes:

1. PHY Reset can be initiated by direct access to the *LCD_RST* bit in the CTRL register or it is asserted while LAN_RST# in inactive. Additionally, when the PHY exits from Power Down (including Auto Connect Battery Saver) state a PHY reset is also issued.
2. MDIO PHY Reset is not recommended for use by software since it bypasses the *RSPCIPHY* bit in the FWSM register.

3. PCI power state transition to/from D0a or direct write access to the OEM fields (LPLU and GbE disable bits in the PHY_CTRL register) that requires a PHY update (Table 143) while the *OEM Write Enable* bit in the EXTCNF_CTRL register is set and the *RSPCIPHY* bit in the FWSM register is also set.
4. The PHY Extended Config image is enabled by the *LCD Write Enable* bit in the EXTCNF_CTRL register.
5. Init the OEM bits are enabled by the *OEM Write Enable* bit in the EXTCNF_CTRL register. Setting of the *Restart AN* bit depends on the source event.

11.5.1 PHY Reset and PHY Reset Init

PHY reset and MDIO PHY reset to the **82566/82567** are followed by a reset completion indication from the PHY to the MAC. After the reset completion, the MAC initializes the PHY as follows:

- Load PHY Extended image from the NVM enabled by the *LCD Write Enable* bit in the EXTCNF_CTRL register.
- Load the OEM bits to PHY register 25d enabled by the *OEM Write Enable* bit in the EXTCNF_CTRL register. The *Go Disconnect* bit is cleared and the *Restart AN* bit is set. The LPLU and GbE setting depends on the PCI device state and the static setting of the LPLU and GbE enabling in the PHY_CTRL register.

Before the MAC accesses the PHY via MDIC register (following the reset completion indication), it first sets and checks the *MDIO HW Ownership* bit.

The **82562V** does provide a reset completion indication to the MAC. If this is the case, the PHY related extended images in the NVM should not be enabled.

Following a reset event (PHY or PHY reset), the PHY negotiates the link according to its setting listed in Table 143.

11.5.2 Setting PHY OEM Bits

The MAC sets the PHY OEM bits only when it is connected to the **82566** and enabled in the NVM word 14h. For the **82566**, the PHY OEM bits are driven as follows:

- The PHY transition from power down to active is a sample case of a reset event.
- PCI function power state transitions to/from D0a enabled by *OEM Write Enable* bit in the EXTCNF_CTRL register and *RSPCIPHT* bit in FWSM register.
- Write access to the OEM bits in the PHY_CTRL register that requires change of the PHY behavior enabled by *OEM Write Enable* bit in the EXTCNF_CTRL register and *RSPCIPHT* bit in FWSM register.

The PHY OEM bits are located in Register 25d in bank 0 of the **82566/82567**. Setting these bits can be accomplished by:

- Sending a MDIO cycle to direct the **82566/82567** to bank 0
- Sending a MDIO cycle setting to register 25d from the PHY_CTRL register.

The detailed content of register 25d in the PHY is as follows:

- *Smart Power Down Enable* (bit 0) is copied from the *SPD Ena* (bit 0) in the PHY_CTRL register[1]
- *SPD_B2B_EN* (bit 7) is copied from the *B2B Ena* (bit 7) in the PHY_CTRL register[1]
- *LPLU* (bit 2) and *Disable 1000* (bit 6) are set
- *Go Disconnect* (bit 5) is cleared to 0b and *Restart AN* (bit 10) is set to 1b during nominal operation. The *Go Disconnect* (bit 5) is set to 1b and *Restart AN* (bit 10) is cleared to 0b during periodic power down sequence.
- All reserved fields and read-only fields are set to 0b (bits 1, 3, 4, 8, 9 and 15:11)

1. Not applicable to the **82567**.

11.5.3 NVM Extended Configuration Structure

The NVM Extended Configuration area is located in consecutive words in the NVM.

The Extended Configuration area is partitioned in the following order:

- PHY Extended Configuration registers

Note: Treat as read only and must be programmed for normal operation.

12.0 Detailed Example Code

12.1 Introduction

This appendix provides detailed example code for the following functions:

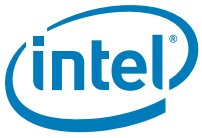
- Multicast hash
- EEPROM readiness

Note: The example code provided in this section is for illustrative purposes only. Although similar to other code, it is not definitive.

12.2 Multicast Hash

This example code explains how to first determine a hash value and then use the resultant to set the MTA register.

```
void
e1000_mta_set(struct e1000_hw *hw,
              uint32_t hash_value)
{
    uint32_t hash_bit, hash_reg;
    uint32_t mta;
    uint32_t temp;

    /* The MTA is a register array of 128 32-bit registers.
     * It is treated like an array of 4096 bits. Bit
     * BitArray[hash_value]needs to be set. Determine what
     * register the bit is in, read it, OR in the new bit, then write
     * back the new value.  The register is determined by the
     * upper seven bits of the hash value and the bit within that
     * register are determined by the lower five bits of the value.
     */
    hash_reg = (hash_value >> 5) & 0x1F;
    hash_bit = hash_value & 0x1F;
```



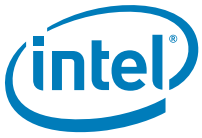
```
    mta = E1000_READ_REG_ARRAY(MTA, hash_reg);

    mta |= (1 << hash_bit);

    E1000_WRITE_REG_ARRAY( MTA, hash_reg, mta);
    E1000_WRITE_FLUSH();
}

/*********************************************************************************
 * Hashes an address to determine its location in the multicast table
 *
 * mc_filter_type - The type of the filter.  See below for details.
 * mc_addr - the multicast address to hash
 *********************************************************************************/
uint32_t
e1000_hash_mc_addr(uint32_t mc_filter_type,
                   uint8_t *mc_addr)
{
    uint32_t hash_value = 0;

    /* The portion of the address that is used for the hash table is
     * determined by the mc_filter_type setting.
     */
    switch (mc_filter_type) {
    /* [0] [1] [2] [3] [4] [5]
     * 01  AA  00  12  34  56
     * LSB                 MSB
     */
    case 0:
            /* [47:38] i.e. 0x158 for above example address */
        hash_value = ((mc_addr[4] >> 6) | (((uint16_t) mc_addr[5]) << 2));
        break;
    case 1:
```

```
            /* [46:37] i.e. 0x2B1 for above example address */
        hash_value = ((mc_addr[4] >> 5) | (((uint16_t) mc_addr[5]) << 3));
        break;
    case 2:
            /*[45:36] i.e. 0x163 for above example address */
        hash_value = ((mc_addr[4] >> 4) | (((uint16_t) mc_addr[5]) << 4));
        break;
    case 3:
            /* [43:34] i.e. 0x18D for above example address */
        hash_value = ((mc_addr[4] >> 2) | (((uint16_t) mc_addr[5]) << 6));
        break;
    }

    hash_value &= 0x3FF;

    return hash_value;
}
```

12.3 EEPROM Readiness

This detailed example code provides EEPROM/NVM/Flash pre-configuration, startup, and initialization instructions.

```
    case e1000_ich8lan:
    {
        int32_t  i = 0;
        uint32_t flash_size = E1000_READ_ICH8_REG(hw, ICH8_FLASH_GFPREG);

        eeprom->type = e1000_eeprom_ich8;
        eeprom->use_eerd = FALSE;
        eeprom->use_eewr = FALSE;
        eeprom->word_size = E1000_SHADOW_RAM_WORDS;

        /* Zero the shadow RAM structure. But don't load it from NVM
```

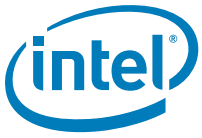


```
         * so as to save time for driver init */
        if (hw->eeprom_shadow_ram != NULL) {
            for (i = 0; i < E1000_SHADOW_RAM_WORDS; i++) {
                hw->eeprom_shadow_ram[i].modified = FALSE;
                hw->eeprom_shadow_ram[i].eeprom_word = 0xFFFF;
            }
        }

        hw->flash_base_addr = (flash_size & ICH8_GFPREG_BASE_MASK) *
                              ICH8_FLASH_SECTOR_SIZE;

        hw->flash_bank_size = ((flash_size >> 16) & ICH8_GFPREG_BASE_MASK) + 1;
        hw->flash_bank_size -= (flash_size & ICH8_GFPREG_BASE_MASK);
        hw->flash_bank_size *= ICH8_FLASH_SECTOR_SIZE;
        hw->flash_bank_size /= 2 * sizeof(uint16_t);

        break;
    }
```

After changes have been made to the shadow RAM, it must be written out. Following is some example code for this function:

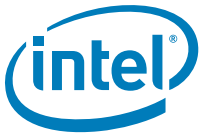
```
/*****************************************************************************
 * Flushes the cached EEPROM to NVM. This is done by saving the modified values
 * in the EEPROM cache and the non-modified values in the currently active bank
 * to the new bank.
 *
 * hw - structure pointing to several variables that store state.
 * offset - offset of the word in the EEPROM to read
 * data - word read from the EEPROM
 * words - number of words to read
 *****************************************************************************/
int32_t
e1000_commit_shadow_ram(struct e1000_hw *hw)
{
```

```
uint32_t attempts = 100000;
uint32_t eecd = 0;
uint32_t flop = 0;
uint32_t i = 0;
int32_t error = E1000_SUCCESS;
uint32_t old_bank_offset = 0;
uint32_t new_bank_offset = 0;
uint32_t sector_retries = 0;
uint8_t low_byte = 0;
uint8_t high_byte = 0;
uint8_t temp_byte = 0;
boolean_t sector_write_failed = FALSE;

if (mac_type == e1000_ich8lan && hw->eeprom_shadow_ram != NULL) {
    /* Programmers need to write to the opposite bank so if on bank 1,
     * write to bank 0 etc.  Programmers also need to erase the segment that
     * is going to be written */
    if (!(E1000_READ_REG(EECD) & E1000_EECD_SEC1VAL)) {
        new_bank_offset = hw->flash_bank_size * 2;
        old_bank_offset = 0;
        e1000_erase_ich8_4k_segment(hw, 1);
    } else {
        old_bank_offset = hw->flash_bank_size * 2;
        new_bank_offset = 0;
        e1000_erase_ich8_4k_segment(hw, 0);
    }

    do {
        sector_write_failed = FALSE;
        /* Loop for every byte in the shadow RAM,
         * which is in units of words. */
        for (i = 0; i < E1000_SHADOW_RAM_WORDS; i++) {
            /* Determine whether to write the value stored
             * in the other NVM bank or a modified value stored
```



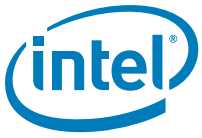
```
     * in the shadow RAM */
    if (hw->eeprom_shadow_ram[i].modified == TRUE) {
        low_byte = (uint8_t)hw->eeprom_shadow_ram[i].eeprom_word;
        e1000_read_ich8_byte(hw, (i << 1) + old_bank_offset,
                             &temp_byte);
        usec_delay(100);
        error = e1000_verify_write_ich8_byte(hw,
                                     (i << 1) + new_bank_offset,
                                     low_byte);
        if (error != E1000_SUCCESS)
            sector_write_failed = TRUE;
        high_byte =
            (uint8_t)(hw->eeprom_shadow_ram[i].eeprom_word >> 8);
        e1000_read_ich8_byte(hw, (i << 1) + old_bank_offset + 1,
                             &temp_byte);
        usec_delay(100);
    } else {
        e1000_read_ich8_byte(hw, (i << 1) + old_bank_offset,
                             &low_byte);
        usec_delay(100);
        error = e1000_verify_write_ich8_byte(hw,
                    (i << 1) + new_bank_offset, low_byte);
        if (error != E1000_SUCCESS)
            sector_write_failed = TRUE;
        e1000_read_ich8_byte(hw, (i << 1) + old_bank_offset + 1,
                             &high_byte);
    }

    /* If the word is 0x13, then make sure the signature bits
     * (15:14) are 11b until the commit has completed.
     * This enables programmers to write 10b which indicates the
     * signature is valid.  Programmers need to do this after the write
     * has completed so that they don't mark the segment valid
     * while the write is still in progress */
```

```
    if (i == E1000_ICH8_NVM_SIG_WORD)
        high_byte = E1000_ICH8_NVM_SIG_MASK | high_byte;

    error = e1000_verify_write_ich8_byte(hw,
                (i << 1) + new_bank_offset + 1, high_byte);
    if (error != E1000_SUCCESS)
        sector_write_failed = TRUE;

    if (sector_write_failed == FALSE) {
        /* Clear the now not used entry in the cache */
        hw->eeprom_shadow_ram[i].modified = FALSE;
        hw->eeprom_shadow_ram[i].eeprom_word = 0xFFFF;
    }
}

/* Don't write to the segment valid bits if sector
 * programming failed. */
if (sector_write_failed == FALSE) {
    /* Finally, validate the new segment by setting bit 15:14
     * to 10b in word 0x13. This can be done without an
     * erase as well since these bits are 11 to start with
     * and we need to change bit 14 to 0b */
    e1000_read_ich8_byte(hw,
        E1000_ICH8_NVM_SIG_WORD * 2 + 1 + new_bank_offset,
        &high_byte);
    high_byte &= 0xBF;
    error = e1000_verify_write_ich8_byte(hw,
                E1000_ICH8_NVM_SIG_WORD * 2 + 1 + new_bank_offset,
                high_byte);
    if (error != E1000_SUCCESS)
        sector_write_failed = TRUE;

    /* And invalidate the previously valid segment by setting
     * its signature word (0x13) high_byte to 0b. This can be
```



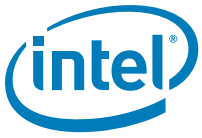
```
                 * done without an erase because flash erase sets all bits
                 * to 1's. We can write 1's to 0's without an erase */
                error = e1000_verify_write_ich8_byte(hw,
                           E1000_ICH8_NVM_SIG_WORD * 2 + 1 + old_bank_offset,
                           0);
                if (error != E1000_SUCCESS)
                    sector_write_failed = TRUE;
            }
        } while (++sector_retries < 10 && sector_write_failed == TRUE);
    }

    return error;
}

/*****************************************************************************
 * Reads a 16-bit word or words from the EEPROM using the ICH8's flash access
 * register.
 *
 * hw - Structure containing variables accessed by shared code
 * offset - offset of word in the EEPROM to read
 * data - word read from the EEPROM
 * words - number of words to read
 *****************************************************************************/
int32_t
e1000_read_eeprom_ich8(struct e1000_hw *hw, uint16_t offset, uint16_t words,
                       uint16_t *data)
{
    int32_t  error = E1000_SUCCESS;
    uint32_t flash_bank = 0;
    uint32_t act_offset = 0;
    uint32_t bank_offset = 0;
    uint16_t word = 0;
    uint16_t i = 0;
```

```
    /* Programmers need to know which is the valid flash bank.  In the event
     * that they didn't allocate eeprom_shadow_ram, they might not be
     * managing flash_bank.  So it cannot be trusted and needs
     * to be updated with each read.
     */
    /* Value of bit 22 corresponds to the flash bank currently on. */
    flash_bank = (E1000_READ_REG(hw, EECD) & E1000_EECD_SEC1VAL) ? 1 : 0;

    /* Adjust offset appropriately if on bank 1 - adjust for word size */
    bank_offset = flash_bank * (hw->flash_bank_size * 2);

    error = e1000_get_software_flag(hw);
    if (error != E1000_SUCCESS)
        return error;

    for (i = 0; i < words; i++) {
        if (hw->eeprom_shadow_ram != NULL &&
            hw->eeprom_shadow_ram[offset+i].modified == TRUE) {
            data[i] = hw->eeprom_shadow_ram[offset+i].eeprom_word;
        } else {
            /* The NVM part needs a byte offset, hence * 2 */
            act_offset = bank_offset + ((offset + i) * 2);
            error = e1000_read_ich8_word(hw, act_offset, &word);
            if (error != E1000_SUCCESS)
                break;
            data[i] = word;
        }
    }

    e1000_release_software_flag(hw);

    return error;
}
```



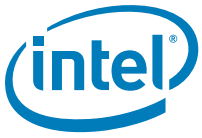
```
/*****************************************************************************
 * Writes a 16 bit word or words to the EEPROM using the ICH8's flash access
 * register.  Actually, writes are written to the shadow ram cache in the hw
 * structure hw->e1000_shadow_ram.  e1000_commit_shadow_ram flushes this to
 * the NVM, which occurs when the NVM checksum is updated.
 *
 * hw - Structure containing variables accessed by shared code
 * offset - offset of word in the EEPROM to write
 * words - number of words to write
 * data - words to write to the EEPROM
 *****************************************************************************/
int32_t
e1000_write_eeprom_ich8(struct e1000_hw *hw, uint16_t offset, uint16_t words,
                        uint16_t *data)
{
    uint32_t i = 0;
    int32_t error = E1000_SUCCESS;

    error = e1000_get_software_flag(hw);
    if (error != E1000_SUCCESS)
        return error;

    /* A driver can write to the NVM only if it has eeprom_shadow_ram
     * allocated.  Subsequent reads to the modified words are read from
     * this cached structure as well.  Writes only go into this
     * cached structure unless it's followed by a call to
     * e1000_update_eeprom_checksum() where it commits the changes
     * and clears the "modified" field.
     */
    if (hw->eeprom_shadow_ram != NULL) {
        for (i = 0; i < words; i++) {
            if ((offset + i) < E1000_SHADOW_RAM_WORDS) {
                hw->eeprom_shadow_ram[offset+i].modified = TRUE;
                hw->eeprom_shadow_ram[offset+i].eeprom_word = data[i];
```

```
            } else {
                error = -E1000_ERR_EEPROM;
                break;
            }
        }
    } else {
        /* Drivers have the option to not allocate eeprom_shadow_ram as long
         * as they don't perform any NVM writes.  An attempt in doing so
         * results in this error.
         */
        error = -E1000_ERR_EEPROM;
    }

    e1000_release_software_flag(hw);

    return error;
}

/******************************************************************************
 * This function does initial flash setup so that a new read/write/erase cycle
 * can be started.
 *
 * hw - The pointer to the hw structure
 ****************************************************************************/
int32_t
e1000_ich8_cycle_init(struct e1000_hw *hw)
{
    union ich8_hws_flash_status hsfsts;
    int32_t error = E1000_ERR_EEPROM;
    int32_t i     = 0;

    DEBUGFUNC("e1000_ich8_cycle_init");

    hsfsts.regval = E1000_READ_ICH8_REG16(hw, ICH8_FLASH_HSFSTS);
```



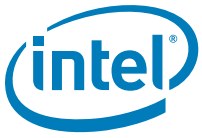
```
/* Maybe check the Flash Des Valid bit in Hw status */
if (hsfsts.hsf_status.fldesvalid == 0) {
    DEBUGOUT("Flash descriptor invalid.  SW Sequencing must be used.");
    return error;
}

/* Clear FCERR in Hw status by writing 1 */
/* Clear DAEL in Hw status by writing a 1 */
hsfsts.hsf_status.flcerr = 1;
hsfsts.hsf_status.dael = 1;

E1000_WRITE_ICH8_REG16(hw, ICH8_FLASH_HSFSTS, hsfsts.regval);

/* Either programmers should have a hardware SPI cycle in progress bit to check
 * against, in order to start a new cycle or FDONE bit should be changed
 * in the hardware so that it is 1 after harware reset, which can then be
 * used as an indication whether a cycle is in progress or has been
 * completed. Programmers should also have some software semaphore mechanism to
 * guard FDONE or the cycle in progress bit so that two threads access to
 * those bits can be sequentiallized or a way so that 2 threads don't
 * start the cycle at the same time */

if (hsfsts.hsf_status.flcinprog == 0) {
    /* There is no cycle running at present, so we can start a cycle */
    /* Begin by setting Flash Cycle Done. */
    hsfsts.hsf_status.flcdone = 1;
    E1000_WRITE_ICH8_REG16(hw, ICH8_FLASH_HSFSTS, hsfsts.regval);
    error = E1000_SUCCESS;
} else {
    /* otherwise poll for sometime so the current cycle has a chance
     * to end before giving up. */
    for (i = 0; i < ICH8_FLASH_COMMAND_TIMEOUT; i++) {
        hsfsts.regval = E1000_READ_ICH8_REG16(hw, ICH8_FLASH_HSFSTS);
```

```
            if (hsfsts.hsf_status.flcinprog == 0) {
                error = E1000_SUCCESS;
                break;
            }
            usec_delay(1);
        }
        if (error == E1000_SUCCESS) {
            /* Successful in waiting for previous cycle to timeout,
             * now set the Flash Cycle Done. */
            hsfsts.hsf_status.flcdone = 1;
            E1000_WRITE_ICH8_REG16(hw, ICH8_FLASH_HSFSTS, hsfsts.regval);
        } else {
            DEBUGOUT("Flash controller busy, cannot get access");
        }
    }
    return error;
}

/*****************************************************************************
 * This function starts a flash cycle and waits for its completion
 *
 * hw - The pointer to the hw structure
 ****************************************************************************/
int32_t
e1000_ich8_flash_cycle(struct e1000_hw *hw, uint32_t timeout)
{
    union ich8_hws_flash_ctrl hsflctl;
    union ich8_hws_flash_status hsfsts;
    int32_t error = E1000_ERR_EEPROM;
    uint32_t i = 0;

    /* Start a cycle by writing 1 in Flash Cycle Go in Hw Flash Control */
    hsflctl.regval = E1000_READ_ICH8_REG16(hw, ICH8_FLASH_HSFCTL);
    hsflctl.hsf_ctrl.flcgo = 1;
```



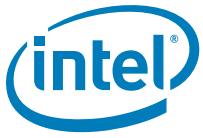
```
    E1000_WRITE_ICH8_REG16(hw, ICH8_FLASH_HSFCTL, hsflctl.regval);

    /* wait till FDONE bit is set to 1 */
    do {
        hsfsts.regval = E1000_READ_ICH8_REG16(hw, ICH8_FLASH_HSFSTS);
        if (hsfsts.hsf_status.flcdone == 1)
            break;
        usec_delay(1);
        i++;
    } while (i < timeout);
    if (hsfsts.hsf_status.flcdone == 1 && hsfsts.hsf_status.flcerr == 0) {
        error = E1000_SUCCESS;
    }
    return error;
}

/******************************************************************************
 * Reads a byte or word from the NVM using the ICH8 flash access registers.
 *
 * hw - The pointer to the hw structure
 * index - The index of the byte or word to read.
 * size - Size of data to read, 1=byte 2=word
 * data - Pointer to the word to store the value read.
 *****************************************************************************/
int32_t
e1000_read_ich8_data(struct e1000_hw *hw, uint32_t index,
                     uint32_t size, uint16_t* data)
{
    union ich8_hws_flash_status hsfsts;
    union ich8_hws_flash_ctrl hsflctl;
    uint32_t flash_linear_address;
    uint32_t flash_data = 0;
    int32_t error = -E1000_ERR_EEPROM;
    int32_t count = 0;
```

```
DEBUGFUNC("e1000_read_ich8_data");

if (size < 1  || size > 2 || data == 0x0 ||
    index > ICH8_FLASH_LINEAR_ADDR_MASK)
    return error;

flash_linear_address = (ICH8_FLASH_LINEAR_ADDR_MASK & index) +
                       hw->flash_base_addr;

do {
    usec_delay(1);
    /* Steps */
    error = e1000_ich8_cycle_init(hw);
    if (error != E1000_SUCCESS)
        break;

    hsflctl.regval = E1000_READ_ICH8_REG16(hw, ICH8_FLASH_HSFCTL);
    /* 0b/1b corresponds to 1 or 2 byte size, respectively. */
    hsflctl.hsf_ctrl.fldbcount = size - 1;
    hsflctl.hsf_ctrl.flcycle = ICH8_CYCLE_READ;
    E1000_WRITE_ICH8_REG16(hw, ICH8_FLASH_HSFCTL, hsflctl.regval);

    /* Write the last 24 bits of index into Flash Linear address field in
     * Flash Address */
    /* Maybe check the index against the size of flash */

    E1000_WRITE_ICH8_REG(hw, ICH8_FLASH_FADDR, flash_linear_address);

    error = e1000_ich8_flash_cycle(hw, ICH8_FLASH_COMMAND_TIMEOUT);

    /* Check if FCERR is set to 1, if set to 1, clear it and try the whole
     * sequence a few more times, else read in (shift in) the Flash Data0,
     * the order is least significant byte first msb to lsb */
```



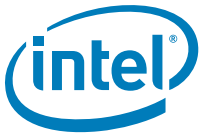
```
            if (error == E1000_SUCCESS) {
                flash_data = E1000_READ_ICH8_REG(hw, ICH8_FLASH_FDATA0);
                if (size == 1) {
                    *data = (uint8_t)(flash_data & 0x000000FF);
                } else if (size == 2) {
                    *data = (uint16_t)(flash_data & 0x0000FFFF);
                }
                break;
            } else {
                /* If programmers get to this point, things are probably
                 * not correct, but if the error condition is detected,
                 * give it another try...ICH8_FLASH_CYCLE_REPEAT_COUNT times.
                 */
                hsfsts.regval = E1000_READ_ICH8_REG16(hw, ICH8_FLASH_HSFSTS);
                if (hsfsts.hsf_status.flcerr == 1) {
                    /* Repeat for some time before giving up. */
                    continue;
                } else if (hsfsts.hsf_status.flcdone == 0) {
                    DEBUGOUT("Timeout error - flash cycle did not complete.");
                    break;
                }
            }
        } while (count++ < ICH8_FLASH_CYCLE_REPEAT_COUNT);

        return error;
    }

    /*****************************************************************************
     * Writes One /two bytes to the NVM using the ICH8 flash access registers.
     *
     * hw - The pointer to the hw structure
     * index - The index of the byte/word to read.
     * size - Size of data to read, 1=byte 2=word
     * data - The byte(s) to write to the NVM.
```

```
 *****************************************************************************/
int32_t
e1000_write_ich8_data(struct e1000_hw *hw, uint32_t index, uint32_t size,
                      uint16_t data)
{
    union ich8_hws_flash_status hsfsts;
    union ich8_hws_flash_ctrl hsflctl;
    uint32_t flash_linear_address;
    uint32_t flash_data = 0;
    int32_t error = -E1000_ERR_EEPROM;
    int32_t count = 0;

    DEBUGFUNC("e1000_write_ich8_data");

    if (size < 1  || size > 2 || data > size * 0xff ||
        index > ICH8_FLASH_LINEAR_ADDR_MASK)
        return error;

    flash_linear_address = (ICH8_FLASH_LINEAR_ADDR_MASK & index) +
                           hw->flash_base_addr;

    do {
        usec_delay(1);
        /* Steps */
        error = e1000_ich8_cycle_init(hw);
        if (error != E1000_SUCCESS)
            break;

        hsflctl.regval = E1000_READ_ICH8_REG16(hw, ICH8_FLASH_HSFCTL);
        /* 0b/1b corresponds to 1 or 2 byte size, respectively. */
        hsflctl.hsf_ctrl.fldbcount = size -1;
        hsflctl.hsf_ctrl.flcycle = ICH8_CYCLE_WRITE;
        E1000_WRITE_ICH8_REG16(hw, ICH8_FLASH_HSFCTL, hsflctl.regval);
```



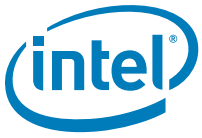
```
        /* Write the last 24 bits of index into Flash Linear address field in
         * Flash Address */
        E1000_WRITE_ICH8_REG(hw, ICH8_FLASH_FADDR, flash_linear_address);

        if (size == 1)
            flash_data = (uint32_t)data & 0x00FF;
        else
            flash_data = (uint32_t)data;

        E1000_WRITE_ICH8_REG(hw, ICH8_FLASH_FDATA0, flash_data);

        /* check if FCERR is set to 1 , if set to 1, clear it and try the whole
         * sequence a few more times else done */
        error = e1000_ich8_flash_cycle(hw, ICH8_FLASH_COMMAND_TIMEOUT);
        if (error == E1000_SUCCESS) {
            break;
        } else {
            /* If programmers get to this point, things are probably not correct,
             * but if the error condition is detected, give
             * it another try...ICH8_FLASH_CYCLE_REPEAT_COUNT times.
             */
            hsfsts.regval = E1000_READ_ICH8_REG16(hw, ICH8_FLASH_HSFSTS);
            if (hsfsts.hsf_status.flcerr == 1) {
                /* Repeat for some time before giving up. */
                continue;
            } else if (hsfsts.hsf_status.flcdone == 0) {
                DEBUGOUT("Timeout error - flash cycle did not complete.");
                break;
            }
        }
    } while (count++ < ICH8_FLASH_CYCLE_REPEAT_COUNT);

    return error;
}
```

```
/*****************************************************************************
 * Reads a single byte from the NVM using the ICH8 flash access registers.
 *
 * hw - pointer to e1000_hw structure
 * index - The index of the byte to read.
 * data - Pointer to a byte to store the value read.
 *****************************************************************************/
int32_t
e1000_read_ich8_byte(struct e1000_hw *hw, uint32_t index, uint8_t* data)
{
    int32_t status = E1000_SUCCESS;
    uint16_t word = 0;

    status = e1000_read_ich8_data(hw, index, 1, &word);
    if (status == E1000_SUCCESS) {
        *data = (uint8_t)word;
    }

    return status;
}

/*****************************************************************************
 * Writes a single byte to the NVM using the ICH8 flash access registers.
 * Performs verification by reading back the value and then going through
 * a retry algorithm before giving up.
 *
 * hw - pointer to e1000_hw structure
 * index - The index of the byte to write.
 * byte - The byte to write to the NVM.
 *****************************************************************************/
int32_t
e1000_verify_write_ich8_byte(struct e1000_hw *hw, uint32_t index, uint8_t byte)
{
```



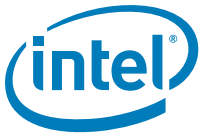
```
    int32_t error = E1000_SUCCESS;
    int32_t program_retries;
    uint8_t temp_byte;

    e1000_write_ich8_byte(hw, index, byte);
    usec_delay(100);

    for (program_retries = 0; program_retries < 100; program_retries++) {
        e1000_read_ich8_byte(hw, index, &temp_byte);
        if (temp_byte == byte)
            break;
        usec_delay(10);
        e1000_write_ich8_byte(hw, index, byte);
        usec_delay(100);
    }
    if (program_retries == 100)
        error = E1000_ERR_EEPROM;

    return error;
}

/******************************************************************************
 * Writes a single byte to the NVM using the ICH8 flash access registers.
 *
 * hw - pointer to e1000_hw structure
 * index - The index of the byte to read.
 * data - The byte to write to the NVM.
 *****************************************************************************/
int32_t
e1000_write_ich8_byte(struct e1000_hw *hw, uint32_t index, uint8_t data)
{
    int32_t status = E1000_SUCCESS;
    uint16_t word = (uint16_t)data;
```

```
    status = e1000_write_ich8_data(hw, index, 1, word);

    return status;
}

/****************************************************************************
 * Reads a word from the NVM using the ICH8 flash access registers.
 *
 * hw - pointer to e1000_hw structure
 * index - The starting byte index of the word to read.
 * data - Pointer to a word to store the value read.
 ****************************************************************************/
int32_t
e1000_read_ich8_word(struct e1000_hw *hw, uint32_t index, uint16_t *data)
{
    int32_t status = E1000_SUCCESS;
    status = e1000_read_ich8_data(hw, index, 2, data);
    return status;
}

/****************************************************************************
 * Writes a word to the NVM using the ICH8 flash access registers.
 *
 * hw - pointer to e1000_hw structure
 * index - The starting byte index of the word to read.
 * data - The word to write to the NVM.
 ****************************************************************************/
int32_t
e1000_write_ich8_word(struct e1000_hw *hw, uint32_t index, uint16_t data)
{
    int32_t status = E1000_SUCCESS;
    status = e1000_write_ich8_data(hw, index, 2, data);
    return status;
}
```



```
/*****************************************************************************
 * Erases the bank specified. Each bank is a 4k block. Segments are 0 based.
 * segment N is 4096 * N + flash_reg_addr.
 *
 * hw - pointer to e1000_hw structure
 * segment - 0 for first segment, 1 for second segment, etc.
 *****************************************************************************/
int32_t
e1000_erase_ich8_4k_segment(struct e1000_hw *hw, uint32_t segment)
{
    union ich8_hws_flash_status hsfsts;
    union ich8_hws_flash_ctrl hsflctl;
    uint32_t flash_linear_address;
    int32_t  count = 0;
    int32_t  error = E1000_ERR_EEPROM;
    int32_t  iteration, seg_size;
    int32_t  sector_size;
    int32_t  j = 0;
    int32_t  error_flag = 0;

    hsfsts.regval = E1000_READ_ICH8_REG16(hw, ICH8_FLASH_HSFSTS);

    /* Determine HW Sector size: Read BERASE bits of Hw flash Status register */
    /* 00: The Hw sector is 256 bytes, hence we need to erase 16
     *     consecutive sectors.  The start index for the nth Hw sector can be
     *     calculated as = segment * 4096 + n * 256
     * 01: The Hw sector is 4K bytes, hence we need to erase 1 sector.
     *     The start index for the nth Hw sector can be calculated
     *     as = segment * 4096
     * 10: Error condition
     * 11: The Hw sector size is much bigger than the size asked to
     *     erase...error condition */
    if (hsfsts.hsf_status.berasesz == 0x0) {
```

```
    /* Hw sector size 256 */
    sector_size = seg_size = ICH8_FLASH_SEG_SIZE_256;
    iteration = ICH8_FLASH_SECTOR_SIZE / ICH8_FLASH_SEG_SIZE_256;
} else if (hsfsts.hsf_status.berasesz == 0x1) {
    sector_size = seg_size = ICH8_FLASH_SEG_SIZE_4K;
    iteration = 1;
} else if (hsfsts.hsf_status.berasesz == 0x3) {
    sector_size = seg_size = ICH8_FLASH_SEG_SIZE_64K;
    iteration = 1;
} else {
    return error;
}

for (j = 0; j < iteration ; j++) {
    do {
        count++;
        /* Steps */
        error = e1000_ich8_cycle_init(hw);
        if (error != E1000_SUCCESS) {
            error_flag = 1;
            break;
        }

        /* Write a value 11 (block Erase) in Flash Cycle field in Hw flash
         * Control */
        hsflctl.regval = E1000_READ_ICH8_REG16(hw, ICH8_FLASH_HSFCTL);
        hsflctl.hsf_ctrl.flcycle = ICH8_CYCLE_ERASE;
        E1000_WRITE_ICH8_REG16(hw, ICH8_FLASH_HSFCTL, hsflctl.regval);

        /* Write the last 24 bits of an index within the block into Flash
         * Linear address field in Flash Address.  This probably needs to
         * be calculated here based off the on-chip segment size and the
         * software segment size assumed (4K) */
        /* TBD */
```



```
                flash_linear_address = segment * sector_size + j * seg_size;
                flash_linear_address &= ICH8_FLASH_LINEAR_ADDR_MASK;
                flash_linear_address += hw->flash_base_addr;

                E1000_WRITE_ICH8_REG(hw, ICH8_FLASH_FADDR, flash_linear_address);

                error = e1000_ich8_flash_cycle(hw, 1000000);
                /* Check if FCERR is set to 1.  If 1, clear it and try the whole
                 * sequence a few more times else Done */
                if (error == E1000_SUCCESS) {
                    break;
                } else {
                    hsfsts.regval = E1000_READ_ICH8_REG16(hw, ICH8_FLASH_HSFSTS);
                    if (hsfsts.hsf_status.flcerr == 1) {
                        /* repeat for some time before giving up */
                        continue;
                    } else if (hsfsts.hsf_status.flcdone == 0) {
                        error_flag = 1;
                        break;
                    }
                }
            } while ((count < ICH8_FLASH_CYCLE_REPEAT_COUNT) && !error_flag);
            if (error_flag == 1)
                break;
        }
        if (error_flag != 1)
            error = E1000_SUCCESS;
        return error;
}
```

Note that after the commitment of the Shadow RAM image, the EEPROM reset must occur for the data to be loaded into shadow RAM function. Otherwise, it is read after the next device or bus level reset. For continuity, it is recommended that the EEPROM reset be done every time.

Note: This page intentionally left blank.