Section 35. Ethernet Controller

HIGHLIGHTS

This section of the manual contains the following topics:

Note:	This family reference manual section is meant to serve as a complement to device data sheets. Depending on the device variant, this manual section may not apply to all PIC32 devices. Please consult the note at the beginning of the **"Ethernet Controller"** chapter in the current device data sheet to check whether this document supports the device you are using. Device data sheets and family reference manual sections are available for download from the Microchip Worldwide Web site at: http://www.microchip.com

35.1 INTRODUCTION

The Ethernet Controller is a bus master module that interfaces with an off-chip PHY in order to implement a complete Ethernet node in a system.

Following are some of the key features of this module:

- Supports 10/100 Mbps data transfer rates
- Supports Full-Duplex and Half-Duplex operation
- Supports Reduced Media Independent Interface (RMII) and Media Independent Interface (MII) PHY interface
- Supports MII Management (MIIM) PHY Management interface
- Supports both manual and automatic flow control
- Supports Auto-MDIX and enabled PHYs
- RAM descriptor based DMA operation for both receive and transmit path
- Fully configurable interrupts
- Configurable receive packet filtering
  - CRC Check
  - 64-byte Pattern Match
  - Broadcast, Multicast and Unicast packets
  - Magic Packet™
  - 64-bit Hash Table
  - Runt Packet
- Supports Packet Payload Checksum calculation
- Supports various hardware statistics counters

35.2 ETHERNET CONTROLLER OVERVIEW

The Ethernet Controller provides the modules needed to implement a 10/100 Mbps Ethernet node using an external PHY chip. In order to off-load the CPU from moving packet data to and from the module, internal descriptor-based DMA engines are included in the controller.

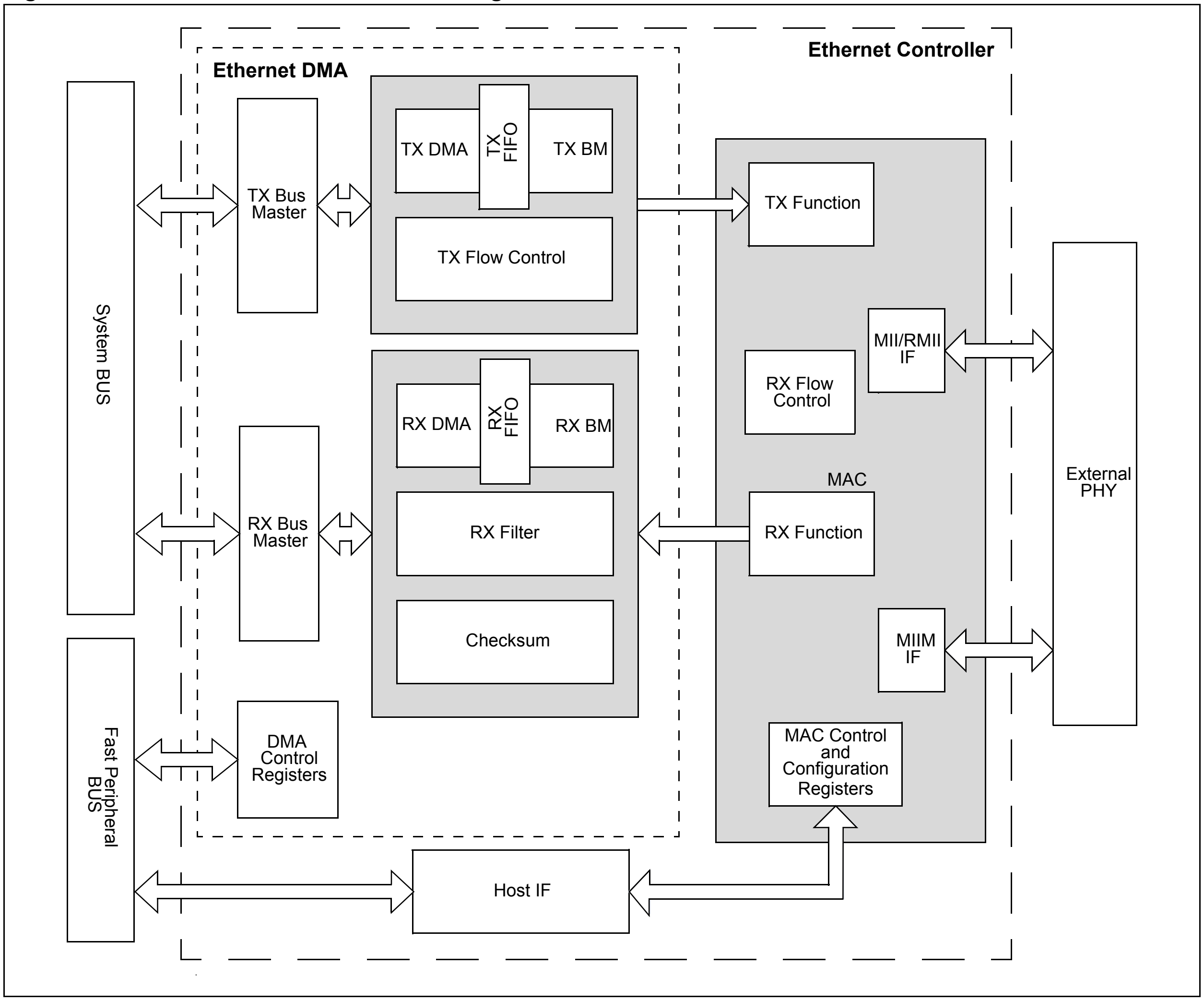
The Ethernet Controller consists of the following modules:

- Media Access Control (MAC) block: Responsible for implementing the MAC functions of the Ethernet IEEE 802.3 Specification
- Flow Control (FC) block: Responsible for control of the transmission of PAUSE frames. Reception of PAUSE frames is handled within the MAC.
- RX Filter (RXF) block: This module performs filtering on every receive packet to determine whether each packet should be accepted or rejected
- TX DMA/TX BM Engine: The TX DMA and TX Buffer Management engines perform data transfers from the memory (using descriptor tables) to the MAC Transmit Interface
- RX DMA/RX BM Engine: The RX DMA and RX Buffer Management engines transfer receive packets from the MAC to the memory (using descriptor tables)

Figure 35-1 shows a block diagram of the Ethernet Controller.

> **Note:** For detailed explanations of Ethernet operation, refer to AN1120 "*Ethernet Theory of Operation*" (DS01120), which is available from the Microchip web site (www.microchip.com), and the IEEE 802.3 Specification (www.ieee.org).

Figure 35-1: Ethernet Controller Block Diagram



35.3 STATUS AND CONTROL REGISTERS

The Ethernet Controller module consists of the following Special Function Registers (SFRs):

Controller and DMA Engine Configuration/Status Registers:

- ETHCON1: Ethernet Controller Control 1 Register
- ETHCON2: Ethernet Controller Control 2 Register[1,2]
- ETHTXST: Ethernet Controller TX Packet Descriptor Start Address Register[1,2]
- ETHRXST: Ethernet Controller RX Packet Descriptor Start Address Register[1,2]
- ETHIEN: Ethernet Controller Interrupt Enable Register
- ETHIRQ: Ethernet Controller Interrupt Request Register
- ETHSTAT: Ethernet Controller Status Register

RX Filtering Configuration Registers:

- ETHRXFC: Ethernet Controller Receive Filter Configuration Register[1,2]
- ETHHT0: Ethernet Controller Hash Table 0 Register[1,2]
- ETHHT1: Ethernet Controller Hash Table 1 Register[1,2]
- ETHPMM0: Ethernet Controller Pattern Match Mask 0 Register[1,2]
- ETHPMM1: Ethernet Controller Pattern Match Mask 1 Register[1,2]
- ETHPMCS: Ethernet Controller Pattern Match Checksum Register[1,2]
- ETHPMO: Ethernet Controller Pattern Match Offset Register[1,2]

Flow Control Configuring Register:

- ETHRXWM: Ethernet Controller Receive Watermarks Register[1]

Ethernet Statistics Registers:

- ETHRXOVFLOW: Ethernet Controller Receive Overflow Statistics Register[1,2]
- ETHFRMTXOK: Ethernet Controller Frames Transmitted OK Statistics Register[1,2]
- ETHSCOLFRM: Ethernet Controller Single Collision Frames Statistics Register[1,2]
- ETHMCOLFRM: Ethernet Controller Multiple Collision Frames Statistics Register[1,2]
- ETHFRMRXOK: Ethernet Controller Frames Received OK Statistics Register[1,2]
- ETHFCSERR: Ethernet Controller Frame Check Sequence Error Statistics Register[1,2]
- ETHALGNERR: Ethernet Controller Alignment Errors Statistics Register[1,2]

MAC Configuration Registers:

- EMAC1CFG1: Ethernet Controller MAC Configuration 1 Register[1]
- EMAC1CFG2: Ethernet Controller MAC Configuration 2 Register[1]
- EMAC1IPGT: Ethernet Controller MAC Back-to-Back Interpacket Gap Register[1]
- EMAC1IPGR: Ethernet Controller MAC Non-Back-to-Back Interpacket Gap Register[1]
- EMAC1CLRT: Ethernet Controller MAC Collision Window/Retry Limit Register[1]
- EMAC1MAXF: Ethernet Controller MAC Maximum Frame Length Register[1,2]
- EMAC1SUPP: Ethernet Controller MAC PHY Support Register[1,2]
- EMAC1TEST: Ethernet Controller MAC Test Register[1,2]
- EMAC1SA0: Ethernet Controller MAC Station Address 0 Register[1,2]
- EMAC1SA1: Ethernet Controller MAC Station Address 1 Register[1,2]
- EMAC1SA2: Ethernet Controller MAC Station Address 2 Register[1,2]

MII Management Registers:

- EMAC1MCFG: Ethernet Controller MAC MII Management Configuration Register[1,2]
- EMAC1MCMD: Ethernet Controller MAC MII Management Command Register[1]
- EMAC1MADR: Ethernet Controller MAC MII Management Address Register[1]
- EMAC1MWTD: Ethernet Controller MAC MII Management Write Data Register[1]
- EMAC1MRDD: Ethernet Controller MAC MII Management Read Data Register[1]
- EMAC1MIND: Ethernet Controller MAC MII Management Indicators Register[1]

Table 35-1 provides a brief summary of the Ethernet Controller registers. Corresponding registers appear after the summary, followed by a detailed description of each register.

Table 35-1: Ethernet Controller Register Summary

Name	Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
ETHCON1[1,2,3]	31:24	PTV <15:8>							
	23:16	PTV <7:0>							
	15:8	ON	—	SIDL	—	—	—	TXRTS	RXEN
	7:0	AUTOFC	—	—	MANFC	—	—	—	BUFCDEC
ETHCON2[1,2,3]	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	—	—	—	—	RXBUFSZ<6:4>		
	7:0	RXBUFSZ<3:0>				—	—	—	—
ETHTXST[1,2,3]	31:24	TXSTADDR<31:24>							
	23:16	TXSTADDR<23:16>							
	15:8	TXSTADDR<15:8>							
	7:0	TXSTADDR<7:2>						—	—
ETHRXST[1,2,3]	31:24	RXSTADDR<31:24>							
	23:16	RXSTADDR<23:16>							
	15:8	RXSTADDR<15:8>							
	7:0	RXSTADDR<7:2>						—	—
ETHHT0[1,2,3]	31:24	HT<31:24>							
	23:16	HT<23:16>							
	15:8	HT<15:8>							
	7:0	HT<7:0>							
ETHHT1[1,2,3]	31:24	HT<63:56>							
	23:16	HT<55:48>							
	15:8	HT<47:40>							
	7:0	HT<39:32>							
ETHPMM0[1,2,3]	31:24	PMM<31:24>							
	23:16	PMM<23:16>							
	15:8	PMM<15:8>							
	7:0	PMM<7:0>							
ETHPMM1[1,2,3]	31:24	PMM<63:56>							
	23:16	PMM<55:48>							
	15:8	PMM<47:40>							
	7:0	PMM<39:32>							
ETHPMCS[1,2,3]	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	PMCS<15:8>							
	7:0	PMCS<7:0>							

Legend: — = unimplemented, read as '0'.

Note 1: This register has an associated Clear register at an offset of 0x4 bytes. These registers have the same name with CLR appended to the end of the register name (e.g., ETHCON1CLR). Writing a '1' to any bit position in the Clear register will clear valid bits in the associated register. Reads from the Clear register should be ignored.

2: This register has an associated Set register at an offset of 0x8 bytes. These registers have the same name with SET appended to the end of the register name (e.g.,ETHCON1SET). Writing a '1' to any bit position in the Set register will set valid bits in the associated register. Reads from the Set register should be ignored.

3: This register has an associated Invert register at an offset of 0xC bytes. These registers have the same name with INV appended to the end of the register name (e.g., ETHCON1INV). Writing a '1' to any bit position in the Invert register will invert valid bits in the associated register. Reads from the Invert register should be ignored.

Table 35-1: Ethernet Controller Register Summary (Continued)

Name	Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
ETHPMO[1,2,3]	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	PMO<15:8>							
	7:0	PMO<7:0>							
ETHRXFC[1,2,3]	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	HTEN	MPEN	—	NOTPM	PMMODE<3:0>			
	7:0	CRCERREN	CRCOKEN	RUNTERREN	RUNTEN	UCEN	NOTMEEN	MCEN	BCEN
ETHRXWM[1,2,3]	31:24	—	—	—	—	—	—	—	—
	23:16	RXFWM<7:0>							
	15:8	—	—	—	—	—	—	—	—
	7:0	RXEWM<7:0>							
ETHIEN[1,2,3]	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	TXBUSEIE	RXBUSEIE	—	—	—	EWMARKIE	FWMARKIE
	7:0	RXDONEIE	PKTPENDIE	RXACTIE	—	TXDONEIE	TXABORTIE	RXBUFNAIE	RXOVFLWIE
ETHIRQ[1,2,3]	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	TXBUSE	RXBUSE	—	—	—	EWMARK	FWMARK
	7:0	RXDONE	PKTPEND	RXACT	—	TXDONE	TXABORT	RXBUFNA	RXOVFLW
ETHSTAT	31:24	—	—	—	—	—	—	—	—
	23:16	BUFCNT<7:0>							
	15:8	—	—	—	—	—	—	—	—
	7:0	ETHBUSY	TXBUSY	RXBUSY	—	—	—	—	—
ETHRXOVFLOW[1,2,3]	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	RXOVFLWCNT<15:8>							
	7:0	RXOVFLWCNT<7:0>							
ETHFRMTXOK[1,2,3]	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	FRMTXOKCNT<15:8>							
	7:0	FRMTXOKCNT<7:0>							
ETHSCOLFRM[1,2,3]	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	SCOLFRMCNT<15:8>							
	7:0	SCOLFRMCNT<7:0>							

Legend: — = unimplemented, read as '0'.

Note 1: This register has an associated Clear register at an offset of 0x4 bytes. These registers have the same name with CLR appended to the end of the register name (e.g., ETHCON1CLR). Writing a '1' to any bit position in the Clear register will clear valid bits in the associated register. Reads from the Clear register should be ignored.

2: This register has an associated Set register at an offset of 0x8 bytes. These registers have the same name with SET appended to the end of the register name (e.g.,ETHCON1SET). Writing a '1' to any bit position in the Set register will set valid bits in the associated register. Reads from the Set register should be ignored.

3: This register has an associated Invert register at an offset of 0xC bytes. These registers have the same name with INV appended to the end of the register name (e.g., ETHCON1INV). Writing a '1' to any bit position in the Invert register will invert valid bits in the associated register. Reads from the Invert register should be ignored.

Table 35-1: Ethernet Controller Register Summary (Continued)

Name	Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
ETHMCOLFRM(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	MCOLFRMCNT<15:8>							
	7:0	MCOLFRMCNT<7:0>							
ETHFRMRXOK(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	FRMRXOKCNT<15:8>							
	7:0	FRMRXOKCNT<7:0>							
ETHFCSERR(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	FCSERRCNT<15:8>							
	7:0	FCSERRCNT<7:0>							
ETHALGNERR(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	ALGNERRCNT<15:8>							
	7:0	ALGNERRCNT<7:0>							
EMAC1CFG1(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	SOFTRESET	SIMRESET	—	—	RESETRMCS	RESETRFUN	RESETTMCS	RESETTFUN
	7:0	—	—	—	LOOPBACK	TXPAUSE	RXPAUSE	PASSALL	RXENABLE
EMAC1CFG2(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	EXCESSDFR	BPNOBKOFF	NOBKOFF	—	—	LONGPRE	PUREPRE
	7:0	AUTOPAD	VLANPAD	PADENABLE	CRCENABLE	DELAYCRC	HUGEFRM	LENGTHCK	FULLDPLX
EMAC1IPGT (1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	—	—	—	—	—	—	—
	7:0	—	B2BIPKTGP<6:0>						
EMAC1IPGR(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	NB2BIPKTGP1<6:0>						
	7:0	—	NB2BIPKTGP2<6:0>						
EMAC1CLRT(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	—	CWINDOW<5:0>					
	7:0	—	—	—	—	RETX<3:0>			

Legend: — = unimplemented, read as '0'.

Note 1: This register has an associated Clear register at an offset of 0x4 bytes. These registers have the same name with CLR appended to the end of the register name (e.g., ETHCON1CLR). Writing a '1' to any bit position in the Clear register will clear valid bits in the associated register. Reads from the Clear register should be ignored.

2: This register has an associated Set register at an offset of 0x8 bytes. These registers have the same name with SET appended to the end of the register name (e.g.,ETHCON1SET). Writing a '1' to any bit position in the Set register will set valid bits in the associated register. Reads from the Set register should be ignored.

3: This register has an associated Invert register at an offset of 0xC bytes. These registers have the same name with INV appended to the end of the register name (e.g., ETHCON1INV). Writing a '1' to any bit position in the Invert register will invert valid bits in the associated register. Reads from the Invert register should be ignored.

Table 35-1: Ethernet Controller Register Summary (Continued)

Name	Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
EMAC1MAXF(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	MACMAXF<15:8>							
	7:0	MACMAXF<7:0>							
EMAC1SUPP(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	—	—	—	RESETRMII	—	—	SPEEDRMII
	7:0	—	—	—	—	—	—	—	—
EMAC1TEST(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	—	—	—	—	—	—	—
	7:0	—	—	—	—	—	TESTBP	TESTPAUSE	SHRTQNTA
EMAC1MCFG(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	RESETMGMT	—	—	—	—	—	—	—
	7:0	—	—	CLKSEL<3:0>				NOPRE	SCANINC
EMAC1MCMD(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	—	—	—	—	—	—	—
	7:0	—	—	—	—	—	—	SCAN	READ
EMAC1MADR(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	—	—	PHYADDR<4:0>				
	7:0	—	—	—	REGADDR<4:0>				
EMAC1MWTD(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	MWTD<15:8>							
	7:0	MWTD<7:0>							
EMAC1MRDD(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	MRDD<15:8>							
	7:0	MRDD<7:0>							
EMAC1MIND(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	—	—	—	—	—	—	—
	7:0	—	—	—	—	LINKFAIL	NOTVALID	SCAN	MIIMBUSY

Legend: — = unimplemented, read as ‘0’.

Note 1: This register has an associated Clear register at an offset of 0x4 bytes. These registers have the same name with CLR appended to the end of the register name (e.g., ETHCON1CLR). Writing a ‘1’ to any bit position in the Clear register will clear valid bits in the associated register. Reads from the Clear register should be ignored.

2: This register has an associated Set register at an offset of 0x8 bytes. These registers have the same name with SET appended to the end of the register name (e.g.,ETHCON1SET). Writing a ‘1’ to any bit position in the Set register will set valid bits in the associated register. Reads from the Set register should be ignored.

3: This register has an associated Invert register at an offset of 0xC bytes. These registers have the same name with INV appended to the end of the register name (e.g., ETHCON1INV). Writing a ‘1’ to any bit position in the Invert register will invert valid bits in the associated register. Reads from the Invert register should be ignored.

Table 35-1: Ethernet Controller Register Summary (Continued)

Name	Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
EMAC1SA0(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	STNADDR6<7:0>							
	7:0	STNADDR5<7:0>							
EMAC1SA1(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	STNADDR4<7:0>							
	7:0	STNADDR3<7:0>							
EMAC1SA2(1,2,3)	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	STNADDR2<7:0>							
	7:0	STNADDR1<7:0>							

Legend: — = unimplemented, read as ‘0’.

Note 1: This register has an associated Clear register at an offset of 0x4 bytes. These registers have the same name with CLR appended to the end of the register name (e.g., ETHCON1CLR). Writing a ‘1’ to any bit position in the Clear register will clear valid bits in the associated register. Reads from the Clear register should be ignored.

2: This register has an associated Set register at an offset of 0x8 bytes. These registers have the same name with SET appended to the end of the register name (e.g.,ETHCON1SET). Writing a ‘1’ to any bit position in the Set register will set valid bits in the associated register. Reads from the Set register should be ignored.

3: This register has an associated Invert register at an offset of 0xC bytes. These registers have the same name with INV appended to the end of the register name (e.g., ETHCON1INV). Writing a ‘1’ to any bit position in the Invert register will invert valid bits in the associated register. Reads from the Invert register should be ignored.

Register 35-1: ETHCON1: Ethernet Controller Control 1 Register

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PTV<15:8>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PTV<7:0>							
15:8	R/W-0	U-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0
	ON	—	SIDL	—	—	—	TXRTS	RXEN(1)
7:0	R/W-0	U-0	U-0	R/W-0	U-0	U-0	U-0	R/W-0
	AUTOFC	—	—	MANFC	—	—	—	BUFCDEC

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **PTV<15:0>:** PAUSE Timer Value bits

PAUSE Timer Value used for Flow Control.
This register should only be written when RXEN (ETHCON1<8>) is not set.

These bits are used for Flow Control operations only.

bit 15 **ON:** Ethernet ON bit

1 = Ethernet module is enabled
0 = Ethernet module is disabled

bit 14 **Unimplemented:** Read as ‘0’

bit 13 **SIDL:** Ethernet Stop in Idle Mode bit

1 = Ethernet module transfers are frozen during Idle
0 = Ethernet module transfers continue during Idle

bit 12-10 **Unimplemented:** Read as ‘0’

bit 9 **TXRTS:** Transmit Request to Send bit

1 = Activate the transmit logic and send the packet(s) defined in the TX EDT
0 = Stop transmit (when cleared by software) or transmit done (when cleared by hardware)

After the bit is written with a ‘1’, it will clear to a ‘0’ whenever the transmit logic has finished transmitting the requested packets in the Ethernet Descriptor Table (EDT). If a ‘0’ is written by the CPU, the transmit logic finishes the current packet’s transmission and then stops any further.

This bit affects TX operations only.

bit 8 **RXEN:** Receive Enable bit(1)

1 = Enable RX logic, packets are received and stored in the RX buffer as controlled by the filter configuration
0 = Disable RX logic, no packets are received in the RX buffer

This bit affects RX operations only.

Note 1: It is not recommended to clear the RXEN bit and then make changes to any RX related field/register. The Ethernet Controller must be reinitialized (ON cleared to ‘0’), and then the RX changes applied.

Register 35-1: ETHCON1: Ethernet Controller Control 1 Register (Continued)

bit 7 **AUTOFC:** Automatic Flow Control bit

1 = Automatic flow control enabled
0 = Automatic flow control disabled

Setting this bit will enable automatic flow control. If set, the full and empty watermarks are used to automatically enable and disable the flow control, respectively. When the number of received buffers BUFCNT (ETHSTAT<16:23>) rises to the full watermark, flow control is automatically enabled. When the BUFCNT falls to the empty watermark, flow control is automatically disabled.

This bit is used for flow control operations only and affects both TX and RX operations.

bit 6-5 **Unimplemented:** Read as '0'

bit 4 **MANFC:** Manual Flow Control bit

1 = Manual Flow Control enabled
0 = Manual Flow Control disabled

Setting this bit will enable manual flow control. If set, the flow control logic will send a PAUSE frame using the PAUSE timer value in the PTV register. It will then resend a PAUSE frame every 128 * PTV<15:0>/2 TX clock cycles until the bit is cleared.

Note that for 10 Mbps operation, TX clock runs at 2.5 MHz. For 100 Mbps operation, TX clock runs at 25 MHz.

When this bit is cleared, the Flow Control logic will automatically send a PAUSE frame with a 0x0000 PAUSE timer value to disable flow control.

This bit is used for flow control operations only and affects both TX and RX operations.

bit 3-1 **Unimplemented:** Read as '0'

bit 0 **BUFCDEC:** Descriptor Buffer Count Decrement bit

The BUFCDEC bit is a write-1 bit that reads out '0'. When written with '1', BUFCNT, the Descriptor Buffer Counter, will decrement by one. If the BUFCNT counter is incremented by the RX logic at the same time that this bit is written, the BUFCNT value will remain unchanged. Writing '0' will have no effect.

This bit is used for RX operations only.

Note 1: It is not recommended to clear the RXEN bit and then make changes to any RX related field/register. The Ethernet Controller must be reinitialized (ON cleared to '0'), and then the RX changes applied.

Register 35-2: ETHCON2: Ethernet Controller Control 2 Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
	—	—	—	—	—	RXBUFSZ<6:4>		
7:0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
	RXBUFSZ<3:0>				—	—	—	—

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: ('0', '1', x = Unknown)		

bit 31-11 **Unimplemented:** Read as '0'

bit 10-4 **RXBUFSZ<6:0>:** RX Data Buffer Size for All RX Descriptors (in 16-byte increments) bits

`0x00` = Reserved
`0x01` = RX data Buffer size for descriptors is 16 bytes
`0x02` = RX data Buffer size for descriptors is 32 bytes
`0x03` = RX data Buffer size for descriptors is 48 bytes
•
•
•
`0x60` = RX data Buffer size for descriptors is 1536 bytes
•
•
•
`0x7F` = RX data Buffer size for descriptors is 2032 bytes

bit 3-0 **Unimplemented:** Read as '0'

Note 1: This register is used for RX operations only.
2: These bits may only be changed while the RXEN bit (ETHCON1<8>) = `0`.

Register 35-3: ETHTXST: Ethernet Controller TX Packet Descriptor Start Address Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	TXSTADDR<31:24>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	TXSTADDR<23:16>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	TXSTADDR<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0
	TXSTADDR<7:2>						—	—

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-2 **TXSTADDR<31:2>:** Starting Address of First Transmit Descriptor bits

This register should not be written while any transmit, receive or DMA operations are in progress.

This address must be 4-byte aligned (i.e., bits 1-0 must be ‘00’).

bit 1-0 **Unimplemented:** Read as ‘0’

Note 1: This register is used for TX operations only.

2: This register will be updated by hardware with the last descriptor used by the last successfully transmitted packet.

Register 35-4: ETHRXST: Ethernet Controller RX Packet Descriptor Start Address Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	RXSTADDR<31:24>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	RXSTADDR<23:16>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	RXSTADDR<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0
	RXSTADDR<7:2>						—	—

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: ('0', '1', x = Unknown)		

bit 31-2 **RXSTADDR<31:2>:** Starting Address of First Receive Descriptor bits

This register should not be written while any transmit, receive or DMA operations are in progress.

This address must be 4-byte aligned (i.e., bits 1-0 must be '00').

bit 1-0 **Unimplemented:** Read as '0'

Note 1: This register is used for RX operations only.

2: This register will be updated by hardware with the last descriptor used by the last successfully transmitted packet.

Register 35-5: ETHHT0: Ethernet Controller Hash Table 0 Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	HT<31:24>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	HT<23:16>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	HT<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	HT<7:0>							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0 **HT<31:0>:** Hash Table Bytes 0-3 bits

Note 1: This register is used for RX operations only.

2: These bits may only be changed while the RXEN bit (ETHCON1<8>) = `0` or the HTEN bit (ETHRXFC<15>) = `0`.

Register 35-6: ETHHT1: Ethernet Controller Hash Table 1 Register[1,2]

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	HT<63:56>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	HT<55:48>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	HT<47:40>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	HT<39:32>							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0 **HT<63:32>:** Hash Table Bytes 4-7 bits

Note 1: This register is used for RX operations only.

2: These bits may only be changed while the RXEN bit (ETHCON1<8>) = `0` or the HTEN bit (ETHRXFC<15>) = `0`.

Register 35-7: ETHPMM0: Ethernet Controller Pattern Match Mask 0 Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PMM<31:24>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PMM<23:16>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PMM<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PMM<7:0>							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-24 **PMM<31:24>:** Pattern Match Mask 3 bits

bit 23-16 **PMM<23:16>:** Pattern Match Mask 2 bits

bit 15-8 **PMM<15:8>:** Pattern Match Mask 1 bits

bit 7-0 **PMM<7:0>:** Pattern Match Mask 0 bits

Note 1: This register is used for RX operations only.

2: These bits may only be changed while the RXEN bit (ETHCON1<8>) = 0 or the PMMODE bit (ETHRXFC<11:8>) = 0.

Register 35-8: ETHPMM1: Ethernet Controller Pattern Match Mask 1 Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PMM<63:56>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PMM<55:48>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PMM<47:40>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PMM<39:32>							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-24 **PMM<63:56>:** Pattern Match Mask 7 bits

bit 23-16 **PMM<55:48>:** Pattern Match Mask 6 bits

bit 15-8 **PMM<47:40>:** Pattern Match Mask 5 bits

bit 7-0 **PMM<39:32>:** Pattern Match Mask 4 bits

Note 1: This register is used for RX operations only.

2: These bits may only be changed while the RXEN bit (ETHCON1<8>) = `0` or the PMMODE bit (ETHRXFC<11:8>) = `0`.

Register 35-9: ETHPMCS: Ethernet Controller Pattern Match Checksum Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PMCS<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PMCS<7:0>							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-8 **PMCS<15:8>:** Pattern Match Checksum 1 bits

bit 7-0 **PMCS<7:0>:** Pattern Match Checksum 0 bits

Note 1: This register is used for RX operations only.

2: These bits may only be changed while the RXEN bit (ETHCON1<8>) = 0 or the PMMODE bit (ETHRXFC<11:8>) = 0.

Register 35-10: ETHPMO: Ethernet Controller Pattern Match Offset Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PMO<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PMO<7:0>							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-0 **PMO<15:0>:** Pattern Match Offset 1 bits

Note 1: This register is used for RX operations only.

2: These bits may only be changed while the RXEN bit (ETHCON1<8>) = 0 or the PMMODE bit (ETHRXFC<11:8>) = 0.

Register 35-11: ETHRXFC: Ethernet Controller Receive Filter Configuration Register[1,2]

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	HTEN	MPEN	—	NOTPM	PMMODE<3:0>			
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CRCERREN	CRCOKEN	RUNTERREN	RUNTEN	UCEN	NOTMEEN	MCEN	BCEN

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **HTEN:** Enable Hash Table Filtering bit
1 = Enable Hash Table Filtering
0 = Disable Hash Table Filtering

bit 14 **MPEN:** Magic Packet™ Enable bit
1 = Enable Magic Packet Filtering
0 = Disable Magic Packet Filtering

bit 13 **Unimplemented:** Read as ‘0’

bit 12 **NOTPM:** Pattern Match Inversion bit
1 = The Pattern Match Checksum must not match for a successful Pattern Match to occur
0 = The Pattern Match Checksum must match for a successful Pattern Match to occur
This bit determines whether Pattern Match Checksum must match in order for a successful Pattern Match to occur.

Note 1: This register is used for RX operations only.
2: These bits may only be changed while the RXEN bit (ETHCON1<8>) = 0.
3: XOR = True when either one or the other conditions are true, but not both.
4: This Hash Table Filter match is active regardless of the value of the HTEN bit.
5: This Magic Packet Filter match is active regardless of the value of the MPEN bit.

Register 35-11: ETHRXFC: Ethernet Controller Receive Filter Configuration Register(1,2) (Continued)

bit 11-8 **PMMODE<3:0>:** Pattern Match Mode bits

0000 = Pattern Match is disabled; pattern match is always unsuccessful
0001 = Pattern match is successful if (NOTPM = 1 XOR Pattern Match Checksum matches)(3)
0010 = Pattern match is successful if (NOTPM = 1 XOR Pattern Match Checksum matches) AND (Destination Address = Station Address)(3)
0011 = Pattern match is successful if (NOTPM = 1 XOR Pattern Match Checksum matches) AND (Destination Address = Station Address)(3)
0100 = Pattern match is successful if (NOTPM = 1 XOR Pattern Match Checksum matches) AND (Destination Address = Unicast Address)(3)
0101 = Pattern match is successful if (NOTPM = 1 XOR Pattern Match Checksum matches) AND (Destination Address = Unicast Address)(3)
0110 = Pattern match is successful if (NOTPM = 1 XOR Pattern Match Checksum matches) AND (Destination Address = Broadcast Address)(3)
0111 = Pattern match is successful if (NOTPM = 1 XOR Pattern Match Checksum matches) AND (Destination Address = Broadcast Address)(3)
1000 = Pattern match is successful if (NOTPM = 1 XOR Pattern Match Checksum matches) AND (Hash Table Filter match)(3,4)
1001 = Pattern match is successful if (NOTPM = 1 XOR Pattern Match Checksum matches) AND (Packet = Magic Packet)(3,5)

bit 7 **CRCERREN:** CRC Error Collection Enable bit

1 = The received packet CRC must be invalid for the packet to be accepted
0 = Disable CRC Error Collection filtering
This bit allows the user to collect all packets that have an invalid CRC.

bit 6 **CRCOKEN:** CRC OK Enable bit

1 = The received packet CRC must be valid for the packet to be accepted
0 = Disable CRC filtering
This bit allows the user to reject all packets that have an invalid CRC.

bit 5 **RUNTERREN:** Runt Error Collection Enable bit

1 = The received packet must be a runt packet for the packet to be accepted
0 = Disable Runt Error Collection filtering

This bit allows the user to collect all packets that are runt packets. For this filter, a runt packet is defined as any packet with a size of less than 64 bytes (when CRCOKEN = 0) or any packet with a size of less than 64 bytes that has a valid CRC (when CRCOKEN = 1).

bit 4 **RUNTEN:** Runt Enable bit

1 = The received packet must not be a runt packet for the packet to be accepted
0 = Disable Runt filtering

This bit allows the user to reject all runt packets. For this filter, a runt packet is defined as any packet with a size of less than 64 bytes.

Note 1: This register is used for RX operations only.
2: These bits may only be changed while the RXEN bit (ETHCON1<8>) = 0.
3: XOR = True when either one or the other conditions are true, but not both.
4: This Hash Table Filter match is active regardless of the value of the HTEN bit.
5: This Magic Packet Filter match is active regardless of the value of the MPEN bit.

Register 35-11: ETHRXFC: Ethernet Controller Receive Filter Configuration Register[1,2] (Continued)

bit 3 **UCEN:** Unicast Enable bit

1 = Enable Unicast Filtering
0 = Disable Unicast Filtering

This bit allows the user to accept all unicast packets whose Destination Address matches the Station Address.

bit 2 **NOTMEEN:** Not Me Unicast Enable bit

1 = Enable Not Me Unicast Filtering
0 = Disable Not Me Unicast Filtering

This bit allows the user to accept all unicast packets whose Destination Address does NOT match the Station Address.

bit 1 **MCEN:** Multicast Enable bit

1 = Enable Multicast Filtering
0 = Disable Multicast Filtering

This bit allows the user to accept all Multicast Address packets.

bit 0 **BCEN:** Broadcast Enable bit

1 = Enable Broadcast Filtering
0 = Disable Broadcast Filtering

This bit allows the user to accept all Broadcast Address packets.

Note 1: This register is used for RX operations only.
2: These bits may only be changed while the RXEN bit (ETHCON1<8>) = 0.
3: XOR = True when either one or the other conditions are true, but not both.
4: This Hash Table Filter match is active regardless of the value of the HTEN bit.
5: This Magic Packet Filter match is active regardless of the value of the MPEN bit.

Register 35-12: ETHRXWM: Ethernet Controller Receive Watermarks Register(1)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	RXFWM<7:0>							
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	RXEWM<7:0>							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: ('0', '1', x = Unknown)		

bit 31-24 **Unimplemented:** Read as '0'

bit 23-16 **RXFWM<7:0>:** Receive Full Watermark bits

The software controlled RX Buffer Full Watermark Pointer is compared against the RX BUFCNT to determine the full watermark condition for the FWMARK interrupt and for enabling flow control when Auto Flow Control is enabled. The Full Watermark Pointer should always be greater than the Empty Watermark Pointer.

bit 15-8 **Unimplemented:** Read as '0'

bit 7-0 **RXEWM<7:0>:** Receive Empty Watermark bits

The software controlled RX Buffer Empty Watermark Pointer is compared against the RX BUFCNT to determine the empty watermark condition for the EWMARK interrupt and for disabling flow control when Auto Flow Control is enabled. The Empty Watermark Pointer should always be less than the Full Watermark Pointer.

Note 1: This register is used for RX operations only.

Register 35-13: ETHIEN: Ethernet Controller Interrupt Enable Register

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	R/W-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0
	—	TXBUSEIE(1)	RXBUSEIE(2)	—	—	—	EWMARKIE(2)	FWMARKIE(2)
7:0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
	RXDONEIE(2)	PKTPENDIE(2)	RXACTIE(2)	—	TXDONEIE(1)	TXABORTIE(1)	RXBUFNAIE(2)	RXOVFLWIE(2)

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-15 **Unimplemented:** Read as ‘0’

bit 14 **TXBUSEIE:** Transmit BVCI Bus Error Interrupt Enable bit(1)
1 = Enable TXBUS Error Interrupt
0 = Disable TXBUS Error Interrupt

bit 13 **RXBUSEIE:** Receive BVCI Bus Error Interrupt Enable bit(2)
1 = Enable RXBUS Error Interrupt
0 = Disable RXBUS Error Interrupt

bit 12-10 **Unimplemented:** Read as ‘0’

bit 9 **EWMARKIE:** Empty Watermark Interrupt Enable bit(2)
1 = Enable EWMARK Interrupt
0 = Disable EWMARK Interrupt

bit 8 **FWMARKIE:** Full Watermark Interrupt Enable bit(2)
1 = Enable FWMARK Interrupt
0 = Disable FWMARK Interrupt

bit 7 **RXDONEIE:** Receiver Done Interrupt Enable bit(2)
1 = Enable RXDONE Interrupt
0 = Disable RXDONE Interrupt

bit 6 **PKTPENDIE:** Packet Pending Interrupt Enable bit(2)
1 = Enable PKTPEND Interrupt
0 = Disable PKTPEND Interrupt

bit 5 **RXACTIE:** RX Activity Interrupt Enable bit
1 = Enable RXACT Interrupt
0 = Disable RXACT Interrupt

bit 4 **Unimplemented:** Read as ‘0’

bit 3 **TXDONEIE:** Transmitter Done Interrupt Enable bit(1)
1 = Enable TXDONE Interrupt
0 = Disable TXDONE Interrupt

bit 2 **TXABORTIE:** Transmitter Abort Interrupt Enable bit(1)
1 = Enable TXABORT Interrupt
0 = Disable TXABORT Interrupt

Note 1: These bits are used for TX operations only.
2: These bits are used for RX operations only.

35

Ethernet Controller

Register 35-13: ETHIEN: Ethernet Controller Interrupt Enable Register (Continued)

bit 1 **RXBUFNAIE:** Receive Buffer Not Available Interrupt Enable bit(2)
1 = Enable RXBUFNA Interrupt
0 = Disable RXBUFNA Interrupt

bit 0 **RXOVFLWIE:** Receive FIFO Overflow Interrupt Enable bit(2)
1 = Enable RXOVFLW Interrupt
0 = Disable RXOVFLW Interrupt

Note 1: These bits are used for TX operations only.
2: These bits are used for RX operations only.

Register 35-14: ETHIRQ: Ethernet Controller Interrupt Request Register

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	R/W-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0
	—	TXBUSE	RXBUSE	—	—	—	EWMARK	FWMARK
7:0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
	RXDONE	PKTPEND	RXACT	—	TXDONE	TXABORT	RXBUFNA	RXOVFLW

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-15 **Unimplemented:** Read as ‘0’

bit 14 **TXBUSE:** Transmit BVCI Bus Error Interrupt bit(2)
1 = BVCI Bus Error occurred
0 = No BVCI Error occurred

This bit is set when the TX DMA encounters a BVCI Bus error during a memory access. It is cleared by either a Reset or CPU write of a ‘1’ to the CLR register.

bit 13 **RXBUSE:** Receive BVCI Bus Error Interrupt bit(2)
1 = BVCI Bus Error occurred
0 = No BVCI Error occurred

This bit is set when the RX DMA encounters a BVCI Bus error during a memory access. It is cleared by either a Reset or CPU write of a ‘1’ to the CLR register.

bit 12-10 **Unimplemented:** Read as ‘0’

bit 9 **EWMARK:** Empty Watermark Interrupt bit(2)
1 = Empty Watermark pointer reached
0 = No interrupt pending

This bit is set when the RX Descriptor Buffer Count is less than or equal to the value in the RXEWM bit (ETHRXWM<0:7>) value. It is cleared by BUFCNT bit (ETHSTAT<16:23>) being incremented by hardware. Writing a ‘0’ or a ‘1’ has no effect.

bit 8 **FWMARK:** Full Watermark Interrupt bit(2)
1 = Full Watermark pointer reached
0 = No interrupt pending

This bit is set when the RX Descriptor Buffer Count is greater than or equal to the value in the RXFWM bit (ETHRXWM<16:23>) field. It is cleared by writing the BUFCDEC (ETHCON1<0>) bit to decrement the BUFCNT counter. Writing a ‘0’ or a ‘1’ has no effect.

bit 7 **RXDONE:** Receive Done Interrupt bit(2)
1 = RX packet was successfully received
0 = No interrupt pending

This bit is set whenever an RX packet is successfully received. It is cleared by either a Reset or CPU write of a ‘1’ to the CLR register.

Note 1: These bits are used only for TX operations.
2: These bits are used only for RX operations.

Register 35-14: ETHIRQ: Ethernet Controller Interrupt Request Register (Continued)

bit 6 **PKTPEND:** Packet Pending Interrupt bit(2)

1 = Received packet pending in memory
0 = No receive packet is pending in memory

This bit is set when the BUFCNT counter has a value other than '0'. It is cleared by either a Reset or by writing the BUFCDEC bit to decrement the BUFCNT counter. Writing a '0' or a '1' has no effect.

bit 5 **RXACT:** Receive Activity Interrupt bit(2)

1 = RX packet data was successfully received
0 = No interrupt pending

This bit is set whenever RX packet data is stored in the RXBM FIFO. It is cleared by either a Reset or CPU write of a '1' to the CLR register.

bit 4 **Unimplemented:** Read as '0'

bit 3 **TXDONE:** Transmit Done Interrupt bit(2)

1 = TX packet successfully sent
0 = No interrupt pending

This bit is set when the currently transmitted TX packet completes transmission, and the Transmit Status Vector is loaded into the first descriptor used for the packet. It is cleared by either a Reset or CPU write of a '1' to the CLR register.

bit 2 **TXABORT:** Transmit Abort Condition Interrupt bit(2)

1 = TX abort condition occurred on the last TX packet
0 = No interrupt pending

This bit is set when the MAC aborts the transmission of a TX packet for one of the following reasons:

- Jumbo TX packet abort
- Underrun abort
- Excessive defer abort
- Late collision abort
- Excessive collisions abort

This bit is cleared by either a Reset or CPU write of a '1' to the CLR register.

bit 1 **RXBUFNA:** Receive Buffer Not Available Interrupt bit(2)

1 = RX Buffer Descriptor Not Available condition occurred
0 = No interrupt pending

This bit is set by a RX Buffer Descriptor Overrun condition. It is cleared by either a Reset or a CPU write of a '1' to the CLR register.

bit 0 **RXOVFLW:** Receive FIFO Over Flow Error bit(2)

1 = RX FIFO Overflow Error condition occurred
0 = No interrupt pending

RXOVFLW is set by the RXBM Logic for an RX FIFO Overflow condition. It is cleared by either a Reset or CPU write of a '1' to the CLR register.

Note 1: These bits are used only for TX operations.
2: These bits are used only for RX operations.

Note:	It is recommended to use the SET, CLR, or INV registers to set or clear any bit in this register. Setting or clearing any bits in this register should be done for debug/test purposes only.

Register 35-15: ETHSTAT: Ethernet Controller Status Register

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	BUFCNT<7:0>(1)							
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
	ETHBUSY(5)	TXBUSY(2,6)	RXBUSY(3,6)	—	—	—	—	—

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: ('0', '1', x = Unknown)		

bit 31-24 **Unimplemented:** Read as '0'

bit 23-16 **BUFCNT<7:0>:** Packet Buffer Count bits(1)

Number of packet buffers received in memory. Once a packet has been successfully received, this register is incremented by hardware based on the number of descriptors used by the packet. Software decrements the counter (by writing to the BUFCDEC bit (ETHCON1<0>) for each descriptor used) after a packet has been read out of the buffer. The register does not roll over (0xFF to 0x00) when hardware tries to increment the register and the register is already at 0xFF. Conversely, the register does not roll under (0x00 to 0xFF) when software tries to decrement the register and the register is already at 0x0000. When software attempts to decrement the counter at the same time that the hardware attempts to increment the counter, the counter value will remain unchanged.

When this register value reaches 0xFF, the RX logic will halt (ONLY if Auto Flow Control is enabled) awaiting software to write the BUFCDEC bit in order to decrement the register below 0xFF.

If Auto Flow Control is disabled, the RXDMA will continue processing and the BUFCNT will saturate at a value of 0xFF.

When this register is non-zero, the PKTPEND status bit will be set and an interrupt may be generated, depending on the value of the ETHIEN bit <PKTPENDIE> register.

When the ETHRXST register is written, the BUFCNT counter is automatically cleared to 0x00.

Note: BUFCNT will NOT be cleared when ON is set to '0'. This enables software to continue to utilize and decrement this count.

bit 15-8 **Unimplemented:** Read as '0'

bit 7 **ETHBUSY:** Ethernet Module busy bit(5)

1 = Ethernet logic has been turned on (ON (ETHCON1<15>) = 1) or is completing a transaction
0 = Ethernet logic is idle

This bit indicates that the module has been turned on or is completing a transaction after being turned off.

Note 1: These bits are used for RX operations only.
2: This bit is only affected by TX operations.
3: This bit is only affected by RX operations.
4: This bit is affected by TX and RX operations.
5: This bit will be *set* when the ON bit (ETHCON1<15>) = 1.
6: This bit will be *cleared* when the ON bit (ETHCON1<15>) = 0.

Register 35-15: ETHSTAT: Ethernet Controller Status Register (Continued)

bit 6 **TXBUSY:** Transmit Busy bit(2,6)

1 = TX logic is receiving data
0 = TX logic is idle

This bit indicates that a packet is currently being transmitted. A change in this status bit is not necessarily reflected by the TXDONE interrupt, as TX packets may be aborted or rejected by the MAC.

bit 5 **RXBUSY:** Receive Busy bit(3,6)

1 = RX logic is receiving data
0 = RX logic is idle

This bit indicates that a packet is currently being received. A change in this status bit is not necessarily reflected by the RXDONE interrupt, as RX packets may be aborted or rejected by the RX filter.

bit 4-0 **Unimplemented:** Read as '0'

Note 1: These bits are used for RX operations only.
2: This bit is only affected by TX operations.
3: This bit is only affected by RX operations.
4: This bit is affected by TX and RX operations.
5: This bit will be *set* when the ON bit (ETHCON1<15>) = 1.
6: This bit will be *cleared* when the ON bit (ETHCON1<15>) = 0.

Register 35-16: ETHRXOVFLOW: Ethernet Controller Receive Overflow Statistics Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	RXOVFLWCNT<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	RXOVFLWCNT<7:0>							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-0 **RXOVFLWCNT<15:0>:** Dropped Receive Frames Count bits
Increment counter for frames accepted by the RX filter and subsequently dropped due to internal receive error (RXFIFO overrun). This event also sets the RXOVFLW bit (ETHIRQ<0>) interrupt flag.

Note 1: This register is used for RX operations only.
2: This register is automatically cleared by hardware after a read operation, unless the byte enables for bytes 0/1 are ‘0’.

Note: It is recommended to use the SET, CLR, or INV registers to set or clear any bit in this register. Setting or clearing any bits in this register should be done for debug/test purposes only.

Register 35-17: ETHFRMTXOK: Ethernet Controller Frames Transmitted OK Statistics Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	FRMTXOKCNT<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	FRMTXOKCNT<7:0>							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-0 **FRMTXOKCNT<15:0>:** Frame Transmitted OK Count bits
Increment counter for frames successfully transmitted.

Note 1: This register is used for TX operations only.
2: This register is automatically cleared by hardware after a read operation, unless the byte enables for bytes 0/1 are ‘0’.

Note: It is recommended to use the SET, CLR, or INV registers to set or clear any bit in this register. Setting or clearing any bits in this register should be done for debug/test purposes only.

Register 35-18: ETHSCOLFRM: Ethernet Controller Single Collision Frames Statistics Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	SCOLFRMCNT<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	SCOLFRMCNT<7:0>							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-0 **SCOLFRMCNT<15:0>:** Single Collision Frame Count bits
Increment count for frames that were successfully transmitted on the second try.

Note 1: This register is used for TX operations only.
2: This register is automatically cleared by hardware after a read operation, unless the byte enables for bytes 0/1 are ‘0’.

Note: It is recommended to use the SET, CLR, or INV registers to set or clear any bit in this register. Setting or clearing any bits in this register should be done for debug/test purposes only.

Register 35-19: ETHMCOLFRM: Ethernet Controller Multiple Collision Frames Statistics Register[1,2]

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	MCOLFRMCNT<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	MCOLFRMCNT<7:0>							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-0 **MCOLFRMCNT<15:0>:** Multiple Collision Frame Count bits
Increment count for frames that were successfully transmitted after there was more than one collision.

Note 1: This register is used for TX operations only.
2: This register is automatically cleared by hardware after a read operation, unless the byte enables for bytes 0/1 are ‘0’.

Note: It is recommended to use the SET, CLR, or INV registers to set or clear any bit in this register. Setting or clearing any bits in this register should be done for debug/test purposes only.

Register 35-20: ETHFRMRXOK: Ethernet Controller Frames Received OK Statistics Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	FRMRXOKCNT<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	FRMRXOKCNT<7:0>							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-0 **FRMRXOKCNT<15:0>:** Frames Received OK Count bits

Increment count for frames received successfully by the RX Filter. This count will not be incremented if there is a Frame Check Sequence (FCS) or Alignment error.

Note 1: This register is used for RX operations only.

2: This register is automatically cleared by hardware after a read operation, unless the byte enables for bytes 0/1 are ‘0’.

Note: It is recommended to use the SET, CLR, or INV registers to set or clear any bit in this register. Setting or clearing any bits in this register should be done for debug/test purposes only.

Register 35-21: ETHFCSERR: Ethernet Controller Frame Check Sequence Error Statistics Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	FCSERRCNT<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	FCSERRCNT<7:0>							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-0 **FCSERRCNT<15:0>:** FCS Error Count bits

Increment count for frames received with FCS error and the frame length in bits is an integral multiple of 8 bits.

Note 1: This register is used for RX operations only.

2: This register is automatically cleared by hardware after a read operation, unless the byte enables for bytes 0/1 are ‘0’.

Note: It is recommended to use the SET, CLR, or INV registers to set or clear any bit in this register. Setting or clearing any bits in this register should be done for debug/test purposes only.

Register 35-22: ETHALGNERR: Ethernet Controller Alignment Errors Statistics Register[1,2]

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	ALGNERRCNT<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	ALGNERRCNT<7:0>							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: ('0', '1', x = Unknown)		

bit 31-16 **Unimplemented:** Read as '0'

bit 15-0 **ALGNERRCNT<15:0>:** Alignment Error Count bits

Increment count for frames with alignment errors. Note that an alignment error is a frame that has an FCS error and the frame length in bits is not an integral multiple of 8 bits (a.k.a., dribble nibble)

Note 1: This register is used for RX operations only.

2: This register is automatically cleared by hardware after a read operation, unless the byte enables for bytes 0/1 are '0'.

Note: It is recommended to use the SET, CLR, or INV registers to set or clear any bit in this register. Setting or clearing any bits in this register should be done for debug/test purposes only.

35

Ethernet Controller

Register 35-23: EMAC1CFG1: Ethernet Controller MAC Configuration 1 Register(1)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-1	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
	SOFTRESET	SIMRESET	—	—	RESETRMCS	RESETRFUN	RESETTMCS	RESETTFUN
7:0	U-0	U-0	U-0	R/W-0	R/W-1	R/W-1	R/W-0	R/W-1
	—	—	—	LOOPBACK	TXPAUSE	RXPAUSE	PASSALL	RXENABLE

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit		-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)	

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **SOFTRESET:** Soft Reset bit
Setting this bit will put the MACMII in reset. Its default value is ‘1’.

bit 14 **SIMRESET:** Simulation Reset bit
Setting this bit will cause a reset to the random number generator within the Transmit Function.

bit 13-12 **Unimplemented:** Read as ‘0’

bit 11 **RESETRMCS:** Reset MCS/RX bit
Setting this bit will put the MAC Control Sub-layer/Receive domain logic in reset.

bit 10 **RESETRFUN:** Reset RX Function bit
Setting this bit will put the MAC Receive function logic in reset.

bit 9 **RESETTMCS:** Reset MCS/TX bit
Setting this bit will put the MAC Control Sub-layer/TX domain logic in reset.

bit 8 **RESETTFUN:** Reset TX Function bit
Setting this bit will put the MAC Transmit function logic in reset.

bit 7-5 **Unimplemented:** Read as ‘0’

bit 4 **LOOPBACK:** MAC Loopback mode bit
1 = MAC Transmit interface is loop backed to the MAC Receive interface
0 = MAC normal operation

bit 3 **TXPAUSE:** MAC TX Flow Control bit
1 = PAUSE Flow Control frames are allowed to be transmitted
0 = PAUSE Flow Control frames are blocked

bit 2 **RXPAUSE:** MAC RX Flow Control bit
1 = The MAC acts upon received PAUSE Flow Control frames
0 = Received PAUSE Flow Control frames are ignored

bit 1 **PASSALL:** MAC Pass all Receive Frames bit
1 = The MAC will accept all frames regardless of type (normal vs. Control)
0 = The received Control frames are ignored

bit 0 **RXENABLE:** MAC Receive Enable bit
1 = Enable the MAC receiving of frames
0 = Disable the MAC receiving of frames

Note 1: Both 16- and 32-bit accesses are allowed to these registers (including the Set, Clear and Invert registers). 8-bit accesses are not allowed and are ignored by the hardware.

Register 35-24: EMAC1CFG2: Ethernet Controller MAC Configuration 2 Register(1)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 25/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	R/W-1	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
	—	EXCESSDFR	BPNOBKOFF	NOBKOFF	—	—	LONGPRE	PUREPRE
7:0	R/W-1	R/W-0	R/W-1	R/W-1	R/W-0	R/W-0	R/W-1	R/W-0
	AUTOPAD(5,6)	VLANPAD(5,6)	PADENABLE(5,7)	CRCENABLE	DELAYCRC	HUGEFRM	LENGTHCK	FULLDPLX

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: ('0', '1', x = Unknown)		

bit 31-15 **Unimplemented:** Read as '0'

bit 14 **EXCESSDFR:** Excess Defer bit
- 1 = The MAC will defer to carrier indefinitely as per the IEEE 802.3 Specification standard
- 0 = The MAC will abort when the excessive deferral limit is reached

bit 13 **BPNOBKOFF:** Backpressure/No Backoff bit
- 1 = The MAC after incidentally causing a collision during backpressure will immediately retransmit without backoff reducing the chance of further collisions and ensuring transmit packets get sent
- 0 = The MAC will not remove the backoff

bit 12 **NOBKOFF:** No Backoff bit
- 1 = Following a collision, the MAC will immediately retransmit rather than using the Binary Exponential Backoff algorithm as specified in the IEEE 802.3 Specification standard
- 0 = Following a collision, the MAC will use the Binary Exponential Backoff algorithm

bit 11-10 **Unimplemented:** Read as '0'

bit 9 **LONGPRE:** Long Preamble Enforcement bit
- 1 = The MAC only allows receive packets which contain preamble fields less than 12 bytes in length
- 0 = The MAC allows any length preamble as per the IEEE 802.3 Specification standard

bit 8 **PUREPRE:** Pure Preamble Enforcement bit
- 1 = The MAC will verify the content of the preamble to ensure it contains 0x55 and is error-free. A packet with errors in its preamble is discarded
- 0 = The MAC does not perform any preamble checking

bit 7 **AUTOPAD:** Auto Detect Pad Enable bit(2,3)
- 1 = The MAC will automatically detect the type of frame, either tagged or untagged, by comparing the two octets following the source address with 0x8100 (VLAN Protocol ID) and pad accordingly
- 0 = The MAC does not perform auto detection

bit 6 **VLANPAD:** VLAN Pad Enable bit(2,3)
- 1 = The MAC will pad all short frames to 64 bytes and append a valid CRC
- 0 = The MAC does not perform padding of short frames

Note 1: Both 16- and 32-bit accesses are allowed to these registers (including the Set, Clear and Invert registers). 8-bit accesses are not allowed and are ignored by the hardware.

2: Table 35-2 provides a description of the pad function based on the configuration of this register.

3: This bit is ignored if the PADENABLE bit is cleared.

4: This bit is used in conjunction with the AUTOPAD and VLANPAD bits.

Register 35-24: EMAC1CFG2: Ethernet Controller MAC Configuration 2 Register(1) (Continued)

bit 5 **PADENABLE:** Pad/CRC Enable bit(2,4)

1 = The MAC will pad all short frames
0 = The frames presented to the MAC have a valid length

bit 4 **CRCENABLE:** CRC Enable1 bit

1 = The MAC will append a CRC to every frame whether padding was required or not. Must be set if the PADENABLE bit is set
0 = The frames presented to the MAC have a valid CRC

bit 3 **DELAYCRC:** Delayed CRC bit

This bit determines the number of bytes, if any, of proprietary header information that exist on the front of the IEEE 802.3 frames.

1 = Four bytes of header (ignored by the CRC function)
0 = No proprietary header

bit 2 **HUGEFRM:** Huge Frame enable bit

1 = Frames of any length are transmitted and received
0 = Huge frames are not allowed for receive or transmit

bit 1 **LENGTHCK:** Frame Length checking bit

1 = Both transmit and receive frame lengths are compared to the Length/Type field. If the Length/Type field represents a length then the check is performed. Mismatches are reported on the Transmit/Receive Statistics Vector
0 = Length/Type field check is not performed

bit 0 **FULLDPLX:** Full-Duplex Operation bit

1 = The MAC operates in Full-Duplex mode
0 = The MAC operates in Half-Duplex mode

Note 1: Both 16- and 32-bit accesses are allowed to these registers (including the Set, Clear and Invert registers). 8-bit accesses are not allowed and are ignored by the hardware.
2: Table 35-2 provides a description of the pad function based on the configuration of this register.
3: This bit is ignored if the PADENABLE bit is cleared.
4: This bit is used in conjunction with the AUTOPAD and VLANPAD bits.

Table 35-2: Pad Operation

Type	AUTOPAD	VLANPAD	PADENABLE	Action
Any	x	x	0	No pad, check CRC
Any	0	0	1	Pad to 60 Bytes, append CRC
Any	x	1	1	Pad to 64 Bytes, append CRC
Any	1	0	1	If untagged: Pad to 60 Bytes, append CRC If VLAN tagged: Pad to 64 Bytes, append CRC

Register 35-25: EMAC1IPGT: Ethernet Controller MAC Back-to-Back Interpacket Gap Register(1)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	U-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-1	R/W-0
	—	B2BIPKTGP<6:0>						

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-7 **Unimplemented:** Read as ‘0’

bit 6-0 **B2BIPKTGP<6:0>:** Back-to-Back Interpacket Gap bits

This is a programmable field representing the nibble time offset of the minimum possible period between the end of any transmitted packet, to the beginning of the next. In Full-Duplex mode, the register value should be the desired period in nibble times minus 3. In Half-Duplex mode, the register value should be the desired period in nibble times minus 6. In Full-Duplex the recommended setting is 0x15 (21d), which represents the minimum IPG of 0.96 μs (in 100 Mbps) or 9.6 μs (in 10 Mbps). In Half-Duplex mode, the recommended setting is 0x12 (18d), which also represents the minimum IPG of 0.96 μs (in 100 Mbps) or 9.6 μs (in 10 Mbps).

Note 1: Both 16- and 32-bit accesses are allowed to these registers (including the Set, Clear and Invert registers). 8-bit accesses are not allowed and are ignored by the hardware.

Register 35-26: EMAC1IPGR: Ethernet Controller MAC Non-Back-to-Back Interpacket Gap Register(1)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-0	R/W-0
	—	NB2BIPKTGP1<6:0>						
7:0	U-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-1	R/W-0
	—	NB2BIPKTGP2<6:0>						

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-15 **Unimplemented:** Read as ‘0’

bit 14-8 **NB2BIPKTGP1<6:0>:** Non-Back-to-Back Interpacket Gap Part 1 bits

This is a programmable field representing the optional carrierSense window referenced in section 4.2.3.2.1 “Deference” of the IEEE 802.3 Specification. If carrier is detected during the timing of IPGR1, the MAC defers to carrier. If, however, carrier becomes after IPGR1, the MAC continues timing IPGR2 and transmits, knowingly causing a collision, thus ensuring fair access to medium. Its range of values is 0x0 to IPGR2. Its recommend value is 0xC (12d).

bit 7 **Unimplemented:** Read as ‘0’

bit 6-0 **NB2BIPKTGP2<6:0>:** Non-Back-to-Back Interpacket Gap Part 2 bits

This is a programmable field representing the non-back-to-back Inter-Packet-Gap. Its recommended value is 0x12 (18d), which represents the minimum IPG of 0.96 μs (in 100 Mbps) or 9.6 μs (in 10 Mbps).

Note 1: Both 16- and 32-bit accesses are allowed to these registers (including the Set, Clear and Invert registers). 8-bit accesses are not allowed and are ignored by the hardware.

Register 35-27: EMAC1CLRT: Ethernet Controller MAC Collision Window/Retry Limit Register(1)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	R/W-1	R/W-1	R/W-0	R/W-1	R/W-1	R/W-1
	—	—	CWINDOW<5:0>					
7:0	U-0	U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1
	—	—	—	—	RETX<3:0>			

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-14 **Unimplemented:** Read as ‘0’

bit 13-8 **CWINDOW<5:0>:** Collision Window bits

This is a programmable field representing the slot time or collision window during which collisions occur in properly configured networks. Since the collision window starts at the beginning of transmission, the preamble and SFD is included. Its default of 0x37 (55d) corresponds to the count of frame bytes at the end of the window.

bit 7-4 **Unimplemented:** Read as ‘0’

bit 3-0 **RETX<3:0>:** Retransmission Maximum bits

This is a programmable field specifying the number of retransmission attempts following a collision before aborting the packet due to excessive collisions. The IEEE 802.3 Specification standard specifies the maximum number of attempts (attemptLimit) to be 0xF (15d). Its default is ‘0xF’.

Note 1: Both 16- and 32-bit accesses are allowed to these registers (including the Set, Clear and Invert registers). 8-bit accesses are not allowed and are ignored by the hardware.

35

Ethernet Controller

Register 35-28: EMAC1MAXF: Ethernet Controller MAC Maximum Frame Length Register[1,2]

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-1
	MACMAXF<15:8>[2]							
7:0	R/W-1	R/W-1	R/W-1	R/W-0	R/W-1	R/W-1	R/W-1	R/W-0
	MACMAXF<7:0>[2]							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-0 **MACMAXF<15:0>:** Maximum Frame Length bits[2]

This field resets to 0x05EE, which represents a maximum receive frame of 1518 octets. An untagged maximum size Ethernet frame is 1518 octets. A tagged frame adds four octets for a total of 1522 octets. If a shorter/longer maximum length restriction is desired, program this 16-bit field.

Note 1: Both 16- and 32-bit accesses are allowed to these registers (including the Set, Clear and Invert registers). 8-bit accesses are not allowed and are ignored by the hardware.

2: If a proprietary header is allowed, this field should be adjusted accordingly. For example, if 4-byte headers are prepended to frames, MACMAXF could be set to 1527 octets. This would allow the maximum VLAN tagged frame plus the 4-byte header.

Register 35-29: EMAC1SUPP: Ethernet Controller MAC PHY Support Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	R/W-0	U-0	U-0	R/W-0
	—	—	—	—	RESETRMII(2)	—	—	SPEEDRMII(2)
7:0	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: ('0', '1', x = Unknown)		

bit 31-12 **Unimplemented:** Read as '0'

bit 11 **RESETRMII:** Reset RMII Logic bit(2)
1 = Reset the MAC RMII module
0 = Normal Operation.

bit 10-9 **Unimplemented:** Read as '0'

bit 8 **SPEEDRMII:** RMII Speed bit(2)
This bit configures the Reduced MII logic for the current operating speed.
1 = RMII running in 100 Mbps
0 = RMII running in 10 Mbps

bit 7-0 **Unimplemented:** Read as '0'

Note 1: Both 16- and 32-bit accesses are allowed to these registers (including the Set, Clear and Invert registers). 8-bit accesses are not allowed and are ignored by the hardware.

2: These bits are used for the RMII module only.

Register 35-30: EMAC1TEST: Ethernet Controller MAC Test Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
	—	—	—	—	—	TESTBP	TESTPAUSE(2)	SHRTQNTA(2)

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: ('0', '1', x = Unknown)		

bit 31-3 **Unimplemented:** Read as '0'

bit 2 **TESTBP:** Test Backpressure bit

1 = The MAC will assert backpressure on the link. Backpressure causes preamble to be transmitted, raising carrier sense. A transmit packet from the system will be sent during backpressure.

0 = Normal Operation

bit 1 **TESTPAUSE:** Test PAUSE bit(2)

1 = The MAC Control sub-layer will inhibit transmissions, just as if a PAUSE Receive Control frame with a non-zero pause time parameter was received

0 = Normal Operation

bit 0 **SHRTQNTA:** Shortcut PAUSE Quanta bit(2)

1 = The MAC reduces the effective PAUSE Quanta from 64 byte-times to 1 byte-time

0 = Normal Operation

Note 1: Both 16- and 32-bit accesses are allowed to these registers (including the Set, Clear and Invert registers). 8-bit accesses are not allowed and are ignored by the hardware.

2: These bits are for testing only.

Register 35-31: EMAC1MCFG: Ethernet Controller MAC MII Management Configuration Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	RESETMGMT	—	—	—	—	—	—	—
7:0	U-0	U-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	CLKSEL<3:0>(2)				NOPRE	SCANINC

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15 **RESETMGMT:** Test Reset MII Management bit
1 = Reset the MII Management module
0 = Normal Operation

bit 14-6 **Unimplemented:** Read as ‘0’

bit 5-2 **CLKSEL<3:0>:** MII Management Clock Select 1 bits(2)
This field is used by the clock divide logic in creating the MII Management Clock (MDC), which the IEEE 802.3 Specification defines to be no faster than 2.5 MHz. Some PHYs support clock rates up to 12.5 MHz.

bit 1 **NOPRE:** Suppress Preamble bit
1 = The MII Management will perform read/write cycles without the 32-bit preamble field. Some PHYs support suppressed preamble
0 = Normal read/write cycles are performed

bit 0 **SCANINC:** Scan Increment bit
1 = The MII Management module will perform read cycles across a range of PHYs. The read cycles will start from address 1 through the value set in EMAC1MADR<PHYADDR>
0 = Continuous reads of the same PHY

Note 1: Both 16- and 32-bit accesses are allowed to these registers (including the Set, Clear and Invert registers). 8-bit accesses are not allowed and are ignored by the hardware.
2: Table 35-3 provides a description of the clock divider encoding.

Table 35-3: MIIM Clock Selection

MIIM Clock Select	EMAC1MCFG<5:2>
SYSCLK divided by 4	000x
SYSCLK divided by 6	0010
SYSCLK divided by 8	0011
SYSCLK divided by 10	0100
SYSCLK divided by 14	0101
SYSCLK divided by 20	0110
SYSCLK divided by 28	0111
SYSCLK divided by 40	1000
Undefined	Any other combination

Register 35-32: EMAC1MCMD: Ethernet Controller MAC MII Management Command Register(1)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
	—	—	—	—	—	—	SCAN	READ

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-2 **Unimplemented:** Read as ‘0’

bit 1 **SCAN:** MII Management Scan Mode bit
1 = The MII Management module will perform read cycles continuously (for example, useful for monitoring the Link Fail)
0 = Normal Operation

bit 0 **READ:** MII Management Read Command bit
1 = The MII Management module will perform a single read cycle. The read data is returned in the EMAC1MRDD register
0 = The MII Management module will perform a write cycle. The write data is taken from the EMAC1MWTD register

Note 1: Both 16- and 32-bit accesses are allowed to these registers (including the Set, Clear and Invert registers). 8-bit accesses are not allowed and are ignored by the hardware.

Register 35-33: EMAC1MADR: Ethernet Controller MAC MII Management Address Register(1)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1
	—	—	—	PHYADDR<4:0>				
7:0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	REGADDR<4:0>				

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-13 **Unimplemented:** Read as ‘0’

bit 12-8 **PHYADDR<4:0>:** MII Management PHY Address bits

This field represents the 5-bit PHY Address field of Management cycles. Up to 31 PHYs can be addressed (0 is reserved).

bit 7-5 **Unimplemented:** Read as ‘0’

bit 4-0 **REGADDR<4:0>:** MII Management Register Address bits

This field represents the 5-bit Register Address field of Management cycles. Up to 32 registers can be accessed.

Note 1: Both 16- and 32-bit accesses are allowed to these registers (including the Set, Clear and Invert registers). 8-bit accesses are not allowed and are ignored by the hardware.

Register 35-34: EMAC1MWTD: Ethernet Controller MAC MII Management Write Data Register(1)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	MWTD<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	MWTD<7:0>							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-0 **MWTD<15:0>:** MII Management Write Data bits

When written, a MII Management write cycle is performed using the 16-bit data and the pre-configured PHY and Register addresses from the EMAC1MADR register.

Note 1: Both 16- and 32-bit accesses are allowed to these registers (including the Set, Clear and Invert registers). 8-bit accesses are not allowed and are ignored by the hardware.

Register 35-35: EMAC1MRDD: Ethernet Controller MAC MII Management Read Data Register(1)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	MRDD<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	MRDD<7:0>							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-0 **MRDD<15:0>:** MII Management Read Data bits
Following a MII Management Read Cycle, the 16-bit data can be read from this location.

Note 1: Both 16- and 32-bit accesses are allowed to these registers (including the Set, Clear and Invert registers). 8-bit accesses are not allowed and are ignored by the hardware.

Register 35-36: EMAC1MIND: Ethernet Controller MAC MII Management Indicators Register(1)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	—	LINKFAIL	NOTVALID	SCAN	MIIMBUSY

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-4 **Unimplemented:** Read as ‘0’

bit 3 **LINKFAIL:** Link Fail bit

When ‘1’ is returned - indicates link fail has occurred. This bit reflects the value last read from the PHY status register.

bit 2 **NOTVALID:** MII Management Read Data Not Valid bit

When ‘1’ is returned - indicates an MII management read cycle has not completed and the Read Data is not yet valid.

bit 1 **SCAN:** MII Management Scanning bit

When ‘1’ is returned - indicates a scan operation (continuous MII Management Read cycles) is in progress.

bit 0 **MIIMBUSY:** MII Management Busy bit

When ‘1’ is returned - indicates MII Management module is currently performing an MII Management Read or Write cycle.

Note 1: Both 16- and 32-bit accesses are allowed to these registers (including the Set, Clear and Invert registers). 8-bit accesses are not allowed and are ignored by the hardware.

Register 35-37: EMAC1SA0: Ethernet Controller MAC Station Address 0 Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P
	STNADDR6<7:0>							
7:0	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P
	STNADDR5<7:0>							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-8 **STNADDR6<7:0>:** Station Address Octet 6 bits
This field holds the sixth transmitted octet of the station address.

bit 7-0 **STNADDR5<7:0>:** Station Address Octet 5 bits
This field holds the fifth transmitted octet of the station address.

Note 1: Both 16- and 32-bit accesses are allowed to these registers (including the Set, Clear and Invert registers). 8-bit accesses are not allowed and are ignored by the hardware.

2: This register is loaded at reset from the factory preprogrammed station address.

Register 35-38: EMAC1SA1: Ethernet Controller MAC Station Address 1 Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P
	STNADDR4<7:0>							
7:0	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P
	STNADDR3<7:0>							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Unimplemented:** Read as ‘0’

bit 15-8 **STNADDR4<7:0>:** Station Address Octet 4 bits
This field holds the fourth transmitted octet of the station address.

bit 7-0 **STNADDR3<7:0>:** Station Address Octet 3 bits
This field holds the third transmitted octet of the station address.

Note 1: Both 16- and 32-bit accesses are allowed to these registers (including the Set, Clear and Invert registers). 8-bit accesses are not allowed and are ignored by the hardware.

2: This register is loaded at reset from the factory preprogrammed station address.

Register 35-39: EMAC1SA2: Ethernet Controller MAC Station Address 2 Register(1,2)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P
	STNADDR2<7:0>							
7:0	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P	R/W-P
	STNADDR1<7:0>							

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Reserved:** Maintain as ‘0’; ignore read

bit 15-8 **STNADDR2<7:0>:** Station Address Octet 2 bits
This field holds the second transmitted octet of the station address.

bit 7-0 **STNADDR1<7:0>:** Station Address Octet 1 bits
This field holds the most significant (first transmitted) octet of the station address.

Note 1: Both 16- and 32-bit accesses are allowed to these registers (including the Set, Clear and Invert registers). 8-bit accesses are not allowed and are ignored by the hardware.

2: This register is loaded at reset from the factory preprogrammed station address.

35.4 OPERATION

The Ethernet Controller provides the system modules needed to implement a 10/100 Mbps Ethernet node using an external PHY chip. In order to off-load the CPU from moving packet data to and from the module, two internal descriptor-based DMA engines are included in the controller.

The Ethernet Controller consists of the following sub-modules:

- 10/100 Megabit Media Access Controller (MAC): Implements the Media Access Control (MAC) sub-layer of the Data Link Layer and performs the CSMA/CD function contained in the ISO/IEC 8802-3 and the IEEE 802.3 specifications. It includes:
  - Media Independent Interface to connect to an external PHY
  - Reduced Media Independent Interface to connect to an external PHY
  - MII Management block that provides control/status connection to the external MII PHY
  - Performs the receive path flow control functions contained in Annex 31B of the IEEE 802.3 Specification
  - Implements the MAC Transmit and MAC Receive interfaces that connect with the TX and RX DMA engines
- Flow Control (FC): Responsible for control of the transmission of PAUSE frames as defined in Annex 31B of the IEEE 802.3 Specification
- RX Filter (RXF): This block performs multiple filters on every receive packet to determine whether each packet should be accepted or rejected
- TX DMA/TXBM Engine: The TX DMA engine and TX Buffer Management engine perform data transfers from the packet buffers to the MAC Transmit Interface and also transfers the Transmit Status Vector (TSV) from the MAC to the packet buffers once the transmission is done. It operates using the TX Descriptor tables.
- RX DMA/RXBM Engine: The RX DMA engine and RX Buffer Management engine transfer receive packets and the Receive Status Vector (RSV) from the MAC to the packet buffers using the RX Descriptor tables

35.4.1 Ethernet Frame Overview

IEEE 802.3-compliant Ethernet frames (packets) are between 64 and 1518 bytes long (Preamble and Start -of-Frame Delimiter not included). Frames containing less than 64 bytes are known as “runt” frames, while frames containing more than 1518 bytes are known as “huge” frames.

An Ethernet frame is made up of the following fields:

- Start-of-Stream/Preamble
- Start-of-Frame Delimiter (SFD)
- Destination MAC address (DA)
- Source MAC address (SA)
- Type/Length field
- Data Payload
- Optional Padding field
- Frame Check Sequence (FCS)

Traffic on the actual physical cable is depicted in Figure 35-2. Please refer to the IEEE 802.3 Specification for detailed information about the Ethernet protocol.

35.4.1.1 START-OF-STREAM/PREAMBLE AND START-OF-FRAME DELIMITER

When transmitted on the Ethernet medium, The Start-of-Stream/Preamble and the SFD fields are appended to the beginning of an Ethernet frame automatically by the MAC.

When receiving, these fields are automatically stripped from the received frames so that these fields are not written into the RX data buffers.

The software does not need to process/generate these fields.

35.4.1.2 DESTINATION MAC ADDRESS

A MAC address is a 6-octet number representing the physical address of the node(s) on an Ethernet network. The destination address contains the MAC address of the device for which the frame is intended. There are different types of addresses in the Ethernet space. For example:

- Unicast Address: Designated for usage by the addressed node only. A Unicast address is an address where the Least Significant bit in the first byte of the address is zero (i.e., the first byte of the address is even). For example, “00 04 a3 00 00 01” is a Unicast address but “01 04 a3 00 00 01” is not.
- Multicast Address: Designated for use by a selected group of Ethernet nodes. A Multicast address is an address where the Least Significant bit in the first byte of this address is set (i.e., the byte is odd). For example, “01 04 a3 00 00 01” is a Multicast address. Note that the Multicast address, “FF-FF-FF-FF-FF-FF”, is reserved (Broadcast address) and is directed to all nodes on the network.

The Ethernet Controller incorporates a Receive Filter module that can be configured to accept or discard Unicast, Multicast and/or Broadcast frames. For details about the receive filters, refer to **35.4.8 “Receive Filtering Overview”**.

35.4.1.3 SOURCE MAC ADDRESS

The source address is the 6-byte field MAC address of the node that transmitted the Ethernet frame. Every Ethernet device must have a globally unique MAC address. Each PIC32 including an Ethernet Controller has a unique address, which is loaded into the MAC registers on power-up. This value can be used as is, or the registers may be reconfigured with a different address at run time.

35.4.1.4 TYPE/LENGTH

This is a 2-byte field indicating the protocol to which the frame belongs. Applications using standards such as Internet Protocol (IP) or Address Resolution Protocol (ARP) should use the type code specified in the appropriate standards document. Alternately, this field can be used as a length field when implementing proprietary networks. Typically, any value of 1500 (0x05DC) or smaller is considered to be a length field and specifies the amount of non-padding data, which follows in the data field.

35.4.1.5 DATA

The data field typically consists of between 0 and 1500 bytes of payload data for each frame. PIC32 devices are capable of transmitting and receiving frames larger than this when the HUGEFRM bit (EMAC1CFG2<2>) is set. However, these larger frames that do not meet the IEEE 802.3 Specification will likely be dropped by most Ethernet nodes.

35.4.1.6 PADDING

The padding field is a variable length field appended to meet IEEE 802.3 Specification requirements when transmitting small data payloads. The minimum payload for an Ethernet frame 46 bytes. Smaller frames must be padded to fill this space. For transmitted frames, the software can instruct the Ethernet Controller to automatically generate the needed padding by using the PADENABLE, VLANPAD and AUTOPAD bits (EMAC1CFG2<5:7>). However, if the auto-padding is not enabled and the application does not provide appropriate padding, the PIC32 device will not prevent the transmission of these “runt” frames. When receiving frames, PIC32 devices accept and write all padding to the receive buffer. Frames shorter than the required 64 bytes can optionally be filtered by the Runt Error Reject filter, described in **35.4.8.4 “Runt Rejection Filter ”**.

35.4.1.7 FRAME CHECK SEQUENCE (FCS)

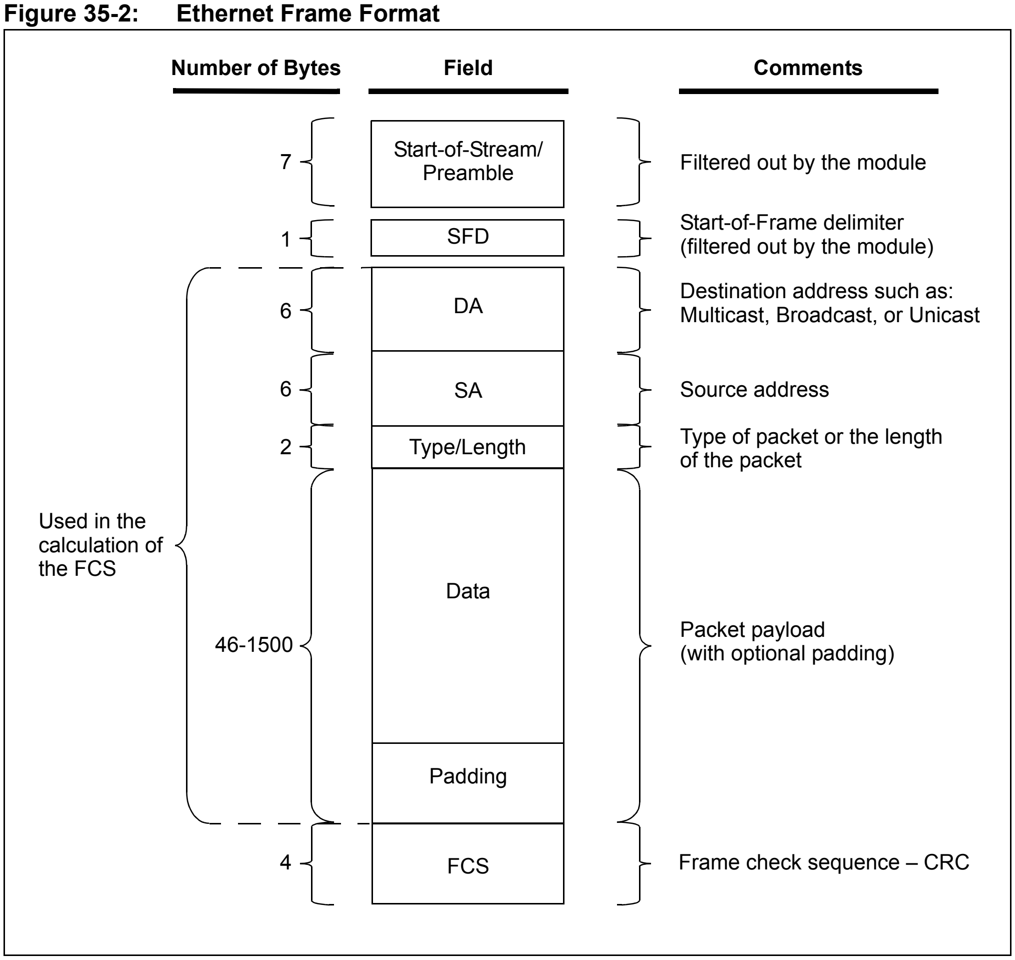
The FCS is a 4-byte field containing a standard 32-bit CRC calculated over the Destination, Source, Type/Length, Data and Padding fields. It allows for the detection of transmission errors.

For transmitted frames, PIC32 devices can automatically generate and append a valid FCS by using the CRCENABLE bit (EMAC1CFG2<4>). Otherwise, the software must calculate the CRC for the frame to be transmitted and append it properly.

For received frames, the FCS field is stored to the receive buffer. Frames with invalid CRC values can either be discarded or accepted using the CRC Error and CRC Check Acceptance filters described in **35.4.8.1 "CRC Error Acceptance Filter "** and **35.4.8.3 "CRC Check Acceptance Filter"**.

> **Note:** The polynomial for generating the FCS is:
>
> $$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$
>
> The FCS is transmitted starting with bit 31 and ending with bit 0.

Figure 35-2: Ethernet Frame Format



35.4.2 Basic Ethernet Controller Operation

The Ethernet Controller is enabled by setting the ON bit (ETHCON1<15>).

The Ethernet Controller is disabled by clearing the ON bit in the ETHCON1 register. This is the default state after any reset. If the Ethernet Controller is disabled, all of the I/O pins used for the MII/RMII and MIIM interfaces operate as port pins and are under the control of the respective PORT latch bit and TRIS bit.

Disabling the controller resets the internal DMA state machines and all transmit and receive operations are aborted. The SFRs are still accessible and their values preserved.

Clearing the ON bit while the Ethernet Controller is active will abort all pending operations and reset the peripheral as defined above.

Re-enabling the ON bit will restart the Ethernet Controller in its clean reset state while preserving the SFRs values.

> **Note 1:** If the ON bit is cleared during an active internal bus transaction, the controller will complete the current bus transaction before entering the disabled state. Once the controller is disabled, the TXBUSY bit (ETHSTAT<6>) and RXBUSY bit (ETHSTAT<5>) will reflect an inactive status.
>
> **2:** Whenever the Ethernet Controller is reset via the ON bit, software should also reset the external PHY via the MIIM registers. This insures the PHY is in a known initialized state. In addition, the MAC should also be soft reset via the EMAC1CFG1 register.

35.4.3 MAC Overview

The MAC sub-layer is part of the functionality described in the OSI model for the Data Link Layer. It defines a medium-independent facility, built on the medium-dependent physical facility provided by the Physical Layer, and under the access-layer-independent LLC sub-layer or other MAC client. It is applicable to a general class of local area broadcast media suitable for use with the Carrier Sense Multiple Access with Collision Detection (CSMA/CD).

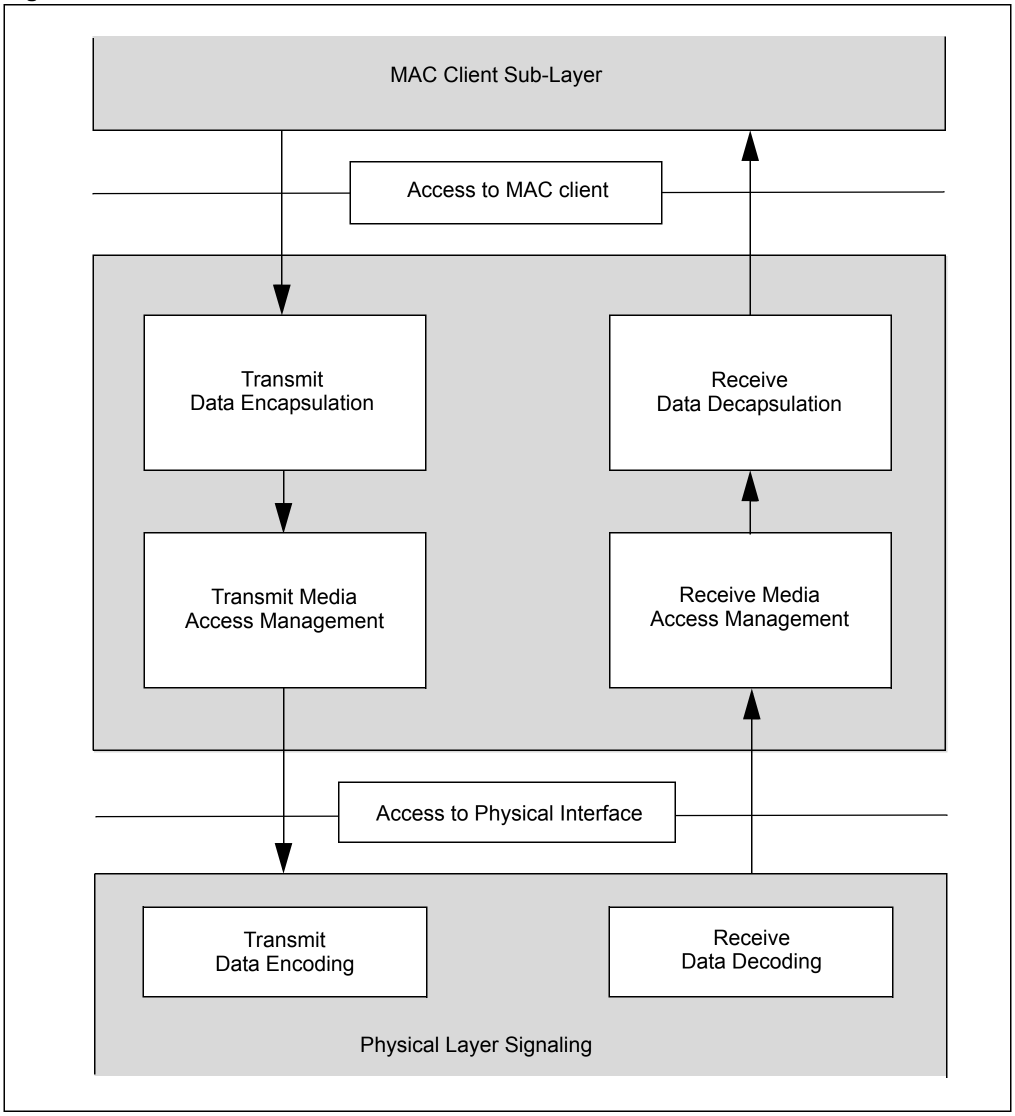
The CSMA/CD MAC sub-layer provides services to the MAC client required for the transmission and reception of frames. The CSMA/CD MAC sub-layer makes a best effort to acquire the medium and transfer a serial stream of bits to the Physical Layer. Although certain errors are reported to the client, error recovery is not provided by the MAC.

The following is a summary of the functional capabilities of the CSMA/CD MAC sub-layer as shown in Figure 35-3:

- For Frame Transmission:
  - Accepts data from the MAC client and constructs a frame
  - Presents a bit-serial data stream to the Physical Layer for transmission on the medium
  - In half-duplex mode, defers transmission of a bit-serial stream whenever the physical medium is busy
  - It can append proper Frame Check Sequence (FCS) value to outgoing frames and verifies full octet boundary alignment
  - Delays transmission of frame bit stream for specified inter-frame gap period
  - In half-duplex mode, halts transmission when collision is detected
  - In half-duplex mode, schedules retransmission after a collision until a specified retry limit is reached
  - In half-duplex mode, enforces collision to ensure propagation throughout network by sending jam message
  - Prepends preamble and Start Frame Delimiter and appends FCS to all frame Appends PAD field for frames whose data length is less than a minimum value

- For Frame Reception:
  - Receives a bit-serial data stream from the Physical Layer
  - Presents the received frames to the MAC client (broadcast, multicast, unicast frames, etc.)
  - Checks incoming frames for transmission errors by way of FCS and verifies octet boundary alignment
  - Removes preamble, Start Frame Delimiter and PAD field (if necessary) from received frames
  - Implements the MII Management block that provides control/status connection to the external MII PHY

Figure 35-3: CSMA/CD Media Access Control Functions



The MAC is accessed using the following SFRs: EMAC1CFG1, EMAC1CFG2, EMAC1IPGT, EMAC1IPGR, EMAC1CLRT, EMAC1MAXF, EMAC1SUPP, EMAC1TEST and the EMAC1SA0 to EMAC1SA3 registers.

Note:	For a detailed explanation of the MAC sub-layer functions and operation, please see Clause 2, Clause 3 and Clause 4 of the IEEE 802.3 Specification.

35.4.4 Media Independent Interface (MII)

The Media Independent Interface (MII) is a standard interconnection between the MAC and the PHY for communicating TX and RX frame data. This MII has the following important characteristics:

- Capable of supporting 10/100 Mbps rates for data transfer, and offers support for management functions
- Provides independent four bit wide transmit and receive data paths
- Uses TTL signal levels, compatible with common digital CMOS processes
- Provides full-duplex operation

In 10 Mbps mode, the MII runs at 2.5 MHz; in 100 Mbps mode it runs at 25 MHz. PHYs that provide MII are not required to support both data rates, and may support either one or both.

Table 35-4 provides a list of the 18 Media Independent Interface signals.

Table 35-4: MII Signals

Signal Name	IEEE 802.3 MII Signals	Width	Type	Description
ETXCLK	TX_CLK	1	Input	The transmit clock signal is a continuous clock that provides the timing reference for the transfer of the ETXEN, ETXD and ETXERR signals from the MAC to the PHY. The ETXCLK frequency is a quarter of the nominal transmit data rate. A PHY operating at 100 Mbps must provide a ETXCLK frequency of 25 MHz, and a PHY operating at 10 Mbps must provide a ETXCLK frequency of 2.5 MHz.
ERXCLK	RX_CLK	1	Input	The receive clock signal is a continuous clock that provides the timing reference for the transfer of the ERXDV, RXD and ERXERR signals from the PHY to the MAC. ERXCLK has a frequency equal to a quarter of the data rate of the received signal.
ETXEN	TX_EN	1	Output	The transmit enable signal indicates that the MAC is presenting nibbles on the MII for transmission. ETXEN transitions synchronously with respect to ETXCLK.
ETXD<3:0>	TXD<3:0>	4	Output	The transmit data signals transition synchronously with respect to the ETXCLK.
ETXERR	TX_ER	1	Output	The transmit coding error signal is synchronous with respect to the ETXCLK. When ETXERR is asserted for one or more ETXCLK periods while ETXEN is also asserted, the PHY will emit one or more symbols that are not part of the valid data or delimiter set somewhere in the frame being transmitted. This signal affects only 100 Mbps data transmission.
ERXDV	RX_DV	1	Input	The receive data valid signal indicates that the PHY is presenting recovered and decoded nibbles on the RXD data lines. ERXDV is synchronous with ERXCLK. ERXDV remains asserted for the entire frame.
ERXD<3:0>	RXD<3:0>	4	Input	The receive data signals represents the four data signals synchronous with respect to ERXCLK. For each ERXCLK period in which ERXDV is asserted, RXD<3:0> transfers four bits of recovered data from the PHY to the MAC.
ERXERR	RX_ER	1	Input	The receive error signal is asserted to indicate to the MAC that a coding error (or any error that the PHY is capable of detecting) has occurred in the frame being transferred from the PHY to the MAC. ERXERR is synchronous with ERXCLK.
ECRS	CRS	1	Input	The carrier sense signal is asserted by the PHY when either the transmit or receive medium is non-idle. CRS will be de-asserted by the PHY when both the transmit and receive media are idle. The CRS remains asserted throughout the duration of a collision condition. It does not have to be synchronous with either the ETXCLK or the ERXCLK.
ECOL	COL	1	Input	The collision detected signal is asserted by the PHY upon detection of a collision on the medium, and remains asserted while the collision condition persists. It does not have to be synchronous with respect to either ETXCLK or ERXCLK.

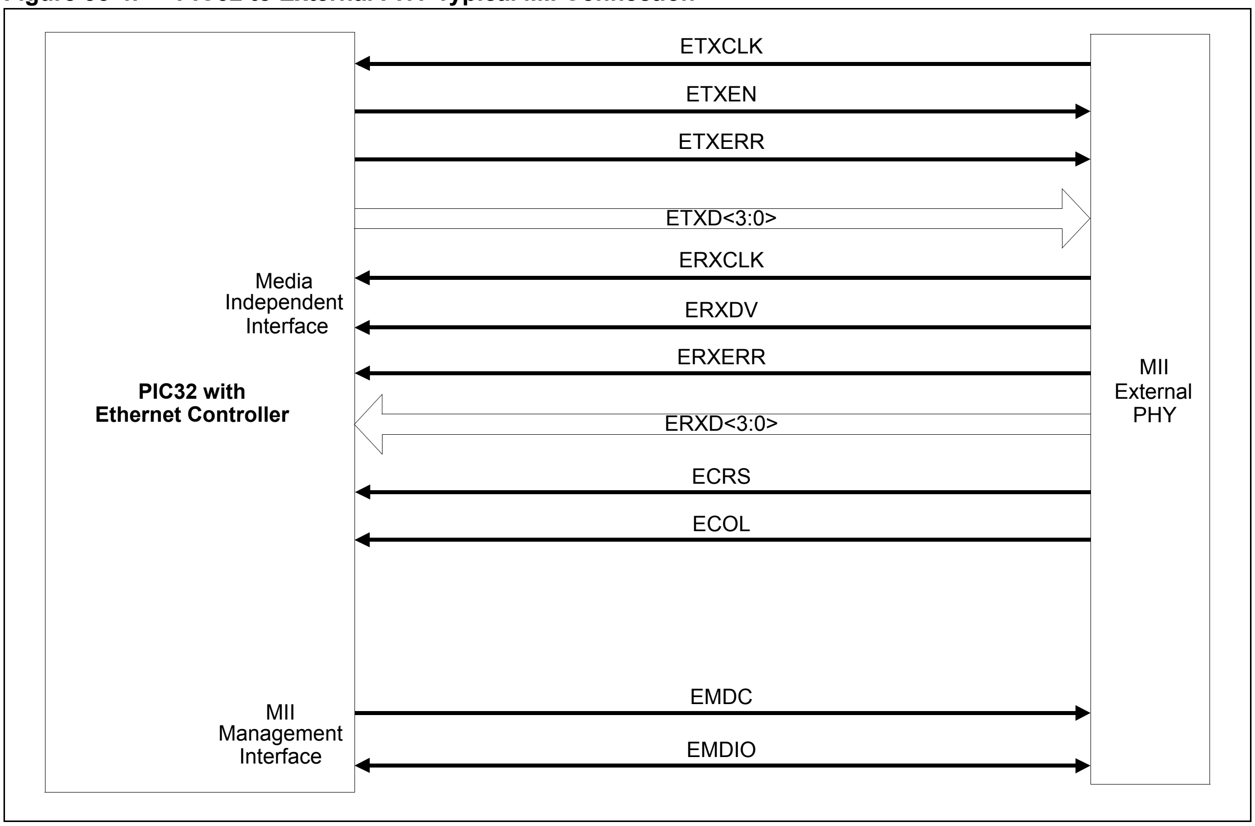
Table 35-4: MII Signals (Continued)

Signal Name	IEEE 802.3 MII Signals	Width	Type	Description
EMDC	MDC	1	Output	The management data clock signal is part of the MII Management interface and is explained in **35.4.6 “Media Independent Interface Management (MIIM)”**.
EMDIO	MDIO	1	Input/ Output	The management data input/output signal is part of the MII Management interface and is explained in **35.4.6 “Media Independent Interface Management (MIIM)”**.

For detailed MII specifications, refer to Clause 22 of the IEEE 802.3 Specification.

Figure 35-4 shows a typical MII connection between the PIC32 and the external PHY.

Figure 35-4: PIC32 to External PHY Typical MII Connection



35.4.5 Reduced Media Independent Interface (RMII)

The Reduced Media Independent Interface (RMII) is intended Specification.

The management interface (MDIO/MDC) is assumed to be identical to that defined in MII.

The RMII has the following characteristics:

- Capable of supporting 10 Mbps and 100 Mbps data rates
- Single clock reference for both MAC and the PHY (can be sourced from the MAC or from an external source)
- Provides independent 2 bit wide transmit and receive data paths
- Uses TTL signal levels, compatible with common digital CMOS processes
- Provides full-duplex operation

The interface runs at 50 MHz.

Table 35-5 provides a list of the 10 Reduced Media Independent Interface (RMII) signals.

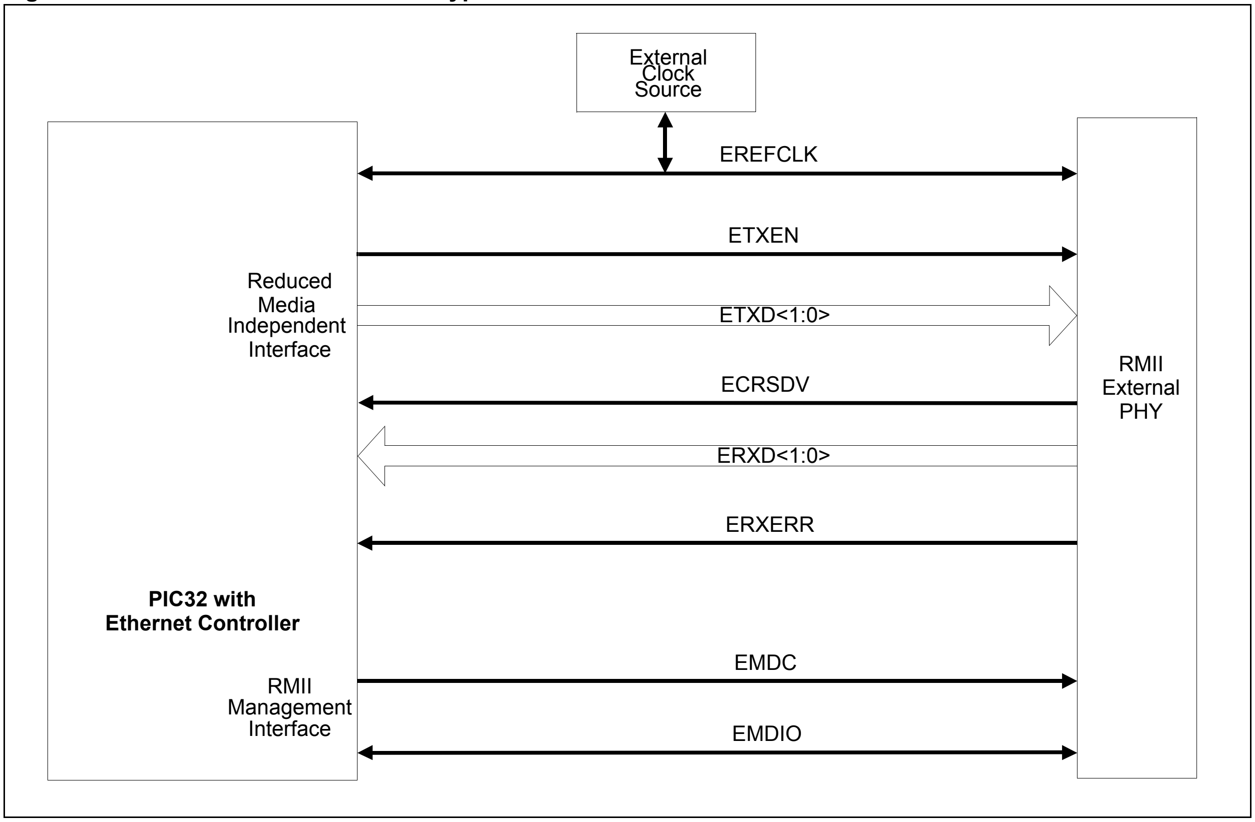
Table 35-5: RMII Signals

Signal Name	IEEE 802.3 RMII Signals	Width	Type	Description
EREFCLK	REF_CLK	1	Input	The reference clock signal is a continuous clock that provides the timing reference for ECRSDV, RXD<1:0>, ETXEN, ETXD<1:0> and ERXERR. EREFCLK is a 50 MHz clock signal sourced by the MAC or an external source. For PIC32 devices, the EREFCLK is an external supplied clock signal.
ECRSDV	CRS_DV	1	Input	The carrier sense/receive data valid signal is asserted by the PHY when the receive medium is non-idle. ECRSDV is asserted asynchronously on detection of carrier. Loss of carrier results in the de-assertion of ECRSDV synchronous to the REF_CLK (only on nibble boundaries). The data on RXD<1:0> is considered valid once ECRSDV is asserted. Using the ECRSDV the MAC can accurately recover ERXDV and CRS.
ERXD<1:0>	RXD<1:0>	2	Input	The receive data signals transition synchronously to EREFCLK. For each clock period in which ECRSDV is asserted, ERXD transfers two bits of recovered data from the PHY. • ERXD is '00' to indicate the idle condition when ECRSDV is deasserted. Since the use of the PHY signal ERXERR is optional, in order to ensure the propagation of errors for the received signal, the ERXD replaces the data in the decoded stream with '01' so that the MAC CRC mechanism will reject the frame. • In 100 Mbps ERXD is synchronous to the EREFCLK • In 10 Mbps the ERXD is sampled every tenth cycle
ETXEN	TX_EN	1	Output	The transmit enable signal indicates that the MAC is presenting di-bits on ETXD<1:0> for transmission. ETXEN is asserted with the first nibble of the preamble and remains asserted while all di-bits are transmitted. ETXEN is synchronous with respect to EREFCLK.
ETXD<1:0>	TXD<1:0>	2	Output	The transmit data signal is transmits data to the PHY when ETXEN is asserted. The ETXD data lines transition synchronously with respect to EREFCLK. ETXD uses the value of '00' to signal idle when the ETXEN is deasserted. • In 100 Mbps mode, ETXD provides valid data for each EREFCLK period while ETXEN is asserted • In 10 Mbps mode since the EREFCLK frequency is 10 times the data rate the value on ETXD is sampled every tenth cycle
ERXERR	RX_ER	1	Input	The receive error signal is asserted for one or more EREFCLK periods to indicate that an error was detected somewhere in the frame presently being transferred from the PHY. The ERXERR is synchronous with respect to EREFCLK.
EMDC	MDC	1	Output	The management data clock signal is part of the MII Management interface and is explained in **35.4.6 "Media Independent Interface Management (MIIM)"**.
EMDIO	MDIO	1	Input/Output	The management data input/output signal is part of the MII Management interface and is explained in **35.4.6 "Media Independent Interface Management (MIIM)"**.

Refer the RMII Specification sponsored by the RMII Consortium for detailed specifications of the RMII.

Figure 35-5 shows a typical RMII connection between the PIC32 and the external PHY.

Figure 35-5: PIC32 to External PHY Typical RMII Connection



35.4.6 Media Independent Interface Management (MIIM)

The MII Management (MIIM) module provides a serial communication link between the PIC32 host and an external MII PHY device. The external serial communications link operates in accordance with Clause 22 of the IEEE 802.3 Specification.

The MIIM input/output signals are:

- Management Data Clock (MDC) – MDC is sourced by the MAC to the PHY as the timing reference for transfer of information on the MDIO signal
- Management Data Input/Output (MDIO) – MDIO is a bidirectional signal between the PHY and the MAC. It is used to transfer control information and status between the PHY and the MAC. Control information is driven by the MAC synchronously with respect to MDC and is sampled synchronously by the PHY. Status information is driven by the PHY synchronously with respect to MDC and is sampled synchronously by the MAC.

The communication over the MIIM link takes place in frames. Frames transmitted on the MII Management Interface have the following structure (see Table 35-6):

- Preamble: At the beginning of each transaction, the MAC sends a sequence of 32 logic one bits on MDIO to provide the PHY with a synchronization pattern
- Start-of-frame: The start-of-frame is indicated by a <01> pattern
- Operation code: <10> for a read transaction, <01> for a write transaction
- PHY Address: Five bits, allowing 32 unique PHY addresses. A PHY will always respond to transactions with address zero.
- Register Address: Five bits, allowing 32 individual registers to be addressed within each PHY
- Turnaround: A 2-bit-time spacing between the Register Address field and the Data field of a management frame to avoid contention during a read transaction
- Data: This 16-bit field carries the data to/from the addressed PHY register

Note:	The IDLE condition on MDIO is a high-impedance state.

Table 35-6: MIIM Frame Format

Operation	Management Frame Fields							
	PRE	ST	OPCODE	PHYAD	REGAD	TA	DATA	IDLE
READ	1….1	01	10	a0…a4	r0…r4	Z0	d0….d15	Z
WRITE	1….1	01	01	a0…a4	r0…r4	10	d0….d15	Z

As indicated previously, the size of a MIIM frame is 64 bits. However, the MIIM module may be configured to suppress the preamble portion of the MII Management serial stream using the NOPRE bit (EMAC1MCFG<1>) when the PHY supports a suppressed preamble operation.

Refer to Clause 22 in the IEEE 802.3 Specification for MIIM details.

35.4.6.1 EXTERNAL PHY REGISTER ACCESS

The PHY registers provide configuration and control of the PHY module, as well as status information about its operation. Unlike the on-chip SFRs, the PHY registers are not directly accessible through the SFR control interface. Instead, access is accomplished through a special set of MAC control registers that implement the Media Independent Interface Management. These control registers are referred to as the MIIM registers. The PHY registers are accessed through the MIIM interface of the MAC. To do this, the MII Management Command, Address and Data registers in the MAC must be used.

The registers that control access to the PHY registers are listed in the Ethernet Control Register Summary (Table 35-1) and include the following registers:

- **EMAC1MCFG: Ethernet Controller MAC MII Management Configuration Register**[1,2]
- **EMAC1MCMD: Ethernet Controller MAC MII Management Command Register**[1]
- **EMAC1MADR: Ethernet Controller MAC MII Management Address Register**[1]
- **EMAC1MWTD: Ethernet Controller MAC MII Management Write Data Register**[1]
- **EMAC1MRDD: Ethernet Controller MAC MII Management Read Data Register**[1]
- **EMAC1MIND: Ethernet Controller MAC MII Management Indicators Register**[1]

> **Note:** All PHY chip registers are treated as 16 bits in width. Writes to unimplemented locations are ignored and any attempts to read these locations will return ‘0’. All reserved locations should be written as ‘0’; their contents should be ignored when read. Refer to the vendor-specific PHY data sheet for register access details.

35.4.6.2 INITIALIZING THE MII MANAGEMENT MODULE

In order for the MAC MIIM module to create the proper interface clock (MDC) frequency the clock speed needs to be configured. The MIIM module uses the SYSCLK as an input clock.

Use the CLKSEL bits (EMAC1MCFG<2:5>) to select the divider for creating the MDC clock signal, which the IEEE 802.3 Specification defines to be no faster than 2.5 MHz. However, some PHYs support clock rates up to 12.5 MHz.

35.4.6.3 READING A PHY REGISTER

When a PHY register is read via the MAC, the entire 16 bits are obtained.

To read from a PHY register, do the following:

1. Write the address of the PHY and of the PHY register to read from into the EMAC1MADR register.
2. Set the READ bit (EMAC1MCMD<0>). The read operation begins and the MIIMBUSY bit (EMAC1MIND<0>) will be set after three SYSCLK periods (this is due to the internal pipeline of the MIIM interface).
3. Poll the MIIMBUSY bit to be certain that the operation is complete (the operation time is the one needed to transfer a full MIIM frame). While busy, the software should not start any MII scan operations or write to the EMAC1MWTD register. When the MAC has obtained the register contents, the MIIMBUSY bit will clear itself.
4. Clear the READ bit (EMAC1MCMD<0>).
5. Read the desired data from the EMAC1MRDD register.

35.4.6.4 WRITING A PHY REGISTER

When a PHY register is written to, all 16 bits are written at once; selective bit writes are not implemented. If it is necessary to reprogram only select bits in the register, the software must first read the PHY register, modify the resulting data, and then write the data back to the PHY register.

To write to a PHY register, do the following:

1. Write the address of the PHY and of the PHY register to read from into the EMAC1MADR register.
2. Write the 16 bits of data to be written into the EMAC1MWTD register. Writing to this register automatically begins the MIIM transaction, which causes the MIIMBUSY bit (EMAC1MIND<0>) to be set after three SYSCLK periods (this is due to the internal pipeline of the MIIM interface).
3. Poll the MIIMBUSY bit until it is cleared, which indicates the write has completed.
4. The PHY register will be written after the MIIM operation completes, which takes a MIIM frame time. When the write operation has completed, the MIIMBUSY bit will clear itself. The software should not start any MII scan or read operations while busy.

35.4.6.5 SCANNING A PHY REGISTER

The MAC can be configured to perform automatic back-to-back read operations on a PHY register. This can significantly reduce the software complexity when periodic status information updates are desired.

To perform the scan operation, do the following:

1. Write the address of the PHY and the address of the PHY register to be read from into the EMAC1MADR register.
2. Set the SCAN bit (EMAC1MCMD<1>). The scan operation begins and the MIIMBUSY (EMAC1MIND<0>) bit is set.
3. The first read operation will complete after the first MIIM frame is transferred. Subsequent reads will be done at the same interval until the operation is cancelled. The NOTVALID bit (EMAC1MIND<2>) may be polled to determine when the first read operation is complete. Read the scanned register data from the EMAC1MRDD register.
4. After setting the SCAN bit, the EMAC1MRDD register will automatically be updated every MIIM frame interval. There is no status information which can be used to determine when the EMAC1MRDD register is updated.
5. When the MIIM scan operation is in progress, the software must not attempt to write to EMAC1MWTD or start a read operation.
6. The MIIM scan operation can be cancelled by clearing the SCAN bit (EMAC1MCMD<1>) and then polling the MIIMBUSY bit. New operations may be started after the MIIMBUSY bit is cleared.

Example 35-1 provides example code for a MIIM initialization and PHY register read, write and scan.

Example 35-1: MIIM Initialization and PHY Access

```
// Assume we're running at 80 MHz and we're working with a PHY that supports a maximum
// 2.5 MHz MIIM frequency

#include <p32xxxx.h>
#define PHY_ADDRESS    0x1f                  // the address of the PHY

    EMAC1MCFG=0x00008000;                    // issue reset
    EMAC1MCFG=0;                             // clear reset
    EMAC1MCFG=(0x8)<<2;                      // program the MIIM clock, divide by 40

    // read the basic status PHY register: 1
    unsigned int phyRegVal;

    while(EMAC1MIND&0x1);                    // wait not busy
    EMAC1MADR=0x1|((PHY_ADDRESS)<<8);        // set the PHY and register address
    EMAC1MCMD=1;                             // issue the read order
    __asm__ __volatile__ ("nop; nop; nop;"); // wait busy to be set
    while(EMAC1MIND&0x1);                    // wait op complete
    EMAC1MCMD=0;                             // clear command register
    phyRegVal=EMAC1MRDD;                     // read the selected register

    // write the basic control PHY register: 0
    while(EMAC1MIND&0x1);                    // wait in case of some previous operation
    EMAC1MADR=0x0|((PHY_ADDRESS)<<8);        // set the PHY and register address
    EMAC1MWT=0x8000;                         // issue the write order (PHY reset)
    __asm__ __volatile__ ("nop; nop; nop;"); // wait busy to be set
    while(EMAC1MIND&0x1);                    // wait write complete

    // Make sure data has been written

    // Perform a scan of the status PHY register: 1
    // Start the scan
    while(EMAC1MIND&0x1);                    // wait in case of some previous operation
    EMAC1MADR=0x1|((PHY_ADDRESS)<<8);        // set the PHY and register address
    EMAC1MCMD=0x2;                           // issue the scan order

    // Read the status register

    // Note that the read can occur now at any time
    // without previously selecting the read operation and the register
    while(EMAC1MIND&0x4);                    // wait data valid
    phyRegVal=EMAC1MRDD;                     // read the scanned register
```

35.4.7 Flow Control Overview

Ethernet flow control consists of the ability to send and receive PAUSE frames, which cause the receiving node to stop transmitting for a specific amount of time.

On the transmit side, the Flow Control (FC) block handles the hardware handshaking between the MAC and the CPU when the transmit flow control is enabled. Flow control for the received packets is part of the MAC functionality.

The PIC32 MAC supports both Symmetric PAUSE and Asymmetric PAUSE as described in Clause 28, Table 28B-2, and Clause 31 and Annex 31B of the IEEE 802.3 Specification.

The FC block supports two modes of operation: manual and automatic. In addition, the mode of transmission (Full-Duplex or Half-Duplex) programmed into the MAC registers, is used by the FC block.

Note:	The software should not change the Full-Duplex or Half-duplex mode of operation while the transmit logic is in the middle of transmitting a package.

Before software can throttle down incoming packets, it must enable flow control. The flow control mechanism operates differently between Full-Duplex and Half-Duplex modes.

35.4.7.1 FULL-DUPLEX

On the transmit side the MAC will send a PAUSE control frame with a PAUSE Timer value. The receiving MAC decodes the control frame, extracts the PAUSE Timer value and stalls transmission for the designated time. Note that this does not imply the transmitting device will pause immediately. There is latency in the activation of the pause mechanism. If flow control is to be deactivated before the PAUSE Timer value expires, another PAUSE frame can be sent that encodes a value of 0x0000 for the PAUSE Timer value.

Looking at the operation from the receiving end, if another device transmits a PAUSE frame and the MAC receives a PAUSE frame, then the transmit operation will be inhibited until the PAUSE Timer expires or the other device cancels the request for pause frames.

> **Note:** A PAUSE frame includes the period of pause time being requested, in the range of 0 through 65535. The pause time is measured in units of pause "quanta", where each unit is equal to 512 bit times.

35.4.7.2 HALF-DUPLEX

Half-duplex flow control is similar in operation. When the software enables the flow control the FC block requests the MAC to apply backpressure. The MAC will continue sending a preamble pattern on the transmit line to prevent any other device from gaining control of the bus. This will continue until flow control is disabled.

35.4.7.3 MANUAL FLOW CONTROL

The manual flow control is enabled via the MANFC (ETHCON1<4>) control bit.

When manual flow control is enabled, the MAC sends PAUSE control frames using the PTV (ETHCON1<31:16>) value. When transmit flow control is disabled, the MAC will send another PAUSE frame that encodes a value of 0x0000 for the PAUSE Timer value in order to disable flow control.

35.4.7.4 AUTOMATIC FLOW CONTROL

The automatic flow control is enabled via the AUTOFC control bit (ETHCON1<7>). When automatic flow control is enabled, PAUSE control frames are sent by hardware based on the current value of the BUFCNT bit (ETHSTAT<23:16>) as follows:

- When BUFCNT reaches the value specified by the RXFWM bit (ETHRXWM<23:16>), a PAUSE frame is automatically sent every 512/2 * PTV (ETHCON1<31:16>) transmit clock cycles.

> **Note 1:** The transmit clock cycle is 10 MHz or 100 MHz depending on the current MAC speed selection: 10 Mbps or 100 Mbps.
>
> **2:** Software must insure that the flow control watermark values allow PAUSE frames to be sent when the amount of free space allocated by the free RX descriptors drops below two times maximum Ethernet frame size (i.e., 1536 * 2). This will insure there is no receive overflow conditions.
>
> **3:** The PTV value may only be changed when the operation is not enabled ETHCON1<15> = 0.

- When BUFCNT reaches the value specified by the RXEWM bit (ETHRXWM<7:0>), a PAUSE frame with the PAUSE Timer Value set to 0x0000 is sent

Note that the BUFCNT value is updated only on a packet boundary; therefore, all automatic flow control changes occur on packet boundaries.

When automatic flow control is enabled, it has the highest priority for setting and clearing flow control operations. Therefore, it is not recommended to mix automatic and manual flow control.

The sequence of steps needed to manually transmit a PAUSE frame are:

1. In the initialization sequence software sets the PAUSE value by writing the PTV value (ETHCON1<31:16>).
2. Software writes the MANFC bit (ETHCON1<4>) to manually start the transmission of a PAUSE frame.
3. The FC block will request the MAC to send a PAUSE frame.
4. The MAC will assemble the complete Flow Control frame, as follows:
   a) Preamble
   b) Start Frame Delimiter (SFD)
   c) Destination Address = 01-80-c2-00-00-01 (Special PAUSE multi-cast address)
   d) Source Address = Station Address from EMAC1SA0-EMAC1SA3 registers
   e) Length = 0x8088 (Control Frame)
   f) Payload:
      - Opcode (2 bytes) = 0x0001
      - PAUSE Value (2 bytes) = PTV
   g) Pad
   h) FCS

Example 35-2: Using Manual Flow Control Code

```
// Note: Setting the new PTV value should be done only when the
// peripheral is not enabled
#include <p32xxxx.h>
ETHCON1CLR=0xffff0000;     // clear PTV
ETHCON1SET=(ptvVal)<<16;   // set the new PTV value

/*....*/
ETHCON1SET=0x10;           // turn on the Manual Flow Control
                           // at this moment PAUSE Frames are being sent
                           // or backpressure is applied
// do some other things
// manage/retrieve all the received packets so far
// ...
// ...

ETHCON1CLR=0x10;           // disable the Manual Flow Control
```

35.4.8 Receive Filtering Overview

The Receive Filter (RXF) block examines all incoming receive packets and accepts or rejects the packet, based on user-selectable filters. The following RX filters are supported:

- CRC Error Acceptance Filter controlled by CRCERREN bit (ETHRXFC<7>)
- Runt Error Acceptance Filter controlled by RUNTERREN bit (ETHRXFC<5>)
- CRC Check Rejection Filter controlled by CRCOKEN bit (ETHRXFC<6>)
- Runt Rejection Filter controlled by RUNTEN bit (ETHRXFC<4>)
- Unicast Acceptance Filter controlled by UCEN bit (ETHRXFC<3>)
- Not Me Unicast Acceptance Filter controlled by NOTMEEN bit (ETHRXFC<2>)
- Multicast Acceptance Filter controlled by MCEN bit (ETHRXFC<1>)
- Broadcast Acceptance Filter controlled by BCEN bit (ETHRXFC<0>)
- Hash Table Acceptance Filter controlled by HTEN bit (ETHRXFC<15>)
- Magic Packet Acceptance Filter controlled by MPEN bit (ETHRXFC<14>)
- Pattern Match Acceptance Filter with logical inversion controlled by PMMODE bit (ETHRXFC<8-11>) and NOTPM bit (ETHRXFC<12>)

> **Note:** Each filter is either an Acceptance filter or a Rejection filter. Acceptance filters force the acceptance of a packet, while rejection filters force the rejection of a packet.

The order of the filters above specifies the priority of the filter from highest-to-lowest, such that if a filter is enabled and accepts or rejects a packet, all lower priority filters will have no effect.

For example, if the Runt Error Acceptance Filter is enabled and a packet of less than 64 bytes is received, it will always be accepted, even if the CRC check fails.

If a received packet is not explicitly accepted or discarded by an enabled filter, the packet will be discarded by default.

Due to the internal design of the RX Filter, the final accept versus abort decision for an Ethernet frame is made at the end of the frame.

When a packet is received, the Receive Status Vector (RSV) for each receive packet contains information about which filters matched the corresponding RX packet, regardless of whether these filters were active at the time. This provides extra "status" information about the packet that may be used to filter packets in software. For example, in Promiscuous mode, the Magic Packet filter RSV bit may be used to quickly identify a Magic Packet without the need to examine the frame contents. Please refer to the Figure 35-9 for more information on the RSV.

All filter settings are done using the ETHRXFC register.

Due to synchronization in the RXF block, the following registers should not be changed while the ON bit (ETHCON1<15>) is set:

- **ETHRXFC: Ethernet Controller Receive Filter Configuration Register[1,2]**
- **ETHHT0: Ethernet Controller Hash Table 0 Register[1,2]**
- **ETHHT1: Ethernet Controller Hash Table 1 Register[1,2]**
- **ETHPMO: Ethernet Controller Pattern Match Offset Register[1,2]**
- **ETHPMCS: Ethernet Controller Pattern Match Checksum Register[1,2]**
- **ETHPMM0: Ethernet Controller Pattern Match Mask 0 Register[1,2]**
- **ETHPMM1: Ethernet Controller Pattern Match Mask 1 Register[1,2]**
- **EMAC1SA0: Ethernet Controller MAC Station Address 0 Register[1,2]**
- **EMAC1SA1: Ethernet Controller MAC Station Address 1 Register[1,2]**
- **EMAC1SA2: Ethernet Controller MAC Station Address 2 Register[1,2]**

To change one or more of these registers or their bits, you must first clear the ON bit (ETHCON1<15>).

The following sections provide short summaries of each receive filter.

35.4.8.1 CRC ERROR ACCEPTANCE FILTER

This filter is used to explicitly accept packets that fail the CRC check. If enabled, all packets that fail the CRC check are accepted, regardless of whether the CRC Check Filter is enabled or not.

35.4.8.2 RUNT ERROR ACCEPTANCE FILTER

This filter is used to explicitly accept packets that fail the Runt check. If enabled, all packets that fail the Runt check are accepted, regardless of whether the Runt Filter is enabled or not.

35.4.8.3 CRC CHECK ACCEPTANCE FILTER

If enabled, results of the MAC CRC check will be examined and used to filter the packet. If this filter is enabled and fails, the packet will be aborted. Conversely, if CRC checking is not enabled, the received packet's CRC is ignored and is not used as an acceptance requirement for the packet.

35.4.8.4 RUNT REJECTION FILTER

The Runt Filter allows filtering on the size of the received packet (Destination Address, Source Address, Length/Type, Payload and FCS). When the Runt Rejection Filter is enabled packets smaller than 64 bytes will be rejected.

35.4.8.5 CAST ACCEPTANCE FILTER

The packet is filtered by its cast, with support for the following:

- Unicast: Accepts any packet that is of type Unicast and whose Destination Address matches the Station Address
- Not Me Unicast: Accepts any packet that is of type Unicast and whose Destination Address does not match the Station Address
- Broadcast: Accepts any packet that is of type Broadcast
- Multicast: Accepts any packet that is of type Multicast

A receive packet may be accepted (depending on the other filters) if any of the active cast filters accept the packet. Enabling both the Unicast and Broadcast filters, for example, would allow either type of packet to be received.

To accept all incoming packets (Promiscuous mode), simply enable the UCEN, NOTMEEN, MCEN and BCEN filters, and disable all other filters.

35.4.8.6 HASH TABLE ACCEPTANCE FILTER

When enabled, the Hash Table filter accepts received packets based on their destination address, with up to 64 different addresses allowed. This is done by using the Destination Address CRC output from the MAC as a lookup key in a user-defined Hash Table.

First, the CRC value is calculated on the received packet Destination Address field. Bits <28:23> of this value are then used as an index into a 64-bit user-programmable table (ETHHT0, ETHHT1) containing single-bit accept (‘1’) or ignore (‘0’) values. For example, if the calculated CRC has a value of 05h, the value in bit 5 of the 64-bit HT register table is examined. If that entry is a logical ‘1’, the packet is accepted; otherwise, the filter results are not taken into account when deciding whether to accept the packet or not.

Note that the Destination Address CRC output used by this filter corresponds to bits <28:23> of the uncomplemented 32-bit CRC over the Destination Address of the RX packet.

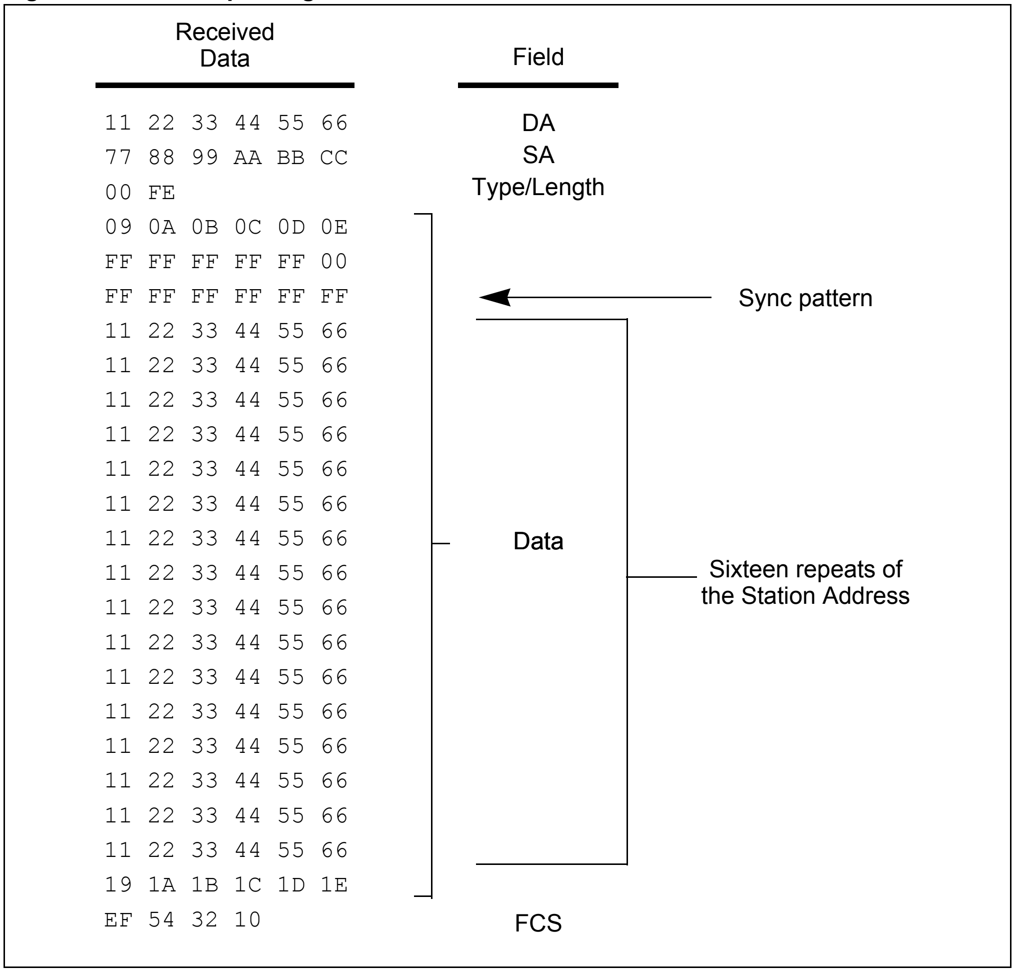
35.4.8.7 MAGIC PACKET ACCEPTANCE FILTER

The Magic Packet filter scans the received packet for a predetermined pattern to accept the packet.

A Magic Packet is defined by the following: the Data field contains a synchronization pattern of six 0xFF bytes followed by the Destination Station Address repeated sixteen times.

The data packet may contain additional payload besides the Magic Packet pattern.

Figure 35-6: Sample Magic Packet™ Format



35.4.8.8 PATTERN MATCH ACCEPTANCE FILTER

When enabled, the Pattern Match Acceptance filter accepts packets that match a certain pattern. The match is accomplished by generating a Checksum of the selected bytes in a 64-byte window.

If the calculated checksum is equal or not equal to the ETHPMCS register, as specified by the NOTPM bit (ETXRXFC<12>), and all conditions associated with PMMODE bit (ETHRXFC<8-11>) are met, the packet is accepted. Otherwise, the packet is aborted.

The 64-byte window is programmed using the PMO value (ETHPMO<15:0>) so that the start of the window can be anywhere from 0 to 65536 bytes. However, if the 64-byte window extends past the end of the packet, the pattern match filter aborts the packet.

The Pattern Match Mask bits PMM<63:0> bit (ETHPMM0<31:0>, ETHPMM1<31:0>) are used to select whether or not the given byte in the 64-byte window is used in the computation of the Checksum. If the Pattern Match Mask bit (PMM<n>) in the Ethernet Pattern Match Mask registers (ETHPMM0, ETHPMM1) is set, the respective byte is used in the Checksum computation (where n = 0, 1, 2,..., 62, 63 and n = 0 points to the first byte after the offset value).

The Checksum algorithm is the same as the TCP/IP checksum calculation. Note that the algorithm requires that the calculation uses a 16-bit word length. This means that the data series used for the calculation will have a zero byte of padding for the last byte if an odd number of bytes are to be matched. Also, the Checksum value is initialized to 0x00000000 before the calculation is started. Figure 35-7 shows an example of Checksum calculation.

Figure 35-7: Checksum Calculation

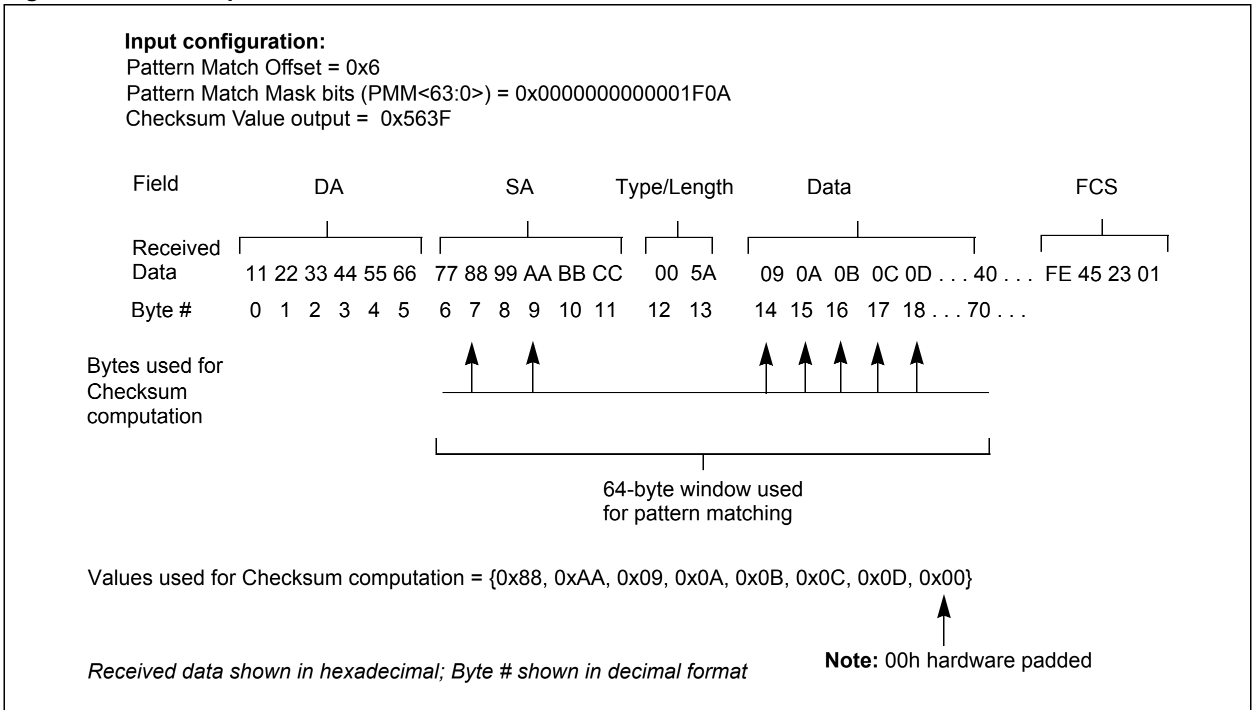
```
12 00 AC 23 92 55 00 00 FE AA FF FF 34 12 CD AB <-- Data Packet (hex)
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 <-- Byte Number

Step 1: Add words of data packet:
       1200h + AC23h + 9255h + 0000h + FEAAh + FFFFh + 3412h + CDABh = 450DEh
       checksum_reg[31:0] = 0x0004_50DE

Step 2a: Add high word with low word of checksum_reg:
        50DEh + 0004h = 50E2h
        checksum_reg[31:0] = 0000_50E2h

Step 2b: If the high order word from Step 2a > 0000h, add high word
         with low word of checksum_reg (i.e., repeat step 2a).

Step 3: NOT(000050E2h) = FFFFAF1Dh
        checksum_reg[31:0] = 0000_50E2h
        output dma_checksum_val[15:0] = AF1Dh
```

Note 1: The calculation shown above assumes an initial seed of 0x0000.

2: If dma_length[DMA_ADDR_MSB:0] = 0, then the final checksum value must be the 1's complement of the checksum seed. For an initial checksum seed of 0x0000, this will result in 0xFFFF. For a user-specified seed, this will result in the same seed value, as user-specified seed values are already 1's complemented before being brought into the DMA.

Figure 35-8 shows a sample Pattern Match format.

Figure 35-8: Sample Pattern Match Format



Example 35-3: Setting the Pattern Match RX Filter

```
// Note: Setting the Pattern Match filter should be done only when receive
// is not enabled
/* Input parameters:
- int matchMode: a value between 0 and 9 describing the Pattern Match Mode
   (see PMMODE (ETHRXFC<8:11>) in the ETHRXFC: Ethernet Controller Receive
  Filter Configuration Register (Register 35-4)
- long long matchMask: the match mask in the 64 Byte packet window
- int matchOffs: the offset applied to the incoming packet data to obtain
  the window
- int matchChecksum: the 16 bit checksum to be used for comparison
- int matchInvert: Boolean to for the Pattern Match Inversion bit NOTPM
(ETHRXFC<12>)
*/
#include <p32xxxx.h>

ETHRXFCCLR = 0x00000F00;                  // disable pattern match mode
ETHPMM0 = (unsigned int)matchMask;
ETHPMM1 = (unsigned int)(matchMask>>32);

ETHPMO = matchOffs;
ETHPMCS = matchChecksum;

if(matchInvert)
{
    ETHRXFCSET = 0x00001000;              // set NOTPM
}
else
{
    ETHRXFCCLR = 0x00001000;              // clear NOTPM
}

ETHRXFCSET=(matchMode)<<0x00000008;       // enable the Pattern Match mode
```

35.4.9 Ethernet DMA and Buffer Management Engines

In order to reduce the overhead on the CPU to move the packet data between data memory and the Ethernet controller, internal RX and TX DMA engines are integrated into the module. The DMA engines are responsible for transferring data from system memory to the MAC for transmit packets and for transferring data from the MAC to system memory for receive packets. The DMA engines each have access to the system memory by acting as two different bus masters, one bus master for transmit and one for receive.

The DMA engines use separate Ethernet Descriptor Tables (EDTs) for TX and RX operations to determine where the TX/RX packet buffer resides in the system memory.

Both transmit and receive descriptors, called Ethernet Descriptors (EDs), used by the DMA engines have a similar format, with only the status word formats being different. The format of the descriptors is shown in Table 35-7 and Table 35-8. The descriptor tables can contain a linked list or linear list of descriptors that point to packet buffers as shown in Figure 35-9 and Figure 35-10.

It is the software's responsibility to set up the RX and TX descriptor tables before enabling an Ethernet transfer.

Table 35-7: Ethernet Controller TX Buffer Descriptor Format

Offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Addr+0	SOP	EOP	U	U	U	BYTE_COUNT<10:0>											U	U	U	U	U	U	U	NPV	EOWN	—	—	—	—	—	—	—
Addr+4	DATA_BUFFER_ADDRESS<31:0>																															
Addr+8	U	U	U	U	U	U	U	U	—	—	—	—	TSV<51:32>																			
Addr+12	TSV<31:0>																															
Addr+16	NEXT_ED<31:0>																															

Note:	The address of the Ethernet Descriptor must be 4-byte aligned.

Offset 0

bit 31 **SOP:** Start of Packet Enable bit
1 = Transmit a start of Packet delimiter with this data buffer
0 = NO Start of Packet delimiter present

bit 30 **EOP:** End of Packet Enable bit
1 = Transmit an End of Packet delimiter with this data buffer
0 = No End of Packet delimiter present

bit 29-27 **User-defined bits**; not used by the Ethernet Controller

bit 26-16 **BYTE_COUNT<10:0>:** Byte Count bits(1)
The Byte Count represents the number of bytes to be transmitted for this descriptor. Valid byte counts are 1-2047 per descriptor entry.

bit 15-9 **User-defined bits**; not used by the Ethernet Controller

bit 8 **NPV:** NEXT ED Pointer Valid Enable bit
1 = Next Descriptor is pointed to by the Next_ED field in this descriptor
0 = Next Descriptor follows this descriptor in memory

bit 7 **EOWN:** Ethernet Controller Own bit(2)
1 = The Ethernet Controller owns the ED and its corresponding data buffer. The software must not modify the ED or the data buffer
0 = The software owns the ED and its corresponding data buffer. The Ethernet Controller ignores all other fields in the ED

bit 6-0 **Reserved:** Maintain as '0'; ignore Read

Note 1: Programming a BYTE_COUNT = 0 can result in undefined behavior.
2: This bit can be written by either the software or the Ethernet Controller and it must be initialized by the user application to the desired value prior to enabling the Ethernet Controller.

Offset 4

bit 31-0 **DATA_BUFFER_ADDRESS<31:0>:** Data Buffer Address bits
The starting point address of the Descriptor data buffer.

Table 35-7: Ethernet Controller TX Buffer Descriptor Format (Continued)

Offset 8

bit 31-24 **User-defined bits**; not used by the Ethernet Controller

bit 23-20 **Reserved:** Maintain as '0'; ignore Read

bit 19-0 **TSV<51:32>:** Transmit Status Vector bits
Status for the transmitted packet:
TSV<51> = Transmit VLAN Tagged Frame
Frame's length/type field contained 0x8100 which is the VLAN Protocol Identifier.
TSV<50> = Transmit Backpressure Applied
Carrier-sense method backpressure was previously applied.
TSV<49> = Transmit PAUSE Control Frame
The frame transmitted was a Control frame with a valid PAUSE Op code.
TSV<48> = Transmit Control Frame
The frame transmitted was a Control frame.
TSV<47-32> = Total Bytes Transmitted on Wire
Total bytes transmitted on the wire for the current packet, including all bytes from collided attempts.

Offset 12

bit 31-0 **TSV<31:0>:** Transmit Status vector
Status for the transmitted packet:
TSV<31> = Transmit Under-run
The system failed to transfer complete packet to the transmit MAC module.
TSV<30> = Transmit Giant
Byte count for frame was greater than MACMAXF (EMAC1MAXF<0:15>).
TSV<29> = Transmit Late Collision
Collision occurred beyond the collision window (512 bit times).
TSV<28> = Transmit Maximum Collision
Packet was aborted due after number of collision exceeded RETX (EMAC1CLRT<0:3>).
TSV<27> = Transmit Excessive Defer
Packet was deferred in excess of 6071 nibble times in 100 Mbps mode or 24,287 bit times in 10 Mbps mode.
TSV<26> = Transmit Packet Defer
Packet was deferred for at least one attempt, but less than an excessive defer.
TSV<25> = Transmit Broadcast
Packet's destination address was a broadcast address.
TSV<24> = Transmit Multicast
Packet's destination address was a multicast address.
TSV<23> = Transmit Done
Transmission of the packet was completed.
TSV<22> = Transmit Length Out Of Range
Indicates that frame Type/Length field was larger than 1500 bytes (Type Field).
TSV<21> = Transmit Length Check Error
Indicates that frame length field value in the packet does not match the actual data byte length and is not a Type field.
TSV<20> = Transmit CRC Error
The attached CRC in the packet did not match the internal generated CRC.
TSV<19:16> = Transmit Collision Count
Number of collisions current packet incurred during transmission attempts.
TSV<15:0> = Transmit Byte Count
Total bytes in frame not counting collided bytes.

Offset 16

bit 31-0 **NEXT_ED<31:0>:** Next Ethernet Descriptor Address bits
When NPV = 1: This field contains the starting point address of the next Ethernet Descriptor.
When NPV = 0: This field is not present in the descriptor.

Table 35-8: Ethernet Controller RX Buffer Descriptor Format

Offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Addr+0	SOP	EOP	—	—	—	BYTE_COUNT<10:0>											U	U	U	U	U	U	U	NPV	EOWN	—	—	—	—	—	—	—
Addr+4	DATA_BUFFER_ADDRESS<31:0>																															
Addr+8	RXF_RSV<7:0>								U	U	U	U	U	U	U	U	PKT_CHECKSUM<15:0>															
Addr+12	RSV<31:0>																															
Addr+16	NEXT_ED<31:0>																															

Note: The address of the Ethernet Descriptor must be 4-byte aligned.

Offset 0

bit 31 **SOP:** Start of Packet Enable bit
1 = Received a start of Packet delimiter with this data buffer
0 = NO Start of Packet delimiter present

bit 30 **EOP:** End of Packet Enable bit
1 = Transmit an End of Packet delimiter with this data buffer
0 = No End of Packet delimiter present

bit 29-27 **Reserved:** Maintain as '0'; ignore Read

bit 26-16 **BYTE_COUNT<10:0>:** Byte Count bits
The Byte Count represents the number of bytes to be transmitted for this descriptor. Valid byte counts are 1-2047 per descriptor entry.

bit 15-9 **User-defined bits**; not used by the Ethernet Controller

bit 8 **NPV:** NEXT ED Pointer Valid Enable bit
1 = Next Descriptor is pointed to by the Next_ED field in this descriptor
0 = Next Descriptor follows this descriptor in memory

bit 7 **EOWN:** Ethernet Controller Own bit[(1)]
1 = The Ethernet Controller owns the ED and its corresponding data buffer. The software must not modify the ED or the data buffer
0 = The software owns the ED and its corresponding data buffer. The Ethernet Controller ignores all other fields in the ED

bit 6-0 **Reserved:** Maintain as '0'; ignore Read

Note 1: This bit can be written by either the software or the Ethernet Controller and it must be initialized by the user application to the desired value prior to enabling the Ethernet Controller.

Offset 4

bit 31-0 **DATA_BUFFER_ADDRESS<31:0>:** Data Buffer Address bits
The starting point address of the Descriptor data buffer.

Offset 8

bit 31-24 **RXF_RSV<7:0>:** Receive Filter Status Vector bits
This field carries extra information about filtering of the received packet:
RXF_RSV<7> = Multicast match
RXF_RSV<6> = Broadcast match
RXF_RSV<5> = Unicast match
RXF_RSV<4> = Pattern Match match
RXF_RSV<3> = Magic Packet match
RXF_RSV<2> = Hash Table match
RXF_RSV<1> = NOT (Unicast match) AND NOT (Multicast Match)
RXF_RSV<0> = Runt packet

bit 23-16 **User-defined bits**; not used by the Ethernet Controller

bit 15-0 **PKT_CHECKSUM<15:0>:** The RX Packet Payload Checksum of this descriptor's packet. The calculated 1's complement of the 16-bit packet checksum value.

Table 35-8: Ethernet Controller RX Buffer Descriptor Format (Continued)

Offset 12

bit 31-0 **RSV<31:0>:** Receive Status Vector bits

Status for the received packet:

RSV<31> = Reserved

RSV<30> = Receive VLAN Type Detected
Current frame was recognized as a VLAN tagged frame.

RSV<29> = Receive Unknown Op code
Current Frame was recognized as a Control Frame but it contained an Unknown Op-code. Packet does not have a CRC error and has a valid length (64-1518).

RSV<28> = Receive Pause Control Frame
Current Frame was recognized as a Control Frame containing a valid Pause Frame Op-code and a valid address. Packet does not have a CRC error and has a valid length (64-1518).

RSV<27> = Receive Control Frame
Current Frame was recognized as a Control Frame for having a valid Type-Length designating it as a Control Frame. Packet does not have a CRC error and has a valid length (64-1518).

RSV<26> = Dribble Nibble
Indicates that after the end of this packet an additional 1 to 7 bits were received. A single nibble, called the dribble nibble, is formed but not sent out.

RSV<25> = Receive Broadcast Packet
Indicates packet received had a valid broadcast address.

RSV<24> = Receive Multicast Packet
Indicates packet received had a valid multicast address.

RSV<23> = Received Ok
Indicates that at the packet had a valid CRC and no symbol errors.

RSV<22> = Length Out of Range
Indicates that frame type/length field was larger than 1500 bytes (Type field).

RSV<21> = Length Check Error
Indicates that frame length field value in the packet does not match the actual data byte-length and specifies a valid length.

RSV<20> = CRC Error
Indicates that frame CRC field value does not match the CRC calculated by the receiver MAC.

RSV<19> = Receive Code Violation
Indicates that the MII data does not represent a valid receive code when MRXER asserts during the data phase of a frame.

RSV<18> = Carrier Event Previously Seen
Indicates that at some time since the last receive statistics, a carrier event was detected, noted and reported with the next receive statistics. The carrier event is not associated with this packet. A carrier event is activity on the receive channel that does not result in a packet receive attempt being made.

RSV<17> = RXDV Event Previously Seen
Indicates that the last receive event seen was not long enough to be a valid packet.

RSV<16> = Long Event/Drop Event
Indicates a packet over 50,000 bit times occurred, or that a packet since the last RSV was dropped.

RSV<15:0> = Received Byte Count
Indicates length of received frame.

Offset 16

bit 31-0 **NEXT_ED<31:0>:** Next Ethernet Descriptor Address bits

When NPV = `1`: This field contains the starting point address of the next Ethernet Descriptor.

When NPV = `0`: This field is not present in the descriptor.

Figure 35-9: Ethernet Descriptor Table Format (Linked List, NPV = 1)

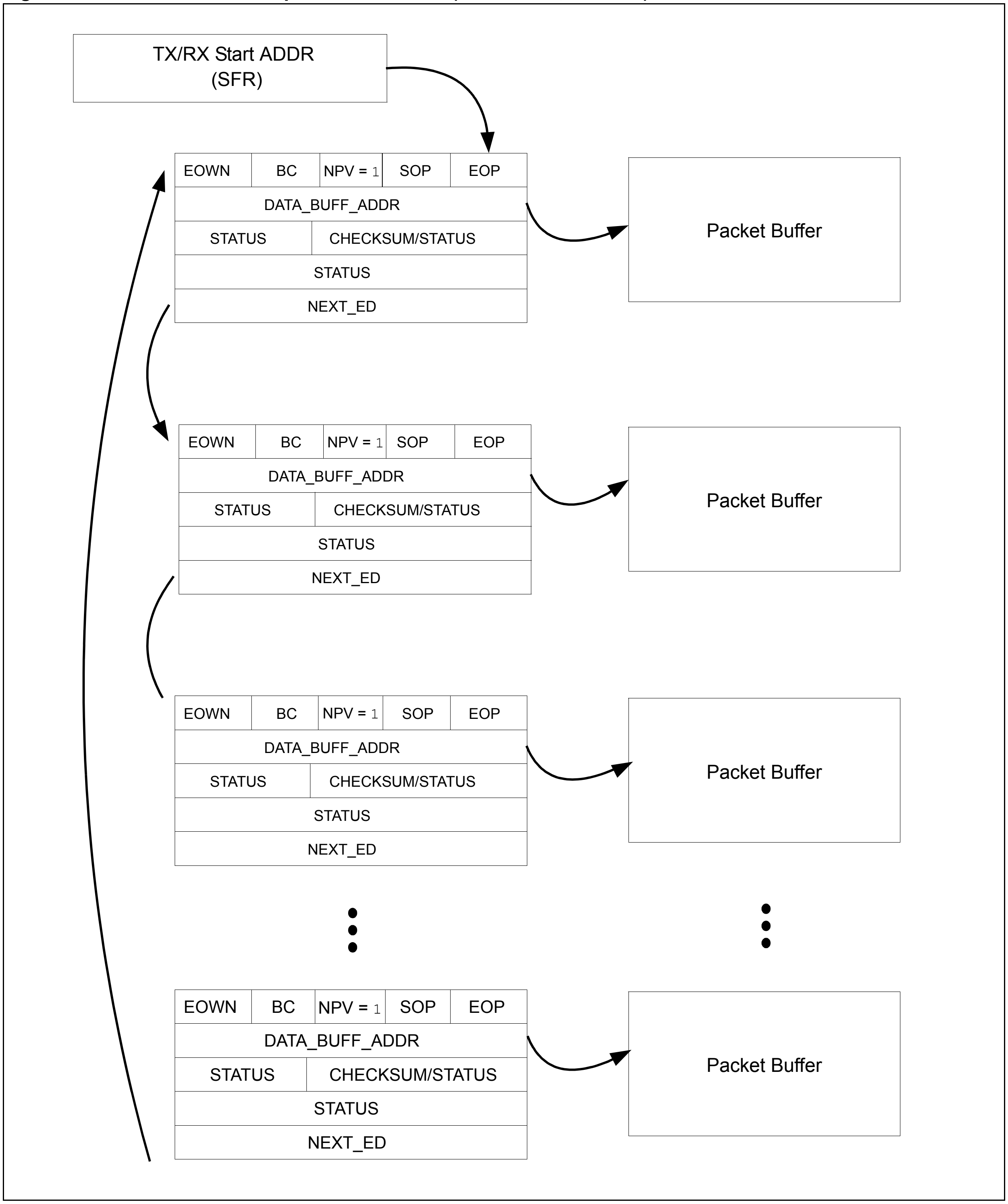
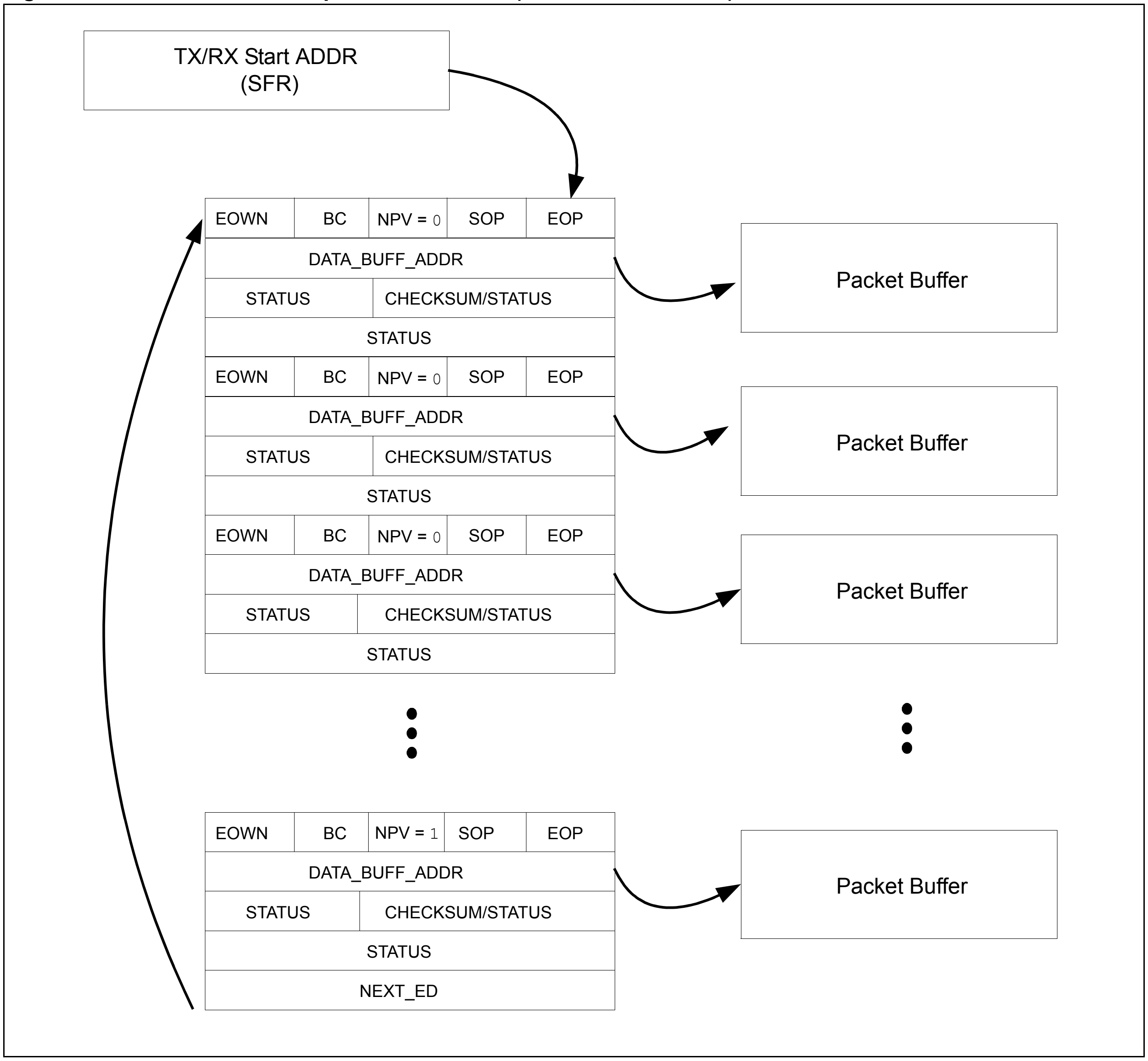


Figure 35-10: Ethernet Descriptor Table Format (Linear List, NPV = 0)



35.4.9.1 ETHERNET DESCRIPTOR BUFFER MANAGEMENT

The descriptor tables and packet data buffers used by the receive and transmit paths reside in the system data memory. The descriptor tables are a linked list of descriptors that reference blocks of packet data buffers in memory. They are physically and logically partitioned into separate Transmit and a Receive Descriptors and Buffers. Note that the start address of both the RX and TX Descriptor tables must be 4-byte-aligned (i.e., bit 1 and bit 0 = 00).

35.4.9.2 ETHERNET DESCRIPTOR OWNERSHIP

Because the descriptors and buffers are shared between the software and the Ethernet Controller, a simple semaphore mechanism is used to distinguish which part is allowed to update the descriptor and associated buffers in memory. This semaphore mechanism is implemented by the EOWN bit in each Buffer Descriptor. When the EOWN bit is clear, the descriptor is owned by the software. The software may modify the descriptor at its discretion. When the EOWN bit is set, the descriptor (and the buffer memory pointed to by the descriptor) is owned by the Ethernet Controller hardware. The software may not modify the descriptor or its corresponding data buffer. The Ethernet Controller will write the Buffer Descriptor and corresponding data buffer at its discretion.

35.4.9.3 ETHERNET DESCRIPTOR TABLE CONFIGURATION

Before enabling the transfers for the Ethernet Controller, the software must set up both the TX and RX descriptor tables as well as allocate the packet data buffer areas pointed to by the descriptors. The number of descriptor entries in each table is determined by the software and the amount of memory available in the system.

There are two different types of Ethernet Descriptor Tables (EDTs) that can be set up by software. One configuration is a descriptor ring; the other is a descriptor list. The only difference is whether the last descriptor in the list is an "empty" descriptor with the EOWN bit = `0`, or the last descriptor has a reference back to the top of the ring. Either of these can be handled by the Ethernet DMA engines. An example of these two types is shown in Figure 35-11.

Note:	It is the responsibility of software to initialize all the fields for each descriptor in the EDT, which includes initializing the status field to `0`'s. Refer to Table 35-7 and Table 35-8 for a detailed descriptions of the Ethernet Descriptor format.

35.4.9.4 ETHERNET TRANSMIT BUFFER MANAGEMENT

The Transmit Buffer Management (TXBM) block along with the TX DMA manages the flow of transmit packets from the system memory to the MAC using the Transmit Buffer Descriptors.

Transmit operation is enabled by setting the TXRTS bit (ETHCON1<9>). In response to the TXRTS bit being set, the TX DMA will fetch an Ethernet Descriptor from the EDT pointed to by ETHTXST. After it has read the location of the data buffer and the control word for the data buffer, the DMA will begin reading the data buffer data and writing it to the transmit port of the MAC. If more than one descriptor is needed, the DMA will move to the next descriptor and will continue sending data.

Note:	Software must insure that the TX descriptor list and ETHTXST register are initialized prior to setting the TXRTS bit.

When a packet has to be transmitted from the system memory to the MAC, the packet data is read from the memory pointed by the descriptor DATA_BUFFER_ADDRESS (see Figure 35-9).

The format of the transmitted packets is the same as the one for the received packets. Each transmit packet buffer contains the standard Ethernet frame fields to be transmitted (DA, SA, Type/Length and Payload).

An example of a TX packet using three descriptors is shown in Figure 35-12.

Once the descriptor table entries and packet data buffers are programmed by the software, the transmit operation is initiated by setting the TXRTS bit (ETHCON1<9>).

Note:	The EOWN bits in the transmitted descriptors should be set by software starting with the last descriptor of the packet to be transmitted and ending with the first. This prevents any race condition between software and the Ethernet controller hardware.

Once a complete packet has been transmitted, the Ethernet Controller will update the transmit status vector (TSV) in the first descriptor entry used for that packet. The EOWN field is updated for all descriptors used by the transmitted packet. Also the hardware will update the ETHTXST register to reflect the next TX descriptor to be used for the next transmitted packet.

As long as there are valid transmit descriptors for packets (the next descriptor is valid, EOWN = `1`), the TX DMA will continue to traverse the descriptor table and send packet data to the MAC for transmission. When all the transmit operations are complete, the transmit logic will clear the TXRTS bit and set the TXDONE bit (ETHIRQ<3>). The TXDONE bit will generate an interrupt if the TXDONEIE bit (ETHIEN<3>) is set. Thus the completion of the transmit operation can be monitored by either polling the TXRTS bit or by using the TXDONE bit (ETHIRQ<3>) interrupt.

The transmit operation can be stopped by clearing the TXRTS bit at any time during the transmission of the packet. When the TXRTS bit is cleared during a packet transmission, the current packet will complete its transmission after which the TXBUSY bit (ETHSTAT<6>) status bit will be cleared, indicating the transmit engine has stopped.

Software should not write any of the TX related SFR registers while the TXRTS bit is set. To change these registers, the TXRTS must be cleared, then the software must wait for TXBUSY to de-assert, after which the registers can be written, and the TXRTS bit set again.

> **Note:** The ETHTXST transmit configuration register (starting address of TX Descriptor Table) is used by the TX DMA, and should only be changed when the TX DMA is not operating (i.e., TXRTS bit (ETHCON1<9>) = `0` and TXBUSY bit (ETHSTAT<6>) = `0`), since continual synchronization to the DMA clock domain is not provided. Ensuring the ETHTXST configuration register does not change when the TX DMA is active is the responsibility of software.

35.4.9.5 ETHERNET TRANSMIT OPERATION DETAILS

The packet transmit process is as follows:

1. Software writes the packet data to a packet buffer in data memory and programs a descriptor entry to point to the packet data buffer. It also programs the SOP, EOP, BYTE_COUNT and EOWN bits to show a valid packet is available and also if there are other descriptors needed to send the complete packet. See Figure 35-12 for an example of a TX packet using multiple descriptor entries. The descriptor is used to define the Transmission Packet data buffer location and length.
2. Software then sets the TXRTS bit (ETHCON1<9>) to start the transmission, which starts the TX DMA and TXBM logic.
3. The TX DMA engine will read the next available descriptor entry from the descriptor table which is pointed to by the ETHTXST register.
4. After the TX DMA engine reads the DATA_BUFFER_ADDRESS value, the engine will begin reading the packet data from the location read from the descriptor.
5. The TXBM indicates the start-of-frame to the MAC and transmits the entire frame data from the transmit buffer, until the end address is reached. The TXBM simply transmits from the start address until the specified number of bytes has been transmitted to the MAC transmit interface.
6. The MAC can retry a transmission due to an early collision.
7. The MAC can abort a transmission due to a late collision, excessive collisions or excessive defers. This condition is signaled by TXABORT bit (ETHIRQ<2>).
8. Once transmission has completed, the TX DMA engine stores the relevant bits of the TSV into the first descriptor entry of the packet.
9. After the packet has been transmitted, all the descriptors used for the transmission are released to the software via the EOWN bits being cleared.
10. If more valid TX descriptors are available (EOWN = `1`), the DMA engine will go back to step 3 to begin the next packet's transmission. Otherwise, if the next descriptor is still owned by hardware (EOWN = `0`), transmission will halt and wait for the software to set the TXRTS bit again.

Note that any collision that occurs within the CWINDOW bit (EMAC1CLRT<8:14>) boundary is an early collision and results in a Retry operation. A collision that happens beyond the CWINDOW boundary will be treated as a late collision and will cause an abort. The CWINDOW bit in the EMAC1CLRT register is typically set to 64 bytes. An abort condition can also result from reaching the maximum collision count RETX bit (EMAC1CLRT<0:3>) for a packet trying to be sent.

The TXBM engine has little information about the content of data it sends to the MAC. However, it should be noted that two kinds of transmit packets can be sent:

1. A complete packet, which includes the following:
   - Addresses (DA, SA), Type/Length and Payload
   - Pad (if required)
   - Frame Check Sequence

   In this case, PADENABLE bit (EMAC1CFG2<5>) is `0`. The MAC will not pad the frame, but will perform a CRC check on the frame, and set TSV<20> in the TX status vector if the CRC check match fails.
2. An incomplete packet that includes:
   - Addresses (DA, SA), Type/Length and Payload

   In this case, PADENABLE = `1`. The MAC will pad the frame (in accordance with the settings of AUTOPAD, VLANPAD bit (EMAC1CFG2<7:6>)), and will insert the calculated CRC (FCS) for the frame.

See Table 35-2 for a description of the pad and CRC options based on the settings of the EMAC1CFG2 register.

Example 35-4: Ethernet Transmit Packet Code

```
/* The following assumptions were made for this example:
    - the packet that has to be sent consists of multiple buffers in memory.
    - the number of available TX descriptors greater than the number of buffers composing the
      packet (there's at least an extra descriptor ending the chain of descriptors)
    - this is the first transmission of a packet, the example enables the transmission process.
Input parameters:
    - sEthTxDcpt* pArrDcpt: pointer to an array that holds free descriptors that we can use for
      the TX operation (see below the definition for sEthTxDcpt).
    - char* pArrBuff: pointer to an array that holds buffers to be transmitted
    - int* pArrSize: pointer to an array that holds the sizes of buffers to be transmitted
    - int nArrayItems: how many buffers to be transmitted are stored in the array. */

#include <p32xxxx.h>
// definition used (see Table 35-7: Ethernet Controller TX Buffer Descriptor Format)

typedef struct
{
    volatile union
    {
        struct
        {
            unsigned: 7;
            unsigned EOWN: 1;
            unsigned NPV: 1;
            unsigned: 7;
            unsigned bCount: 11;
            unsigned: 3;
            unsigned EOP: 1;
            unsigned SOP: 1;
        };
        unsigned intw;
    }hdr;                                       // descriptor header
    unsigned char* pEDBuff;                     // data buffer address
    volatile unsigned long longstat;            // tx packet status
    unsigned int   next_ed;                     // next descriptor (hdr.NPV==1);
}__attribute__ ((__packed__)) sEthTxDcpt;      // hardware Tx descriptor (linked).

extern void* VA_TO_PA(char* pBuff);             // extern function that returns the physical
                                                // address of the virtual address input parameter

int     ix;                                     // loop index
sEthTxDcpt* pEDcpt;                             // current Ethernet descriptor
sEthTxDcpt* tailDcpt;                           // last Ethernet descriptor
char* pBuff;                                    // current data buffer to be transmitted
int* pSize;                                     // current data buffer size

pEDcpt=pArrDcpt;
pBuff=pArrBuff;
pSize=pArrSize;
tailDcpt=0;

for(ix=0; ix< nArrayItems; ix++, pEDcpt++, pBuff++, pSize++)
{
    // pass the descriptor to hw, use linked descriptors, set proper size
    pEDcpt->pEDBuff=(unsigned char*)VA_TO_PA(pBuff);        // set buffer
    pEDcpt->hdr.w=0;                                        // clear all the fields
    pEDcpt-> hdr.NPV=1;                                     // set next pointer valid
    pEDcpt-> hdr.EOWN=1;                                    // set hw ownership
    pEDcpt->hdr.bCount=* pSize;                             // set proper size

    if(tailDcpt)
    {
    tailDcpt->next_ed=VA_TO_PA(&pEDcpt);
    }
    tailDcpt=pEDcpt;
}
// at this moment pEDcpt is an extra descriptor we use to end the descriptors list
pEDcpt->hdr.w=0;                                // software ownership
tailDcpt->next_ed= VA_TO_PA(&pEDcpt);
pArrDcpt[0].hdr.SOP=1;                          // start of packet
pArrDcpt[nArrayItems-1].hdr.EOP=1;              // end of packet

ETHTXST=VA_TO_PA(pArrDcpt);                     // set the transmit address
ETHCON1SET= 0x00008200;                         // set the ON and the TXRTS

// the ETHC will transmit the buffers we just programmed
/* do something else in between */

while(!(ETHCON1&0x00000200));                   // wait transmission to be done

// check the ETHSTAT register to see the transfer result
```

Note:	This code example uses MPLAB® C32 C compiler specific syntax. Refer to your compiler manual regarding support for packed data structures.

35.4.9.6 ETHERNET RECEIVE BUFFER MANAGEMENT

The Receive Buffer Management (RXBM) block along with the RX DMA manages the flow of receive packets from the MAC to the system memory using the Receive Buffer Descriptors.

The receive operation is enabled by setting the RXEN bit (ETHCON1<8>). Once the RXEN bit is set, the RX DMA will respond to incoming packets by reading the next available descriptor entry in the table and writing the packet data into the packet buffer pointed to by the descriptor and writing the receive packet status into the descriptor entry itself. If the incoming packet requires more space than is allocated by a single buffer, the packet may span multiple descriptors. If the RX DMA reads the next packet descriptor in the table and does not own it, this may be an overflow condition and will be reported via the status registers.

Note:	When the RXEN bit is first enabled, the RXDMA engine will initially fetch a RX descriptor from memory in preparation of receiving packet data. Software must insure the descriptor list is initialized prior to setting the RXEN bit.

When a packet is successfully received (the packet is not aborted by the filter or an overrun error), the packet data is stored in the memory pointed to by the descriptor DATA_BUFFER_ADDRESS (see Figure 35-9). The receive status vector (RSV), the RX filter receive status (RXF_RSV), the packet checksum (PKT_CHECKSUM) and control word (SOP and EOP) are updated in the first descriptor entry used for that packet. The EOWN field is updated for all descriptors used by the received packet. Additionally, for improved packet management, the BUFCNT bit (ETHSTAT<16:23>) keeps a running count of the number of received packet buffers stored in data memory.

Finally, the ETHRXST register is updated to reflect the next RX descriptor to be used for the next received packet. Once hardware has stored the packet in memory, software is responsible for checking the packet's RSV for errors before the packet is processed.

Once software processes a received packet, it should write the BUFCDEC bit (ETHCON1<0>) bit once for each descriptor used for that packet in order to decrement the packet buffer count BUFCNT. This provides an accurate count of unprocessed packet buffers pending in data memory that is used in automatic flow control (see **35.4.7 "Flow Control Overview"**).

Software should also update the descriptors and clear the EOWN field in the descriptors used for that packet to free them up for another received packet.

When automatic flow control is enabled, an overrun condition occurs if the RX logic receives more packets than the maximum number that the BUFCNT bit in the ETHSTAT register can reflect. If an attempt is made to increment the BUFCNT field (by RXBM having received a packet), and the register has reached its maximum value (0xFF), the register will not rollover; an overrun error condition will be generated and the RX logic will halt. The proper way to handle this situation is to read out packets and decrement the BUFCNT counter.

If automatic flow control is disabled, the RXDMA will continue processing and BUFCNT will saturate at a value of 0xFF.

If the RX engine stops due to a lack of available descriptors, it will not start again until it detects a write to the BUFCDEC bit (ETHCON1<0>). This signals that the software has made available additional RX descriptor buffers.

Note 1:	The ETHRXST receive configuration register (starting address of RX Descriptor Table) is used by the RX DMA, and should only be changed when the RX DMA is not operating (i.e., RXEN (ETHCON1<8>) = 0 and RXBUSY (ETHSTAT<5>) = 0), since continual synchronization to the DMA clock domain is not provided. Ensuring the ETHRXST configuration register does not change when the RX DMA is active is the responsibility of software.
2:	When using an RX EDT ring, the software must make sure that the RX EDT contains (at a minimum) enough entries that are required to buffer the largest Ethernet frame. This is needed because the RX DMA engine does not detect the wrap-around condition.

35.4.9.7 ETHERNET RECEIVE OPERATION DETAILS

A received packet is stored in the packet buffer along with a status vector, which is stored in the descriptor. The status vector has two components:

- The Receive Status Vector (RSV), which is driven by the MAC at the end of a received packet, and contains information about the packet received.
- The Receive Filter Status vector (RXF_RSV) driven out by the RX Filter block.

This combined status is stored in the first descriptor used to store the packet data buffer.

The sequence of steps involved in the receiving process is as follows:

1. Software sets up the RX descriptor table with RX descriptor entries and the associated packet data buffers pointed to by each descriptor entry.
2. Software writes to the ETHRXST register (pointing to the start of the RX descriptor table) and enables the RX port by setting the RXEN bit (ETHCON1<8>).
3. The RX DMA engine reads a valid descriptor from the table and determines the start of the packet data buffer.
4. When a packet is received, the MAC indicates the start of a new frame and presents the data.
5. The RXBM receives the data and stores it in an RX FIFO. Writes to the system memory are postponed until 32 bits of data have been received. The RX DMA takes the data from the RX FIFO and stores it in the Packet data buffer pointed to by the descriptor it just read. Once all the bytes are written that have been allocated by the descriptor, the RX DMA reads another descriptor in order to write more packet data.
6. If the RX DMA fetches a descriptor with EOWN = 0 when needed to store more packet data, a RXBUFNA bit (ETHIRQ<1>) interrupt occurs.
7. If any of the following events occur during a packet reception, the packet is aborted and the descriptor is not updated with the packet status, which leaves it available for the next packet:
   - The RX Filter aborts the packet
   - RXFIFO overrun, which can be caused by:
     - Excessive system level latency
     - BUFCNT bit (ETHSTAT<16:23>) reaching maximal value
     - No descriptors are available for hardware processing
8. Once the frame completes, the MAC presents the Receive Status Vector (RSV) and the RX Filter presents the Receive Filter Status Vector (RXF_RSV).
9. The RX DMA will traverse the descriptors a second time:
   a) The first descriptor used for the current packet will be updated with the RSV, RXF_RSV, PKT_CHECKSUM and BYTE_COUNT values.
   b) All the descriptors belonging to the current packet get their SOP and EOP updated. Also the EOWN bit is changed to signal to software that it can read the packet data.
10. Once the RSV is passed to the RX DMA, the RXBM is ready to receive another packet.

Example 35-5: Ethernet Receive Packet Code

```
/* The following assumptions were made for this example:
    - the packet that has to be received might consist of multiple Ethernet frames.
    - the number of available RX descriptors is greater than the number of receive buffers that
      have to be made available for the incoming frames (there's at least an extra descriptor
      ending the chain of descriptors)
    - all the RX filtering is already programmed
    - this is the first receive operation to take place, the example enables the receive process.
    Input parameters:
    - sEthRxDcpt* pArrDcpt: pointer to an array that holds free descriptors that we can use for
      the RX operation (see below the definition for sEthRxDcpt).
    - char* pArrBuff: pointer to an array that holds buffers to receive the incoming data traffic
    - int rxBuffSize: size of the receive buffers
    - int nArrayItems: how many receive buffers are stored in the array. */

#include <p32xxxx.h>
// definition used for this example (see Table 35-7)

typedef struct
{
    volatile union
    {
        struct
        {
            unsigned: 7;
            unsigned EOWN: 1;
            unsigned NPV: 1;
            unsigned: 7;
            unsigned bCount: 11;
            unsigned: 3;
            unsigned EOP: 1;
            unsigned SOP: 1;
        };
        unsigned int w;
    }hdr;                                       // descriptor header
    unsigned char* pEDBuff;                     // data buffer address
    volatile unsigned long long   stat;         // tx packet status
    unsigned int next_ed;                       // next descriptor (hdr.NPV==1);
}__attribute__ ((__packed__)) sEthRxDcpt;      // hardware RX descriptor (linked).

extern void* VA_TO_PA(char* pBuff);             // extern function that returns the physical
                                                // address of the input virtual address parameter

int     ix;                                     // index
sEthRxDcpt* pEDcpt;                             // current Ethernet descriptor
sEthRxDcpt* tailDcpt;                           // last Ethernet descriptor
char* pBuff;                                    // current data buffer to be transmitted

pEDcpt=pArrDcpt;
pBuff=pArrBuff;
tailDcpt=0;

ETHCON2=(rxBuffSize/16)<<4;                     // set the RX data buffer size

for(ix=0; ix< nArrayItems; ix++, pEDcpt++, pBuff++)
{
    // pass the descriptor to hw, use linked descriptors, set proper size
    pEDcpt->pEDBuff=(unsigned char*)VA_TO_PA(pBuff); // set buffer
    pEDcpt->hdr.w=0;                                 // clear all the fields
    pEDcpt-> hdr.NPV= 1;                             // set next pointer valid
    pEDcpt-> hdr.EOWN= 1;                            // set hw ownership

    if(tailDcpt)
    {
    tailDcpt->next_ed=VA_TO_PA(&pEDcpt);
    }
    tailDcpt=pEDcpt;
}
// at this moment pEDcpt is an extra descriptor we use to end the descriptors list
pEDcpt->hdr.w=0;                                // software ownership
tailDcpt->next_ed= VA_TO_PA(&pEDcpt);

ETHRXST=VA_TO_PA(pArrDcpt);                     // set the address of the first RX descriptor
ETHCON1SET= 0x00008100;                         // set the ON and the RXEN

// the Ethernet Controller will receive frames and place them in the receive buffers
// we just programmed
/* do something else in between */

// can check the BUFCNT (ETHSTAT<16:23>) or RXDONE (ETHIRQ<7>)
// to see if there are packets received
```

Note: This code example uses MPLAB C32 C compiler specific syntax. Refer to your compiler manual regarding support for packed data structures.

Figure 35-11: Ethernet Descriptor Table List and Ring Format

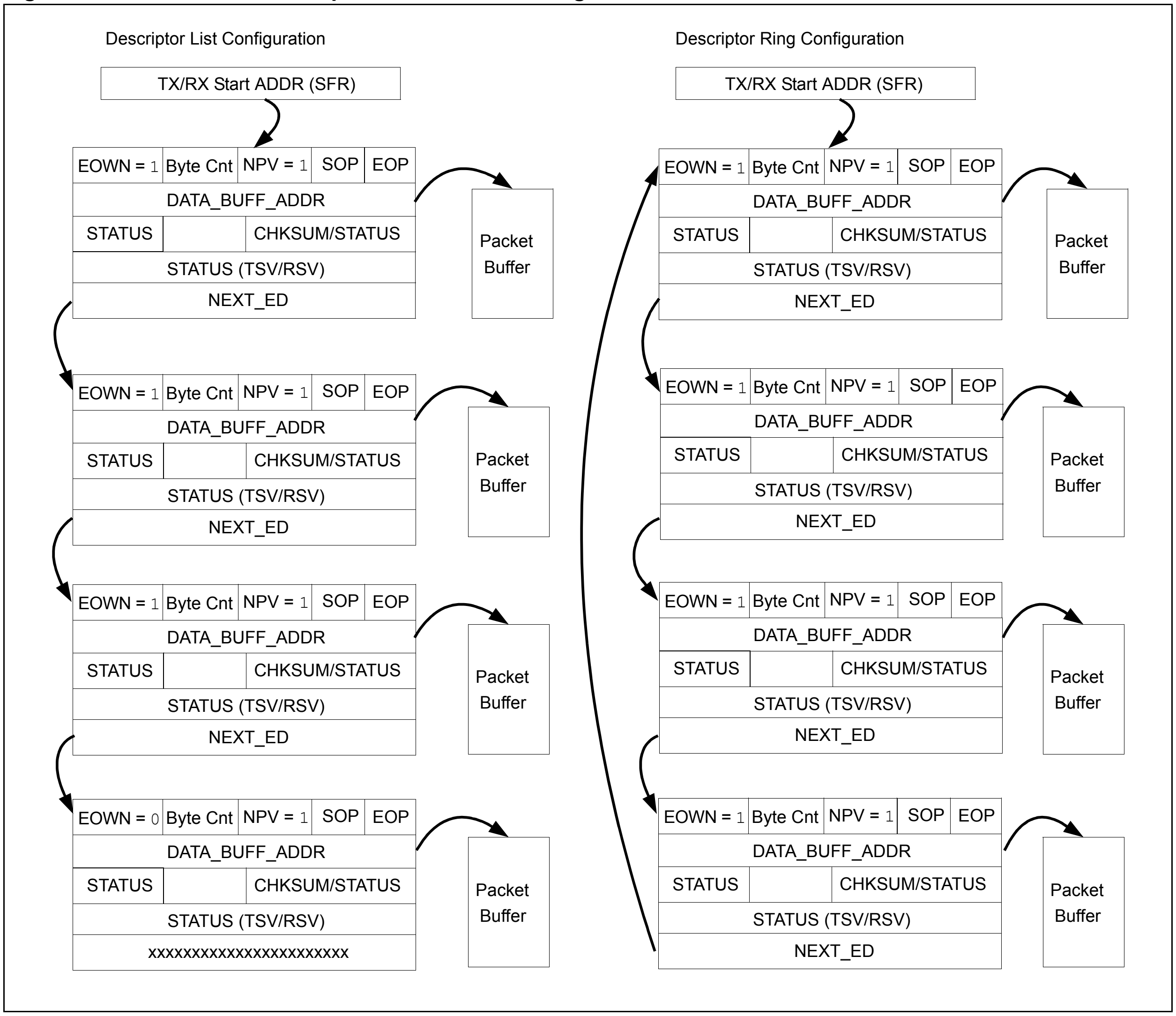
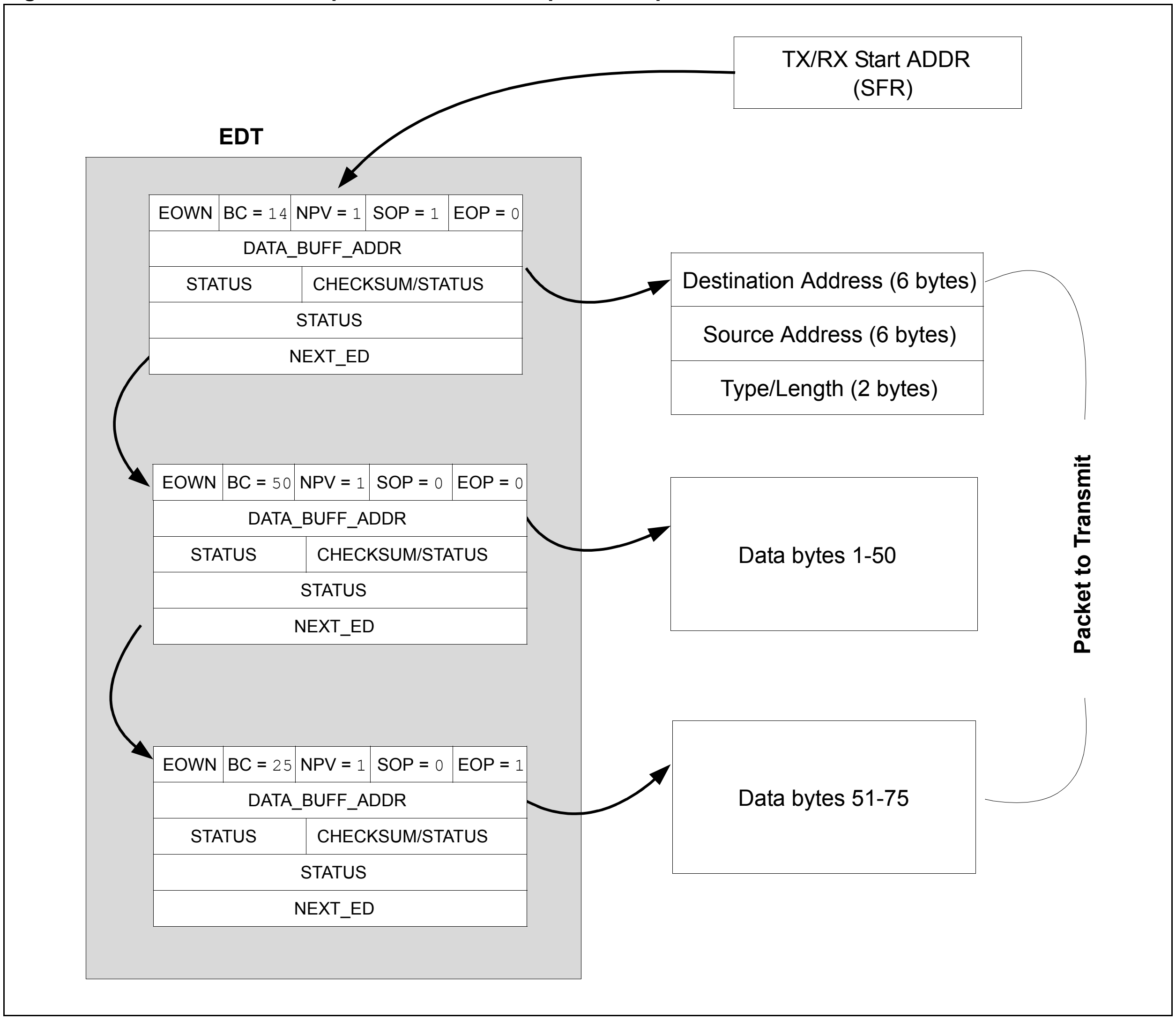


Figure 35-12: Ethernet Descriptor Table with Multiple Descriptors Per Packet



35
Ethernet Controller

35.4.10 Ethernet Initialization Sequence

To initialize the Ethernet Controller to receive and transmit Ethernet messages, Microchip recommends that this sequence of steps be followed:

1. Ethernet Controller Initialization:
   a) Disable Ethernet interrupts in the EVIC by clearing bit ETHIE (IEC1<28>).
   b) Turn the Ethernet Controller off, and then clear the ON, RXEN and TXEN bits (ETHCON1<15>, ETHCON1<8> and ETHCON1<9>).
   c) Wait activity abort by polling the ETHBUSY bit (ETHSTAT<7>).
   d) Clear the Ethernet Interrupt Flag (ETHIF) bit in the Interrupts module (IFS1<28>).
   e) Disable any Ethernet Controller interrupt generation by clearing the ETHIRQIE register.
   f) Clear the TX and RX start addresses by using ETHTXSTCLR and ETHRXSTCLR.
2. MAC Init:
   a) Reset the MAC using SOFTRESET (EMAC1CFG1<15>) or individually reset the modules setting the RESETRMCS, RESETRFUN, RESETTMCS, RESETTFUN bits (EMAC1CFG1<11:8>).
   b) Use the configuration fuse setting FETHIO bit (DEVCFG3<25>) to detect the alternate/default I/O configuration (Refer to **Section 32. “Configuration”** (DS61124) for details).
   c) Use the configuration fuse setting FMIIEN (DEVCFG3<24>) to detect the MII/RMII operation mode.
   d) Properly initialize as digital, all the pins used by the MAC - PHY interface (normally only those pins that have shared analog functionality need to be configured).
   e) Initialize the MIIM interface:
      i. If the RMII operation is selected, reset the RMII module by using the RESETRMII (EMAC1SUPP<11>) bit and set the proper speed in the SPEEDRMII bit (EMAC1SUPP<8>) bit.
      ii. Issue an MIIM block reset, by setting the RESETMGMT (EMAC1MCFG<15>) bit, and then clear the reset bit.
      iii. Select a proper divider in the CLKSEL bit (EMAC1CFG<5:2>) for the MIIM PHY communication based on the system running clock frequency and the external PHY supported clock.
3. PHY Init:

   This depends on the actual external PHY used. All PHYs should implement the basic register set as stated in Table 22-6 “MII Management Register Set” in Clause 22 of the IEEE 802.3 Specification.

   In addition to the basic register set, PHYs may provide an extended set of nine registers and capabilities that may be accessed and controlled via the MIIM interface. They provide a PHY-specific identifier, control and monitoring for the Auto-Negotiation process, etc.

   The IEEE 802.3 Specification provides room for 16 extended registers, which implement vendor-specific capabilities.

   The following is a list of common initialization steps that should be taken. Adjust accordingly by referring to the vendor-specific PHY data sheet:
   a) Reset the PHY (use Control Register 0).
   b) Set the MII/RMII operation mode. This usually requires access to a vendor-specific control register.
   c) Set the normal, swapped or auto (preferred) MDIX. This usually requires access to a vendor-specific control register.
   d) Check the PHY capabilities by investigating the Status Register 1.
   e) Preferably the auto-negotiation should be selected if the PHY supports it. Expose the supported capabilities: Half/Full Duplex, 10BaseT/100Base TX, etc. (Extended Register 4). Start the negotiation (Control Register 0) and wait for the negotiation complete and get the link partner capabilities (Extended Register 5) and negotiation result (vendor-specific register).
   f) If auto-negotiation is not supported/selected, update the PHY Duplex and Speed settings directly (use Control Register 0 and possibly some vendor-specific registers).

4. MAC Configuration:

   Having available the Duplex and Speed settings, configure the MAC accordingly, using the following steps:

   a) Enable the RXENABLE bit (EMAC1CFG1<0>), selecting both the TXPAUSE and RXPAUSE bit (EMAC1CFG1<3,2>) (the PIC32 MAC supports both).
   b) Select the desired auto-padding and CRC capabilities, and the enabling of the huge frames and the Duplex type in the EMAC1CFG2 register.
   c) Program EMAC1IPGT with the back-to-back inter-packet gap.
   d) Use EMAC1IPGR for setting the non back-to-back inter-packet gap.
   e) Set the collision window and the maximum number of retransmissions in EMAC1CLRT.
   f) Set the maximum frame length in EMAC1MAXF.
   g) Optionally set the station MAC address in the EMAC1SA0, EMAC1SA1 and EMAC1SA2 registers (these registers are loaded at reset from the factory preprogrammed station address).

5. Continue the Ethernet Controller Initialization:

   a) If planning to turn on the flow control, update the PTV value (ETHCON1<31:16>).
   b) If using the auto-flow control, set the full and empty watermarks: RXFWM and RXEWM (ETHRXWM<23:16> and ETHRXWM<7:0>).
   c) If needed, enable the auto-flow control by setting AUTOFC (ETHCON1<7>).
   d) Set the RX filters by updating the ETHHT0, ETHHT1, ETHPMM0, ETHPMM1, ETHPMCS and ETHRXFC registers.
   e) Set the size of the RX buffers in the RXBUFSZ bit (ETHCON2<10:4>) (all receive descriptors use the same buffer size). Keep in mind that using packets that are too small leads to packet fragmentation and has a noticeable impact on the performance.
   f) Prepare a list/ring of TX descriptors for messages to be transmitted. Properly update all the fields in the TX descriptor (NPV, EOWN = 1, NEXT_ED) (see Table 35-7, “Ethernet Controller TX Buffer Descriptor Format”). If using a list, end it properly with a software own descriptor (EOWN = 0).

      The SOP, EOP, DATA_BUFFER_ADDRESS and BYTE_COUNT will be updated when a particular message has to be transmitted. The DATA_BUFFER_ADDRESS will contain the physical address of the message, the BYTE_COUNT message size. SOP and EOP are set depending on how many packets are needed to transmit the message.

   g) Prepare a list of RX descriptors populated with valid buffers for messages to be received. Properly update the NPV, EOWN = 1 and DATA_BUFFER_ADDRESS fields in the RX descriptors (see Table 35-8, “Ethernet Controller RX Buffer Descriptor Format”). The DATA_BUFFER_ADDRESS should contain the physical address of the corresponding RX buffer.
   h) The actual number of RX/TX descriptors and RX previously allocated buffers depends on your actual system memory availability and on the intended Ethernet traffic you anticipate and want to handle.
   i) Update the ETHTXST register with the physical address of the head of the TX descriptors list.
   j) Update the ETHRXST register with the physical address of the head of the RX descriptors list.
   k) Enable the Ethernet Controller by setting the ON bit (ETHCON1<15).
   l) Enable the receiving of messages by setting the RXEN bit (ETHCON1<8>).
   m) Inspect the list of RX descriptors to see if the EOWN bit is cleared. If it is, this descriptor is now under software control and a message was received. Use SOP and EOP to extract the message, use BYTE_COUNT, RXF_RSV, RSV and PKT_CHECKSUM to get the message characteristics.

n) In order to transmit a message:
   i. First make sure that the message has the proper format according the Ethernet Frame specifications.
   ii. Update the necessary number of TX descriptors, starting with the head of the list, by setting the DATA_BUFFER_ADDRESS to be the physical address of the corresponding buffer in the message to be transmitted.
   iii. Keep in mind that large packet fragmentation has an impact on the performance.
   iv. Update BYTE_COUNT for each descriptor with the number of bytes contained in each buffer.
   v. Set EOWN = `1` for each descriptor that belongs to the packet.
   vi. Use SOP and EOP to specify that the message uses one or more TX descriptors.

o) Enable the transmission of the message, set the TXRTS bit (ETHCON1<9>).

p) Inspect the list of TX descriptors to see if the EOWN bit is cleared. If it is, this descriptor is now under software control and the message was transmitted. Use TSV to check for the transmission result.

35.4.11 Ethernet Statistics Registers

To comply with the 802.3 Layer Management specification, the Ethernet Controller implements various statistics registers in hardware. These registers are incremented by hardware when various conditions are detected in a transmitted/received packet. Once a register reached its maximum value, it will roll over to all zeros the next time it is incremented. Therefore, it is the responsibility of software to read these in a timely manner to avoid losing any data.

A read by software will automatically cause the corresponding register to be cleared.

Statistics counters can be written by software using the SET, CLR and INV registers. Writes to the normal registers are also supported. In normal operation, the statistics registers should just be read on a periodic basis to collect data on the Ethernet link traffic.

> **Note 1:** The SET, CLR and INV Statistics registers are meant only for supporting software debugging and testing.
>
> **2:** When the device is put in Sleep mode, updates to the statistics registers are suspended as the system clock is not running. The only exception to this is an overflow case, which will increment the overflow counter and set the overflow flag RXOVFLW bit (ETHIRQ<0>). This is done to signal to software upon a wake-up that some packets have been lost.
>
> **3:** Some statistical counters may immediately increment when exiting Sleep due to pending events.

35.4.11.1 PAYLOAD CHECKSUM CALCULATION

The Ethernet Controller automatically calculates a 16-bit packet checksum for all received packets and stores the 16-bit value along with the received packets status vector in the packet descriptor PKT_CHECKSUM field (RX_DCPT<96:111>). This checksum can be useful for TCP/IP software implementations.

The payload checksum is calculated over the complete received packet except for the first 14 bytes (destination address, source address and length/type fields). If software needs to exclude more bytes from the checksum, it must subtract the values out.

A payload checksum is a simple checksum used to provide basic protection against bit corruption during transmission. It is typically used in TCP and UDP packets of the TCP/IP protocols. The checksum is calculated by dividing the byte stream into 16-bit words and adding them together. Any overflow is also added back into the sum. The checksum calculation begins after the first 14 bytes of the frame are received and includes the FCS bytes. The result is the 1's complement of the calculated sum.

Refer to Figure 35-7 for an example of payload checksum calculation.

35.5 ETHERNET INTERRUPTS

The PIC32 device has the ability to generate interrupts reflecting the events that occur during the Ethernet Controller's transfer of frames. Each of the Ethernet Controller interrupt events has a corresponding interrupt enable bit in the ETHIEN register, which must be set for an interrupt to be generated. However, regardless of the value of the ETHIEN register, the status of all interrupt events is directly readable via the ETHIRQ register. Therefore, the software has visibility of an event generating a potential interrupt by polling the register and not having an interrupt propagate out of the module.

Ethernet interrupts are persistent. This means that as long as the event that generated the interrupt is pending, the interrupt signal from the Ethernet Controller module will remain asserted.

Following is a description of the interrupt events generated by the transmission and receive of Ethernet frames.

Transmit path related interrupt events:

- TX DMA engine transfer error interrupt, signaled by the TXBUSE bit (ETHIRQ<14>) and enabled using the TXBUSEIE bit (ETHIEN<14>). This event occurs when the TX DMA encounters a bus error during a memory access and is caused by an addressing error (usually because of a bad pointer).
- Transmission done interrupt, signaled by the TXDONE bit (ETHIRQ<3>) and enabled using the TXDONEIE bit (ETHIEN<3>). This event occurs when the currently transmitted TX packet completes transmission and the Transmit Status Vector is loaded into the first descriptor of the packet.
- Transmission aborted interrupt, signaled by the TXABORT bit (ETHIRQ<2>) and enabled using the TXABORTIE bit (ETHIEN<2>). This event occurs when the MAC aborts the transmission because of one of the following reasons:
  - Jumbo TX packet abort (The size of the packet is greater than the maximum size MACMAXF bit (EMAC1MAXF<15:0>)
  - Underrun abort (The transmit engine cannot keep up with the requested data flow. This usually happens when the system bus is overloaded.)
  - Excessive defer abort (Packet was deferred in excess of 6071 nibble times in 100 Mbps mode or 24,287 bit times in 10 Mbps mode)
  - Late collision abort (Collision occurred beyond the collision window)
  - Excessive collisions abort (Packet was aborted because the number of collisions exceeded RETX bit (EMAC1CLRT<3:0>)

> **Note:** An early collision will cause the MAC to assert the Retry, but not the Abort. This condition will therefore not cause an interrupt.

Receive path related interrupt events:

- RX DMA engine transfer error interrupt, signaled by the RXBUSE bit (ETHIRQ<13>) and enabled using the RXBUSEIE bit (ETHIEN<13>). This event occurs when the RX DMA encounters a bus error during a memory access and is caused by an addressing error (usually because of a bad pointer).
- Receive done interrupt, signaled by the RXDONE bit (ETHIRQ<7>) and enabled using the RXDONEIE bit (ETHIEN<7>). This event occurs whenever a packet is successfully received.
- Packet pending interrupt, signaled by the PKTPEND bit (ETHIRQ<6>) and enabled using the PKTPENDIE bit (ETHIEN<6>). This event occurs whenever the buffer counter BUFCNT bit (ETHSTAT<16:23>) has a value greater than 0.
- Receive Activity interrupt, signaled by the RXACT bit (ETHIRQ<5>) and enabled using the RXACTIE bit (ETHIEN<5>). This event occurs whenever there is data stored in the RXBM FIFO.
- Receive buffer not available interrupt, signaled by the RXBUFNA bit (ETHIRQ<1>) and enabled using the RXBUFNAIE bit (ETHIEN<1>). This event occurs whenever the RX DMA runs out of descriptors by fetching a descriptor not owned by hardware (EOWN = 0).

- Receive FIFO overflow error interrupt, signaled by the RXOVFLW bit (ETHIRQ<0>) and enabled using the RXOVFLIE bit (ETHIEN<0>). This event occurs whenever the RXBM is unable to transfer data out of the receive FIFO to the system memory and the internal FIFO overflows because of one of the following reasons:
  - Excessive system level latency
  - BUFCNT bit (ETHSTAT<16:23>) reaching maximal value
  - No descriptors are available for hardware processing
- Empty Watermark interrupt, signaled by the EWMARK bit (ETHIRQ<9>) and enabled using the EWMARKIE bit (ETHIEN<9>). This event occurs whenever the RX descriptor buffer count BUFCNT bit (ETHSTAT<16:23>) is less than or equal to the RXEWM bit ETHRXWM<0:7>) value.
- Full Watermark interrupt, signaled by the FWMARK bit (ETHIRQ<8>) bit and enabled using the FWMARKIE bit (ETHIEN<8>). This event occurs whenever the RX descriptor buffer count BUFCNT bit (ETHSTAT<16:23>) is greater than or equal to the RXFWM bit (ETHRXWM<16:23>) value.

Also, interrupts in the Ethernet peripheral could be divided into two types, depending on how and where the interrupt is cleared. They are software cleared interrupt events and hardware cleared interrupt events. Note that all the interrupt events are cleared by a device Reset.

Software cleared interrupt events are cleared by writing the corresponding bit in the ETHIRQCLR register or the ETHIRQ register directly, and include the following:

- TXBUSE
- RXBUSE
- RXDONE
- RXACT
- TXDONE
- TXABORT
- RXBUFNA
- RXOVFLW

Hardware cleared interrupt events are cleared by removing the condition that caused the interrupt, and include the following:

- EWMARK – This interrupt can be cleared when an RX packet is successfully received and the BUFCNT value is greater than the RXEWM bit (ETHRXWM<0:7>) value.
- FWMARK – This interrupt can be cleared by writing to the BUFCDEC bit (ETHCON1<0>), thereby decrementing the buffer descriptor count (BUFCNT) below the RXFWM (ETHRXWM<16:23>) value.
- PKTPEND – This interrupt can be cleared by writing the BUFCDEC bit until BUFCNT (ETHSTAT<16:23>) reaches '0'.

All the interrupts belonging to the Ethernet controller map to the Ethernet interrupt vector.

The corresponding Ethernet Controller interrupt flag is ETHIF (IFS1<28>). This interrupt flag must be cleared in software once the cause generating the interrupt is processed.

The Ethernet controller is enabled as a source of interrupts via the respective Ethernet Controller interrupt enable bit, ETHIE (IEC1<28>).

The interrupt priority-level bits and interrupt subpriority-level bits must also be configured:

- ETHIP<2:0> (IPC12<4:2>)
- ETHIS<1:0> (IPC12<1:0>)

Note:	Refer to **Section 8. "Interrupts"** (DS61108) for detailed descriptions of the IFSx, IECx and IPCx interrupt bits.

35.5.1 Interrupt Configuration

The Ethernet Controller module has multiple internal interrupt flags (TXBUSE, RXBUSE, EWMARK, FWMARK, RXDONE, PKTPEND, RXACT, TXDONE, TXABORT, RXBUFNA, RXOVFLW) and corresponding enable interrupt control bits (TXBUSEIE, RXBUSEIE, EWMARKIE, FWMARKIE, RXDONEIE, PKTPENDIE, RXACTIE, TXDONEIE, TXABORTIE, RXBUFNAIE, RXOVFLIE). However, for the interrupt controller, there is just one dedicated interrupt flag bit for the Ethernet Controller: ETHIF (IFS1<28>) and the corresponding interrupt enable/mask bit: ETHIE (IEC1<28>).

> **Note:** All of the interrupt conditions for the Ethernet Controller module share just one interrupt vector.

The Ethernet Controller module has its own priority and sub-priority levels independent of other peripherals. Note that the ETHIF bit will be set without regard to the state of the corresponding enable bit ETHIE. The ETHIF bit can be polled by software if desired.

The ETHIE bit is used to define the behavior of the Vector Interrupt Controller (INT) module when a corresponding ETHIF bit is set. When the corresponding ETHIE bit is clear, the Interrupts module does not generate a CPU interrupt for the event. If the ETHIE bit is set, the Interrupts module will generate an interrupt to the CPU when the ETHIF bit is set (subject to the priority and sub-priority as follows).

It is the responsibility of the user's software routine that services a particular interrupt to clear the interrupt flag bit before the service routine is complete.

The priority of the Ethernet Controller module interrupt can be set using the IPC12 register of the INT controller. This priority defines the priority group to which the interrupt source will be assigned. The priority groups range from a value of 7 (the highest priority), to a value of 0, which does not generate an interrupt. An interrupt being serviced will be preempted by an interrupt in a higher priority group.

The sub-priority bits allow setting the priority of an interrupt source within a priority group. The values for the sub-priority range from 3 (the highest priority), to 0 the lowest priority. An interrupt with the same priority group but having a higher sub-priority value will not preempt a lower sub-priority interrupt that is in progress.

The priority group and sub-priority bits allow more than one interrupt source to share the same priority and sub-priority. If simultaneous interrupts occur in this configuration, the natural order of the interrupt sources within a priority/sub-priority group pair determine the interrupt generated.

The natural priority is based on the vector numbers of the interrupt sources. The lower the vector number the higher the natural priority of the interrupt. Any interrupts that were overridden by natural order will then generate their respective interrupts based on Priority, sub-priority and natural order after the interrupt flag for the current interrupt is cleared.

After an enabled interrupt is generated, the CPU will jump to the vector assigned to that interrupt. The vector number for the interrupt is the same as the natural order number. The CPU will then begin executing code at the vector address. The user's code at this vector address should perform any application-specific operations and clear the ETHIF interrupt flags (as well as the corresponding event in the ETHIRQ register if a software clearable interrupt) and then exit.

Refer to the vector address table details in **Section 8. "Interrupts"** (DS61108) for more information.

Table 35-9: Ethernet Interrupt Vectors for Various Offsets with EBASE = 0x8000:0000

Interrupt	Vector/ Natural Order	IRQ Number	Vector Address IntCtl.VS = 0x01	Vector Address IntCtl.VS = 0x02	Vector Address IntCtl.VS = 0x04	Vector Address IntCtl.VS = 0x08	Vector Address IntCtl.VS = 0x10
ETH	48	60	8000 0800	8000 0e00	8000 1a00	8000 3200	8000 6200

Example 35-6: Ethernet Initialization with Interrupts Enabled Code

```
// this code example assumes that the system vectored interrupts are properly configured
#include <p32xxxx.h>

    IEC1CLR =0x10000000;            // disable Ethernet interrupts

    ETHCON1CLR=0x00008300;          // reset: disable ON, clear TXRTS, RXEN

    while(ETHSTAT&0x80);            // wait everything down

    IFS1CLR=0x10000000;                 // clear the interrupt controller flag
    ETHIENCLR=0x000063ef;           // disable all events
    ETHIRQCLR=0x000063ef;           // clear any existing interrupt event

    ETHCON1SET=0x00008000;          // turn device ON
    /*
    Init the MAC
    Init the PHY
    Init RX Filtering
    Init Flow Control
    */

    ETHIENSET=0x0000400c;           // enable the TXBUSE, TXDONE
                                    // and TXABORT interrupt events
    IPC12CLR = 0x0000001f;          // clear the Ethernet Controller priority and sub-priority
    IPC12SET = 0x00000016;          // set IPL 5, sub-priority 2
    IEC1SET=0x10000000;             // enable the Ethernet Controller interrupt

    // start transmit packets
    // whenever a packet completes transmission or an transmission error occurs
    // an interrupt will be generated
```

Example 35-7: Ethernet Controller ISR Code

```
/*
The following code example demonstrates a simple Interrupt Service Routine for Ethernet
Controller interrupts. The user’s code at this vector should perform any application specific
operations and must clear the Ethernet Controller interrupt flags before exiting.
*/

#include <p32xxxx.h>
void __ISR(_ETH_VECTOR, IPL5) __EthInterrupt(void)
{
    int ethFlags=ETHIRQ;          // read the interrupt flags (requests)

    // the sooner we acknowledge, the smaller the chance to miss another event of the
    // same type because of a lengthy ISR
    ETHIRQCLR= ethFlags;          // acknowledge the interrupt flags

    /*
    perform application specific operations in response
    to any interrupt flag set in ethFlags
    */

    IFS1CLR= 0x10000000;          // Be sure to clear the Ethernet Controller Interrupt
                                  // Controller flag before exiting the service routine.
}
```

Note:	The Ethernet Controller ISR code example shows MPLAB C32 C compiler-specific syntax. Refer to your compiler manual regarding support for ISRs.

35.5.2 External PHY Interrupt

Some PHYs have the option of generating an interrupt signal when a specific event occurs. A PHY interrupt is usually asserted for the following types of events/conditions:

- Energy Detect
- Power-Down mode exited
- Auto-negotiation complete
- Remote fault detected
- Link down
- Auto-negotiation LP acknowledge
- Parallel detection fault
- Auto-negotiation page received

Refer the vendor-specific PHY data sheet for the details about the events generating interrupts.

If the PIC32 device must be made aware and respond to this interrupt, the PHY interrupt signal should be tied to a PIC32 external interrupt pin. Note that this interrupt does not go through the Ethernet Controller module.

The software must clear the PHY generated interrupt event by writing a register in the PHY via the MAC MIIM registers. Note that in this case, the PHY interrupt is the only software-clearable interrupt that is not directly clearable in the ETHIRQ register.

35.6 OPERATION IN POWER-SAVING AND DEBUG MODES

35.6.1 Ethernet Operation in Sleep Mode

When the PIC32 device enters Sleep mode, the system clock is disabled. No Ethernet transfers can occur in this mode. For the Ethernet Controller module, all clocks are stopped except for the external MII RX_CLK and TX_CLK signals or REF_CLK if operating in RMII mode. The Ethernet Controller module is in Sleep mode with asynchronous wake-up events allowed.

If the user application enters Sleep mode while the Ethernet Controller is operating on an active transfer, it will be suspended in its current state until clock execution resumes. The software should avoid this situation as it might result in unexpected pin timings.

Software is responsible for determining when the link is in a state that is safe for the Ethernet Controller to enter Sleep mode. Execution of the `WAIT` instruction by the CPU to place the device in Sleep mode is only recommended in two cases:

- The Ethernet Controller is disabled
- The Ethernet Controller has no pending TX packets and all the incoming RX packets are processed

Placing the Ethernet Controller in Sleep mode while transmit transactions on the bus are active, may result in improper Ethernet device behavior, which may cause dropped packets or a broken link connection.

Once the device has safely entered Sleep Mode, the Ethernet Controller will generate a wake-up interrupt when an asynchronous enabled event occurs.

35.6.2 Ethernet Operation in Idle Mode

When the device enters Idle mode, the system clock sources remain functional. The SIDL bit (ETHCON1<13>) selects whether the Ethernet Controller will stop or continue functioning in Idle mode:

- If SIDL = `0`, the Ethernet Controller will continue normal operation in Idle mode and will have the clocks turned on
- If SIDL = `1`, the Ethernet Controller will discontinue operation in Idle mode. The Ethernet controller will turn off the clocks (except RX_CLK, TX_CLK or REF_CLK) so that power consumption is more efficient. When the Ethernet Controller is stopped in Idle mode, it behaves the same as when in Sleep mode.

Note:	The Ethernet Controller treats Idle mode with SIDL = `1` (ETHCON1<13>) the same as the Sleep mode. Therefore, the same restrictions apply.

35.6.3 Wake-on-LAN (WOL) Operation

There is no specific WOL functionality implemented in the Ethernet Controller. Instead, WOL functionality is implemented by setting the appropriate RX filters and enabling the RXDONE bit (ETHIRQ<7>) interrupt to wake this system when a receive packet is accepted. Normally a Magic Packet filter is used for this purpose.

If the system is in Sleep mode or in a slow-clock mode, the software can enable the RXACT bit (ETHIRQ<5>) interrupt. This prevents the receive buffer from overflowing while the device is in a low power mode.

Special care must be taken when using RXACT to wake-up the system from Sleep or Idle (and SIDL bit (ETHCON1<13>) is set).

Example 35-8 provides a sequence of instructions to put the system into Sleep or Idle mode.

Example 35-8: Using the Ethernet Controller to Wake-up the System

```
/*
The following code example illustrates a simple sequence of instructions to put the system into
Sleep or Idle state so that the incoming Ethernet activity, signaled by RXACT, is used to
wake-up the system. It assumes that either the Sleep state is enabled or, if the Idle state is
enabled, the Ethernet Controller SIDL bit is set.
*/
#include <p32xxxx.h>

    if(want_to_sleep)
    {
        ETHIRQCLR = 0x20;      // clear RXACT flag
        ETHIENSET = 0x20;      // enable RXACT interrupt
        asm("wait");           // go to Sleep/Idle
    }
```

Under these circumstances, the wake-up ISR should look like Example 35-9:

Example 35-9: Ethernet Controller Wake-up on RXACT ISR

```
/*
The following code example demonstrates a simple Interrupt Service Routine for Ethernet
Controller interrupts. The user's code at this vector should perform any application specific
operations and must clear the Ethernet Controller interrupt flags before exiting.
*/
#include <p32xxxx.h>

void __ISR(_ETH_VECTOR, IPL5) __EthInterrupt(void)
{
int ethFlags=ETHIRQ;           // read the interrupt flags (requests)

    ethFlags =ETHIRQ;
    if(ethFlags &0x20)
    { // RXACT interrupt
        ETHIENCLR = 0x20;      // disable further RXACT interrupts
        ETHCON1CLR= 0x2000;    // disable SIDL
                               // suspend any activity in your system that could interfere
                               // with the critical system unlock sequence such as:
                               // disable higher priority interrupts, DMA transfers, etc.
                               // now unlock the system
        SYSKEY = 0, SYSKEY = 0xAA996655, SYSKEY = 0x556699AA;
        OSCCONCLR = 0x10;      // disable SLEEP mode, enable IDLE
        SYSKEY = 0x33333333;   // relock the system
                               // resume the activity previously stopped: re-enable interrupts,
                               // DMA, etc.
    }

    /*
    perform other application specific operations in response
    to other interrupt flag set in ethFlags
    */

    ETHIRQCLR= ethFlags;       // acknowledge the interrupt flags

    IFS1CLR= 0x10000000;       // clear the Ethernet Controller Interrupt Controller
                               // flag before exiting the service routine.
}
```

Note:	The Ethernet Controller ISR code example shows MPLAB C32 C compiler-specific syntax. Refer to your compiler manual regarding support for ISRs.

The disabling of further RXACT interrupts in this ISR is needed because of the way the Ethernet Controller generates this interrupt request, which is active as long as there is data in the RX FIFO. Otherwise, the control will continuously get back to the ISR and execution will be locked up.

However, instead of simply disabling the RXACT interrupt, further action is needed: disable the Sleep mode and enable the Idle mode and make sure that the Ethernet Controller is enabled (SIDL = `0`). This is needed to prevent the situation where a RXACT interrupt was taken exactly after the enabling of the RXACT, but just before the execution of the `WAIT` instruction in the main loop of Example 35-9.

If activity was received right before the `WAIT` instruction call, the ISR will disable the RX activity interrupt and when the ISR returns the main loop will execute the `WAIT` instruction. At this point, the Ethernet Controller will never be able to wake the part.

The workaround is to have the ISR disable Sleep mode and clear the Ethernet SIDL bits so that the Ethernet Controller will not go to sleep.

Please note that another Ethernet interrupt needs to be enabled after the RXACT interrupt is disabled. Usually, the RXDONE bit is the best candidate.

35.7 EFFECTS OF VARIOUS RESETS

35.7.1 Device Reset

All Ethernet registers are forced to their reset states upon a device Reset. When the asynchronous reset input goes active, the Ethernet logic does the following:

- Resets all fields in the SFRs (ETHCON1, ETHCON2, ETHSTAT, etc.)
- Loads the EMAC1SA0, EMAC1SA1 and EMAC1SA2 registers containing the station address from the factory pre-programmed station address
- Resets the TX and RX DMA engines, and puts the corresponding FIFOs in the empty state
- Aborts any on-going data transfers

35.7.2 Power-on Reset

All Ethernet Controller registers are forced to their reset states upon a Power-on Reset.

35.7.3 Watchdog Timer Reset

All Ethernet Controller registers are forced to their reset states upon a Watchdog Timer Reset.

35.8 I/O PIN CONTROL

Enabling the Ethernet Controller will configure the I/O pin direction as defined by the Ethernet Controller control bits (see Table 35-10). The port TRIS and LATCH registers will be overridden.

Table 35-10: I/O Pin Configuration for Use with the Ethernet Controller(1)

I/O Pin Name	MII Required	RMII Required	Module Control	TRIS(2)	Pin Type	Description
EMDC	Yes	Yes	ON	X	O	Ethernet MII Management Clock
EMDIO	Yes	Yes	ON	X	I/O	Ethernet MII Management IO
ETXCLK	Yes	No	ON	X	I	Ethernet MII TX Clock
ETXEN	Yes	Yes	ON	X	O	Ethernet Transmit Enable
ETXD0	Yes	Yes	ON	X	O	Ethernet Data Transmit 0
ETXD1	Yes	Yes	ON	X	O	Ethernet Data Transmit 1
ETXD2	Yes	No	ON	X	O	Ethernet Data Transmit 2
ETXD3	Yes	No	ON	X	O	Ethernet Data Transmit 3
ETXERR	Yes	No	ON	X	O	Ethernet Transmit Error
ERXCLK	Yes	No	ON	X	I	Ethernet MII RX Clock
EREF_CLK	No	Yes	ON	X	I	Ethernet RMII Ref Clock
ERXDV	Yes	No	ON	X	I	Ethernet MII Receive Data Valid
ECRS_DV	No	Yes	ON	X	I	Ethernet RMII Carrier Sense/Receive Data Valid
ERXD0	Yes	Yes	ON	X	I	Ethernet Data Receive 0
ERXD1	Yes	Yes	ON	X	I	Ethernet Data Receive 1
ERXD2	Yes	No	ON	X	I	Ethernet Data Receive 2
ERXD3	Yes	No	ON	X	I	Ethernet Data Receive 3
ERXERR	Yes	Yes	ON	X	I	Ethernet Receive Error
ECRS	Yes	No	ON	X	I	Ethernet Carrier Sense
ECOL	Yes	No	ON	X	I	Ethernet Collision Detected

Note 1: Ethernet controller pins that are not used by selected interface but can be used by other peripherals.

2: The setting of the TRIS bit is irrelevant. However, if the pin is shared with an analog input, the AD1PCFG and corresponding TRIS register must be properly set to configure this pin as digital.

35.9 RELATED APPLICATION NOTES

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the PIC32 device family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to Ethernet Controller module are:

Title	Application Note #
Ethernet Theory of Operation	AN1120

Note: Please visit the Microchip web site (www.microchip.com) for additional application notes and code examples for the PIC32 family of devices.

35.10 REVISION HISTORY

Revision A (August 2009)

This is the initial released version of this document.

Revision B (October 2011)

This revision includes the following updates:

- Added a Note at the beginning of the section, which provides information on the complementary documentation
- Changed the document running header from PIC32MX Family Reference Manual to PIC32 Family Reference Manual
- Changed all occurrences of PIC32MX to PIC32
- Updated all occurrences of MACx as MAC1
- Changed all occurrences of SCLK to SYSCLK
- Updated the Signal Name column and added new column for IEEE 802.3 signals in Table 35-4 and Table 35-5
- Updated signal names in Figure 35-4 and Figure 35-5
- Added a note in Table 35-10 indicating that the Ethernet controller pins that are not used by selected interface but can be used by other peripherals
- Removed the following Clear, Set and Invert registers:
  - RCONCLR: Reset Control Clear Register
  - RCONSET: Reset Control Set Register
  - RONINV: Reset Control Invert Register
  - RSWRSTCLR: Software Reset Clear Register
  - RSWRSTSET: Software Reset Set Register
  - RSWRSTINV: Software Reset Invert Register
- Updated all r-x bits as U-0 bits in Register 35-1 and Register 35-39
- Modifications to register formatting and minor text updates have been made throughout the document

NOTES:

ISBN: 978-1-61341-711-9

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2009

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Hangzhou
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-330-9305

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

08/02/11