第 35 章 以太网控制器

目录

本章包括下列主题：

35.1 简介

以太网控制器是与片外 PHY 连接的总线主模块，用以在系统中实现完整的以太网节点。

下面列出了该模块的部分主要特性：

- 支持 10/100 Mbps 数据传输速率
- 支持全双工和半双工操作
- 支持精简型介质无关接口 （Reduced Media Independent Interface， RMII）和介质无关接口 （Media Independent Interface， MII） PHY 接口
- 支持 MII 管理 （MIIM） PHY 管理接口
- 支持手动和自动流量控制
- 支持自动 MDIX 和使能的 PHY
- 接收和发送路径均采用基于 RAM 描述符的 DMA 操作
- 可完全配置的中断
- 可配置的接收数据包过滤
  - CRC 校验
  - 64 字节模式匹配
  - 广播、组播和单播数据包
  - Magic Packet™
  - 64 位哈希表
  - 过短 （runt）数据包
- 支持数据包有效载荷校验和计算
- 支持各种硬件统计计数器

35.2 以太网控制器概述

以太网控制器提供了使用外部PHY芯片实现10/100 Mbps以太网节点所需的模块。为了分担CPU从模块中移出和向模块中移入数据包数据所产生的开销，控制器中包含了基于内部描述符的DMA引擎。

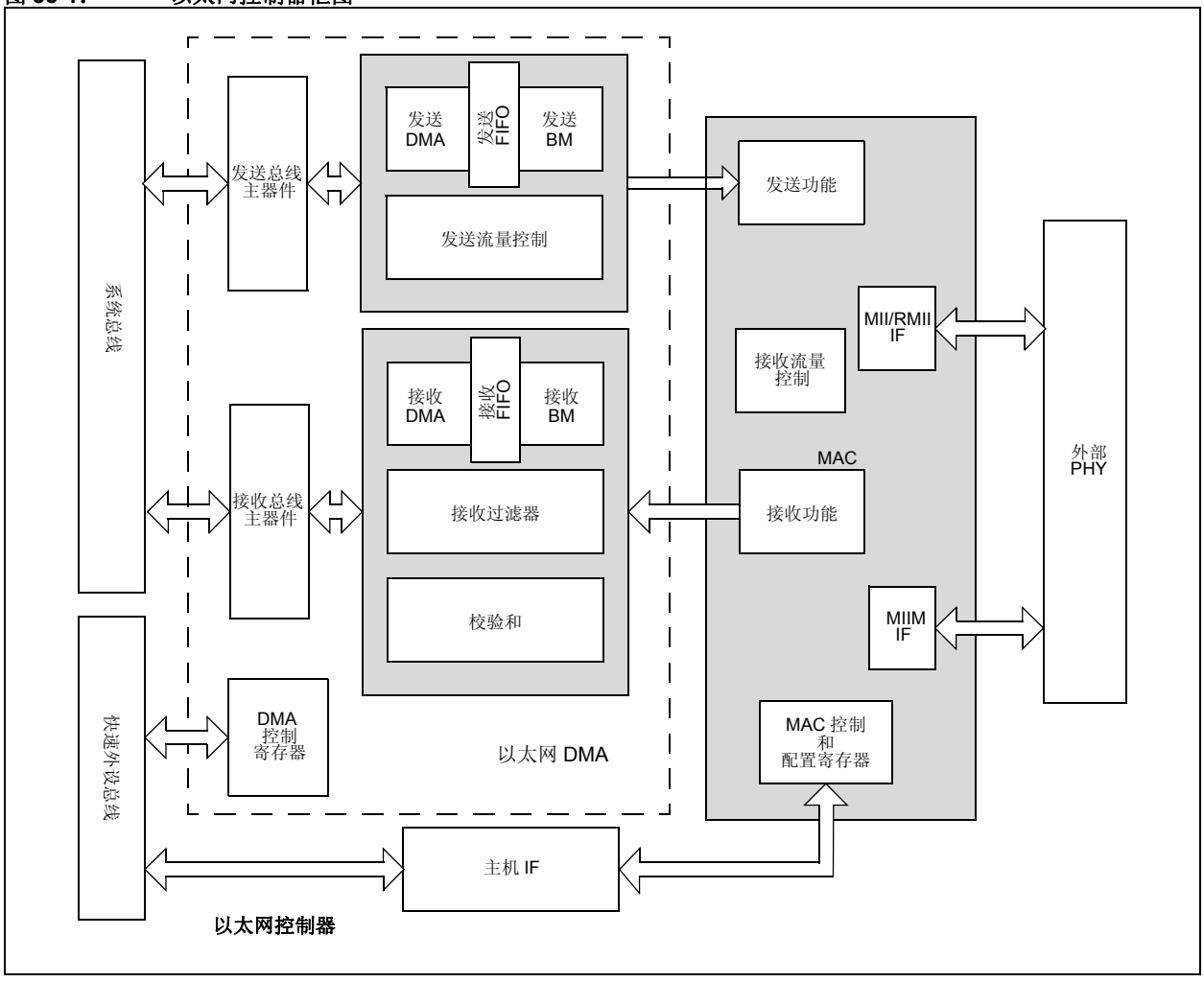
以太网控制器包含以下模块：

- 介质访问控制 （Media Access Control， MAC）模块：负责实现以太网规范的 MAC 功能。
- 流量控制 （Flow Control， FC）模块：负责控制暂停帧的发送。暂停帧的接收在 MAC 中进行处理。
- 接收过滤器 （RXF）模块：该模块用于对每个接收数据包执行过滤，以确定对于每个数据包是接收还是拒绝。
- 发送 DMA/ 发送 BM 引擎：发送 DMA 和发送缓冲区管理引擎用于执行从存储器 （使用描述符表）到 MAC 发送接口的数据传输。
- 接收 DMA/ 接收 BM 引擎：接收 DMA 和接收缓冲区管理引擎用于将接收数据包从 MAC 传输到存储器中 （使用描述符表）。

图 35-1 给出了以太网控制器的框图。

注： 关于以太网操作的详细说明，请参见 AN1120 *“Ethernet Theory of Operation”*（可从 Microchip 网站 www.microchip.com 获取）和 IEEE 802.3 规范（www.ieee.org）。

图 35-1： 以太网控制器框图



35

以太网控制器

35.3 状态和控制寄存器

以太网控制器模块包含以下特殊功能寄存器 （Special Function Register， SFR）：

控制器和 DMA 引擎配置 / 状态寄存器：

- ETHCON1：以太网控制器控制 1 寄存器
- ETHCON2：以太网控制器控制 2 寄存器
- ETHTXST：以太网控制器发送数据包描述符起始地址寄存器
- ETHRXST：以太网控制器接收数据包描述符起始地址寄存器
- ETHIEN：以太网控制器中断允许寄存器
- ETHIRQ：以太网控制器中断请求寄存器
- ETHSTAT：以太网控制器状态寄存器

接收过滤配置寄存器：

- ETHRXFC：以太网控制器接收过滤器配置寄存器
- ETHHT0：以太网控制器哈希表 0 寄存器
- ETHHT1：以太网控制器哈希表 1 寄存器
- ETHPMM0：以太网控制器模式匹配屏蔽器 0 寄存器
- ETHPMM1：以太网控制器模式匹配屏蔽器 1 寄存器
- ETHPMCS：以太网控制器模式匹配校验和寄存器
- ETHPMO：以太网控制器模式匹配偏移寄存器

流量控制配置寄存器：

- ETHRXWM：以太网控制器接收水印寄存器

以太网统计寄存器：

- ETHRXOVFLOW：以太网控制器接收溢出统计寄存器
- ETHFRMTXOK：以太网控制器正常发送帧统计寄存器
- ETHSCOLFRM：以太网控制器单次冲突帧统计寄存器
- ETHMCOLFRM：以太网控制器多次冲突帧统计寄存器
- ETHFRMRXOK：以太网控制器正常接收帧统计寄存器
- ETHFCSERR：以太网控制器帧校验序列错误统计寄存器
- ETHALGNERR：以太网控制器对齐错误统计寄存器

MAC 配置寄存器：

- EMACxCFG1：以太网控制器 MAC 配置 1 寄存器
- EMACxCFG2：以太网控制器 MAC 配置 2 寄存器
- EMACxIPGT：以太网控制器 MAC 背靠背包间间隔寄存器
- EMACxIPGR：以太网控制器 MAC 非背靠背包间间隔寄存器
- EMACxCLRT：以太网控制器 MAC 冲突窗口 / 重试限制寄存器
- EMACxMAXF：以太网控制器 MAC 最大帧长寄存器
- EMACxSUPP：以太网控制器 MAC PHY 支持寄存器
- EMACxTEST：以太网控制器 MAC 测试寄存器
- EMACxSA0：以太网控制器 MAC 站点地址 0 寄存器
- EMACxSA1：以太网控制器 MAC 站点地址 1 寄存器
- EMACxSA2：以太网控制器 MAC 站点地址 2 寄存器

MII 管理寄存器：

- EMACxMCFG：以太网控制器 MAC MII 管理配置寄存器
- EMACxMCMD：以太网控制器 MAC MII 管理命令寄存器
- EMACxMADR：以太网控制器 MAC MII 管理地址寄存器
- EMACxMWTD：以太网控制器 MAC MII 管理写数据寄存器
- EMACxMRDD：以太网控制器 MAC MII 管理读数据寄存器
- EMACxMIND：以太网控制器 MAC MII 管理指示寄存器

表 35-1 简要汇总了以太网控制器寄存器。该汇总表之后列出了相应的寄存器，并且每个寄存器均附有详细的说明。

表 35-1：　以太网控制器寄存器汇总

地址偏移	名称	位范围	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
0x0000	ETHCON1(1,2,3)	31:24	PTV <15:8>							
		23:16	PTV <7:0>							
		15:8	ON	FRZ	SIDL	—	—	—	TXRTS	RXEN
		7:0	AUTOFC	—	—	MANFC	—	—	—	BUFCDEC
0x0010	ETHCON2(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	RXBUFSZ<6:4>		
		7:0	RXBUFSZ<3:0>				—	—	—	—
0x0020	ETHTXST(1,2,3)	31:24	TXSTADDR<31:24>							
		23:16	TXSTADDR<23:16>							
		15:8	TXSTADDR<15:8>							
		7:0	TXSTADDR<7:2>						—	—
0x0030	ETHRXST(1,2,3)	31:24	RXSTADDR<31:24>							
		23:16	RXSTADDR<23:16>							
		15:8	RXSTADDR<15:8>							
		7:0	RXSTADDR<7:2>						—	—
0x0040	ETHHT0(1,2,3)	31:24	HT<31:24>							
		23:16	HT<23:16>							
		15:8	HT<15:8>							
		7:0	HT<7:0>							
0x0050	ETHHT1(1,2,3)	31:24	HT<63:56>							
		23:16	HT<55:48>							
		15:8	HT<47:40>							
		7:0	HT<39:32>							
0x0060	ETHPMM0(1,2,3)	31:24	PMM<31:24>							
		23:16	PMM<23:16>							
		15:8	PMM<15:8>							
		7:0	PMM<7:0>							
0x0070	ETHPMM1(1,2,3)	31:24	PMM<63:56>							
		23:16	PMM<55:48>							
		15:8	PMM<47:40>							
		7:0	PMM<39:32>							
0x0080	ETHPMCS(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	PMCS<15:8>							
		7:0	PMCS<7:0>							
0x0090	ETHPMO(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	PMO<15:8>							
		7:0	PMO<7:0>							
0x00A0	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
0x00B0	ETHRXWM(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	RXFWM<7:0>							
		15:8	—	—	—	—	—	—	—	—
		7:0	RXEWM<7:0>							
0x00C0	ETHIEN(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	TXBUSEIE	RXBUSEIE	—	—	—	EWMARKIE	FWMARKIE
		7:0	RXDONEIE	PKTPENDIE	RXACTIE	—	TXDONEIE	TXABORTIE	RXBUFNAIE	RXOVFLWIE

图注： — = 未实现，读为 0。地址偏移值以十六进制显示。

注 1： 该寄存器具有关联的清零寄存器，位于 0x4 字节偏移处。这些清零寄存器的命名方式是在关联寄存器的名称末尾附加 CLR（例如， ETHCON1CLR）。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2： 该寄存器具有关联的置 1 寄存器，位于 0x8 字节偏移处。这些置 1 寄存器的命名方式是在关联寄存器的名称末尾附加 SET（例如， ETHCON1SET）。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3： 该寄存器具有关联的取反寄存器，位于 0xC 字节偏移处。这些取反寄存器的命名方式是在关联寄存器的名称末尾附加 INV （例如， ETHCON1INV）。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

表 35-1：　以太网控制器寄存器汇总 （续）

地址偏移	名称	位范围	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
0x00D0	ETHIRQ(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	TXBUSE	RXBUSE	—	—	—	EWMARK	FWMARK
		7:0	RXDONE	PKTPEND	RXACT	—	TXDONE	TXABORT	RXBUFNA	RXOVFLW
0x00E0	ETHSTAT	31:24	—	—	—	—	—	—	—	—
		23:16	BUFCNT<7:0>							
		15:8	—	—	—	—	—	—	—	—
		7:0	ETHBUSY	TXBUSY	RXBUSY	—	—	—	—	—
0x0100	ETHRXOVFLOW(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	RXOVFLWCNT<15:8>							
		7:0	RXOVFLWCNT<7:0>							
0x0110	ETHFRMTXOK(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	FRMTXOKCNT<15:8>							
		7:0	FRMTXOKCNT<7:0>							
0x0120	ETHSCOLFRM(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	SCOLFRMCNT<15:8>							
		7:0	SCOLFRMCNT<7:0>							
0x0130	ETHMCOLFRM(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	MCOLFRMCNT<15:8>							
		7:0	MCOLFRMCNT<7:0>							
0x0140	ETHFRMRXOK(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	FRMRXOKCNT<15:8>							
		7:0	FRMRXOKCNT<7:0>							
0x0150	ETHFCSERR(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	FCSERRCNT<15:8>							
		7:0	FCSERRCNT<7:0>							
0x0160	ETHALGNERR(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	ALGNERRCNT<15:8>							
		7:0	ALGNERRCNT<7:0>							
0x0200	EMACxCFG1(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	SOFTRESET	SIMRESET	—	—	RESETRMCS	RESETRFUN	RESETTMCS	RESETTFUN
		7:0	—	—	—	LOOPBACK	TXPAUSE	RXPAUSE	PASSALL	RXENABLE
0x0210	EMACxCFG2(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	EXCESSDFR	BPNOBKOFF	NOBKOFF	—	—	LONGPRE	PUREPRE
		7:0	AUTOPAD	VLANPAD	PADENABLE	CRCENABLE	DELAYCRC	HUGEFRM	LENGTHCK	FULLDPLX
0x0220	EMACxIPGT (1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	—	B2BIPKTGP<6:0>						
0x0230	EMACxIPGR(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	NB2BIPKTGP1<6:0>						
		7:0	—	NB2BIPKTGP2<6:0>						

图注： — = 未实现，读为 0。地址偏移值以十六进制显示。

注 1： 该寄存器具有关联的清零寄存器，位于 0x4 字节偏移处。这些清零寄存器的命名方式是在关联寄存器的名称末尾附加 CLR （例如， ETHCON1CLR）。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2： 该寄存器具有关联的置 1 寄存器，位于 0x8 字节偏移处。这些置 1 寄存器的命名方式是在关联寄存器的名称末尾附加 SET （例如， ETHCON1SET）。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3： 该寄存器具有关联的取反寄存器，位于 0xC 字节偏移处。这些取反寄存器的命名方式是在关联寄存器的名称末尾附加 INV （例如， ETHCON1INV）。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

表 35-1： 以太网控制器寄存器汇总 （续）

地址偏移	名称	位范围	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
0x0240	EMACxCLRT(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	CWINDOW<5:0>					
		7:0	—	—	—	—	RETX<3:0>			
0x0250	EMACxMAXF(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	MACMAXF<15:8>							
		7:0	MACMAXF<7:0>							
0x0260	EMACxSUPP(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	RESETRMII	—	—	SPEEDRMII
		7:0	—	—	—	—	—	—	—	—
0x0270	EMACxTEST(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	—	—	—	—	—	TESTBP	TESTPAUSE	SHRTQNTA
0x0280	EMACxMCFG(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	RESETMGMT	—	—	—	—	—	—	—
		7:0	—	—	CLKSEL<3:0>				NOPRE	SCANINC
0x0290	EMACxMCMD(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	—	—	—	—	—	—	SCAN	READ
0x02A0	EMACxMADR(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	PHYADDR<4:0>				
		7:0	—	—	—	REGADDR<4:0>				
0x02B0	EMACxMWTD(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	MWTD<15:8>							
		7:0	MWTD<7:0>							
0x02C0	EMACxMRDD(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	MRDD<15:8>							
		7:0	MRDD<7:0>							
0x02D0	EMACxMIND(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	—	—	—
		7:0	—	—	—	—	LINKFAIL	NOTVALID	SCAN	MIIMBUSY
0x0300	EMACxSA0(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	STNADDR6<7:0>							
		7:0	STNADDR5<7:0>							
0x0310	EMACxSA1(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	STNADDR4<7:0>							
		7:0	STNADDR3<7:0>							
0x0320	EMACxSA2(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	STNADDR2<7:0>							
		7:0	STNADDR1<7:0>							

图注： — = 未实现，读为 0。地址偏移值以十六进制显示。

注 1： 该寄存器具有关联的清零寄存器，位于 0x4 字节偏移处。这些清零寄存器的命名方式是在关联寄存器的名称末尾附加 CLR （例如， ETHCON1CLR）。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2： 该寄存器具有关联的置 1 寄存器，位于 0x8 字节偏移处。这些置 1 寄存器的命名方式是在关联寄存器的名称末尾附加 SET （例如， ETHCON1SET）。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3： 该寄存器具有关联的取反寄存器，位于 0xC 字节偏移处。这些取反寄存器的命名方式是在关联寄存器的名称末尾附加 INV （例如， ETHCON1INV）。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

寄存器 35-1： ETHCON1：以太网控制器控制 1 寄存器 (1,2,3)

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>							
bit 31							bit 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<7:0>							
bit 23							bit 16

R/W-0	R/W-0	R/W-0	r-x	r-x	r-x	R/W-0	R/W-0
ON	FRZ	SIDL	—	—	—	TXRTS	RXEN(4)
bit 15							bit 8

R/W-0	r-x	r-x	R/W-0	r-x	r-x	r-x	R/W-0
AUTOFC	—	—	MANFC	—	—	—	BUFCDEC
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **PTV<15:0>：**暂停定时器值位
用于流量控制的暂停定时器值。
只有 RXEN （ETHCON1<8>）未置 1 时，才能写入该寄存器。
这些位仅用于流量控制操作。

bit 15 **ON：**以太网使能位
1 = 使能以太网模块
0 = 禁止以太网模块

bit 14 **FRZ：**以太网冻结位
1 = 以太网模块在 Debug （调试）模式下冻结运行
0 = 以太网模块在 Debug （调试）模式下继续运行

bit 13 **SIDL：**以太网空闲模式停止位
1 = 以太网模块在 Idle （空闲）期间冻结传输
0 = 以太网模块在 Idle （空闲）期间继续传输

bit 12-10 **保留：**保持为 0；忽略读操作

注 1： 该寄存器具有关联的清零寄存器 （ETHCON1CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2： 该寄存器具有关联的置 1 寄存器 （ETHCON1SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3： 该寄存器具有关联的取反寄存器 （ETHCON1INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4： 建议不要执行这种操作：清零 RXEN 位，然后对与接收相关的任意字段 / 寄存器进行更改。必须先重新初始化以太网控制器 （ON 位清零），然后再应用接收更改。

寄存器 35-1： ETHCON1：以太网控制器控制 1 寄存器 (1,2,3) （续）

bit 9 **TXRTS**：发送请求发送位

1 = 激活发送逻辑并发送在发送 EDT 中定义的数据包
0 = 停止发送 （由软件清零时）或发送已完成 （由硬件清零时）

在该位写入 1 之后，每次发送逻辑完成发送在以太网描述符表 （Ethernet Descriptor Table， EDT）中请求的数据包时，该位会清零。如果 CPU 向该位写入 0，则发送逻辑会完成当前数据包的发送，然后停止之后所有其他数据包的发送。

该位仅影响发送操作。

bit 8 **RXEN**：接收使能位 (4)

1 = 使能接收逻辑，数据包将按照过滤器配置接收并存储到接收缓冲区中
0 = 禁止接收逻辑，不在接收缓冲区中接收任何数据包

该位仅影响接收操作。

bit 7 **AUTOFC**：自动流量控制位

1 = 使能自动流量控制
0 = 禁止自动流量控制

将该位置 1 会使能自动流量控制。如果置 1，则满水印和空水印分别用于自动使能和禁止流量控制。当已接收缓冲区数量 BUFCNT（ETHSTAT<16:23>）上升至满水印时，会自动使能流量控制。当 BUFCNT 下降至空水印时，会自动禁止流量控制。

该位仅用于流量控制操作，会同时影响发送和接收操作。

bit 6-5 **保留**：保持为 0 ；忽略读操作

bit 4 **MANFC**：手动流量控制位

1 = 使能手动流量控制
0 = 禁止手动流量控制

将该位置 1 会使能手动流量控制。如果置 1，流量控制逻辑会使用 PTV 寄存器中的暂停定时器值发送暂停帧。然后，它会每隔 128 * PTV<15:0>/2 个发送时钟周期重新发送暂停帧，直到该位清零为止。

请注意，对于 10 Mbps 操作，发送时钟以 2.5 MHz 运行。对于 100 Mbps 操作，发送时钟以 25 MHz 运行。

当该位清零时，流量控制逻辑会自动发送暂停定时器值为 0x0000 的暂停帧，以禁止流量控制。

该位仅用于流量控制操作，会同时影响发送和接收操作。

bit 3-1 **保留**：保持为 0 ；忽略读操作

bit 0 **BUFCDEC**：描述符缓冲区计数递减位

BUFCDEC 位是读为 0 的写 1 位。写入 1 时，描述符缓冲区计数器 BUFCNT 会递减 1。如果在写入该位的同时由于 BUFCNT 计数器接收逻辑递增，则 BUFCNT 值将保持不变。写入 0 没有任何作用。

该位仅用于接收操作。

注 1： 该寄存器具有关联的清零寄存器 （ETHCON1CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2： 该寄存器具有关联的置 1 寄存器 （ETHCON1SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3： 该寄存器具有关联的取反寄存器 （ETHCON1INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4： 建议不要执行这种操作：清零 RXEN 位，然后对与接收相关的任意字段 / 寄存器进行更改。必须先重新初始化以太网控制器 （ON 位清零），然后再应用接收更改。

寄存器 35-2： ETHCON2：以太网控制器控制 2 寄存器 (1,2,3,4,5)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	r-x	r-x	r-x	r-x	R/W-0	R/W-0	R/W-0
—	—	—	—	—	RXBUFSZ<6:4>		
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	r-x	r-x	r-x	r-x
RXBUFSZ<3:0>				—	—	—	—
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-11 **保留：**保持为 0 ； 忽略读操作

bit 10-4 **RXBUFSZ<6:0>：**所有接收描述符的接收数据缓冲区大小 （按 16 字节递增）位

0x00 = 保留
0x01 = 描述符的接收数据缓冲区大小为 16 字节
0x02 = 描述符的接收数据缓冲区大小为 32 字节
0x03 = 描述符的接收数据缓冲区大小为 48 字节
•
•
•
0x60 = 描述符的接收数据缓冲区大小为 1536 字节
•
•
•
0x7F = 描述符的接收数据缓冲区大小为 2032 字节

bit 3-0 **保留：**保持为 0 ； 忽略读操作

注 1： 该寄存器具有关联的清零寄存器（ETHCON2CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2： 该寄存器具有关联的置 1 寄存器（ETHCON2SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3： 该寄存器具有关联的取反寄存器（ETHCON2INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4： 该寄存器仅用于接收操作。

5： 只有在 RXEN （ETHCON1<8>） = 0 时，才能更改这些位。

寄存器 35-3： ETHTXST：以太网控制器发送数据包描述符起始地址寄存器 (1,2,3,4,5)

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>							
bit 31							bit 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<23:16>							
bit 23							bit 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<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	r-0	r-0
TXSTADDR<7:2>						—	—
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-2 **TXSTADDR<31:2>：**第一个发送描述符起始地址位
在任意发送、接收或 DMA 操作正在进行时，不能写入该寄存器。
该地址必须按 4 字节对齐 （即， bit 1-0 必须为 00）。

bit 1-0 **保留：**保持为 0；忽略读操作

注 1： 该寄存器具有关联的清零寄存器（ETHTXSTCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2： 该寄存器具有关联的置 1 寄存器（ETHTXSTSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3： 该寄存器具有关联的取反寄存器（ETHTXSTINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4： 该寄存器仅用于发送操作。

5： 硬件会使用上次成功发送数据包所用的最后一个描述符更新该寄存器。

寄存器 35-4： ETHRXST：以太网控制器接收数据包描述符起始地址寄存器 (1,2,3,4,5)

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>							
bit 31							bit 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<23:16>							
bit 23							bit 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<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	r-0	r-0
RXSTADDR<7:2>						—	—
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-2 **RXSTADDR<31:2>**：第一个接收描述符起始地址位
在任意发送、接收或 DMA 操作正在进行时，不能写入该寄存器。
该地址必须按 4 字节对齐 （即， bit 1-0 必须为 00）。

bit 1-0 **保留**：保持为 0；忽略读操作

注 1：该寄存器具有关联的清零寄存器（ETHRXSTCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（ETHRXSTSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（ETHRXSTINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：该寄存器仅用于接收操作。

5：硬件会使用上次成功发送数据包所用的最后一个描述符更新该寄存器。

寄存器 35-5： ETHHT0：以太网控制器哈希表 0 寄存器 (1,2,3,4,5)

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>							
bit 31							bit 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<23:16>							
bit 23							bit 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<15:8>							
bit 15							bit 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<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-0 **HT<31:0>：**哈希表字节 0-3 位

注 1：该寄存器具有关联的清零寄存器 （ETHHT0CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器 （ETHHT0SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器 （ETHHT0INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：该寄存器仅用于接收操作。

5：只有在 RXEN （ETHCON1<8>） = 0 或 HTEN （ETHRXFC<15>） = 0 时，才能更改这些位。

寄存器 35-6： ETHHT1：以太网控制器哈希表 1 寄存器 (1,2,3,4,5)

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>							
bit 31							bit 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<55:48>							
bit 23							bit 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<47:40>							
bit 15							bit 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<39:32>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-0 **HT<63:32>：**哈希表字节 4-7 位

注 1：该寄存器具有关联的清零寄存器 （ETHHT1CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器 （ETHHT1SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器 （ETHHT1INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：该寄存器仅用于接收操作。

5：只有在 RXEN （ETHCON1<8>） = 0 或 HTEN （ETHRXFC<15>） = 0 时，才能更改这些位。

寄存器 35-7： ETHPMM0：以太网控制器模式匹配屏蔽器 0 寄存器 (1,2,3,4,5)

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>							
bit 31							bit 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<23:16>							
bit 23							bit 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<15:8>							
bit 15							bit 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<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-24 **PMM<31:24>：**模式匹配屏蔽器 3 位

bit 23-16 **PMM<23:16>：**模式匹配屏蔽器 2 位

bit 15-8 **PMM<15:8>：**模式匹配屏蔽器 1 位

bit 7-0 **PMM<7:0>：**模式匹配屏蔽器 0 位

注 1： 该寄存器具有关联的清零寄存器（ETHPMM0CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2： 该寄存器具有关联的置 1 寄存器（ETHPMM0SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3： 该寄存器具有关联的取反寄存器（ETHPMM0INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4： 该寄存器仅用于接收操作。

5： 只有在 RXEN （ETHCON1<8>） = 0 或 PMMODE （ETHRXFC<11:8>） = 0 时，才能更改这些位。

寄存器 35-8： ETHPMM1：以太网控制器模式匹配屏蔽器 1 寄存器 (1,2,3,4,5)

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>							
bit 31							bit 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<55:48>							
bit 23							bit 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<47:40>							
bit 15							bit 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<39:32>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-24 **PMM<63:56>：**模式匹配屏蔽器 7 位

bit 23-16 **PMM<55:48>：**模式匹配屏蔽器 6 位

bit 15-8 **PMM<47:40>：**模式匹配屏蔽器 5 位

bit 7-0 **PMM<39:32>：**模式匹配屏蔽器 4 位

注 1：该寄存器具有关联的清零寄存器（ETHPMM1CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（ETHPMM1SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器 （ETHPMM1INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：该寄存器仅用于接收操作。

5：只有在 RXEN （ETHCON1<8>） = 0 或 PMMODE （ETHRXFC<11:8>） = 0 时，才能更改这些位。

寄存器 35-9： ETHPMCS：以太网控制器模式匹配校验和寄存器 (1,2,3,4,5)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 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
PMCS<15:8>							
bit 15							bit 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<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **保留：**保持为 0；忽略读操作

bit 15-8 **PMCS<15:8>：**模式匹配校验和 1 位

bit 7-0 **PMCS<7:0>：**模式匹配校验和 0 位

注 1：该寄存器具有关联的清零寄存器（ETHPMCSCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（ETHPMCSSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（ETHPMCSINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：该寄存器仅用于接收操作。

5：只有在 RXEN （ETHCON1<8>） = 0 或 PMMODE （ETHRXFC<11:8>） = 0 时，才能更改这些位。

寄存器 35-10： ETHPMO：以太网控制器模式匹配偏移寄存器 (1,2,3,4,5)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 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
PMO<15:8>							
bit 15							bit 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<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **保留：**保持为 0 ； 忽略读操作

bit 15-0 **PMO<15:0>：**模式匹配偏移 1 位

注 1：该寄存器具有关联的清零寄存器 （ETHPMOCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器 （ETHPMOSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器 （ETHPMOINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：该寄存器仅用于接收操作。

5：只有在 RXEN （ETHCON1<8>） = 0 或 PMMODE （ETHRXFC<11:8>） = 0 时，才能更改这些位。

寄存器 35-11： ETHRXFC：以太网控制器接收过滤器配置寄存器 (1,2,3,4,5)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 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
HTEN	MPEN	—	NOTPM	PMMODE<3:0>			
bit 15							bit 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
CRCERREN	CRCOKEN	RUNTERREN	RUNTEN	UCEN	NOTMEEN	MCEN	BCEN
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **保留：**保持为 0；忽略读操作

bit 15 **HTEN：**哈希表过滤使能位
1 = 使能哈希表过滤
0 = 禁止哈希表过滤

bit 14 **MPEN：**Magic Packet™ 使能位
1 = 使能 Magic Packet 过滤
0 = 禁止 Magic Packet 过滤

bit 13 **保留：**保持为 0；忽略读操作

bit 12 **NOTPM：**模式匹配反相位
1 = 只有模式匹配校验和不匹配时，才会发生成功的模式匹配
0 = 只有模式匹配校验和匹配时，才会发生成功的模式匹配
该位决定要发生成功的模式匹配，模式匹配校验和是否必须匹配。

注 1： 该寄存器具有关联的清零寄存器（ETHRXFCCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2： 该寄存器具有关联的置 1 寄存器（ETHRXFCSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3： 该寄存器具有关联的取反寄存器（ETHRXFCINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4： 该寄存器仅用于接收操作。

5： 只有在 RXEN （ETHCON1<8>） = 0 时，才能更改这些位。

6： 当两个条件中有一个为真，但不同时为真时， XOR 的结果为真。

7： 无论 HTEN 的值如何，该哈希表过滤器匹配总是工作。

8： 无论 MPE 的值如何，该 Magic Packet 过滤器匹配总是工作。

寄存器 35-11： ETHRXFC：以太网控制器接收过滤器配置寄存器 (1,2,3,4,5) （续）

bit 11-8 **PMMODE<3:0>**：模式匹配模式位

0000 = 禁止模式匹配；模式匹配总是不成功
0001 = 如果 (NOTPM = 1 XOR 模式匹配校验和匹配) 的结果为真，则模式匹配成功 (6)
0010 = 如果 (NOTPM = 1 XOR 模式匹配校验和匹配) AND (目标地址 = 站点地址) 的结果为真，则模式匹配成功 (6)
0011 = 如果 (NOTPM = 1 XOR 模式匹配校验和匹配) AND (目标地址 = 站点地址) 的结果为真，则模式匹配成功 (6)
0100 = 如果 (NOTPM = 1 XOR 模式匹配校验和匹配) AND (目标地址 = 单播地址) 的结果为真，则模式匹配成功 (6)
0101 = 如果 (NOTPM = 1 XOR 模式匹配校验和匹配) AND (目标地址 = 单播地址) 的结果为真，则模式匹配成功 (6)
0110 = 如果 (NOTPM = 1 XOR 模式匹配校验和匹配) AND (目标地址 = 广播地址) 的结果为真，则模式匹配成功 (6)
0111 = 如果 (NOTPM = 1 XOR 模式匹配校验和匹配) AND (目标地址 = 广播地址) 的结果为真，则模式匹配成功 (6)
1000 = 如果 (NOTPM = 1 XOR 模式匹配校验和匹配) AND (哈希表过滤器匹配) 的结果为真，则模式匹配成功 (6,7)
1001 = 如果 (NOTPM = 1 XOR 模式匹配校验和匹配) AND (数据包 = Magic Packet) 的结果为真，则模式匹配成功 (6,8)

bit 7 **CRCERREN**：CRC 错误收集使能位

1 = 只有接收数据包 CRC 无效时，才会接收数据包
0 = 禁止 CRC 错误收集过滤
该位使用户可以收集所有具有无效 CRC 的数据包。

bit 6 **CRCOKEN**：CRC 有效使能位

1 = 只有接收数据包 CRC 有效时，才会接收数据包
0 = 禁止 CRC 过滤
该位使用户可以拒绝所有具有无效 CRC 的数据包。

bit 5 **RUNTERREN**：过短错误收集使能位

1 = 只有接收数据包为过短数据包时，才会接收数据包
0 = 禁止过短错误收集过滤

该位使用户可以收集所有属于过短数据包的数据包。对于该过滤器，过短数据包定义为大小小于 64 字节的任意数据包 （当 CRCOKEN = 0 时），或者具有有效 CRC 但大小小于 64 字节的任意数据包 （当 CRCOKEN = 1 时）。

bit 4 **RUNTEN**：过短使能位

1 = 只有接收数据包不是过短数据包时，才会接收数据包
0 = 禁止过短过滤

该位使用户可以拒绝所有过短数据包。对于该过滤器，过短数据包定义为大小小于64字节的任意数据包。

注 1：该寄存器具有关联的清零寄存器（ETHRXFCCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（ETHRXFCSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器 （ETHRXFCINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：该寄存器仅用于接收操作。

5：只有在 RXEN （ETHCON1<8>） = 0 时，才能更改这些位。

6：当两个条件中有一个为真，但不同时为真时， XOR 的结果为真。

7：无论 HTEN 的值如何，该哈希表过滤器匹配总是工作。

8：无论 MPE 的值如何，该 Magic Packet 过滤器匹配总是工作。

寄存器 35-11： ETHRXFC：以太网控制器接收过滤器配置寄存器 (1,2,3,4,5) （续）

bit 3 **UCEN**：单播使能位
1 = 使能单播过滤
0 = 禁止单播过滤
该位使用户可以接收目标地址与站点地址匹配的所有单播数据包。

bit 2 **NOTMEEN**：非己单播使能位
1 = 使能非己单播过滤
0 = 禁止非己单播过滤
该位使用户可以接收目标地址与站点地址不匹配的所有单播数据包。

bit 1 **MCEN**：组播使能位
1 = 使能组播过滤
0 = 禁止组播过滤
该位使用户可以接收所有组播地址数据包。

bit 0 **BCEN**：广播使能位
1 = 使能广播过滤
0 = 禁止广播过滤
该位使用户可以接收所有广播地址数据包。

注 1：该寄存器具有关联的清零寄存器（ETHRXFCCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（ETHRXFCSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（ETHRXFCINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：该寄存器仅用于接收操作。

5：只有在 RXEN （ETHCON1<8>） = 0 时，才能更改这些位。

6：当两个条件中有一个为真，但不同时为真时， XOR 的结果为真。

7：无论 HTEN 的值如何，该哈希表过滤器匹配总是工作。

8：无论 MPE 的值如何，该 Magic Packet 过滤器匹配总是工作。

寄存器 35-12： ETHRXWM：以太网控制器接收水印寄存器 (1,2,3,4)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 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
RXFWM<7:0>							
bit 23							bit 16

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 15							bit 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
RXEWM<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-24 **保留：**保持为 0；忽略读操作

bit 23-16 **RXFWM<7:0>：**接收满水印位

软件控制的接收缓冲区满水印指针与接收 BUFCNT 进行比较，用以确定满水印条件，从而产生 FWMARK 中断和使能流量控制（使能自动流量控制时）。满水印指针应总是大于空水印指针。

bit 15-8 **保留：**保持为 0；忽略读操作

bit 7-0 **RXEWM<7:0>：**接收空水印位

软件控制的接收缓冲区空水印指针与接收 BUFCNT 进行比较，用以确定空水印条件，从而产生 EWMARK 中断和禁止流量控制（使能自动流量控制时）。空水印指针应总是小于满水印指针。

注 1：该寄存器具有关联的清零寄存器（ETHRXWMCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（ETHRXWMSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（ETHRXWMINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：该寄存器仅用于接收操作。

寄存器 35-13： ETHIEN：以太网控制器中断允许寄存器 (1,2,3,4,5)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	R/W-0	R/W-0	r-x	r-x	r-x	R/W-0	R/W-0
—	TXBUSEIE(4)	RXBUSEIE(5)	—	—	—	EWMARKIE(5)	FWMARKIE(5)
bit 15							bit 8

R/W-0	R/W-0	R/W-0	r-x	R/W-0	R/W-0	R/W-0	R/W-0
RXDONEIE(5)	PKTPENDIE(5)	RXACTIE(5)	—	TXDONEIE(4)	TXABORTIE(4)	RXBUFNAIE(5)	RXOVFLWIE(5)
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-15 **保留：**保持为 0；忽略读操作

bit 14 **TXBUSEIE：**发送 BVCI 总线错误中断允许位 (4)
1 = 允许 TXBUS 错误中断
0 = 禁止 TXBUS 错误中断

bit 13 **RXBUSEIE：**接收 BVCI 总线错误中断允许位 (5)
1 = 允许 RXBUS 错误中断
0 = 禁止 RXBUS 错误中断

bit 12-10 **保留：**保持为 0；忽略读操作

bit 9 **EWMARKIE：**空水印中断允许位 (5)
1 = 允许 EWMARK 中断
0 = 禁止 EWMARK 中断

bit 8 **FWMARKIE：**满水印中断允许位 (5)
1 = 允许 FWMARK 中断
0 = 禁止 FWMARK 中断

bit 7 **RXDONEIE：**接收器完成中断允许位 (5)
1 = 允许 RXDONE 中断
0 = 禁止 RXDONE 中断

注 1：该寄存器具有关联的清零寄存器（ETHIENCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（ETHIENSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（ETHIENINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：这些位仅用于发送操作。

5：这些位仅用于接收操作。

寄存器 35-13： ETHIEN：以太网控制器中断允许寄存器 (1,2,3,4,5) （续）

bit 6 **PKTPENDIE**：数据包待处理中断允许位 (5)
1 = 允许 PKTPEND 中断
0 = 禁止 PKTPEND 中断

bit 5 **RXACTIE**：接收活动中断允许位 (2)
1 = 允许 RXACT 中断
0 = 禁止 RXACT 中断

bit 4 **保留**：保持为 0；忽略读操作

bit 3 **TXDONEIE**：发送器完成中断允许位 (4)
1 = 允许 TXDONE 中断
0 = 禁止 TXDONE 中断

bit 2 **TXABORTIE**：发送器中止中断允许位 (4)
1 = 允许 TXABORT 中断
0 = 禁止 TXABORT 中断

bit 1 **RXBUFNAIE**：接收缓冲区不可用中断允许位 (5)
1 = 允许 RXBUFNA 中断
0 = 禁止 RXBUFNA 中断

bit 0 **RXOVFLWIE**：接收 FIFO 溢出中断允许位 (5)
1 = 允许 RXOVFLW 中断
0 = 禁止 RXOVFLW 中断

注 1：该寄存器具有关联的清零寄存器（ETHIENCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（ETHIENSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（ETHIENINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：这些位仅用于发送操作。

5：这些位仅用于接收操作。

寄存器 35-14： ETHIRQ：以太网控制器中断请求寄存器 (1,2,3,4,7)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	R/W-0	R/W-0	r-x	r-x	r-x	R/W-0	R/W-0
—	TXBUSE	RXBUSE	—	—	—	EWMARK	FWMARK
bit 15							bit 8

R/W-0	R/W-0	R/W-0	r-x	R/W-0	R/W-0	R/W-0	R/W-0
RXDONE	PKTPEND	RXACT	—	TXDONE	TXABORT	RXBUFNA	RXOVFLW
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-15 **保留：**保持为 0；忽略读操作

bit 14 **TXBUSE：**发送 BVCI 总线错误中断位 (5)

1 = 发生了 BVCI 总线错误
0 = 未发生 BVCI 总线错误

在存储器访问期间，当发送 DMA 遇到 BVCI 总线错误时，该位会置 1。它通过复位或用 CPU 向清零寄存器中写入 1 来清零。

bit 13 **RXBUSE：**接收 BVCI 总线错误中断位 (6)

1 = 发生了 BVCI 总线错误
0 = 未发生 BVCI 总线错误

在存储器访问期间，当接收 DMA 遇到 BVCI 总线错误时，该位会置 1。它通过复位或用 CPU 向清零寄存器中写入 1 来清零。

bit 12-10 **保留：**保持为 0；忽略读操作

注 1：该寄存器具有关联的清零寄存器 （ETHIRQCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器 （ETHIRQSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器 （ETHIRQINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：要置 1 或清零该寄存器中的位，应使用置 1、清零或取反寄存器。

5：这些寄存器位仅用于发送操作。

6：这些寄存器位仅用于接收操作。

7：只有在进行调试 / 测试时，才应对该寄存器中的位进行设置。

寄存器 35-14： ETHIRQ：以太网控制器中断请求寄存器 (1,2,3,4,7) （续）

bit 9 **EWMARK**：空水印中断位 (6)

1 = 已达到空水印指针
0 = 没有待处理中断

当接收描述符缓冲区计数小于等于 RXEWM （ETHRXWM<0:7>）值时，该位会置 1。在硬件递增 BUFCNT （ETHSTAT<16:23>）时，它会清零。写入 0 或 1 没有任何作用。

bit 8 **FWMARK**：满水印中断位 (6)

1 = 已达到满水印指针
0 = 没有待处理中断

当接收描述符缓冲区计数大于等于 RXFWM （ETHRXWM<16:23>）字段值时，该位会置 1。写入 BUFCDEC （ETHCON1<0>）位以递减 BUFCNT 计数器时，它会清零。写入 0 或 1 没有任何作用。

bit 7 **RXDONE**：接收完成中断位 (6)

1 = 已成功接收到接收数据包
0 = 没有待处理中断

每次成功接收到一个接收数据包时，该位会置 1。它通过复位或用 CPU 向清零寄存器中写入 1 来清零。

bit 6 **PKTPEND**：数据包待处理中断位 (6)

1 = 存储器中有待处理的接收数据包
0 = 存储器中没有待处理的接收数据包

当 BUFCNT 计数器值不为 0 时，该位会置 1。复位或写入 BUFCDEC 位来递减 BUFCNT 计数器时，它会清零。写入 0 或 1 没有任何作用。

bit 5 **RXACT**：接收活动中断位 (6)

1 = 已成功接收到接收数据包数据
0 = 没有待处理中断

每次接收数据包数据存储到 RXBM FIFO 中时，该位会置 1。它通过复位或用 CPU 向清零寄存器中写入 1 来清零。

bit 4 **保留**：保持为 0；忽略读操作

bit 3 **TXDONE**：发送完成中断位 (5)

1 = 已成功发送了发送数据包
0 = 没有待处理中断

在当前发送的发送数据包完成发送，并且发送状态向量装入用于数据包的第一个描述符中时，该位会置 1。它通过复位或用 CPU 向清零寄存器中写入 1 来清零。

注 1：该寄存器具有关联的清零寄存器 （ETHIRQCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器 （ETHIRQSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器 （ETHIRQINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：要置 1 或清零该寄存器中的位，应使用置 1、清零或取反寄存器。

5：这些寄存器位仅用于发送操作。

6：这些寄存器位仅用于接收操作。

7：只有在进行调试 / 测试时，才应对该寄存器中的位进行设置。

寄存器 35-14： ETHIRQ：以太网控制器中断请求寄存器 (1,2,3,4,7) （续）

bit 2 **TXABORT**：发送中止条件中断位 (5)

1 = 上一个发送数据包中产生了发送中止条件
0 = 没有待处理中断

当 MAC 由于以下原因之一而中止发送数据包的发送时，该位会置 1：

- 超大发送数据包中止
- 数据不足中止
- 延时过长中止
- 迟冲突中止
- 冲突过量中止

该位通过复位或用 CPU 向清零寄存器中写入 1 来清零。

bit 1 **RXBUFNA**：接收缓冲区不可用中断位 (6)

1 = 产生了接收缓冲区描述符不可用条件
0 = 没有待处理中断

产生发送缓冲区描述符溢出条件时，该位会置 1。它通过复位或用 CPU 向清零寄存器中写入 1 来清零。

bit 0 **RXOVFLW**：接收 FIFO 溢出错误位 (6)

1 = 产生了接收 FIFO 溢出错误条件
0 = 没有待处理中断

在产生接收 FIFO 溢出条件时， RXBM 逻辑会将 RXOVFLW 置 1。它通过复位或用 CPU 向清零寄存器中写入 1 来清零。

注 1： 该寄存器具有关联的清零寄存器 （ETHIRQCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2： 该寄存器具有关联的置 1 寄存器 （ETHIRQSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3： 该寄存器具有关联的取反寄存器 （ETHIRQINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4： 要置 1 或清零该寄存器中的位，应使用置 1、清零或取反寄存器。

5： 这些寄存器位仅用于发送操作。

6： 这些寄存器位仅用于接收操作。

7： 只有在进行调试 / 测试时，才应对该寄存器中的位进行设置。

寄存器 35-15： ETHSTAT：以太网控制器状态寄存器

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 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
BUFCNT<7:0>(1)							
bit 23							bit 16

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	r-x	r-x	r-x	r-x	r-x
ETHBUSY(5)	TXBUSY(2,6)	RXBUSY(3,6)	—	—	—	—	—
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-24 **保留：**保持为 0；忽略读操作

bit 23-16 **BUFCNT<7:0>：**数据包缓冲区计数位 (1)

存储器中已接收的数据包缓冲区的数量。成功接收到一个数据包之后，硬件会根据该数据包使用的描述符数量递增该寄存器。从缓冲区中读出一个数据包之后，软件需要递减计数器（对于每个使用的描述符，均写入 BUFCDEC（ETHCON1<0>）位）。在寄存器值为 0xFF 时，如果硬件递增寄存器，寄存器不会计满返回（从 0xFF 变为 0x00）。相反地，在寄存器值为 0x0000 时，如果软件递减寄存器，寄存器不会触底回弹（从 0x00 变为 0xFF）。如果在硬件递增计数器的同时，软件递减计数器，则计数器值将保持不变。

当该寄存器值达到 0xFF 时，接收逻辑会暂停（仅当使能自动流量控制时），等待软件通过写入 BUFCDEC 位来将寄存器递减为小于 0xFF。

如果禁止了自动流量控制，则 RXDMA 会继续进行处理， BUFCNT 将一直为饱和值 0xFF。

当该寄存器非零时， PKTPEND 状态位将置 1，并且可能会产生中断；是否产生中断取决于 ETHIEN <PKTPENDIE> 寄存器位的值。

当写入 ETHRXST 寄存器时， BUFCNT 计数器会自动清除为 0x00。

注： 当 ON 设置为 0 时， BUFCNT 不会被清零。这使软件可以继续使用并递减该计数。

bit 15-8 **保留：**保持为 0；忽略读操作

注 1：这些位仅用于接收操作。

2：该位仅受发送操作影响。

3：该位仅受接收操作影响。

4：该位会受发送和接收操作影响。

5：当 ON （ETHCON1<15>） = 1 时，该位会*置 1*。

6：当 ON （ETHCON1<15>） = 0 时，该位会*清零*。

寄存器 35-15： ETHSTAT：以太网控制器状态寄存器 （续）

bit 7	**ETHBUSY**：以太网模块忙状态位 **(5)** 1 = 以太网逻辑已开启 （ON （ETHCON1<15>） = 1）或正在完成某个事务 0 = 以太网逻辑空闲 该位指示模块已开启，或者在模块被关闭之后仍在完成某个事务。
bit 6	**TXBUSY**：发送忙状态位 **(2,6)** 1 = 发送逻辑正在发送数据 0 = 发送逻辑空闲 该位指示当前正在发送数据包。该状态位的变化不一定会反映为 TXDONE 中断，因为发送数据包可能会被 MAC 中止或拒绝。
bit 5	**RXBUSY**：接收忙状态位 **(3,6)** 1 = 接收逻辑正在接收数据 0 = 接收逻辑空闲 该位指示当前正在接收数据包。该状态位的变化不一定会反映为 RXDONE 中断，因为接收数据包可能会被接收过滤器中止或拒绝。
bit 4-0	**保留**：保持为 0 ；忽略读操作

注 1：这些位仅用于接收操作。

2：该位仅受发送操作影响。

3：该位仅受接收操作影响。

4：该位会受发送和接收操作影响。

5：当 ON （ETHCON1<15>） = 1 时，该位会*置 1*。

6：当 ON （ETHCON1<15>） = 0 时，该位会*清零*。

寄存器 35-16： ETHRXOVFLOW：以太网控制器接收溢出统计寄存器 (1,2,3,4,5,6)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 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
RXOVFLWCNT<15:8>							
bit 15							bit 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<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **保留：**保持为 0；忽略读操作

bit 15-0 **RXOVFLWCNT<15:0>：**已丢弃接收帧计数位
已由接收过滤器接收、但随后由于内部接收错误（RXFIFO 溢出）而被丢弃的帧的递增计数器。该事件也会将 RXOVFLW （ETHIRQ<0>）中断标志置 1。

注 1：该寄存器具有关联的清零寄存器 （ETHRXOVFLOWCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器 （ETHRXOVFLOWSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器 （ETHRXOVFLOWINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：该寄存器仅用于接收操作。

5：除非字节 0/1 的字节使能设置为 0，否则在执行读操作之后，硬件会自动清零该寄存器。

6：只有在进行调试 / 测试时，才应对该寄存器中的位进行设置。

寄存器 35-17： ETHFRMTXOK：以太网控制器正常发送帧统计寄存器 (1,2,3,4,5,6)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 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
FRMTXOKCNT<15:8>							
bit 15							bit 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<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **保留：**保持为 0；忽略读操作

bit 15-0 **FRMTXOKCNT<15:0>：**正常发送帧计数位
成功发送帧的递增计数器。

注 1：该寄存器具有关联的清零寄存器（ETHFRMTXOKCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（ETHFRMTXOKSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（ETHFRMTXOKINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：该寄存器仅用于发送操作。

5：除非字节 0/1 的字节使能设置为 0，否则在执行读操作之后，硬件会自动清零该寄存器。

6：只有在进行调试 / 测试时，才应对该寄存器中的位进行设置。

寄存器 35-18： ETHSCOLFRM：以太网控制器单次冲突帧统计寄存器 (1,2,3,4,5,6)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 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
SCOLFRMCNT<15:8>							
bit 15							bit 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<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **保留：**保持为 0 ；忽略读操作

bit 15-0 **SCOLFRMCNT<15:0>：**单次冲突帧计数位
用于第二次尝试时成功发送的帧的递增计数。

注 1：该寄存器具有关联的清零寄存器 （ETHSCOLFRMCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器 （ETHSCOLFRMSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器 （ETHSCOLFRMINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：该寄存器仅用于发送操作。

5：除非字节 0/1 的字节使能设置为 0，否则在执行读操作之后，硬件会自动清零该寄存器。

6：只有在进行调试 / 测试时，才应对该寄存器中的位进行设置。

寄存器 35-19： ETHMCOLFRM：以太网控制器多次冲突帧统计寄存器 (1,2,3,4,5,6)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 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
MCOLFRMCNT<15:8>							
bit 15							bit 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<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **保留：**保持为 0；忽略读操作

bit 15-0 **MCOLFRMCNT<15:0>：**多次冲突帧计数位
用于发生多次冲突之后成功发送的帧的递增计数。

注 1：该寄存器具有关联的清零寄存器（ETHMCOLFRMCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（ETHMCOLFRMSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（ETHMCOLFRMINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：该寄存器仅用于发送操作。

5：除非字节 0/1 的字节使能设置为 0，否则在执行读操作之后，硬件会自动清零该寄存器。

6：只有在进行调试 / 测试时，才应对该寄存器中的位进行设置。

寄存器 35-20： ETHFRMRXOK：以太网控制器正常接收帧统计寄存器 (1,2,3,4,5,6)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 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
FRMRXOKCNT<15:8>							
bit 15							bit 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<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **保留：**保持为 0；忽略读操作

bit 15-0 **FRMRXOKCNT<15:0>：**正常接收帧计数位
接收过滤器成功接收的帧的递增计数。如果存在帧校验序列（Frame Check Sequence，FCS）或对齐错误，则不会递增该计数。

注 1：该寄存器具有关联的清零寄存器（ETHFRMRXOKCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（ETHFRMRXOKSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（ETHFRMRXOKINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：该寄存器仅用于接收操作。

5：除非字节 0/1 的字节使能设置为 0，否则在执行读操作之后，硬件会自动清零该寄存器。

6：只有在进行调试 / 测试时，才应对该寄存器中的位进行设置。

寄存器 35-21： ETHFCSERR：以太网控制器帧校验序列错误统计寄存器 (1,2,3,4,5,6)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 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
FCSERRCNT<15:8>							
bit 15							bit 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<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **保留：**保持为 0；忽略读操作

bit 15-0 **FCSERRCNT<15:0>：**FCS 错误计数位
这种帧的递增计数：接收时带有 FCS 错误，并且帧长 （以位为单位）是 8 位的整数倍。

注 1： 该寄存器具有关联的清零寄存器 （ETHFCSERRCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2： 该寄存器具有关联的置 1 寄存器 （ETHFCSERRSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3： 该寄存器具有关联的取反寄存器 （ETHFCSERRINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4： 该寄存器仅用于接收操作。

5： 除非字节 0/1 的字节使能设置为 0，否则在执行读操作之后，硬件会自动清零该寄存器。

6： 只有在进行调试 / 测试时，才应对该寄存器中的位进行设置。

寄存器 35-22： ETHALGNERR：以太网控制器对齐错误统计寄存器 (1,2,3,4,5,6)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 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
ALGNERRCNT<15:8>							
bit 15							bit 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<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **保留：**保持为 0；忽略读操作

bit 15-0 **ALGNERRCNT<15:0>：**对齐错误计数位

带有对齐错误的帧的递增计数。请注意，对齐错误表示帧带有 FCS 错误，并且以位为单位的帧长不是 8 位的整数倍 （也称为多余数据）。

注 1：该寄存器具有关联的清零寄存器 （ETHALGNERRCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器 （ETHALGNERRSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器 （ETHALGNERRINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：该寄存器仅用于接收操作。

5：除非字节 0/1 的字节使能设置为 0，否则在执行读操作之后，硬件会自动清零该寄存器。

6：只有在进行调试 / 测试时，才应对该寄存器中的位进行设置。

寄存器 35-23： EMACxCFG1：以太网控制器 MAC 配置 1 寄存器 (1,2,3,4)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-1	R/W-0	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0
SOFTRESET	SIMRESET	—	—	RESETRMCS	RESETRFUN	RESETTMCS	RESETTFUN
bit 15							bit 8

r-x	r-x	r-x	R/W-0	R/W-1	R/W-1	R/W-0	R/W-1
—	—	—	LOOPBACK	TXPAUSE	RXPAUSE	PASSALL	RXENABLE
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **保留：**保持为 0；忽略读操作

bit 15 **SOFTRESET：**软复位位
该位置 1 时，会将 MACMII 置为复位状态。其默认值为 1。

bit 14 **SIMRESET：**模拟复位位
该位置 1 时，会导致发送功能内的随机数发生器复位。

bit 13-12 **保留：**保持为 0；忽略读操作

bit 11 **RESETRMCS：**复位 MCS/RX 位
该位置 1 时，会将 MAC 控制子层 / 接收域逻辑置为复位状态。

bit 10 **RESETRFUN：**复位接收功能位
该位置 1 时，会将 MAC 接收功能逻辑置为复位状态。

bit 9 **RESETTMCS：**复位 MCS/TX 位
该位置 1 时，会将 MAC 控制子层 / 发送域逻辑置为复位状态。

bit 8 **RESETTFUN：**复位发送功能位
该位置 1 时，会将 MAC 发送功能逻辑置为复位状态。

bit 7-5 **保留：**保持为 0；忽略读操作

注 1：该寄存器具有关联的清零寄存器（EMACxCFG1CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（EMACxCFG1SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（EMACxCFG1INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：这些寄存器（包括置 1、清零和取反寄存器）允许 16 位和 32 位访问。它们不允许 8 位访问，硬件会忽略 8 位访问。

寄存器 35-23： EMACxCFG1：以太网控制器 MAC 配置 1 寄存器 (1,2,3,4) （续）

bit 4 **LOOPBACK**：MAC 环回模式位
1 = MAC 发送接口环回到 MAC 接收接口
0 = MAC 正常工作

bit 3 **TXPAUSE**：MAC 发送流量控制位
1 = 允许发送暂停流量控制帧
0 = 阻止暂停流量控制帧

bit 2 **RXPAUSE**：MAC 接收流量控制位
1 = MAC 根据接收到的暂停流量控制帧执行操作
0 = 忽略接收到的暂停流量控制帧

bit 1 **PASSALL**：MAC 通过所有接收帧位
1 = MAC 将接收所有帧，无论帧的类型如何 （普通帧与控制帧）
0 = 忽略接收到的控制帧

bit 0 **RXENABLE**：MAC 接收使能位
1 = 使能 MAC 接收帧
0 = 禁止 MAC 接收帧

注 1：该寄存器具有关联的清零寄存器 （EMACxCFG1CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器 （EMACxCFG1SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器 （EMACxCFG1INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：这些寄存器（包括置 1、清零和取反寄存器）允许 16 位和 32 位访问。它们不允许 8 位访问，硬件会忽略 8 位访问。

寄存器 35-24： EMACxCFG2：以太网控制器 MAC 配置 2 寄存器 (1,2,3,4)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	R/W-1	R/W-0	R/W-0	r-x	r-x	R/W-0	R/W-0
—	EXCESSDFR	BPNOBKOFF	NOBKOFF	—	—	LONGPRE	PUREPRE
bit 15							bit 8

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
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-15 **保留：**保持为 0；忽略读操作

bit 14 **EXCESSDER：**延时过长位
1 = MAC 将根据标准无期限地延迟至载波消失时
0 = MAC 将在达到延时过长限制时中止

bit 13 **BPNOBKOFF：**背压 / 无后退位
1 = MAC 在背压过程中偶然导致冲突之后将立即重发，无需后退，从而降低进一步冲突的机会，确保发送数据包能够发送出去
0 = MAC 不会去除后退

bit 12 **NOBKOFF：**无后退位
1 = 在发生冲突之后， MAC 将立即重发，而不是使用标准中规定的二进制指数后退算法
0 = 在发生冲突之后， MAC 将使用二进制指数后退算法

bit 11-10 **保留：**保持为 0；忽略读操作

注 1： 该寄存器具有关联的清零寄存器 （EMACxCFG2CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2： 该寄存器具有关联的置 1 寄存器 （EMACxCFG2SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3： 该寄存器具有关联的取反寄存器 （EMACxCFG2INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4： 这些寄存器 （包括置 1、清零和取反寄存器）允许 16 位和 32 位访问。它们不允许 8 位访问，硬件会忽略 8 位访问。

5： 表 35-2 给出了基于该寄存器配置的填充功能的说明。

6： 如果 PADENABLE 清零，该位会被忽略。

7： 该位与 AUTOPAD 和 VLANPAD 配合使用。

寄存器 35-24： EMACxCFG2：以太网控制器 MAC 配置 2 寄存器 (1,2,3,4) （续）

bit 9 **LONGPRE**：长前导强制位
1 = MAC 仅允许所包含的前导字段长度小于 12 字节的接收数据包
0 = MAC 按照标准允许任意长度的前导

bit 8 **PUREPRE**：纯前导强制位
1 = MAC 将对前导的内容进行验证，确保它包含 0x55 且无错误。前导中带有错误的数据包会被丢弃
0 = MAC 不执行任何前导检查

bit 7 **AUTOPAD**：自动检测填充使能位 (5,6)
1 = MAC 通过将源地址之后的两个八位字节与 0x8100（VLAN 协议 ID）进行比较，自动检测帧的类型（标记或未标记），并相应地进行填充
0 = MAC 不执行自动检测

bit 6 **VLANPAD**：VLAN 填充使能位 (5,6)
1 = MAC 将所有短帧填充至 64 字节，并附加有效 CRC
0 = MAC 不执行短帧填充

bit 5 **PADENABLE**：填充 /CRC 使能位 (5,7)
1 = MAC 将填充所有短帧
0 = 送到 MAC 的帧具有有效长度

bit 4 **CRCENABLE**：CRC 使能位
1 = 无论是否需要填充， MAC 都向每个帧附加 CRC。如果 PADENABLE 置 1，则它必须置 1
0 = 送到 MAC 的帧具有有效 CRC

bit 3 **DELAYCRC**：延迟 CRC 位
该位决定在 IEEE 802.3 帧前面存在的专用报头信息 （如果有）的字节数。
1 = 报头为 4 字节 （会被 CRC 功能忽略）
0 = 无专用报头

bit 2 **HUGEFRM**：超大帧使能位
1 = 允许发送和接收任意长度的帧
0 = 不允许接收或发送超大帧

bit 1 **LENGTHCK**：帧长检查位
1 = 发送和接收帧长都与长度 / 类型字段进行比较。如果长度 / 类型字段代表长度，则执行检查。不匹配情况在发送 / 接收统计向量中报告
0 = 不执行长度 / 类型字段检查

bit 0 **FULLDPLX**：全双工操作位
1 = MAC 工作于全双工模式
0 = MAC 工作于半双工模式

注 1：该寄存器具有关联的清零寄存器 （EMACxCFG2CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器 （EMACxCFG2SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器 （EMACxCFG2INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：这些寄存器 （包括置 1、清零和取反寄存器）允许 16 位和 32 位访问。它们不允许 8 位访问，硬件会忽略 8 位访问。

5：表 35-2 给出了基于该寄存器配置的填充功能的说明。

6：如果 PADENABLE 清零，该位会被忽略。

7：该位与 AUTOPAD 和 VLANPAD 配合使用。

表 35-2： 填充操作

类型	AUTOPAD	VLANPAD	PADENABLE	操作
任意	X	X	0	无填充，检查 CRC
任意	0	0	1	填充至 60 字节，附加 CRC
任意	X	1	1	填充至 64 字节，附加 CRC
任意	1	0	1	如果未标记：填充至 60 字节，附加 CRC 如果带有 VLAN 标记：填充至 64 字节，附加 CRC

寄存器 35-25： EMACxIPGT：以太网控制器 MAC 背靠背包间间隔寄存器 (1,2,3,4)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 15							bit 8

r-x	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-1	R/W-0
—	B2BIPKTGP<6:0>						
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-7 **保留：**保持为 0；忽略读操作

bit 6-0 **B2BIPKTGP<6:0>：**背靠背包间间隔位

这是一个可编程字段，代表从任意发送数据包结束到下一个数据包开始之间的最小可能周期所对应的半字节时间偏移。在全双工模式下，寄存器值应为所需周期（以半字节时间为单位）减 3。在半双工模式下，寄存器值应为所需周期（以半字节时间为单位）减 6。在全双工模式下，建议设置为 0x15（21d），它代表 0.96 µs （100 Mbps 时）或 9.6 µs （10 Mbps 时）的最小 IPG。在半双工模式下，建议设置为 0x12 （18d），它也代表 0.96 µs （100 Mbps 时）或 9.6 µs （10 Mbps 时）的最小 IPG。

注 1： 该寄存器具有关联的清零寄存器 （EMACxIPGTCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2： 该寄存器具有关联的置 1 寄存器 （EMACxIPGTSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3： 该寄存器具有关联的取反寄存器 （EMACxIPGTINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4： 这些寄存器 （包括置 1、清零和取反寄存器）允许 16 位和 32 位访问。它们不允许 8 位访问，硬件会忽略 8 位访问。

寄存器 35-26： EMACxIPGR：以太网控制器 MAC 非背靠背包间间隔寄存器 (1,2,3,4)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-0	R/W-0
—	NB2BIPKTGP1<6:0>						
bit 15							bit 8

r-x	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-1	R/W-0
—	NB2BIPKTGP2<6:0>						
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-15 **保留：**保持为 0；忽略读操作

bit 14-8 **NB2BIPKTGP1<6:0>：**非背靠背包间间隔第 1 部分位

这是一个可编程字段，代表在 IEEE 802.3/4.2.3.2.1 “Carrier Deference”中提及的可选 carrierSense 窗口。如果在 IPGR1 时间段中检测到载波，则 MAC 延迟到载波消失时。但如果在 IPGR1 之后出现载波，则 MAC 继续计时 IPGR2 并发送，有意引起冲突，从而确保对介质的公平访问。值的范围为 0x0 至 IPGR2。建议值为 0xC （12d）。

bit 7 **保留：**保持为 0；忽略读操作

bit 6-0 **NB2BIPKTGP2<6:0>：**非背靠背包间间隔第 2 部分位

这是一个可编程字段，代表非背靠背包间间隔。它的建议值为 0x12（18d），该值代表 0.96 μs（100 Mbps 时）或 9.6 μs （10 Mbps 时）的最小 IPG。

注 1：该寄存器具有关联的清零寄存器 （EMACxIPGRCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器 （EMACxIPGRSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器 （EMACxIPGRINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：这些寄存器 （包括置 1、清零和取反寄存器）允许 16 位和 32 位访问。它们不允许 8 位访问，硬件会忽略 8 位访问。

寄存器 35-27： EMACxCLRT：以太网控制器 MAC 冲突窗口 / 重试限制寄存器 (1,2,3,4)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	r-x	R/W-1	R/W-1	R/W-0	R/W-1	R/W-1	R/W-1
—	—	CWINDOW<5:0>					
bit 15							bit 8

r-x	r-x	r-x	r-x	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	—	RETX<3:0>			
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-14 **保留：**保持为 0；忽略读操作

bit 13-8 **CWINDOW<5:0>：**冲突窗口位

这是一个可编程字段，代表在适当配置的网络中发生冲突的时隙或冲突窗口。由于冲突窗口从发送开始时开始计时，所以它包含前导和 SFD。它的默认值 0x37（55d）对应于在窗口结束时的帧字节的计数。

bit 7-4 **保留：**保持为 0；忽略读操作

bit 3-0 **RETX<3:0>：**最大重发次数位

这是一个可编程字段，它指定在发生冲突之后的重发尝试次数，超过该次数之后会由于冲突过量而中止数据包。标准规定最大尝试次数 （attemptLimit）为 0xF （15d）。其默认值为 “0xF”。

注 1：该寄存器具有关联的清零寄存器 （EMACxCLRTCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器 （EMACxCLRTSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器 （EMACxCLRTINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：这些寄存器 （包括置 1、清零和取反寄存器）允许 16 位和 32 位访问。它们不允许 8 位访问，硬件会忽略 8 位访问。

寄存器 35-28： EMACxMAXF：以太网控制器 MAC 最大帧长寄存器 (1,2,3,4)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

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> (5)							
bit 15							bit 8

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> (5)							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **保留：**保持为 0；忽略读操作

bit 15-0 **MACMAXF<15:0>：**最大帧长位 (5)

该字段复位为 0x05EE，该值代表最大接收帧为 1518 个八位字节。未标记最大大小以太网帧为 1518 个八位字节。标记帧会增加 4 个八位字节，总共 1522 个八位字节。如果需要较短 / 较长的最大长度限制，则可以编程这个 16 位字段。

注 1：该寄存器具有关联的清零寄存器（EMACxMAXFCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（EMACxMAXFSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（EMACxMAXFINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：这些寄存器（包括置 1、清零和取反寄存器）允许 16 位和 32 位访问。它们不允许 8 位访问，硬件会忽略 8 位访问。

5：如果允许有专用报头，则应相应地调整该字段。例如，如果要在帧前附加 4 字节报头，则可以将 MACMAXF 设置为 1527 个八位字节。这将允许最大 VLAN 标记帧加上 4 字节报头。

寄存器 35-29： EMACxSUPP：以太网控制器 MAC PHY 支持寄存器 (1,2,3,4)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-0	r-x	r-x	r-1	R/W-0	r-x	r-x	R/W-0
—	—	—	—	RESETRMII(5)	—	—	SPEEDRMII(5)
bit 15							bit 8

r-0	r-0	r-0	r-0	r-0	r-x	r-0	r-0
—	—	—	—	—	—	—	—
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-12 **保留：**保持为 0；忽略读操作

bit 11 **RESETRMII：**复位 RMII 逻辑位 (5)
1 = 复位 MAC RMII 模块
0 = 正常工作

bit 10-9 **保留：**保持为 0；忽略读操作

bit 8 **SPEEDRMII：**RMII 速度位 (5)
该位配置精简型 MII 逻辑的当前工作速度。
1 = RMII 以 100 Mbps 运行
0 = RMII 以 10 Mbps 运行

bit 7-0 **保留：**保持为 0；忽略读操作

注 1：该寄存器具有关联的清零寄存器（EMACxSUPPCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（EMACxSUPPSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（EMACxSUPPINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：这些寄存器（包括置 1、清零和取反寄存器）允许 16 位和 32 位访问。它们不允许 8 位访问，硬件会忽略 8 位访问。

5：这些位仅用于 RMII 模块。

寄存器 35-30： EMACxTEST：以太网控制器 MAC 测试寄存器 (1,2,3,4)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 15							bit 8

r-x	r-x	r-x	r-x	r-x	R/W-0	R/W-0	R/W-0
—	—	—	—	—	TESTBP	TESTPAUSE(5)	SHRTQNTA(5)
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-3 **保留：**保持为 0；忽略读操作

bit 2 **TESTBP：**测试背压位
1 = MAC 将在链路上产生背压。背压会导致发送前导，引发载波侦听。来自系统的发送数据包将在背压期间发送出去。
0 = 正常工作

bit 1 **TESTPAUSE：**测试暂停位 (5)
1 = MAC 控制子层将禁止发送，就如同接收到了带非零暂停时间参数的暂停接收控制帧
0 = 正常工作

bit 0 **SHRTQNTA：**短切暂停份额位 (5)
1 = MAC 会将有效暂停份额从 64 字节时间减为 1 字节时间
0 = 正常工作

注 1：该寄存器具有关联的清零寄存器（EMACxTESTCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（EMACxTESTSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（EMACxTESTINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：这些寄存器（包括置 1、清零和取反寄存器）允许 16 位和 32 位访问。它们不允许 8 位访问，硬件会忽略 8 位访问。

5：这些位仅用于测试。

寄存器 35-31： EMACxMCFG：以太网控制器 MAC MII 管理配置寄存器 (1,2,3,4)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	r-x	r-x	r-x	r-x	r-x	r-x	r-x
RESETMGMT	—	—	—	—	—	—	—
bit 15							bit 8

r-x	r-x	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CLKSEL<3:0>(5)				NOPRE	SCANINC
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **保留：**保持为 0；忽略读操作

bit 15 **RESETMGMT：**测试复位 MII 管理位
1 = 复位 MII 管理模块
0 = 正常工作

bit 14-6 **保留：**保持为 0；忽略读操作

bit 5-2 **CLKSEL<3:0>：**MII 管理时钟选择位 (5)
时钟分频逻辑使用该字段来产生 MII 管理时钟（MDC），IEEE 802.3u 规定该时钟不能超过 2.5 MHz。一些 PHY 最高支持 12.5 MHz 的时钟速率。

bit 1 **NOPRE：**抑制前导位
1 = MII 管理将执行不带 32 位前导字段的读 / 写周期。一些 PHY 支持抑制前导
0 = 执行正常的读 / 写周期

bit 0 **SCANINC：**扫描递增位
1 = MII 管理模块将在一系列 PHY 之间执行读周期。读周期将从地址 1 开始，一直读到在 EMACxMADR <PHYADDR> 中设置的值为止
0 = 连续读取同一 PHY

注 1：该寄存器具有关联的清零寄存器（EMACxMCFGCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（EMACxMCFGSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（EMACxMCFGINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：这些寄存器（包括置 1、清零和取反寄存器）允许 16 位和 32 位访问。它们不允许 8 位访问，硬件会忽略 8 位访问。

5：表 35-3 给出了时钟分频比编码的说明。

表 35-3： MIIM 时钟选择

MIIM 时钟选择	EMACxMCFG<5:2>
SCLK 4 分频	000x
SCLK 6 分频	0010
SCLK 8 分频	0011
SCLK 10 分频	0100
SCLK 14 分频	0101
SCLK 20 分频	0110
SCLK 28 分频	0111
SCLK 40 分频	1000
未定义	任何其他组合

寄存器 35-32： EMACxMCMD：以太网控制器 MAC MII 管理命令寄存器 (1,2,3,4)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 15							bit 8

r-x	r-x	r-x	r-x	r-x	r-x	R/W-0	R/W-0
—	—	—	—	—	—	SCAN	READ
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-2 **保留**：保持为 0 ； 忽略读操作

bit 1 **SCAN**：MII 管理扫描模式位
1 = MII 管理模块将连续地执行读周期 （例如，可用于监视链路故障）
0 = 正常工作

bit 0 **READ**：MII 管理读命令位
1 = MII 管理模块将执行单个读周期。读取的数据在 EMACxMRDD 寄存器中返回
0 = MII 管理模块将执行写周期。写入的数据取自 EMACxMWTD 寄存器

注 1：该寄存器具有关联的清零寄存器（EMACxMCMDCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（EMACxMCMDSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（EMACxMCMDINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：这些寄存器（包括置 1、清零和取反寄存器）允许 16 位和 32 位访问。它们不允许 8 位访问，硬件会忽略 8 位访问。

寄存器 35-33： EMACxMADR：以太网控制器 MAC MII 管理地址寄存器 (1,2,3,4)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1
—	—	—	PHYADDR<4:0>				
bit 15							bit 8

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	REGADDR<4:0>				
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-13 **保留：**保持为 0；忽略读操作

bit 12-8 **PHYADDR<4:0>：**MII 管理 PHY 地址位
该字段代表管理周期的 5 位 PHY 地址字段。最多可寻址 31 个 PHY （0 是保留地址）。

bit 7-5 **保留：**保持为 0；忽略读操作

bit 4-0 **REGADDR<4:0>：**MII 管理寄存器地址位
该字段代表管理周期的 5 位寄存器地址字段。最多可访问 32 个寄存器。

注 1：该寄存器具有关联的清零寄存器（EMACxMADRCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（EMACxMADRSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（EMACxMADRINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：这些寄存器（包括置 1、清零和取反寄存器）允许 16 位和 32 位访问。它们不允许 8 位访问，硬件会忽略 8 位访问。

寄存器 35-34： EMACxMWTD：以太网控制器 MAC MII 管理写数据寄存器 (1,2,3,4)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 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
MWTD<15:8>							
bit 15							bit 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<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **保留：**保持为 0；忽略读操作

bit 15-0 **MWTD<15:0>：**MII 管理写数据位
在写入时，将使用 16 位数据和 EMACxMADR 寄存器中预先配置的 PHY 地址和寄存器地址来执行 MII 管理写周期。

注 1：该寄存器具有关联的清零寄存器（EMACxMWTDCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（EMACxMWTDSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（EMACxMWTDINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：这些寄存器（包括置 1、清零和取反寄存器）允许 16 位和 32 位访问。它们不允许 8 位访问，硬件会忽略 8 位访问。

寄存器 35-35： EMACxMRDD：以太网控制器 MAC MII 管理读数据寄存器 (1,2,3,4)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 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
MRDD<15:8>							
bit 15							bit 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<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **保留：**保持为 0；忽略读操作

bit 15-0 **MRDD<15:0>：**MII 管理读数据位
在 MII 管理读周期之后，可以从该单元中读取 16 位数据。

注 1： 该寄存器具有关联的清零寄存器（EMACxMRDDCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2： 该寄存器具有关联的置 1 寄存器（EMACxMRDDSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3： 该寄存器具有关联的取反寄存器（EMACxMRDDINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4： 这些寄存器（包括置 1、清零和取反寄存器）允许 16 位和 32 位访问。它们不允许 8 位访问，硬件会忽略 8 位访问。

寄存器 35-36： EMACxMIND：以太网控制器 MAC MII 管理指示寄存器 (1,2,3,4)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 15							bit 8

r-x	r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	LINKFAIL	NOTVALID	SCAN	MIIMBUSY
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-4 **保留：**保持为 0；忽略读操作

bit 3 **LINKFAIL：**链路故障位
返回 1 时，指示发生了链路故障。该位反映上次从 PHY 状态寄存器读取的值。

bit 2 **NOTVALID：**MII 管理读数据无效位
返回 1 时，指示 MII 管理读周期尚未完成，读数据尚未变为有效。

bit 1 **SCAN：**MII 管理扫描位
返回 1 时，指示正在进行扫描操作 （连续 MII 管理读周期）。

bit 0 **MIIMBUSY：**MII 管理忙状态位
返回 1 时，指示 MII 管理模块当前正在执行 MII 管理读周期或写周期。

注 1：该寄存器具有关联的清零寄存器 （EMACxMINDCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器 （EMACxMINDSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器 （EMACxMINDINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：这些寄存器 （包括置 1、清零和取反寄存器）允许 16 位和 32 位访问。它们不允许 8 位访问，硬件会忽略 8 位访问。

寄存器 35-37： EMACxSA0：以太网控制器 MAC 站点地址 0 寄存器 (1,2,3,4,5)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

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>							
bit 15							bit 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
STNADDR5<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **保留：**保持为 0；忽略读操作

bit 15-8 **STNADDR6<7:0>：**站点地址八位字节 6 位
该字段保存站点地址的第 6 个发送的八位字节。

bit 7-0 **STNADDR5<7:0>：**站点地址八位字节 5 位
该字段保存站点地址的第 5 个发送的八位字节。

注 1：该寄存器具有关联的清零寄存器 （EMACxSA0CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器 （EMACxSA0SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器 （EMACxSA0INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：这些寄存器 （包括置 1、清零和取反寄存器）允许 16 位和 32 位访问。它们不允许 8 位访问，硬件会忽略 8 位访问。

5：在复位时，会向该寄存器中装入出厂时预先设定的站点地址。

寄存器 35-38： EMACxSA1：以太网控制器 MAC 站点地址 1 寄存器 (1,2,3,4,5)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

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>							
bit 15							bit 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
STNADDR3<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **保留：**保持为 0；忽略读操作

bit 15-8 **STNADDR4<7:0>：**站点地址八位字节 4 位
该字段保存站点地址的第 4 个发送的八位字节。

bit 7-0 **STNADDR3<7:0>：**站点地址八位字节 3 位
该字段保存站点地址的第 3 个发送的八位字节。

注 1：该寄存器具有关联的清零寄存器（EMACxSA1CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（EMACxSA1SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（EMACxSA1INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：这些寄存器（包括置 1、清零和取反寄存器）允许 16 位和 32 位访问。它们不允许 8 位访问，硬件会忽略 8 位访问。

5：在复位时，会向该寄存器中装入出厂时预先设定的站点地址。

寄存器 35-39： EMACxSA2：以太网控制器 MAC 站点地址 2 寄存器 (1,2,3,4,5)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

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>							
bit 15							bit 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
STNADDR1<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-16 **保留：**保持为 0；忽略读操作

bit 15-8 **STNADDR2<7:0>：**站点地址八位字节 2 位
该字段保存站点地址的第 2 个发送的八位字节。

bit 7-0 **STNADDR1<7:0>：**站点地址八位字节 1 位
该字段保存站点地址的最高 （第 1 个发送的）八位字节。

注 1：该寄存器具有关联的清零寄存器 （EMACxSA2CLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器 （EMACxSA2SET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器 （EMACxSA2INV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

4：这些寄存器 （包括置 1、清零和取反寄存器）允许 16 位和 32 位访问。它们不允许 8 位访问，硬件会忽略 8 位访问。

5：在复位时，会向该寄存器中装入出厂时预先设定的站点地址。

35.4 工作原理

以太网控制器提供了使用外部 PHY 芯片实现 10/100 Mbps 以太网节点所需的系统模块。为了分担 CPU 从模块中移出和向模块中移入数据包数据所产生的开销，控制器中包含了两个基于内部描述符的 DMA 引擎。

以太网控制器包含以下子模块：

- 10/100 Mb 介质访问控制器 （MAC）：实现数据链路层的介质访问控制 （MAC）子层，并执行 ISO/IEC 8802-3 和 IEEE 802.3 规范中包含的 CSMA/CD 功能。它包括：
  - 用于连接外部 PHY 的介质无关接口
  - 用于连接外部 PHY 的精简型介质无关接口
  - 用于与外部 MII PHY 进行控制 / 状态连接的 MII 管理模块
  - 执行 IEEE 802.3 附录 31B 中包含的接收路径流量控制功能
  - 实现与发送和接收 DMA 引擎连接的 MAC 发送和 MAC 接收接口
- 流量控制 （FC）：负责根据 IEEE 802.3 附录 31B 中定义的控制暂停帧的发送。
- 接收过滤器 （RXF）：该模块用于对每个接收数据包执行多重过滤，以确定对于每个数据包是接收还是拒绝。
- 发送 DMA/TXBM 引擎：发送 DMA 引擎和发送缓冲区管理引擎执行从数据包缓冲区到 MAC 发送接口的数据传输，以及在传输完成之后将发送状态向量 （TSV）从 MAC 传输到数据包缓冲区。它使用发送描述符表工作。
- 接收 DMA/RXBM 引擎：接收 DMA 引擎和接收缓冲区管理引擎会通过接收描述符表，将接收数据包和接收状态向量 （RSV）从 MAC 传输到数据包缓冲区。

35.4.1 以太网帧概述

符合 IEEE 802.3 规范的以太网帧（数据包）的长度介于 64 和 1518 字节之间（未包括前导和帧起始定界符）。长度小于 64 字节的帧称为“过短”帧，而长度大于 1518 字节的帧称为“超大”帧。

以太网帧由以下字段组成：

- 流起始 / 前导
- 帧起始定界符 （Start-of-Frame Delimiter， SFD）
- 目标 MAC 地址 （DA）
- 源 MAC 地址 （SA）
- 类型 / 长度字段
- 数据有效载荷
- 可选的填充字段
- 帧校验序列 （FCS）

图 35-2 给出了实际物理电缆上的通信图示。关于以太网协议的详细信息，请参见 IEEE 802.3 规范。

35.4.1.1 流起始 / 前导和帧起始定界符

在以太网介质上发送时， MAC 自动在以太网帧起始位置附加流起始 / 前导和 SFD 字段。

在接收时，会自动从接收帧中除去这些字段，从而这些字段不会被写入接收数据缓冲区中。

软件不需要处理 / 生成这些字段。

35.4.1.2 目标 MAC 地址

MAC 地址是由 6 个八位字节组成的数字，代表以太网网络上节点的物理地址。目标地址包含帧所期望发送到的器件的 MAC 地址。在以太网空间中，存在一些不同类型的地址。例如：

- 单播地址：指定为仅供所寻址的节点使用。在单播地址中，地址第一字节的最高有效位为 0（即，地址第一字节为偶数）。例如，“00 04 a3 00 00 01”是单播地址，而 “01 04 a3 00 00 01”不是。
- 组播地址：指定为供一组选定以太网节点使用。在组播地址中，地址第一字节的最高有效位置 1 （即，该字节为奇数）。例如，“01 04 a3 00 00 01”是组播地址。请注意，组播地址“FF-FF-FF-FF-FF-FF”是保留地址 （广播地址），会被定向到网络上的所有节点。

以太网控制器具有可配置为接收或丢弃单播、组播和 / 或广播帧的接收过滤器模块。关于接收过滤器的详细信息，请参见**第 35.4.8 节 “接收过滤概述”**。

35.4.1.3 源 MAC 地址

源地址是发送以太网帧的节点的 6 字节字段 MAC 地址。每个以太网器件必须具有全局唯一的 MAC 地址。每个包含以太网控制器的 PIC32MX 都具有一个唯一地址，该地址会在上电时装入 MAC 寄存器。该值可按原样使用，也可以在运行时使用不同地址重新配置寄存器。

35.4.1.4 类型 / 长度

它是一个 2 字节字段，指示帧所属的协议。使用诸如因特网协议 （Internet Protocol， IP）或地址解析协议 （Address Resolution Protocol， ARP）之类标准的应用程序应使用相应标准文档中规定的类型代码。或者，在实现专用网络时，该字段也可以用作长度字段。通常，在值小于等于 1500 （0x05DC）时，该字段会被视为长度字段，指定随后的数据字段中包含的非填充数据量。

35.4.1.5 数据

对于每个帧，数据字段通常包含0至1500字节的有效载荷数据。当HUGEFRM（EMACxCFG2<2>）位置 1 时， PIC32MX 器件能够发送和接收大于该长度的帧。但是，大多数以太网节点很可能会丢弃这些不满足 IEEE 802.3 规范的较大帧。

35.4.1.6 填充

填充字段是一个可变长度的字段，附加该字段是为了在发送小的数据有效载荷时也能满足 IEEE 802.3 规范要求。以太网帧的最小有效载荷为 46 字节。较小的帧必须进行填充，以填满该空间。对于发送帧，软件可以通过使用PADENABLE、VLANPAD和AUTOPAD位（EMACxCFG2<5:7>）指示以太网控制器自动生成所需的填充数据。但是，如果未使能自动填充，并且应用程序未进行相应填充，PIC32MX 器件也不会阻止发送这些“过短”帧。在接收帧时，PIC32MX 器件会接收并将所有填充数据写入接收缓冲区。可以选择通过过短错误拒绝过滤器，对小于所要求的 64 字节的帧进行过滤，如**第 35.4.8.4 节 “过短拒绝过滤器”**中所述。

35.4.1.7 帧校验序列 （FCS）

FCS 是一个 4 字节字段，包含针对目标、源、类型 / 长度、数据和填充字段计算的标准 32 位 CRC。它可用于检测发送错误。

对于发送帧， PIC32MX 器件可以通过使用 CRCENABLE （EMACxCFG2<4>）位自动生成并附加有效的 FCS。否则，软件必须计算待发送帧的 CRC，并正确地附加它。

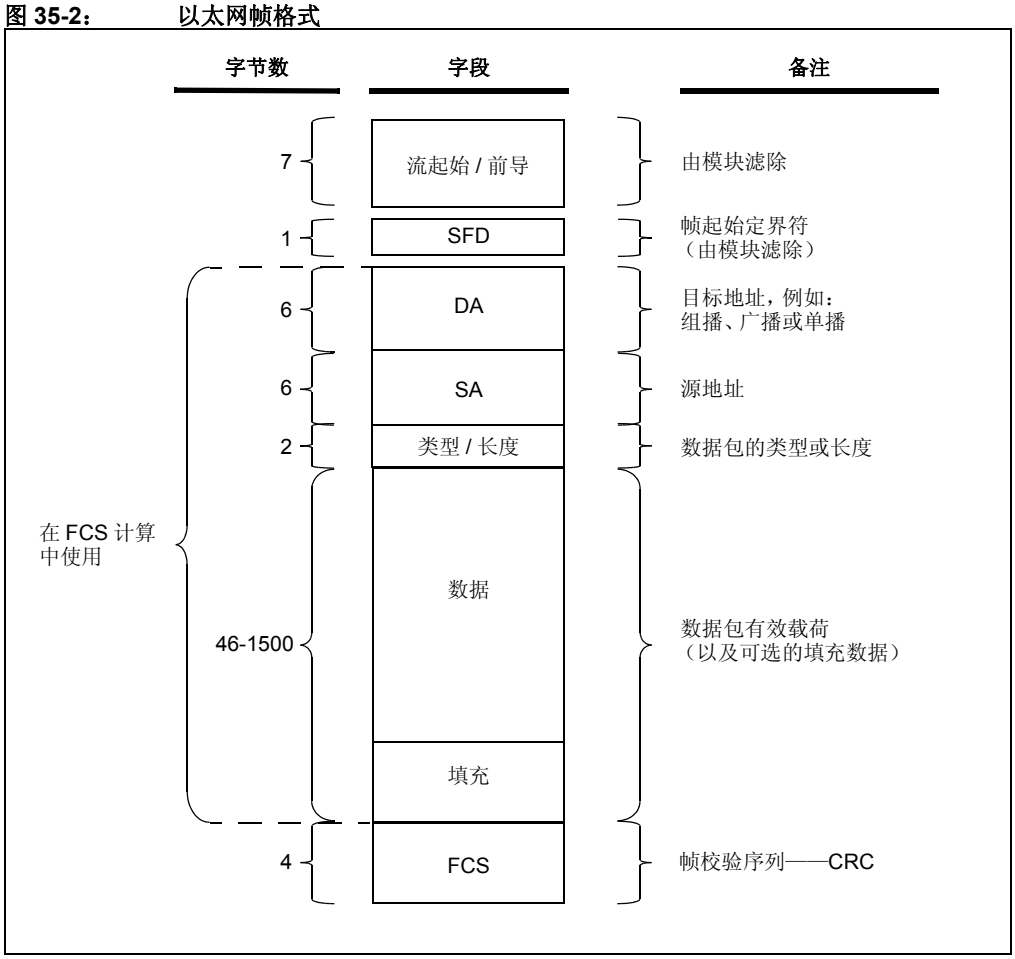
对于接收帧， FCS 字段会被存储到接收缓冲区中。通过使用**第 35.4.8.1 节 “CRC 错误接收过滤器”**和**第 35.4.8.3 节 “CRC 校验接收过滤器”**中描述的 CRC 错误和 CRC 校验接收过滤器，可以丢弃或接收带有无效 CRC 值的帧。

注： 用于生成 FCS 的多项式为：

G(x) = x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1

FCS 从 bit 31 开始发送，以 bit 0 结束。

图 35-2： 以太网帧格式



35.4.2 基本以太网控制器操作

通过将 ETHCON1 寄存器中的 ON （ETHCON1<15>）位置 1，可以使能以太网控制器。

通过将 ETHCON1 寄存器中的 ON 位清零，可以禁止以太网控制器。这是任何复位后的默认状态。如果禁止了以太网控制器，则用于 MII/RMII 和 MIIM 接口的所有 I/O 引脚都将用作端口引脚，并由相应的 PORT 锁存器位和 TRIS 位控制。

禁止控制器会使内部 DMA 状态机复位，并且所有发送和接收操作都会被中止。SFR 仍然可供访问，它们的值会被保留。

在以太网控制器处于活动状态时清零 ON 位会中止所有待处理操作，并按上面的描述复位外设。

重新使能 ON 位会以主复位状态重新启动以太网控制器，并同时保留 SFR 值。

注 1： 如果在活动内部总线事务期间清零 ON 位，控制器会在进入禁止状态之前先完成当前总线事务。禁止控制器之后，TXBUSY（ETHSTAT<6>）和RXBUSY（ETHSTAT<5>）位将反映为不活动状态。

2： 每次通过 ON 位复位以太网控制器时，软件也应通过 MIIM 寄存器复位外部 PHY。这可以确保 PHY 处于已知的初始化状态。此外，还应通过 EMACxCFG1 寄存器对 MAC 进行软复位。

35.4.3 MAC 概述

MAC 子层是 OSI 模型中所描述功能的一部分，用于数据链路层。它定义一个介质无关的设施，基于物理层提供的介质无关物理设施而构建，并位于访问层无关的 LLC 子层或其他 MAC 客户端之下。它适用于一般的适合使用载波侦听多路访问 / 冲突检测 （CSMA/CD）的局域广播介质。

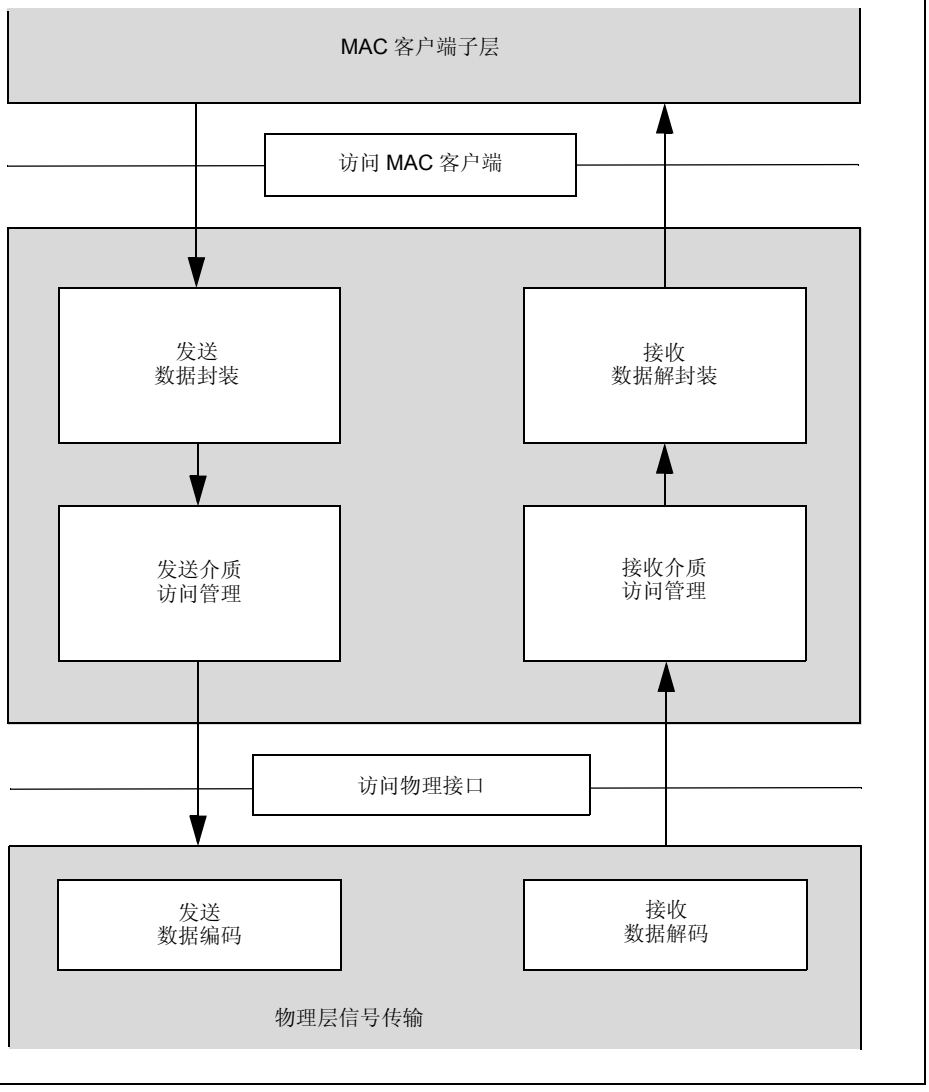
CSMA/CD MAC 子层为 MAC 客户端提供一些发送和接收帧所需的服务。CSMA/CD MAC 子层会尽最大努力来取得介质，并向物理层传输串行位流。虽然 MAC 会向客户端报告一些特定错误，但 MAC 不提供错误恢复。

以下总结了 CSMA/CD MAC 子层的功能，如图 35-3 中所示：

- 对于帧发送：
  - 从 MAC 客户端接收数据并构造一个帧
  - 将串行位数据流送到物理层，用于在介质上发送
  - 在半双工模式下，在每次物理介质忙时，延迟串行位流的发送
  - 它可以在外发帧中附加适当的帧校验序列 （FCS）值，并验证是否完全按八位字节边界对齐
  - 将帧位流的发送延迟一个指定的帧间间隔周期
  - 在半双工模式下，在检测到冲突时暂停发送
  - 在半双工模式下，在发生冲突之后安排重发，直到达到指定的重试限制为止
  - 在半双工模式下，通过发送阻塞报文来强制产生冲突，确保传播到整个网络
  - 在帧前附加前导和帧起始定界符，并在所有帧后附加 FCS ；对于数据长度小于最小值的帧，附加 PAD 字段

- 对于帧接收：
  - 从物理层接收串行位数据流
  - 将接收帧送到 MAC 客户端 （广播、组播和单播等）
  - 通过 FCS 和验证是否按八位字节边界对齐的方式，检查传入帧是否存在发送错误
  - 从接收帧中除去前导、帧起始定界符和 PAD 字段 （如果需要）
  - 实现与外部 MII PHY 进行控制 / 状态连接的 MII 管理模块

图 35-3： CSMA/CD 介质访问控制功能



MAC 使用以下 SFR 进行访问：EMACxCFG1、 EMACxCFG2、 EMACxIPGT、 EMACxIPGR、 EMACxCLRT、EMACxMAXF、EMACxSUPP、EMACxTEST 和 EMACxSA0 至 EMACxSA3 寄存器。

注： 关于 MAC 子层功能和操作的详细说明，请参见 IEEE 802.3 规范的条款 2、3 和 4。

35.4.4 介质无关接口 （MII）

介质无关接口 （MII）是 MAC 和 PHY 之间的标准互连方式，用于传输发送帧和接收帧数据。

该 MII 具有以下重要特性：

- 它能够支持 10 Mbps 和 100 Mbps 的数据传输速率，并为管理功能提供支持
- 它提供了独立的 4 位宽发送和接收数据路径
- 它使用 TTL 信号电平，与常用的数字 CMOS 工艺兼容
- 它提供了全双工工作模式

在 10 Mbps 模式下， MII 以 2.5 MHz 运行；在 100 Mbps 模式下，它以 25 MHz 运行。提供 MII 的 PHY 不需要同时支持这两种数据速率，可以支持任意一种，但也可以同时支持两种。

表 35-4 列出了 18 种介质无关接口信号。

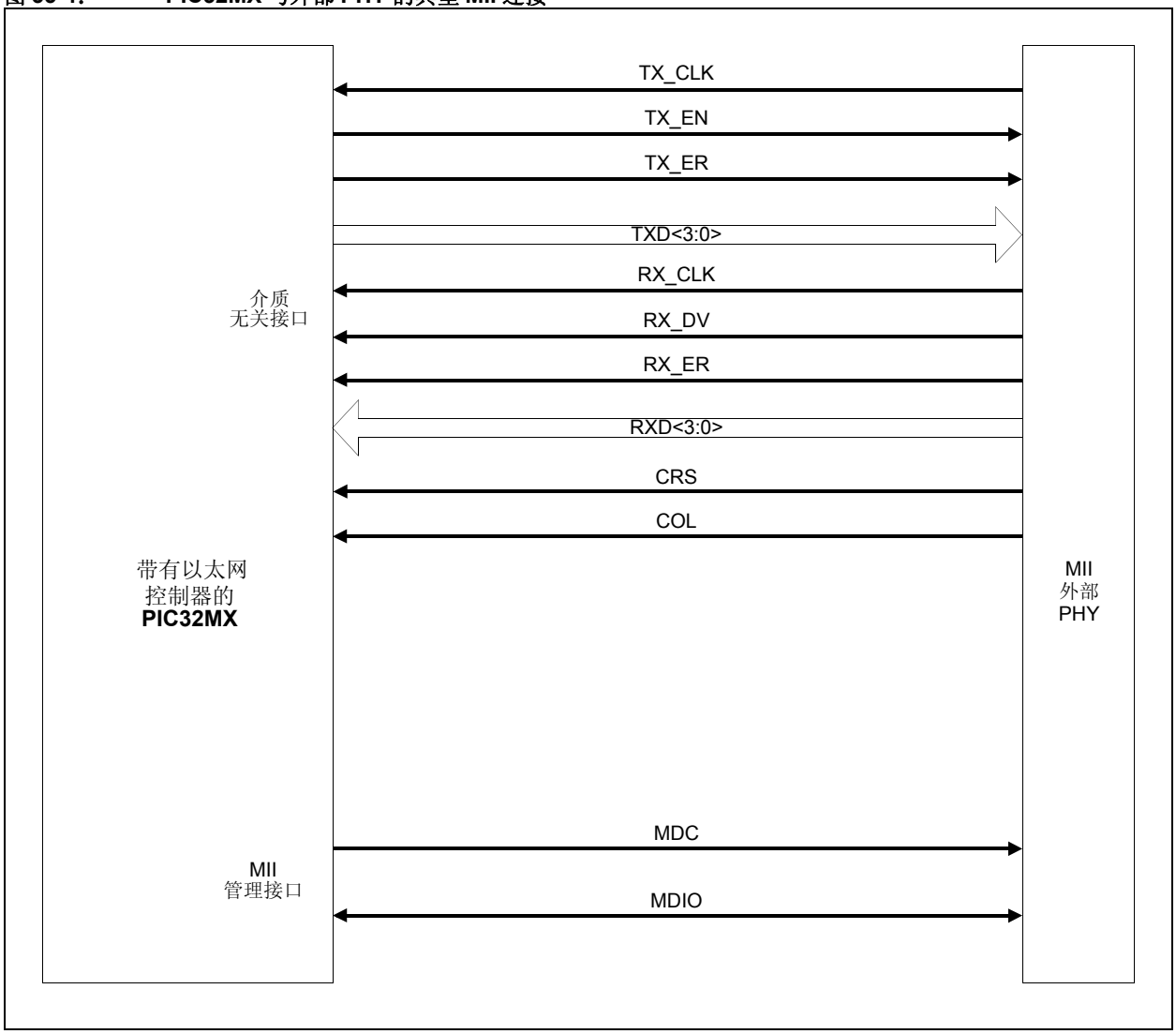
表 35-4： MII 信号

信号名称	宽度	类型	说明
TX_CLK	1	输入	发送时钟信号——它是连续时钟，为从 MAC 向 PHY 传输 TX_EN、 TXD 和 TX_ER 信号提供参考时序。 TX_CLK 频率是标称发送数据速率的四分之一。以 100 Mbps 工作的 PHY 必须提供 25 MHz 的 TX_CLK 频率，以 10 Mbps 工作的 PHY 必须提供 2.5 MHz 的 TX_CLK 频率。
RX_CLK	1	输入	接收时钟信号——它是连续时钟，为从 PHY 向 MAC 传输 RX_DV、 RXD 和 RX_ER 信号提供参考时序。 RX_CLK 的频率等于接收信号数据速率的四分之一。
TX_EN	1	输出	发送使能信号——指示 MAC 正在 MII 上送出用于发送的半字节数据。 TX_EN 与 TX_CLK 同步转换。
TXD<3:0>	4	输出	发送数据信号——它与 TX_CLK 同步转换。
TX_ER	1	输出	发送编码错误信号——它与 TX_CLK 进行同步。当 TX_ER 在一个或多个 TX_CLK 周期中置为有效，同时 TX_EN 也置为有效时， PHY 将会忽略正在发送的帧中不属于有效数据或定界符集一部分的一个或多个符号。该信号仅影响 100 Mbps 数据发送。
RX_DV	1	输入	接收数据有效信号——指示 PHY 正在 RXD 数据线上发送恢复和解码的半字节数据。 RX_DV 与 RX_CLK 进行同步。 RX_DV 在整个帧期间保持有效。
RXD<3:0>	4	输入	接收数据信号——代表与 RX_CLK 进行同步的 4 个数据信号。对于 RX_DV 置为有效时的每个 RX_CLK 周期， RXD<3:0> 会从 PHY 向 MAC 发送 4 位的恢复数据。
RX_ER	1	输入	接收错误信号——在它置为有效时，将向 MAC 指示，从 PHY 传输到 MAC 的帧中出现了编码错误 （或者 PHY 能够检测到的任意错误）。 RX_ER 与 RX_CLK 进行同步。
CRS	1	输入	载波侦听信号——在发送或接收介质非空闲时， PHY 会将该信号置为有效。当发送和接收介质均空闲时， PHY 会将 CRS 置为无效。在存在冲突条件的整个过程中， CRS 会一直保持有效。它不一定要与 TX_CLK 或 RX_CLK 进行同步。
COL	1	输入	检测到冲突信号——在介质上检测到冲突时， PHY 会将该信号置为有效，并在冲突条件存在期间保持有效。它不一定要与 TX_CLK 或 RX_CLK 进行同步。
MDC	1	输出	管理数据时钟信号——它是 MII 管理接口的一部分，**第 35.4.6 节 “介质无关接口管理 （MIIM）”** 对它进行了介绍。
MDIO	1	输入 / 输出	管理数据输入 / 输出信号——它是 MII 管理接口的一部分，**第 35.4.6 节 “介质无关接口管理 （MIIM）”** 对它进行了介绍。

关于详细的 MII 规范，请参见 IEEE 802.3 规范的条款 22。

图 35-4 给出了 PIC32MX 和外部 PHY 之间的典型 MII 连接图。

图 35-4： PIC32MX 与外部 PHY 的典型 MII 连接



35.4.5 精简型介质无关接口 （RMII）

精简型介质无关接口 （RMII）旨在根据 IEEE 802.3 规范的条款 22 中的规定，用作 MII 的低成本、低引脚数替代方案。

管理接口 （MDIO/MDC）假定为与 MII 中定义的相同。

RMII 具有以下特性：

- 能够支持 10 Mbps 和 100 Mbps 数据速率
- 对于 MAC 和 PHY 使用单个参考时钟 （可以源自 MAC 或外部时钟源）
- 提供了独立的 2 位宽发送和接收数据路径
- 使用 TTL 信号电平，与常用的数字 CMOS 工艺兼容
- 提供了全双工工作模式

该接口以 50 MHz 运行。

表 35-5 列出了 10 种精简型介质无关接口信号。

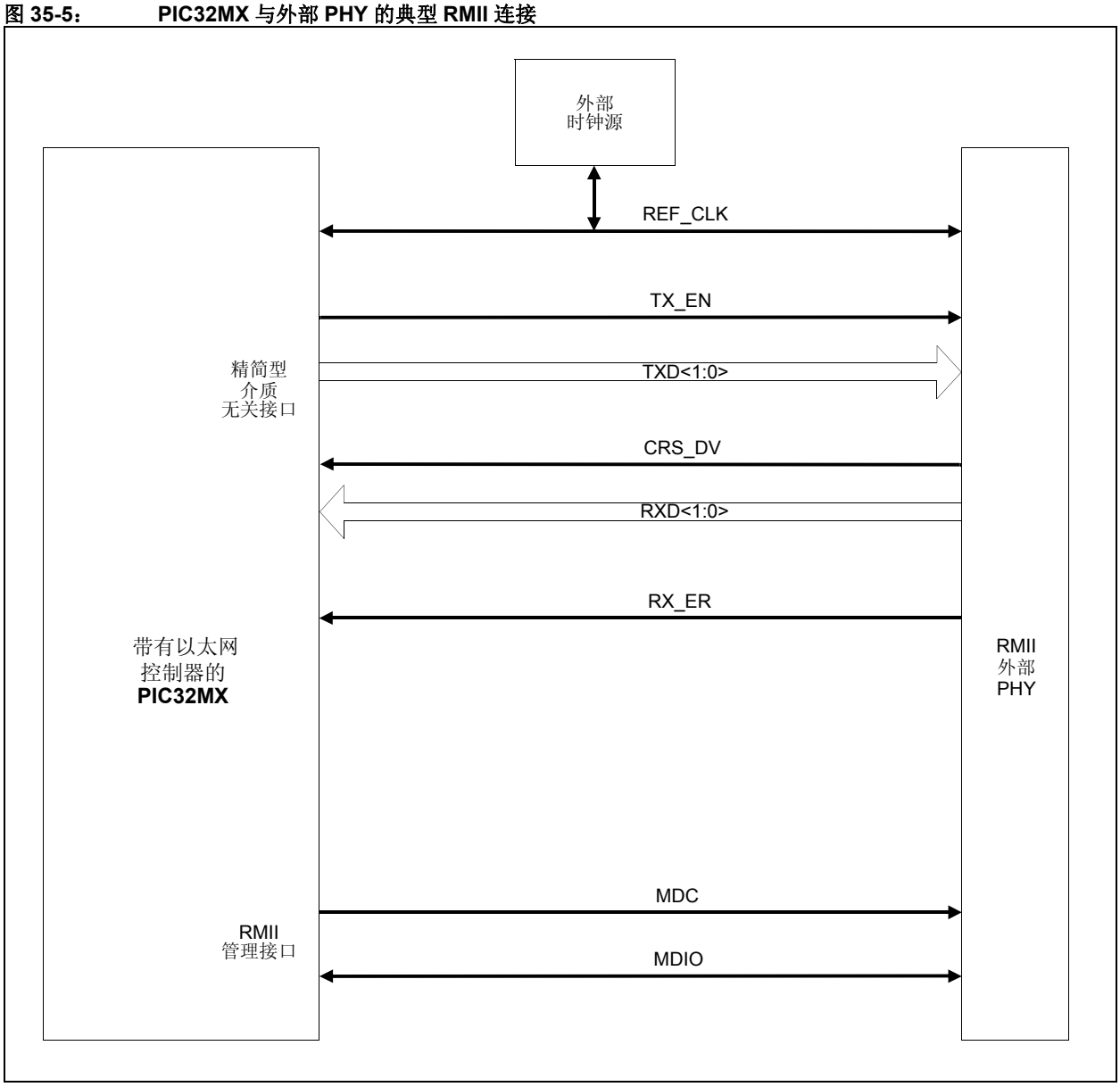
表 35-5： RMII 信号

信号名称	宽度	类型	说明
REF_CLK	1	输入	参考时钟信号——它是连续时钟，为 CRS_DV、 RXD<1:0>、 TX_EN、 TXD<1:0> 和 RX_ER 提供参考时序。 REF_CLK 是源自 MAC 或外部时钟源的 50 MHz 时钟信号。对于 PIC32MX 器件， REF_CLK 是外部提供的时钟信号。
CRS_DV	1	输入	载波侦听 / 接收数据有效信号——在接收介质非空闲时， PHY 会将该信号置为有效。 CRS_DV 在检测到载波时异步置为有效。载波消失会使 CRS_DV 同步于 REF_CLK 置为无效（仅在半字节边界处）。在 CRS_DV 置为有效之后， RXD<1:0> 上的数据会被视为有效数据。通过使用 CRS_DV， MAC 可以准确地恢复 RX_DV 和 CRS。
RXD<1:0>	2	输入	接收数据信号——它与 REF_CLK 同步转换。对于 CRS_DV 置为有效时的每个时钟周期， RXD 会从 PHY 传输 2 位的恢复数据。 • 当 CRS_DV 置为无效时， RXD 为 00 指示空闲条件。由于 PHY 信号 RX_ER 的使用是可选的，为了确保传输所接收信号的错误， RXD 会将已解码数据流中的数据替换为 01，使 MAC CRC 机制拒绝该帧。 • 在 100 Mbps 模式下， RXD 与 REF_CLK 进行同步。 • 在 10 Mbps 模式下，每隔 10 个周期采样 RXD 一次。
TX_EN	1	输出	发送使能信号——指示 MAC 正在 TXD<1:0> 上送出用于发送的二位数据。 TX_EN 在发送前导的第一个半字节时置为有效，并在发送所有二位数据时保持有效。 TX_EN 与 REF_CLK 进行同步。
TXD<1:0>	2	输出	发送数据信号——当 TX_EN 置为有效时，它是送到 PHY 的发送数据。 TXD 数据线与 REF_CLK 同步转换。当 TX_EN 置为无效时， TXD 使用 00 值来指示空闲状态。 • 在 100 Mbps 模式下， TXD 会在 TX_EN 置为有效时的每个 REF_CLK 周期提供有效数据。 • 在 10 Mbps 模式下，由于 REF_CLK 频率为数据速率的 10 倍，所以 TXD 上的值每隔 10 个周期采样一次。
RX_ER	1	输入	接收错误信号——在一个或多个 REF_CLK 周期中置为有效时，将指示在当前正从 PHY 传输的帧中检测到错误。 RX_ER 与 REF_CLK 进行同步。
MDC	1	输出	管理数据时钟信号——它是 MII 管理接口的一部分，**第 35.4.6 节 “介质无关接口管理（MIIM）”** 对它进行了介绍。
MDIO	1	输入 / 输出	管理数据输入 / 输出信号——它是 MII 管理接口的一部分，**第 35.4.6 节 “介质无关接口管理 （MIIM）”** 对它进行了介绍。

关于 RMII 的详细规范，请参见 RMII 联盟倡导的 RMII 规范。

图 35-5 给出了 PIC32MX 和外部 PHY 之间的典型 RMII 连接图。

图 35-5： PIC32MX 与外部 PHY 的典型 RMII 连接



35.4.6 介质无关接口管理 （MIIM）

MII 管理 （MIIM）模块用于提供 PIC32MX 主机和外部 MII PHY 器件之间的串行通信链路。外部串行通信链路按照 IEEE 802.3 的条款 22 标准工作。

MIIM 输入 / 输出信号有：

- 管理数据时钟 （MDC）——MDC 由 MAC 向 PHY 提供，用作在 MDIO 信号上传输信息时的参考时序。
- 管理数据输入 / 输出 （MDIO）——MDIO 是 PHY 和 MAC 之间的双向信号。它用于在 PHY 和 MAC 之间传输控制信息和状态。控制信息由 MAC 根据 MDC 同步驱动，并由 PHY 同步采样。状态信息由 PHY 根据 MDC 同步驱动，并由 MAC 同步采样。

MIIM 链路上的通信以帧的形式进行。在 MII 管理接口上发送的帧具有以下结构 （见表 35-6）：

- 前导：在每个事务开始时， MAC 会在 MDIO 上发送由 32 个逻辑 1 位组成的序列，为 PHY 提供同步模式。
- 帧起始：帧起始通过 <01> 模式指示。
- 操作码：<10> 代表读事务， <01> 代表写事务。
- PHY 地址：5 个位，支持 32 个唯一的 PHY 地址。 PHY 总是会响应地址为 0 的事务。
- 寄存器地址：5 个位，支持在每个 PHY 中寻址 32 个独立寄存器。
- 周转：管理帧的寄存器地址字段和数据字段之间的 2 位时间的间隔，用以避免在读事务期间发生争用。
- 数据：这个 16 位字段携带送至 / 来自所寻址 PHY 寄存器的数据。

注： MDIO 上的 IDLE （空闲）条件为高阻抗状态。

表 35-6： MIIM 帧格式

操作	管理帧字段							
	PRE	ST	OPCODE	PHYAD	REGAD	TA	DATA	IDLE
读	1….1	01	10	a0…a4	r0…r4	Z0	d0….d15	Z
写	1….1	01	01	a0…a4	r0…r4	10	d0….d15	Z

如上所示，MIIM 帧的大小为 64 位。但是，当 PHY 支持抑制前导操作时，可以通过使用 NOPRE （EMACxMCFG<1>）位，将 MIIM 模块配置为抑制 MII 管理串行流的前导部分。

关于 MIIM 的详细信息，请参见 IEEE 802.3 规范中的条款 22 标准。

35.4.6.1 外部 PHY 寄存器访问

PHY 寄存器可对 PHY 模块进行配置和控制，并提供操作状态信息。不同于片上 SFR，PHY 寄存器不能通过 SFR 控制接口直接访问。访问需要通过实现介质无关接口管理的一组特殊的 MAC 控制寄存器来实现。这些控制寄存器称为 MIIM 寄存器。PHY 寄存器通过 MAC 的 MIIM 接口进行访问。要实现该目的，必须在 MAC 中使用 MII 管理命令、地址和数据寄存器。

“以太网控制寄存器汇总”（表 35-1）中列出了用于控制 PHY 寄存器访问的寄存器，包括以下寄存器：

- EMACxMCFG：以太网控制器 MAC MII 管理配置寄存器
- EMACxMCMD：以太网控制器 MAC MII 管理命令寄存器
- EMACxMADR：以太网控制器 MAC MII 管理地址寄存器
- EMACxMWTD：以太网控制器 MAC MII 管理写数据寄存器
- EMACxMRDD：以太网控制器 MAC MII 管理读数据寄存器
- EMACxMIND：以太网控制器 MAC MII 管理指示寄存器

注：	所有 PHY 芯片寄存器都被视为 16 位宽。对未实现单元进行写操作将被忽略，而对这些单元的所有读操作都将返回 0。所有保留的单元都应写为 0；当读取时应忽略其内容。关于寄存器访问的详细信息，请参见特定供应商的 PHY 数据手册。

35.4.6.2 初始化 MII 管理模块

为了让 MAC MIIM 模块产生适当的接口时钟（MDC）频率，需要配置时钟速度。MIIM 模块使用 SCLK 作为输入时钟。

使用 CLKSEL（EMACxMCFG<2:5>）位可选择用于产生 MDC 时钟信号的分频比，IEEE 802.3 规范中规定它不能大于 2.5 MHz。但是，一些 PHY 最高支持 12.5 MHz 的时钟速率。

35.4.6.3 读取 PHY 寄存器

通过 MAC 读取 PHY 寄存器时，将会获取全部 16 位内容。

要读取 PHY 寄存器，请执行以下步骤：

1. 将 PHY 的地址和要读取的 PHY 寄存器的地址写入 EMACxMADR 寄存器。
2. 将READ（EMACxMCMD<0>）位置1。此时读操作会开始，MIIMBUSY（EMACxMIND<0>）位将在 3 个 SCLK 周期之后置 1（这是由于 MIIM 接口内部流水线的原因）。
3. 通过查询 MIIMBUSY 位来确定操作是否完成（操作时间是传输一个完整 MIIM 帧需要的时间）。处于忙状态时，软件不应启动任何 MII 扫描操作或写入 EMACxMWTD 寄存器。当 MAC 已获取寄存器的内容时，MIIMBUSY 位将自动清零。
4. 清零 READ（EMACxMCMD<0>）位。
5. 从 EMACxMRDD 寄存器读取所需的数据。

35.4.6.4 写入 PHY 寄存器

当写入 PHY 寄存器时，会一次性写入全部 16 位；不能选择性地对某些位进行写操作。如果只需要对寄存器中的选定位进行再编程，软件必须首先读取 PHY 寄存器，修改结果数据，然后再将数据写回 PHY 寄存器。

要写入 PHY 寄存器，请执行以下步骤：

1. 将 PHY 的地址和要读取的 PHY 寄存器的地址写入 EMACxMADR 寄存器。
2. 将要写入的 16 位数据写入 EMACxMWTD 寄存器。对该寄存器的这一写操作会自动开始 MIIM 事务，这会导致 MIIMBUSY（EMACxMIND<0>）位在 3 个 SCLK 周期之后置 1（这是由于 MIIM 接口内部流水线的原因）。
3. 查询 MIIMBUSY 位，直到它清零为止；清零指示写操作已完成。
4. PHY 寄存器将在 MIIM 操作完成之后被写入数据，该 MIIM 操作需要一个 MIIM 帧的时间。当写操作完成时，MIIMBUSY 位会将自身清零。处于忙状态时，软件不应启动任何 MII 扫描或读操作。

35.4.6.5 扫描 PHY 寄存器

MAC 可配置为对 PHY 寄存器执行自动背靠背读操作。这可以显著降低在需要定期更新状态信息时的软件复杂度。

要执行扫描操作，请执行以下步骤：

1. 将 PHY 的地址和要读取的 PHY 寄存器的地址写入 EMACxMADR 寄存器。
2. 将 SCAN（EMACxMCMD<1>）位置 1。扫描操作开始，MIIMBUSY（EMACxMIND<0>）位置 1。
3. 第一个读操作将在传输第一个 MIIM 帧之后完成。后续读操作将按同一时间间隔完成，直到操作被取消为止。可以通过查询 NOTVALID （EMACxMIND<2>）位来确定第一个读操作何时完成。从 EMACxMRDD 寄存器读取已扫描的寄存器数据。
4. 将 SCAN 位置 1 之后，每隔一个 MIIM 帧的时间间隔，都会自动更新 EMACxMRDD 寄存器。没有可以用来确定 EMACxMRDD 寄存器何时更新的相关状态信息。
5. 当 MIIM 扫描操作正在进行时，软件一定不能尝试写入 EMACxMWTD 或启动读操作。
6. 取消 MIIM 扫描操作的方式是，清零 SCAN（EMACxMCMD<1>）位，然后查询 MIIMBUSY 位。在 MIIMBUSY 位清零之后，可以启动新的操作。

例 35-1 给出了 MIIM 初始化和 PHY 寄存器读、写和扫描操作的示例代码。

例 35-1： MIIM 初始化和 PHY 访问

```
// 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

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

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

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

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

    // Make sure data has been written

    // Perform a scan of the status PHY register: 1
    // Start the scan
    while(EMACxMIND&0x1);                      // wait in case of some previous operation
    EMACxMADR=0x1|((PHY_ADDRESS)<<8);          // set the PHY and register address
    EMACxMCMD=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(EMACxMIND&0x4);                      // wait data valid
    phyRegVal=EMACxMRDD;                       // read the scanned register
```

35.4.7 流量控制概述

以太网流量控制包含发送和接收暂停帧的功能，暂停帧会导致接收节点在一个特定时长内停止传输。

在发送端，当使能发送流量控制时，流量控制（FC）模块会处理 MAC 和 CPU 之间的硬件握手。接收数据包的流量控制属于 MAC 功能的一部分。

PIC32MX MAC 同时支持 IEEE 802.3 标准的条款 28、表 28B-2、条款 31 和附录 31B 中描述的对称暂停和非对称暂停。

FC 模块支持两种工作模式：手动和自动。此外， FC 模块还会使用在 MAC 寄存器中设定的发送模式 （全双工或半双工）。

注：	在发送逻辑正在发送数据包时，软件不应更改全双工或半双工工作模式。

软件必须先使能流量控制，然后才能减少传入的数据包。在全双工和半双工模式下，流量控制机制的工作方式不同。

35.4.7.1 全双工

在发送端，MAC 会发送带有暂停定时器值的暂停控制帧。接收 MAC 会对控制帧进行解码，提取暂停定时器值，并在指定时间内暂停发送。请注意，这并不意味着发送器件会立即暂停。暂停机制的激活存在一定延时。如果要在暂停定时器值到期之前取消激活流量控制，可以发送另一个暂停定时器值编码为 0x0000 的暂停帧。

从接收端的角度看，如果有另一个器件发送暂停帧，并且 MAC 接收到暂停帧，那么接收操作会被禁止，直到暂停定时器到期或另一个器件取消暂停帧请求为止。

注：	暂停帧中包含所请求的暂停时间周期，其范围为 0 至 65535。暂停时间以暂停 “份额”为单位测量，其中的每个单位等于 512 个位时间。

35.4.7.2 半双工

半双工流量控制的工作方式类似。当软件使能流量控制时，FC 模块会请求 MAC 应用背压。MAC 会不断在发送线上发送前导模式，以防止任何其他器件获得总线控制权。这会不断进行，直到流量控制被禁止为止。

35.4.7.3 手动流量控制

手动流量控制通过 MANFC （ETHCON1<4>）控制位来使能。

当使能手动流量控制时，MAC 会使用 PTV（ETHCON1<31:16>）值发送暂停控制帧。当禁止发送流量控制时， MAC 会发送另一个暂停定时器值编码为 0x0000 的暂停帧，以禁止流量控制。

35.4.7.4 自动流量控制

自动流量控制通过 AUTOFC （ETHCON1<7>）控制位来使能。当使能自动流量控制时，硬件会根据 BUFCNT （ETHSTAT<23:16>）位的当前值发送暂停控制帧，方式如下：

- 当 BUFCNT 达到 RXFWM （ETHRXWM<23:16>）位指定的值时，它会每隔 512/2 * PTV （ETHCON1<31:16>）个发送时钟周期自动发送暂停帧。

注 1： 发送时钟周期为10 MHz或100 MHz，取决于当前MAC速度选择. 10 Mbps或100 Mbps。

2： 软件必须确保在由可用接收描述符分配的可用空间量降至低于最大以太网帧大小两倍 （即， 1536 * 2）时，流量控制水印值可以允许发送暂停帧。这可以确保不会产生接收溢出条件。

3： 只有未使能该操作 （ETHCON1<15> = 0）时，才能更改 PTV 值。

- 当 BUFCNT 达到 RXEWM （ETHRXWM<7:0>）指定的值时，将会发送暂停定时器值设置为 0x0000 的暂停帧。

请注意，BUFCNT 值仅在数据包边界处发生更新；因此，所有自动流量控制更改都发生在数据包边界处。

当使能自动流量控制时，它具有设置和清除流量控制操作的最高优先级。因此，建议不要混合使用自动和手动流量控制。

手动发送暂停帧需要的步骤序列有：

1. 在初始化序列中，软件通过写入 PTV 值 （ETHCON1<31:16>）来设置暂停值。
2. 软件通过写入 MANFC 位 （ETHCON1<4>）来手动启动暂停帧的发送。
3. FC 模块会请求 MAC 发送暂停帧。
4. MAC 将按如下方式组合完整的流量控制帧：
   a) 前导
   b) 帧起始定界符 （SFD）
   c) 目标地址 = 01-80-c2-00-00-01 （特殊的暂停组播地址）
   d) 源地址 = 来自 EMACxSA0-EMACxSA3 寄存器的站点地址
   e) 长度 = 0x8088 （控制帧）
   f) 有效载荷：
      - 操作码 （2 字节） = 0x0001
      - 暂停值 （2 字节） = PTV
   g) 填充
   h) FCS

例 35-2： 使用手动流量控制代码

```
// 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 接收过滤概述

接收过滤器（RXF）模块会检查所有传入的接收数据包，并根据用户可选的过滤器接收或拒绝数据包。支持以下接收过滤器：

- 由 CRCERREN （ETHRXFC<7>）控制的 CRC 错误接收过滤器
- 由 RUNTERREN （ETHRXFC<5>）控制的过短错误接收过滤器
- 由 CRCOKEN （ETHRXFC<6>）控制的 CRC 校验拒绝过滤器
- 由 RUNTEN （ETHRXFC<4>）控制的过短拒绝过滤器
- 由 UCEN （ETHRXFC<3>）控制的单播接收过滤器
- 由 NOTMEEN （ETHRXFC<2>）控制的非己单播接收过滤器
- 由 MCEN （ETHRXFC<1>）控制的组播接收过滤器
- 由 BCEN （ETHRXFC<0>）控制的广播接收过滤器
- 由 HTEN （ETHRXFC<15>）控制的哈希表接收过滤器
- 由 MPEN （ETHRXFC<14>）控制的 Magic Packet 接收过滤器
- 由 PMMODE （ETHRXFC<8-11>）和 NOTPM （ETHRXFC<12>）控制的带逻辑反相的模式匹配接收过滤器

注：	每个过滤器或为接收过滤器，或为拒绝过滤器。接收过滤器会强制接收数据包，而拒绝过滤器会强制拒绝数据包。

以上过滤器的顺序从高到低指定过滤器的优先级，从而如果使能了某个过滤器，并且该过滤器接收或拒绝了某个数据包，则所有优先级较低的过滤器不起任何作用。

例如，如果使能了过短错误接收过滤器，并且接收到小于 64 字节的数据包，则即使 CRC 校验失败，也总是会接收它。

如果已使能的过滤器未明确地接收或丢弃某个接收数据包，则默认情况下会丢弃该数据包。

由于接收过滤器内部设计的原因，最终是接收还是中止以太网帧的决定将在帧结束时作出。

当接收到数据包时，每个接收数据包的接收状态向量（RSV）中都会包含关于哪些过滤器与相应接收数据包匹配的信息，无论这些过滤器当时是否处于活动状态。这可以提供关于数据包的额外“状态“信息，可在软件中用于过滤数据包。例如，在混杂模式下， Magic Packet 过滤器 RSV 位可用于快速地识别 Magic Packet，而无需检查帧内容。关于 RSV 的更多信息，请参见以太网描述符表格式 （链表， NPV = 1）（表 35-9）。

所有过滤器设置都使用 ETHRXFC 寄存器来完成。

由于 RXF 模块中同步的原因，在 ON 位 （ETHCON1<15>）置 1 时，不应更改以下寄存器：

- ETHRXFC：以太网控制器接收过滤器配置寄存器
- ETHHT0：以太网控制器哈希表 0 寄存器
- ETHHT1：以太网控制器哈希表 1 寄存器
- ETHPMO：以太网控制器模式匹配偏移寄存器
- ETHPMCS：以太网控制器模式匹配校验和寄存器
- ETHPMM0：以太网控制器模式匹配屏蔽器 0 寄存器
- ETHPMM1：以太网控制器模式匹配屏蔽器 1 寄存器
- EMACxSA0：以太网控制器 MAC 站点地址 0 寄存器
- EMACxSA1：以太网控制器 MAC 站点地址 1 寄存器
- EMACxSA2：以太网控制器 MAC 站点地址 2 寄存器

要更改其中一个或多个寄存器 / 寄存器位，必须先清零 ON （ETHCON1<15>）位。

以下是每个接收过滤器的简要总结。

35.4.8.1 CRC 错误接收过滤器

该过滤器用于明确接收未通过 CRC 校验的数据包。如果使能，则无论是否使能 CRC 校验过滤器，都会接收未通过 CRC 校验的所有数据包。

35.4.8.2 过短错误接收过滤器

该过滤器用于明确接收未通过过短检查的数据包。如果使能，则无论是否使能过短过滤器，都会接收未通过过短检查的所有数据包。

35.4.8.3 CRC 校验接收过滤器

如果使能，则会检查 MAC CRC 校验的结果，并将结果用于过滤数据包。如果使能了该过滤器，并且未通过，则数据包会被中止。相反地，如果未使能 CRC 校验，则接收数据包的 CRC 会被忽略，不用作数据包的接收要求。

35.4.8.4 过短拒绝过滤器

通过过短过滤器，可以根据接收数据包（目标地址、源地址、长度 / 类型、有效载荷和 FCS）的大小进行过滤。当使能过短拒绝过滤器时，所有小于 64 字节的数据包都会被拒绝。

35.4.8.5 传播类型接收过滤器

数据包按照传播类型进行过滤，支持以下类型：

- 单播：接收类型为单播，且目标地址与站点地址匹配的所有数据包
- 非己单播：接收类型为单播，且目标地址与站点地址不匹配的所有数据包
- 广播：接收类型为广播的所有数据包
- 组播：接收类型为组播的所有数据包

如果任意处于活动状态的传播类型过滤器接收了某个接收数据包，则该数据包可能会被接收（取决于其他过滤器）。例如，如果同时使能了单播和广播过滤器，将允许接收其中任一类型的数据包。

要接收所有传入数据包（混杂模式），只需使能 UCEN、NOTMEEN、MCEN 和 BCEN 过滤器，并禁止所有其他过滤器即可。

35.4.8.6 哈希表接收过滤器

在使能时，哈希表过滤器会根据接收数据包的目标地址（最多允许 64 个不同地址）接收数据包。实现方式是使用来自 MAC 的目标地址 CRC 输出作为用户定义的哈希表中的查询键。

首先，根据接收数据包目标地址字段计算 CRC 值。然后，该值的 bit <28:23> 用作一个 64 位的用户可编程表 （ETHHT0 和 ETHHT1）的索引，该表中包含一些单个位值，代表接收 （1）还是忽略 （0）。例如，如果计算得到的 CRC 值为 05h，则会检查 64 位 HT 寄存器表中的 bit 5 的值。如果该条目为逻辑 1，则会接收数据包；否则，在决定是否接收数据包时不会考虑该过滤器的结果。

请注意，该过滤器使用的目标地址 CRC 输出对应于接收数据包目标地址的 32 位 CRC（非补码）的 bit <28:23>。

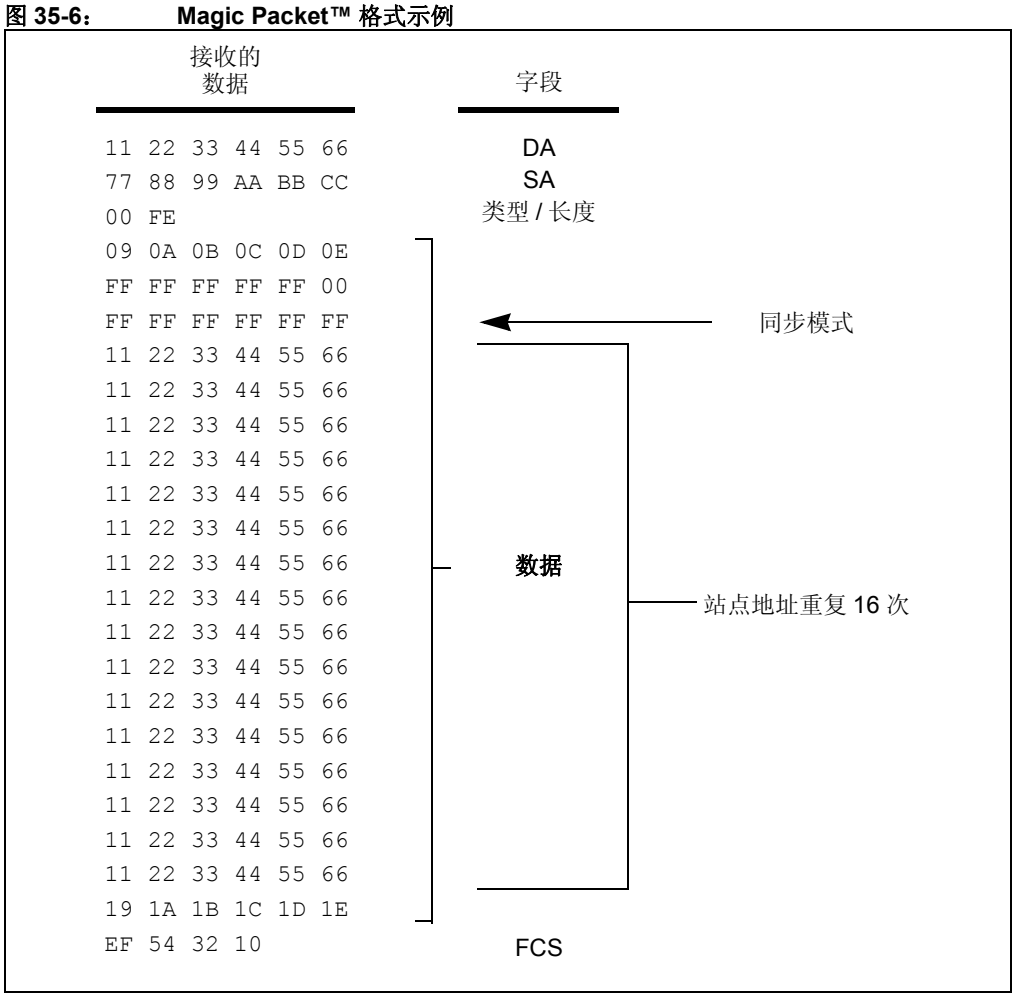
35.4.8.7 MAGIC PACKET 接收过滤器

Magic Packet 过滤器会扫描接收数据包是否存在一种预先确定的模式，以决定是否接收数据包。

Magic Packet 定义为以下形式：数据字段包含同步模式：6 个 0xFFh 字节，后面跟随重复 16 次的目标站点地址。

数据包可以包含除 Magic Packet 模式之外的其他有效载荷。

图 35-6： Magic Packet™ 格式示例



35.4.8.8 模式匹配接收过滤器

在使能时，模式匹配接收过滤器会接收与特定模式匹配的数据包。匹配通过生成一个 64 字节窗口中一些选定字节的校验和来实现。

如果计算得到的校验和等于或不等于 ETHPMCS 寄存器（由 NOTPM（ETXRXFC<12>）位指定），并且满足与 PMMODE（ETHRXFC<8-11>）关联的所有条件，则会接收数据包。否则，数据包会被中止。

64 字节窗口使用 PMO 值（ETHPMO<15:0>）进行设定，从而窗口起始位置可以位于从 0 至 65536 字节的任意位置。但是，如果 64 字节窗口的延展范围超出数据包结束位置，则模式匹配过滤器会中止数据包。

模式匹配屏蔽器位 PMM<63:0>（ETHPMM0<31:0> 和 ETHPMM1<31:0>）用于选择在校验和计算中是否使用 64 字节窗口中的给定字节。如果以太网模式匹配屏蔽器寄存器（ETHPMM0 和 ETHPMM1）中的模式匹配屏蔽器位（PMM<n>）置 1，则在校验和计算中会使用相应的字节（其中， n = 0、 1、 2、 ...、 62、 63， n = 0 指向偏移值之后的第一个字节）。

校验和算法与 TCP/IP 校验和计算相同。请注意，该算法要求计算使用 16 位的字长。这意味着，如果要匹配奇数个字节，则用于计算的数据串的最后一个字节将使用值为 0 的字节填充。此外，在开始计算之前，校验和值会初始化为 0x00000000h。图 35-7 给出了一个校验和计算的示例。

图 35-7： 校验和计算

```
12 00 AC 23 92 55 00 00 FE AA FF FF 34 12 CD AB <-- 数据包 （十六进制）
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 <-- 字节编号
```

步骤 1：将数据包的字相加：

1200h + AC23h + 9255h + 0000h + FEAAh + FFFFh + 3412h + CDABh = 450DEh

checksum_reg[31:0] = 0x0004_50DE

步骤 2a：将 checksum_reg 的高位字与低位字相加：

50DEh + 0004h = 50E2h

checksum_reg[31:0] = 0000_50E2h

步骤 2b：如果来自步骤 2a 的高位字 > 0000h，则将 checksum_reg 的高位字与低位字相加（即，重复步骤 2a）。

步骤 3： NOT(000050E2h) = FFFFAF1Dh

checksum_reg[31:0] = 0000_50E2h

output dma_checksum_val[15:0] = AF1Dh

注 1： 以上计算采用的初始种子为 0000h。

2： 如果 dma_length[DMA_ADDR_MSB:0] = 0，则最终的校验和值必须为校验和种子的补码。对于初始校验和种子 0000h，它将产生 FFFFh。对于用户指定的种子，这会产生相同的种子值，因为在送入 DMA 之前，用户指定的种子值已经是补码。

图 35-8 给出了模式匹配格式示例。

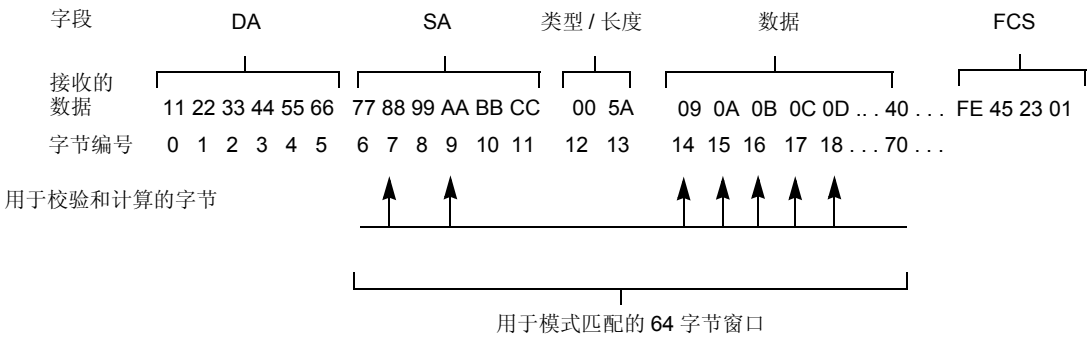
图 35-8： 模式匹配格式示例

输入配置：

模式匹配偏移 = 6h

模式匹配屏蔽器位（PMM<63:0>）= 0000000000001F0Ah

校验和值输出 = 563Fh



用于校验和计算的值 = {88h, AAh, 09h, 0Ah, 0Bh, 0Ch, 0Dh, 00h}

接收的数据以十六进制显示；字节编号以十进制格式显示。

注：硬件填充 00h

例 35-3：　　设置模式匹配接收过滤器

```
// 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 Register 35-4 ETHRXFC: Ethernet Controller
   Receive Filter Configuration Register
- 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　以太网 DMA 和缓冲区管理引擎

为了降低 CPU 在数据存储器和以太网控制器之间移动数据包数据的开销，在模块中集成了接收和发送 DMA 引擎。对于发送数据包，DMA 引擎负责将数据从系统存储器传输到 MAC 中；对于接收数据包，它负责将数据从 MAC 传输到系统存储器中。每个 DMA 引擎都可以通过充当两种不同的总线主器件来访问系统存储器，一种总线主器件用于发送，一种用于接收。

对于接收和发送操作，DMA 引擎会使用独立的以太网描述符表（EDT），用以确定发送 / 接收数据包缓冲区位于系统存储器中的什么位置。

DMA 引擎使用的发送和接收描述符（称为以太网描述符（ED））具有类似的格式，只有状态字的格式不同。表 35-7 和表 35-8 给出了描述符的格式。如图 35-9 和图 35-10 中所示，描述符表可以包含由指向数据包缓冲区的描述符组成的链表或线性表。

在使能以太网传输之前，软件负责设置接收和发送描述符表。

表 35-7：　以太网控制器发送缓冲区描述符格式

偏移	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>																															

注：　以太网描述符的地址必须按 4 字节对齐。

偏移 0

bit 31　**SOP：**数据包起始使能位
1 = 随该数据缓冲区发送数据包起始定界符
0 = 不存在数据包起始定界符

bit 30　**EOP：**数据包结束使能位
1 = 随该数据缓冲区发送数据包结束定界符
0 = 不存在数据包结束定界符

bit 29-27　**用户定义位：**以太网控制器不使用这些位

bit 26-16　**BYTE_COUNT<10:0>：**字节计数位 **(1)**
字节计数代表要对于该描述符发送的字节数。每个描述符条目的有效字节计数为 1-2047。

bit 15-9　**用户定义位：**以太网控制器不使用这些位

bit 8　**NPV：**下一个 ED 指针有效使能位
1 = 该描述符中的 Next_ED 字段指向的下一个描述符
0 = 下一个描述符跟随在该描述符之后 （在存储器中）

bit 7　**EOWN：**以太网控制器所有权位 **(2)**
1 = 以太网控制器拥有 ED 及其对应的数据缓冲区。软件不得修改 ED 或数据缓冲区
0 = 软件拥有 ED 及其对应的数据缓冲区。以太网控制器忽略 ED 中的所有其他字段

bit 6-0　**保留：**保持为 0 ；忽略读操作

注　1：设定 BYTE_COUNT = 0 会导致未定义的行为。
2：该位可以由软件或以太网控制器写入，并且在使能以太网控制器之前，用户应用程序必须先将其初始化为所需的值。

偏移 4

bit 31-0　**DATA_BUFFER_ADDRESS<31:0>：**数据缓冲区地址位
描述符数据缓冲区的起点地址。

表 35-7： 以太网控制器发送缓冲区描述符格式 （续）

偏移 8

bit 31-24 **用户定义位**：以太网控制器不使用这些位

bit 23-20 **保留**：保持为 0 ；忽略读操作

bit 19-0 **TSV<51:32>**：发送状态向量位
发送数据包的状态：
TSV<51> = 发送 VLAN 标记帧
帧的长度 / 类型字段包含的值为 VLAN 协议标识符 0x8100。
TSV<50> = 已应用发送背压
已应用载波侦听式背压。
TSV<49> = 发送暂停控制帧
发送的帧是一个带有有效暂停操作码的控制帧。
TSV<48> = 发送控制帧
发送的帧是一个控制帧。
TSV<47-32> = 在线上发送的总字节数
为当前在线上发送的数据包的总字节数，包括所有冲突尝试的字节。

偏移 12

bit 31-0 **TSV<31:0>**：发送状态向量位
发送数据包的状态：
TSV<31> = 发送数据不足
系统未能向发送 MAC 模块传输完整的数据包。
TSV<30> = 发送超大帧
帧字节计数大于 MACMAXF （EMACxMAXF<0:15>）。
TSV<29> = 发送延迟冲突
在超出冲突窗口 （512 个位时间）之后发生冲突。
TSV<28> = 发送最大冲突
数据包由于冲突数超出 RETX （EMACxCLRT<0:3>）而中止。
TSV<27> = 发送延时过长
数据包的发送延时超出了 6071 个半字节时间 （在 100 Mbps 模式下）或 24,287 个位时间（在 10 Mbps 模式下）。
TSV<26> = 发送数据包延迟
数据包至少延迟了一次，但延时未超出规定值。
TSV<25> = 发送广播
数据包的目标地址是广播地址。
TSV<24> = 发送组播
数据包的目标地址是组播地址。
TSV<23> = 发送完成
数据包发送已经完成。
TSV<22> = 发送长度超出范围
指示帧的类型 / 长度字段大于 1500 字节 （类型字段）。
TSV<21> = 发送长度校验错误
指示数据包中帧长字段的值与实际数据字节长度不匹配，并且不是类型字段。
TSV<20> = 发送 CRC 错误
数据包中的 CRC 与内部生成的 CRC 不匹配。
TSV<19:16> = 发送冲突计数
在尝试发送期间，当前数据包引起的冲突次数。
TSV<15:0> = 发送字节计数
帧中的总字节数，不包括冲突字节。

偏移 16

bit 31-0 **NEXT_ED<31:0>**：下一个以太网描述符地址位
当 NPV = 1 时：该字段包含下一个以太网描述符的起点地址。
当 NPV = 0 时：描述符中不存在该字段。

表 35-8：　以太网控制器接收缓冲区描述符格式

偏移	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>																															

注：　以太网描述符的地址必须按 4 字节对齐。

偏移 0

bit 31　**SOP**：数据包起始使能位
1 = 随该数据缓冲区接收到数据包起始定界符
0 = 不存在数据包起始定界符

bit 30　**EOP**：数据包结束使能位
1 = 随该数据缓冲区接收到数据包结束定界符
0 = 不存在数据包结束定界符

bit 29-27　**保留**：保持为 0；忽略读操作

bit 26-16　**BYTE_COUNT<10:0>**：字节计数位
字节计数代表要对于该描述符发送的字节数。每个描述符条目的有效字节计数为 1-2047。

bit 15-9　**用户定义位**：以太网控制器不使用这些位

bit 8　**NPV**：下一个 ED 指针有效使能位
1 = 该描述符中的 Next_ED 字段指向的下一个描述符
0 = 下一个描述符在存储器中跟随在该描述符之后

bit 7　**EOWN**：以太网控制器所有权位 **(1)**
1 = 以太网控制器拥有 ED 及其对应的数据缓冲区。软件不得修改 ED 或数据缓冲区
0 = 软件拥有 ED 及其对应的数据缓冲区。以太网控制器忽略 ED 中的所有其他字段

bit 6-0　**保留**：保持为 0；忽略读操作

注　1：该位可以由软件或以太网控制器写入，并且在使能以太网控制器之前，用户应用程序必须先将其初始化为所需的值。

偏移 4

bit 31-0　**DATA_BUFFER_ADDRESS<31:0>**：数据缓冲区地址位
描述符数据缓冲区的起点地址。

偏移 8

bit 31-24　**RXF_RSV<7:0>**：接收过滤器状态向量位
该字段携带关于接收数据包过滤的额外信息：
RXF_RSV<7> = 组播匹配
RXF_RSV<6> = 广播匹配
RXF_RSV<5> = 单播匹配
RXF_RSV<4> = 模式匹配匹配
RXF_RSV<3> = Magic Packet 匹配
RXF_RSV<2> = 哈希表匹配
RXF_RSV<1> = NOT(单播匹配) AND NOT(组播匹配)
RXF_RSV<0> = 过短数据包

bit 23-16　**用户定义位**：以太网控制器不使用这些位

bit 15-0　**PKT_CHECKSUM<15:0>**：该描述符数据包的接收数据包有效载荷校验和。计算得到的 16 位数据包校验和值的补码。

表 35-8： 以太网控制器接收缓冲区描述符格式 （续）

偏移 12

bit 31-0 **RSV<31:0>**：接收状态向量位
接收数据包的状态：
RSV<31> = 保留
RSV<30> = 检测到接收 VLAN 类型
确认当前帧为 VLAN 标记帧。
RSV<29> = 接收未知操作码
确认当前帧为控制帧，但它包含未知操作码。数据包没有 CRC 错误，并且具有有效的长度（64-1518）。
RSV<28> = 接收暂停控制帧
确认当前帧为控制帧，其中包含有效暂停帧操作码和有效地址。数据包没有 CRC 错误，并且具有有效的长度 （64-1518）。
RSV<27> = 接收控制帧
确认当前帧为控制帧，因为有效的类型 / 长度字段指明它为控制帧。数据包没有 CRC 错误，并且具有有效的长度 （64-1518）。
RSV<26> = 多余数据
指示在数据包末尾，额外接收到 1 至 7 位数据。已构造了单个半字节数据 （称为多余数据），但未送出。
RSV<25> = 接收广播数据包
指示接收到的数据包具有有效的广播地址。
RSV<24> = 接收组播数据包
指示接收到的数据包具有有效的组播地址。
RSV<23> = 正常接收
指示数据包具有有效的 CRC，并且无符号错误。
RSV<22> = 长度超出范围
指示帧的类型 / 长度字段大于 1500 字节 （类型字段）。
RSV<21> = 长度校验错误
指示数据包中帧长字段的值与实际数据字节长度不匹配，并且指定了一个有效的长度。
RSV<20> = CRC 错误
指示帧 CRC 字段的值与接收器 MAC 计算的 CRC 值不匹配。
RSV<19> = 接收代码违例
指示在一个帧的数据阶段，当 MRXER 置为有效时， MII 数据代表的不是有效的接收代码。
RSV<18> = 先前检测到载波事件
指示在上一个接收统计之后的某个时间，检测并记录了一个载波事件，并在下一个接收统计中报告。载波事件与该数据包无关。载波事件是指接收通道上与数据包接收尝试操作无关的活动。
RSV<17> = 先前检测到 RXDV 事件
指示上一次检测到的接收事件长度不够，不足以构成一个有效的数据包。
RSV<16> = 长事件 / 丢弃事件
指示数据包接收时间超过 50,000 个位时间，或上次 RSV 之后丢弃了数据包。
RSV<15:0> = 接收字节计数
指示接收帧的长度。

偏移 16

bit 31-0 **NEXT_ED<31:0>**：下一个以太网描述符地址位
当 NPV = 1 时：该字段包含下一个以太网描述符的起点地址。
当 NPV = 0 时：描述符中不存在该字段。

图 35-9： 以太网描述符表格式（链表， NPV = 1）

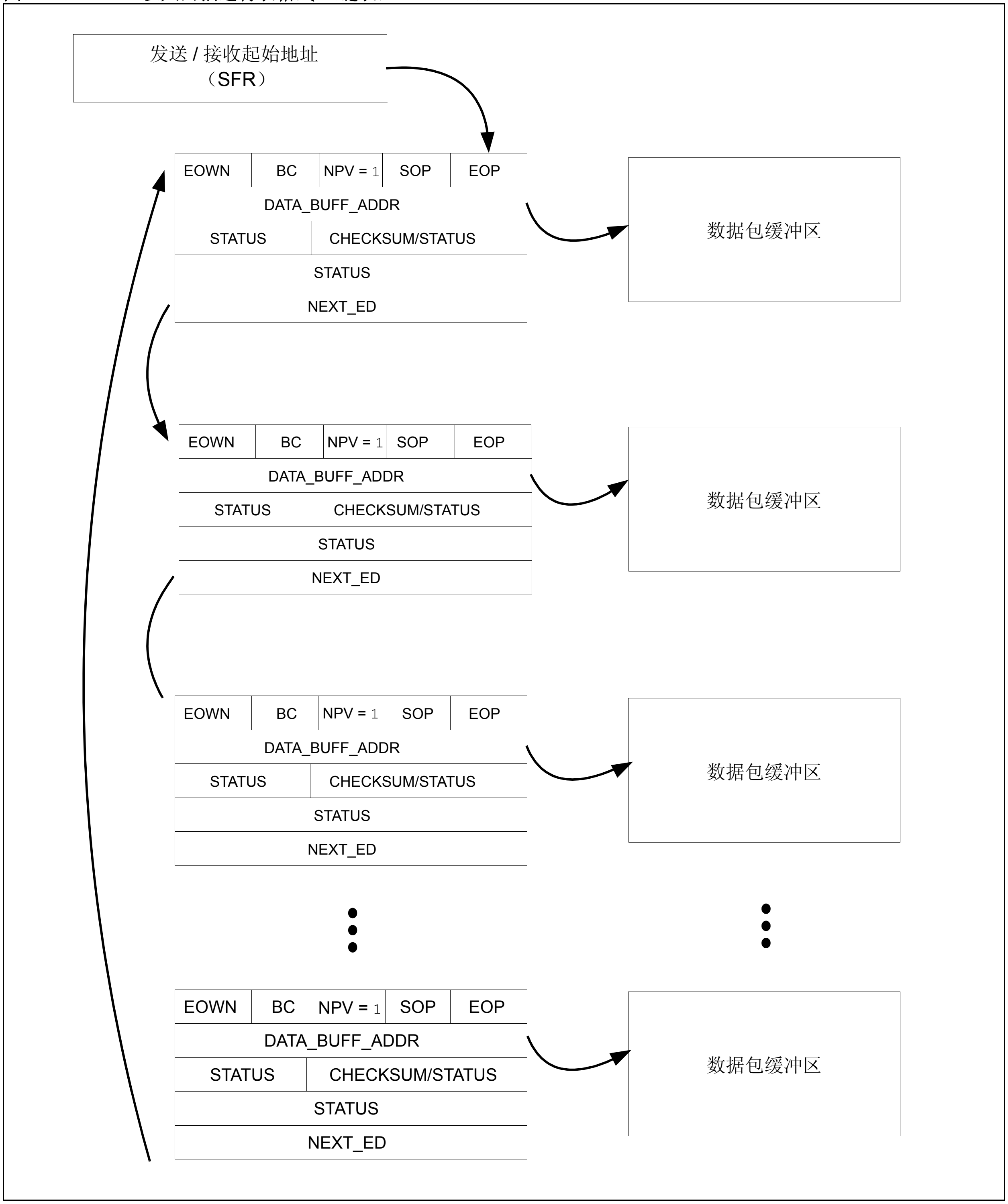
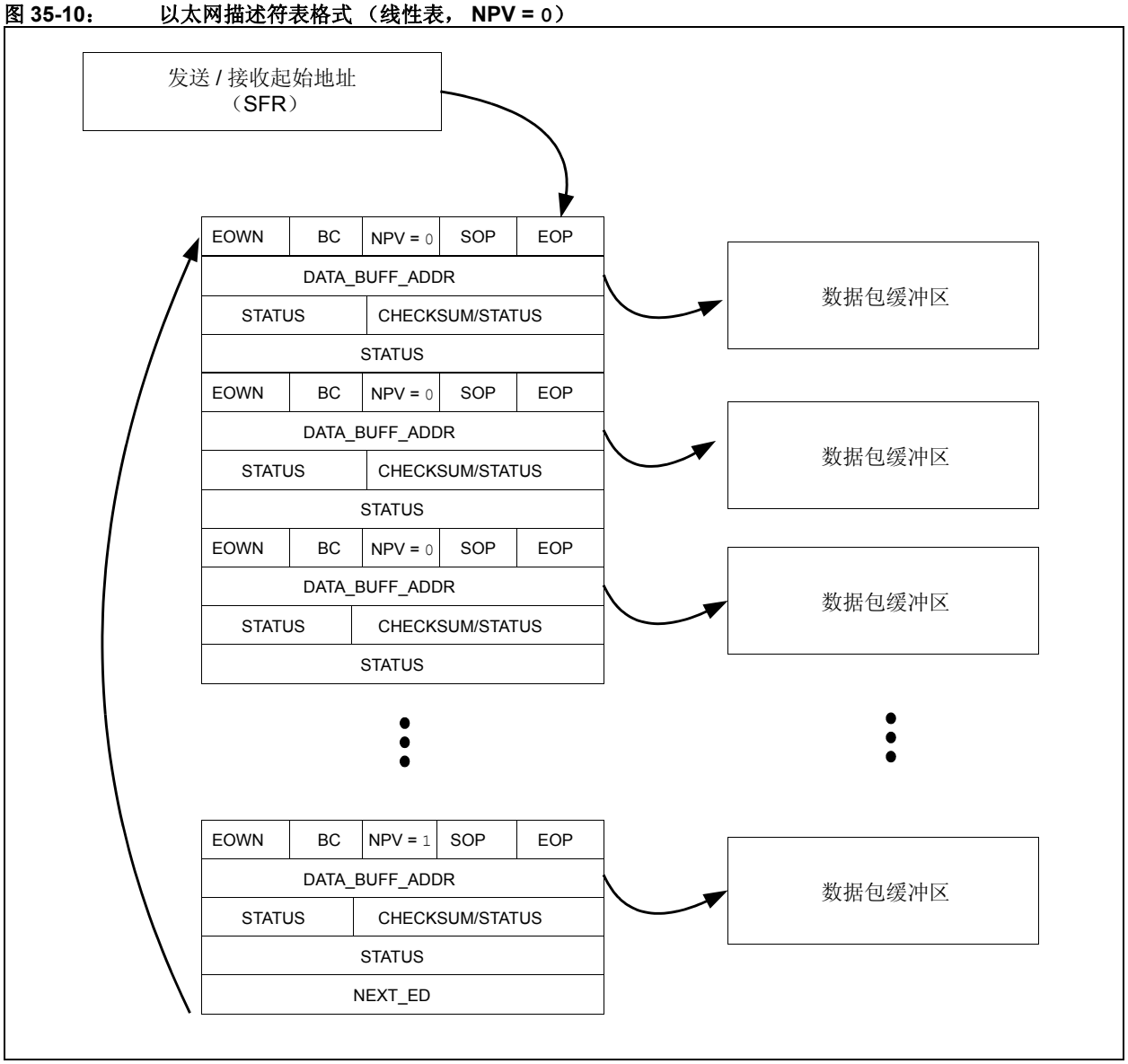


图 35-10： 以太网描述符表格式 （线性表， NPV = 0）



35.4.9.1 以太网描述符缓冲区管理

接收和发送路径使用的描述符表和数据包数据缓冲区位于系统数据存储器中。描述符表是一个由描述符组成的链表，这些描述符指向存储器中的数据包数据缓冲区块。它们按物理方式和逻辑方式划分为独立的发送和接收描述符和缓冲区。请注意，接收和发送描述符表的起始地址都必须按 4 字节对齐 （即， bit 1 和 bit 0 = 00）。

35.4.9.2 以太网描述符所有权

由于描述符和缓冲区是在软件和以太网控制器之间共用的，因此采用了简单的信号量机制来区分允许软件还是以太网控制器更新存储器中的描述符和相关缓冲区。这种信号量机制通过每个缓冲区描述符中的 EOWN 位实现。当 EOWN 位清零时，描述符由软件拥有。软件可以自行修改描述符。当 EOWN 位置 1 时，描述符（以及描述符指向的缓冲区存储器）由以太网控制器硬件拥有。此时软件不能修改描述符或其相应的数据缓冲区。而以太网控制器则可以自行写入缓冲区描述符和相应的数据缓冲区。

35.4.9.3 以太网描述符表配置

在使能以太网控制器进行传输之前，软件必须先设置发送和接收描述符表，以及分配描述符所指向的数据包数据缓冲区。每个表中的描述符条目数量由软件和系统中的可用存储器量决定。

软件可以设置两种不同类型的以太网描述符表 （EDT）。一种配置是描述符环形表；另一种是描述符列表。两者的唯一区别在于表中的最后一个描述符是 EOWN 位 = 0 的 “空”描述符，还是最后一个描述符指向环形表顶部。以太网 DMA 引擎均可处理这两种类型。图 35-11 给出了这两种类型的示例。

注：	软件负责初始化 EDT 中每个描述符中的所有字段，包括将状态字段初始化为 0。关于以太网描述符格式的详细说明，请参见表 35-7 和表 35-8。

35.4.9.4 以太网发送缓冲区管理

带发送 DMA 的发送缓冲区管理 （TXBM）模块通过使用发送缓冲区描述符来管理从系统存储器到 MAC 的发送数据包传输。

通过将 TXRTS 位 （ETHCON1<9>）置 1 来使能发送操作。在 TXRTS 位置 1 时，发送 DMA 会从 ETHTXST 指向的 EDT 中取出一个以太网描述符。在它读取数据缓冲区的位置和数据缓冲区的控制字之后，DMA 会开始读取数据缓冲区数据，并将数据写入 MAC 的发送端口。如果需要多个描述符， DMA 会移到下一个描述符并继续发送数据。

注：	软件必须确保在将 TXRTS 位置 1 之前，先初始化发送描述符列表和 ETHTXST 寄存器。

当必须将某个数据包从系统存储器发送到 MAC 时，需要从描述符 DATA_BUFFER_ADDRESS 指向的存储器读取数据包数据 （见图 35-9）。

发送数据包的格式与接收数据包的格式相同。每个发送数据包缓冲区都包含要发送的标准以太网帧字段 （DA、 SA、类型 / 长度和有效载荷）。

图 35-12 给出了一个使用 3 个描述符的发送数据包的示例。

在软件设定描述符表条目和数据包数据缓冲区之后，可以通过将 TXRTS（ETHCON1<9>）位置 1 来启动发送操作。

注：	软件应从待发送数据包的最后一个描述符开始，到第一个描述符结束，将发送描述符中的 EOWN 位置 1。这可以防止在软件和以太网控制器硬件之间出现任何竞争条件。

发送一个完整数据包之后，以太网控制器会更新用于该数据包的第一个描述符条目中的发送状态向量（TSV）。对于发送数据包使用的所有描述符， EOWN 字段都会发生更新。此外，硬件还会更新 ETHTXST 寄存器，以反映要用于下一个发送数据包的下一个发送描述符。

只要数据包存在有效的发送描述符 （下一个描述符有效， EOWN = 1），发送 DMA 就会继续遍历描述符表并向 MAC 发送要发送的数据包数据。当所有发送操作都完成时，发送逻辑会清零 TXRTS 位并将 TXDONE （ETHIRQ<3>）位置 1。如果 TXDONEIE （ETHIEN<3>）位置 1，则 TXDONE 位会产生中断。因而，可以通过查询 TXRTS 位或使用 TXDONE （ETHIRQ<3>）中断来监视发送操作是否完成。

在发送数据包期间的任意时刻，都可以通过清零 TXRTS 位来停止发送操作。在数据包发送期间清零 TXRTS 位时，当前数据包会完成发送，之后 TXBUSY （ETHSTAT<6>）状态位会清零，指示发送引擎已停止。

在 TXRTS 位置 1 时，软件不应写入任意与发送相关的 SFR 寄存器。要更改这些寄存器，必须先清零 TXRTS，然后软件必须等待 TXBUSY 置为无效，之后才能写入这些寄存器，然后再次将 TXRTS 位置 1。

注：	ETHTXST 发送配置寄存器（发送描述符表的起始地址）由发送 DMA 使用，只有在发送 DMA 不工作（即，TXRTS（ETHCON1<9>）= 0 且 TXBUSY（ETHSTAT<6>）= 0）时才能进行更改，因为并未提供与 DMA 时钟域进行连续同步的功能。 当发送 DMA 处于活动状态时，软件需要确保 ETHTXST 配置寄存器不发生改变。

35.4.9.5 以太网发送操作细节

数据包发送过程如下：

1. 软件将数据包数据写入数据存储器中的数据包缓冲区，并将某个描述符条目设定为指向数据包数据缓冲区。它还需要设定 SOP、EOP、BYTE_COUNT 和 EOWN 位，以指示存在有效的数据包，以及发送完整数据包是否还需要其他描述符。关于使用多个描述符条目的发送数据包的示例，请参见图 35-12。描述符用于定义发送数据包数据缓冲区的位置和长度。
2. 然后，软件通过将 TXRTS （ETHCON1<9>）位置 1 来启动发送，这会启动发送 DMA 和 TXBM 逻辑。
3. 发送 DMA 引擎将会从 ETHTXST 寄存器指向的描述符表中读取下一个可用的描述符条目。
4. 在发送 DMA 引擎读取 DATA_BUFFER_ADDRESS 值之后，引擎会开始从描述符中读取的位置读取数据包数据。
5. TXBM 向 MAC 指示帧起始，并发送来自发送缓冲区的全部帧数据，直到达到结束地址为止。 TXBM 只需从起始地址开始发送，直到向 MAC 发送接口发送的字节数达到指定字节数为止。
6. MAC 可能会由于过早冲突而重新尝试发送。
7. MAC 可能会由于延迟冲突、冲突过量和延时过长而中止发送。该条件通过 TXABORT （ETHIRQ<2>）指示。
8. 完成发送之后，发送 DMA 引擎会将 TSV 的相关位存储到数据包的第一个描述符条目中。
9. 发送数据包之后，硬件会将用于发送的所有描述符的 EOWN 位清零，将这些描述符释放给软件。
10. 如果还存在其他有效发送描述符（EOWN = 1），DMA 引擎会回到步骤 3，开始下一个数据包的发送。否则，如果下一个描述符仍然由软件拥有 （EOWN = 0），则发送会暂停，并等待软件再次将 TXRTS 位置 1。

请注意，在 CWINDOW （EMACxCLRT<8:14>）边界内发生的所有冲突都是过早冲突，会导致重试操作。在 CWINDOW 边界之外发生的冲突会被视为延迟冲突，并导致中止。 EMACxCLRT 寄存器中的 CWINDOW 位通常设置为 64 字节。对于尝试发送的数据包，达到最大冲突计数 RETX （EMACxCLRT<0:3>）也会导致中止条件。

TXBM 引擎几乎没有关于它发送给 MAC 的数据内容的信息。但是，应注意它可以发送的两种发送数据包：

1. 完整数据包，它包括以下内容：
   - 地址 （DA 和 SA）、类型 / 长度和有效载荷
   - 填充 （如果需要）
   - 帧校验序列

   这种情况下， PADENABLE （EMACxCFG2<5>）位为 0。 MAC 不会对帧进行填充，但会对帧执行 CRC 校验，如果 CRC 校验匹配失败，则将发送状态向量中的 TSV<20> 置 1。
2. 不完整数据包，它包括以下内容：
   - 地址 （DA 和 SA）、类型 / 长度和有效载荷

   这种情况下， PADENABLE = 1。MAC 会对帧进行填充（根据 AUTOPAD 和 VLANPAD （EMACxCFG2<7:6>）的设置），并插入对于帧计算得到的 CRC （FCS）。

关于基于 EMACxCFG2 寄存器设置的填充和 CRC 选项的说明，请参见表 35-2。

例 35-4：　以太网发送数据包代码

```
/*
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 for this example (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 intnext_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;
}
```

例 35-4：　　以太网发送数据包代码 （续）

```
// 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
```

注： 该代码示例使用了 MPLAB® C32 C 编译器的特定语法。关于对打包数据结构的支持，请参见编译器手册。

35.4.9.6 以太网接收缓冲区管理

带接收 DMA 的接收缓冲区管理 （RXBM）模块通过使用接收缓冲区描述符来管理从 MAC 到系统存储器的接收数据包传输。

通过将 RXEN 位 （ETHCON1<8>）置 1 来使能接收操作。在 RXEN 位置 1 之后，接收 DMA 会通过读取表中的下一个可用描述符条目来响应传入的数据包，并将数据包数据写入描述符指向的数据包缓冲区，将接收数据包状态写入描述符条目自身中。如果传入数据包需要的空间大于单个缓冲区分配的空间，则数据包可能会横跨多个描述符。如果接收 DMA 读取了表中的下一个数据包描述符，但并不拥有它，则这可能是溢出条件，将通过状态寄存器进行报告。

注： 当第一次使能 RXEN 位时， RXDMA 引擎会在准备接收数据包数据时从存储器中取回一个接收描述符。软件必须确保在将 RXEN 位置 1 之前先初始化描述符列表。

当成功接收到数据包时 （数据包未被过滤器中止或由于溢出错误而中止），数据包数据会被存储到描述符 DATA_BUFFER_ADDRESS 指向的存储器中（见图 35-9）。在用于该数据包的第一个描述符条目中，接收状态向量 （RSV） 、接收过滤器接收状态 （RXF_RSV） 、数据包校验和（PKT_CHECKSUM）和控制字 （SOP 和 EOP）会发生更新。对于接收数据包使用的所有描述符， EOWN 字段都会发生更新。此外，为了改善数据包管理， BUFCNT （ETHSTAT<16:23>）位会保存已存储到数据存储器中的接收数据包缓冲区数量的动态计数。

最后， ETHRXST 寄存器会被更新，以反映要用于下一个接收数据包的下一个接收描述符。在硬件将数据包存储到存储器中之后，软件负责在处理数据包之前检查数据包的 RSV，确定是否存在错误。

在软件处理一个接收数据包之后，应对于用于该数据包的每个描述符写入 BUFCDEC（ETHCON1<0>）位一次，以递减数据包缓冲区计数 BUFCNT。这可以提供数据存储器中等待处理的未处理数据包缓冲区的准确计数，用于自动流量控制 （见**第 35.4.7 节 “流量控制概述”**）。

软件还应更新用于该数据包的描述符并将描述符中的 EOWN 字段置 1，以释放它们，使它们可供其他接收数据包使用。

当使能自动流量控制时，如果接收逻辑接收到的数据包数量大于 ETHSTAT 寄存器中 BUFCNT 位可以反映的最大数量，则会发生溢出条件。如果尝试递增 BUFCNT 字段（由 RXBM 在接收到数据包时执行），而寄存器已达到它的最大值 （0xFF），寄存器不会发生计满返回；此时会产生溢出错误条件，接收逻辑会暂停。处理这种情况的正确方式是读出数据包并递减 BUFCNT 计数器。

如果禁止了自动流量控制，则 RXDMA 会继续进行处理， BUFCNT 将一直为饱和值 0xFF。

如果接收引擎由于缺少可用描述符而停止，则只有在检测到 BUFCDEC （ETHCON1<0>）位发生写操作时，它才会再次启动。该操作指示软件已释放了另外的接收描述符缓冲区。

注 1： ETHRXST 接收配置寄存器 （接收描述符表的起始地址）由接收 DMA 使用，只有在接收 DMA 不工作（即，RXEN（ETHCON1<8>）= 0 且 RXBUSY（ETHSTAT<5>）= 0）时才能进行更改，因为并未提供与 DMA 时钟域进行连续同步的功能。
当接收 DMA 处于活动状态时，软件需要确保 ETHRXST 配置寄存器不发生改变。

2： 当使用接收 EDT 环形表时，软件必须确保接收 EDT 包含 （至少）缓冲最大以太网帧所需的足够条目。要求这一点是因为接收 DMA 引擎不会检测折回条件。

35.4.9.7 以太网接收操作细节

接收数据包存储在数据包缓冲区中，它带有一个状态向量，该状态向量存储在描述符中。状态向量由两方面组成：

- 接收状态向量 （RSV），它由 MAC 在接收数据包结束时驱送，其中包含关于接收数据包的信息。
- 接收过滤器状态向量 （RXF_RSV），由接收过滤器模块驱送。

该组合状态存储在用于存储数据包数据缓冲区的第一个描述符中。

接收过程中涉及的步骤序列如下：

1. 软件使用该信息设置接收描述符表：接收描述符条目和每个描述符条目所指向的关联数据包数据缓冲区。
2. 软件写入ETHRXST寄存器（指向接收描述符表的起始位置），并通过将RXEN（ETHCON1<8>）位置 1 来使能接收端口。
3. 接收 DMA 引擎从表中读取有效的描述符，并确定数据包数据缓冲区的起始位置。
4. 当接收到数据包时， MAC 指示新帧开始并送出数据。
5. RXBM 接收数据并将它存储在接收 FIFO 中。对系统存储器的写操作会被推迟到接收到 32 位数据时。接收 DMA 从接收 FIFO 中获取数据，并将它存储到刚刚读取的描述符指向的数据包数据缓冲区中。写入描述符所分配的所有字节之后，接收 DMA 会读取另一个描述符，以写入更多数据包数据。
6. 如果在需要存储更多数据包数据时，接收 DMA 获取到 EOWN = 0 的描述符，则会产生 RXBUFNA （ETHIRQ<1>）中断。
7. 如果在数据包接收期间发生以下任意事件，数据包会被中止，并且不会使用数据包状态更新描述符，这会使它可供下一个数据包使用：
   - 接收过滤器中止了数据包
   - RXFIFO 发生溢出，原因有：
     - 系统级别的延时过长
     - BUFCNT （ETHSTAT<16:23>）达到最大值
     - 没有可用于硬件处理的描述符
8. 在帧结束之后， MAC 会提供接收状态向量 （RSV），接收过滤器会提供接收过滤器状态向量 （RXF_RSV）。
9. 接收 DMA 将再次遍历描述符：
   a) 使用 RSV、 RXF_RSV、 PKT_CHECKSUM 和 BYTE_COUNT 值更新用于当前数据包的第一个描述符。
   b) 所有属于当前数据包的描述符的 SOP 和 EOP 会发生更新。此外， EOWN 位也会发生更改，指示软件可以读取数据包数据。
10. 将 RSV 传递给接收 DMA 之后， RXBM 就可以开始接收另一个数据包。

例 35-5：　　以太网接收数据包代码

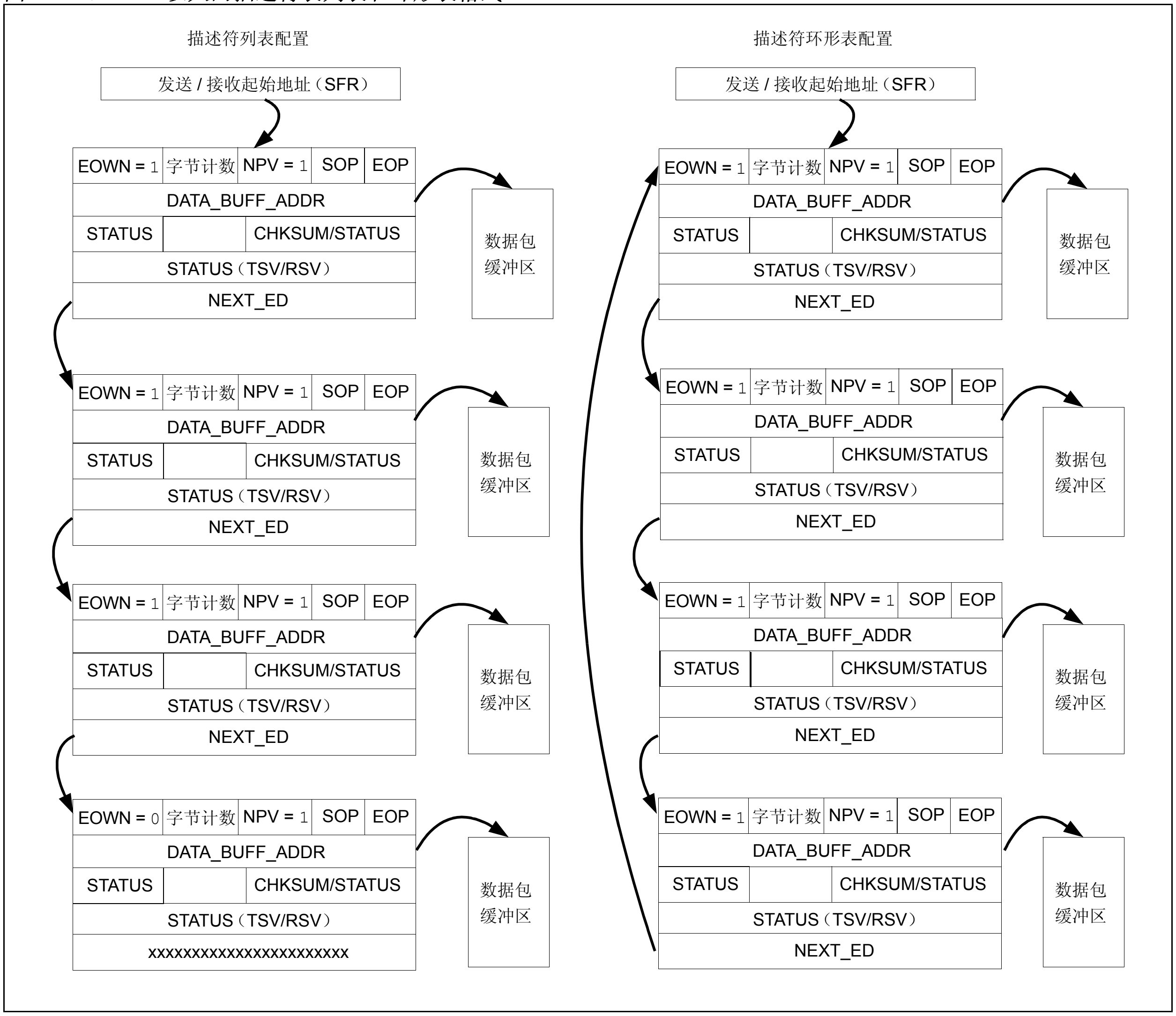
```
/* The following assumptions were made for this example:
- the packet that has to be received might consist of multiple Ethernet frames.
- the number of available RX descriptors is greater than the number of receive buffers that
  have to be made available for the incoming frames (there's at least 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: 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 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;
}
```

例 35-5： 以太网接收数据包代码 （续）

```
// 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
```

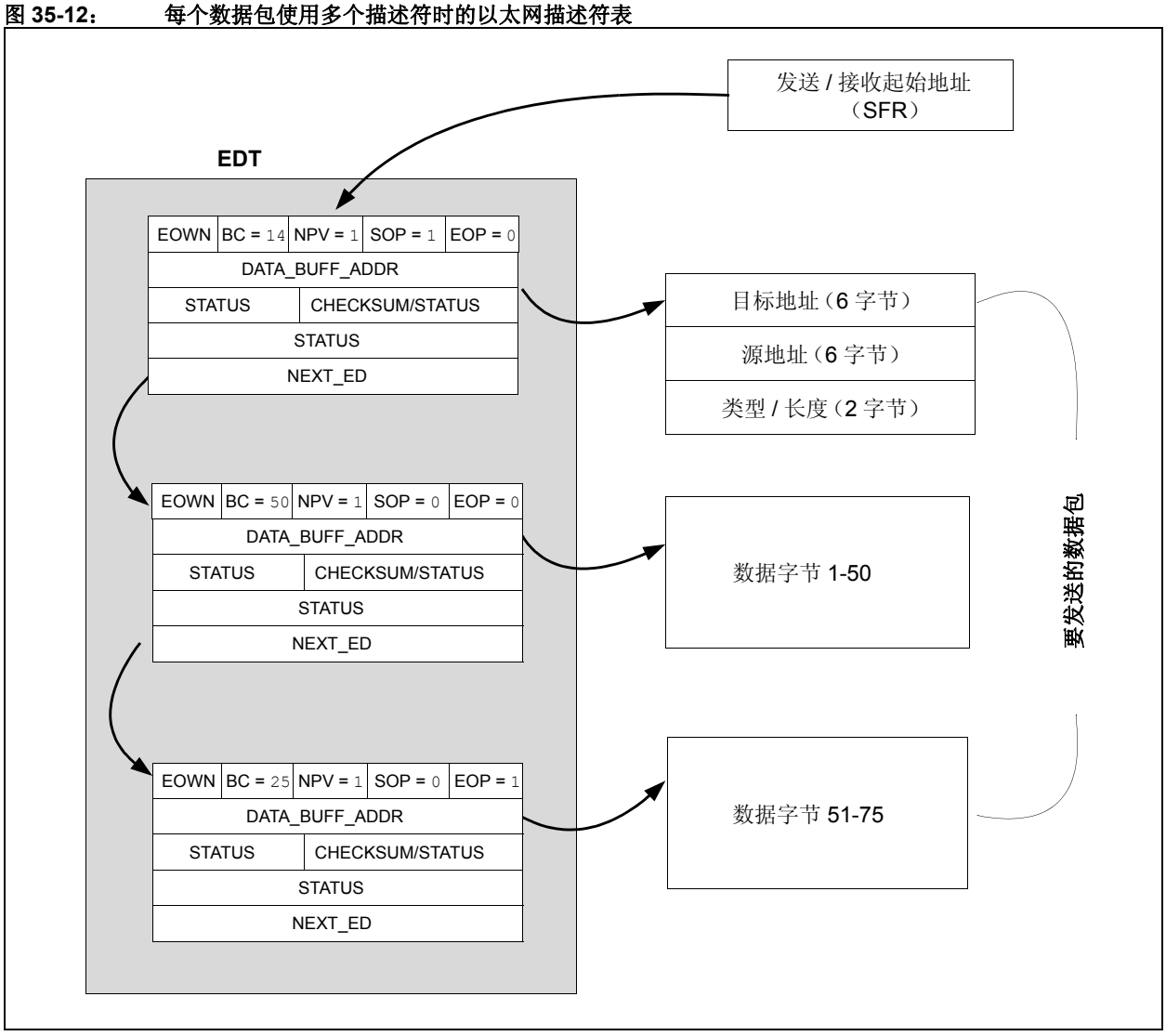
注： 该代码示例使用了 MPLAB C32 C 编译器的特定语法。关于对打包数据结构的支持，请参见编译器手册。

图 35-11： 以太网描述符表列表和环形表格式



 初稿

图 35-12： 每个数据包使用多个描述符时的以太网描述符表



35.4.10 以太网初始化序列

为了初始化以太网控制器来接收和发送以太网报文， Microchip 建议执行以下步骤序列：

1. 以太网控制器初始化：
   a) 通过在 EVIC 中清零 ETHIE 位 （IEC1<28>）来禁止以太网中断。
   b) 关闭以太网控制器，然后清零 ON、RXEN 和 TXEN 位 （ETHCON1<15>、ETHCON1<8> 和 ETHCON1<9>）。
   c) 通过查询 ETHBUSY 位 （ETHSTAT<7>）来等待活动中止。
   d) 清零中断模块中的以太网中断标志位 ETHIF （IFS1<28>）。
   e) 通过清零 ETHIRQIE 寄存器来禁止产生任何以太网控制器中断。
   f) 通过使用 ETHTXSTCLR 和 ETHRXSTCLR 清零发送和接收起始地址。
2. MAC 初始化：
   a) 使用 SOFTRESET （EMACxCFG1<15>）复位 MAC，或者通过将 RESETRMCS、RESETRFUN、RESETTMCS 和 RESETTFUN 位（EMACxCFG1<11:8>）置 1 来分别复位各个模块。
   b) 使用配置熔丝设置 FETHIO （DEVCFG3<25>）来检测备用 / 默认 I/O 配置 （详情请参见**第 32 章 “配置”**）。
   c) 使用配置熔丝设置 FMIIEN （DEVCFG3<24>）来检测 MII/RMII 工作模式。
   d) 将 MAC/PHY 接口使用的所有引脚正确初始化为数字引脚（通常只需要配置具有共用模拟功能的那些引脚）。
   e) 初始化 MIIM 接口：
      i. 如果选择了 RMII 工作模式，则通过使用 RESETRMII （EMACxSUPP<11>）位复位 RMII 模块，并在 SPEEDRMII （EMACxSUPP<8>）位中设置适当的速度。
      ii. 通过将 RESETMGMT （EMACxMCFG<15>）位置 1 来发出 MIIM 模块复位，然后清零复位位。
      iii. 根据系统运行时钟频率和外部PHY支持的时钟，在CLKSEL（EMACxCFG<5:2>）位中为 MIIM PHY 通信选择适当的分频比。
3. PHY 初始化：

   这取决于实际使用的外部 PHY。所有 PHY 都应实现在 IEEE 802.3 标准的条款 22、表 22-6 “MII Management Register Set”中描述的基本寄存器集。

   除了基本寄存器集之外，PHY 还可能提供一组 9 个寄存器的扩展集和一些功能，并可通过 MIIM 接口访问和控制它们。它们提供特定于 PHY 的标识符、用于自动协商过程的控制和监视功能等。

   IEEE 802.3 标准提供了 16 个扩展寄存器的空间，它们用于实现特定于供应商的功能。

   以下列出了应执行的通用初始化步骤。请参见特定于供应商的 PHY 数据手册，进行相应调整。

   a) 复位 PHY （使用控制寄存器 0）。
   b) 设置 MII/RMII 工作模式。这通常需要访问特定于供应商的控制寄存器。
   c) 设置正常、交换或自动（首选）MDIX。这通常需要访问特定于供应商的控制寄存器。
   d) 通过查看状态寄存器 1 来检查 PHY 功能。
   e) 如果PHY支持，应首选自动协商方式。展示支持的功能 半/全双工、10BaseT/100BaseTX 等 （扩展寄存器 4）。启动协商 （控制寄存器 0），并等待协商完成，然后获取链路对端功能 （扩展寄存器 5）和协商结果 （特定于供应商的寄存器）。
   f) 如果不支持 / 未选择自动协商，则直接更新 PHY 双工和速度设置（使用控制寄存器 0，可能还有一些特定于供应商的寄存器）。

4. MAC 配置：

   提供双工和速度设置之后，使用以下步骤相应地配置 MAC：

   a) 使能 RXENABLE （EMACxCFG1<0>）位，同时选择 TXPAUSE 和 RXPAUSE （EMACxCFG1<3,2>）（PIC32MX MAC 同时支持两者）。
   b) 选择所需的自动填充和 CRC 功能，然后在 EMACxCFG2 寄存器中使能超大帧和双工类型。
   c) 使用 EMACxIPGT 设置背靠背包间间隔。
   d) 使用 EMACxIPGR 设置非背靠背包间间隔。
   e) 在 EMACxCLRT 中设置冲突窗口和最大重发次数。
   f) 在 EMACxMAXF 中设置最大帧长。
   g) （可选）在 EMACxSA0、 EMACxSA1 和 EMACxSA2 寄存器中设置站点 MAC 地址（在复位时，这些寄存器中会装入出厂时预先设定的站点地址）。

5. 继续以太网控制器初始化：

   a) 如果计划开启流量控制，则需更新 PTV 值 （ETHCON1<31:16>）。
   b) 如果要使用自动流量控制，则需设置满水印和空水印：RXFWM 和 RXEWM （ETHRXWM<23:16> 和 ETHRXWM<7:0>）。
   c) 如果需要，通过将 AUTOFC （ETHCON1<7>）置 1 来使能自动流量控制。
   d) 通过更新 ETHHT0、ETHHT1、ETHPMM0、ETHPMM1、ETHPMCS 和 ETHRXFC 寄存器来设置接收过滤器。
   e) 在 RXBUFSZ（ETHCON2<10:4>）位中设置接收缓冲区的大小（所有接收描述符使用相同的缓冲区大小）。请记住，使用太小的数据包会导致数据包分段，会对性能产生明显影响。
   f) 准备要用于发送报文的发送描述符列表 / 环形表。正确更新发送描述符中的所有字段（NPV、EOWN = 1 和 NEXT_ED）（请参见表 35-7 “以太网控制器发送缓冲区描述符格式”）。如果要使用列表，则将它正确设置为以软件控制的描述符（EOWN = 0）结尾。

      当必须发送特定报文时，SOP、EOP、DATA_BUFFER_ADDRESS 和 BYTE_COUNT 会被更新。 DATA_BUFFER_ADDRESS 将包含报文的物理地址， BYTE_COUNT 则包含报文大小。 SOP 和 EOP 根据发送报文需要多少数据包而设置。

   g) 准备接收描述符列表，列表中填入要接收报文的有效缓冲区。正确更新接收描述符中的 NPV、 EOWN = 1 和 DATA_BUFFER_ADDRESS 字段（请参见表 35-8 “以太网控制器接收缓冲区描述符格式”）。DATA_BUFFER_ADDRESS 应包含相应接收缓冲区的物理地址。
   h) 接收/发送描述符和接收逻辑预分配缓冲区的实际数量取决于实际可用的系统存储器，以及您预取并希望处理的以太网通信量。
   i) 使用发送描述符列表头部的物理地址更新 ETHTXST 寄存器。
   j) 使用接收描述符列表头部的物理地址更新 ETHRXST 寄存器。
   k) 通过将 ON 位 （ETHCON1<15）置 1 来使能以太网控制器。
   l) 通过将 RXEN 位 （ETHCON1<8>）置 1 来使能报文接收。
   m) 检查接收描述符列表，确定 EOWN 位是否清零。如果清零，则该描述符现在由软件控制，说明已接收到一个报文。使用 SOP 和 EOP 提取报文，使用 BYTE_COUNT、RXF_RSV、 RSV 和 PKT_CHECKSUM 来获取报文特征。

n) 要发送报文，需要执行以下步骤：
   i. 首先确保报文具有对应于以太网规范的正确格式。
   ii. 从列表头部开始，通过将 DATA_BUFFER_ADDRESS 设置为待发送报文中相应缓冲区的物理地址，更新所需的发送描述符数量。
   iii. 请记住，大数据包分段会对性能产生影响。
   iv. 对于每个描述符，使用每个缓冲区中包含的字节数更新 BYTE_COUNT。
   v. 对于属于数据包的每个描述符，设置 EOWN = 1。
   vi. 使用 SOP 和 EOP 来指定报文使用一个或多个发送描述符。
o) 使能报文发送，将 TXRTS （ETHCON1<9>）位置 1。
p) 检查发送描述符列表，确定 EOWN 位是否清零。如果清零，则该描述符现在由软件控制，说明已发送了一个报文。使用 TSV 来检查发送结果。

35.4.11 以太网统计寄存器

为了符合 802.3 层管理规范，以太网控制器在硬件中实现了各种统计寄存器。在发送 / 接收数据包中检测到各种条件时，硬件会递增这些寄存器。如果某个寄存器已达到它的最大值，在下一次递增时它会计满返回为全 0。因此，软件负责及时地读取这些寄存器，以避免丢失任何数据。

软件读操作会自动导致相应的寄存器清零。

软件可以通过使用置 1、清零和取反寄存器来写入统计计数器。此外也支持写入普通寄存器。在正常工作时，应当通过定期读取统计寄存器来收集关于以太网链路通信的数据。

注 1： 置 1、清零和取反统计寄存器仅用于支持软件调试和测试。

2： 将器件置为 Sleep （休眠）模式时，因为系统时钟不运行，统计寄存器的更新会被暂停。它的唯一例外是溢出条件，溢出条件会使溢出计数器递增，并使溢出标志 RXOVFLW （ETHIRQ<0>）位置 1。这么做是为了在唤醒时通知软件，有一些数据包已丢失。

3： 在由于待处理事件而退出 Sleep（休眠）模式时，一些统计计数器可能会立即递增。

35.4.11.1 有效载荷校验和计算

对于所有接收数据包，以太网控制器会自动计算一个 16 位的数据包校验和，并将该 16 位值与接收数据包状态向量一起存储在数据包描述符的 PKT_CHECKSUM 字段 （RX_DCPT<96:111>）中。该校验和对于 TCP/IP 软件实现很有用。

有效载荷校验和的计算范围为除前 14 字节 （目标地址、源地址和长度 / 类型字段）之外的完整接收数据包。如果软件需要从校验和中排除更多字节，它必须减去相应的值。

有效载荷校验和是用于防止发送期间位损坏的基本保护。在 TCP/IP 协议的 TCP 和 UDP 数据包中通常会使用它。校验和的计算方式是将字节流除以一些 16 位字，并将它们相加在一起。任何溢出都会再次与和相加。校验和计算从接收帧的前 14 字节之后开始，包括 FCS 字节。结果是计算得到的和的补码。

关于有效载荷校验和计算的示例，请参见图 35-7。

35.5 以太网中断

PIC32MX 器件能够产生一些中断，以反映在以太网控制器的帧传输期间发生的事件。每个以太网控制器中断事件都在 ETHIEN 寄存器中具有相应的中断允许位，只有该位置 1 时，才会产生中断。但是，无论 ETHIEN 寄存器的值如何，所有中断事件的状态都可以直接通过 ETHIRQ 寄存器读取。因此，软件可以通过查询该寄存器来确定会产生潜在中断的事件，不让中断传出模块。

以太网中断是持久性中断。这意味着，只要产生中断的事件等待处理，来自以太网控制器模块的中断信号就会保持有效。

以下描述了由以太网帧发送和接收产生的中断事件。

与发送路径相关的中断事件：

- 发送 DMA 引擎传输错误中断，通过 TXBUSE （ETHIRQ<14>）位指示，并使用 TXBUSEIE （ETHIEN<14>）位来允许该中断。当发送 DMA 在存储器访问期间遇到总线错误，并且是由于寻址错误 （通常是由于指针有错）而导致时，会发生该事件。
- 发送完成中断，通过 TXDONE （ETHIRQ<3>）位指示，并使用 TXDONEIE （ETHIEN<3>）位来允许该中断。在当前发送的发送数据包完成发送，并且发送状态向量装入数据包的第一个描述符中时，会发生该事件。
- 发送中止中断，通过 TXABORT （ETHIRQ<2>）位指示，并使用 TXABORTIE （ETHIEN<2>）位来允许该中断。在 MAC 由于以下原因之一而中止发送时，会发生该事件：
  - 超大发送数据包中止 （数据包的大小大于最大的 MACMAXF （EMACxMAXF<15:0>））
  - 数据不足中止 （发送引擎无法跟上所请求的数据流。这通常在系统总线超载时发生。）
  - 延时过长中止 （数据包的发送延时超出了 6071 个半字节时间 （在 100 Mbps 模式下）或 24,287 个位时间 （在 10 Mbps 模式下））
  - 延迟冲突中止 （在冲突窗口之外发生冲突）
  - 冲突过量中止 （数据包由于冲突数超出 RETX （EMACxCLRT<3:0>）而中止）

注：	过早冲突会导致 MAC 将重新尝试发送，而不是中止发送。因此，该条件不会导致中断。

与接收路径相关的中断事件：

- 接收 DMA 引擎传输错误中断，通过 RXBUSE （ETHIRQ<13>）位指示，并使用 RXBUSEIE （ETHIEN<13>）位来允许该中断。当接收 DMA 在存储器访问期间遇到总线错误，并且是由于寻址错误 （通常是由于指针有错）而导致时，会发生该事件。
- 接收完成中断，通过 RXDONE （ETHIRQ<7>）位指示，并使用 RXDONEIE （ETHIEN<7>）位来允许该中断。每次成功接收到一个数据包时，会发生该事件。
- 数据包待处理中断，通过 PKTPEND （ETHIRQ<6>）位指示，并使用 PKTPENDIE （ETHIEN<6>）位来允许该中断。每当缓冲区计数器 BUFCNT （ETHSTAT<16:23>）的值大于 0 时，会发生该事件。
- 接收活动中断，通过 RXACT （ETHIRQ<5>）位指示，并使用 RXACTIE （ETHIEN<5>）位来允许该中断。每次有数据存储到 RXBM FIFO 中时，会发生该事件。
- 接收缓冲区不可用中断，通过 RXBUFNA （ETHIRQ<1>）位指示，并使用 RXBUFNAIE （ETHIEN<1>）位来允许该中断。每次接收 DMA 获取不是由硬件拥有的描述符 （EOWN = 0）而无描述符可用时，会发生该事件。

- 接收 FIFO 溢出错误中断，通过 RXOVFLW （ETHIRQ<0>）位指示，并使用 RXOVFLIE （ETHIEN<0>）位来允许该中断。每当由于以下原因之一，使 RXBM 无法将数据从接收 FIFO 传输到系统存储器中，内部 FIFO 发生溢出时，会发生该事件：
  - 系统级别的延时过长
  - BUFCNT （ETHSTAT<16:23>）达到最大值
  - 没有可用于硬件处理的描述符
- 空水印中断，通过 EWMARK （ETHIRQ<9>）位指示，并使用 EWMARKIE （ETHIEN<9>）位来允许该中断。每当接收描述符缓冲区计数 BUFCNT （ETHSTAT<16:23>）小于等于 RXEWM （ETHRXWM<0:7>）值时，会发生该事件。
- 满水印中断，通过 FWMARK （ETHIRQ<8>）位指示，并使用 FWMARKIE （ETHIEN<8>）位来允许该中断。每当接收描述符缓冲区计数 BUFCNT （ETHSTAT<16:23>）大于等于 RXFWM （ETHRXWM<16:23>）值时，会发生该事件。

此外，根据中断的清除方式和清除位置，以太网外设中的中断可以分为两类。它们是软件清除中断事件和硬件清除中断事件。请注意，通过器件复位可以清除所有中断事件。

软件清除中断事件通过直接写入 ETHIRQCLR 寄存器或 ETHIRQ 寄存器中的相应位来进行清除，包括以下：

- TXBUSE
- RXBUSE
- RXDONE
- RXACT
- TXDONE
- TXABORT
- RXBUFNA
- RXOVFLW

硬件清除中断事件通过消除中断产生条件来进行清除，包括以下：

- EWMARK——当成功接收到接收数据包，并且 BUFCNT 值大于 RXEWM （ETHRXWM<0:7>）值时，可以清除该中断。
- FWMARK——通过写入 BUFCDEC （ETHCON1<0>）位，从而将缓冲区描述符计数（BUFCNT）递减至小于 RXFWM （ETHRXWM<16:23>）值，可以清除该中断。
- PKTPEND——通过写入 BUFCDEC 位，直到 BUFCNT （ETHSTAT<16:23>）达到 0，可以清除该中断。

所有属于以太网控制器的中断都映射到以太网中断向量。

相应的以太网控制器中断标志为 ETHIF （IFS1<28>）。在处理中断产生原因之后，必须用软件清零该中断标志。

以太网控制器通过相应的以太网控制器中断允许位 ETHIE （IEC1<28>）来使能为中断源。

此外，还必须配置中断优先级位和中断子优先级位：

- ETHIP<2:0> （IPC12<4:2>）
- ETHIS<1:0> （IPC12<1:0>）

注： 关于 IFSx、IECx 和 IPCx 中断位的详细说明，请参见**第 8 章“中断”**（DS61108）。

35.5.1 中断配置

以太网控制器在内部具有多个中断标志位（TXBUSE、RXBUSE、EWMARK、FWMARK、RXDONE、PKTPEND、RXACT、TXDONE、TXABORT、RXBUFNA 和 RXOVFLW）以及相应的中断允许控制位（TXBUSEIE、RXBUSEIE、EWMARKIE、FWMARKIE、RXDONEIE、PKTPENDIE、RXACTIE、TXDONEIE、TXABORTIE、RXBUFNAIE 和 RXOVFLIE）。但是，对于中断控制器，只有一个用于以太网控制器的专用中断标志位 ETHIF （IFS1<28>），以及相应的中断允许 / 屏蔽位 ETHIE （IEC1<28>）。

注： 以太网控制器的所有中断条件仅共用一个中断向量。

以太网控制器具有自己的优先级和子优先级，独立于其他外设。请注意，ETHIF 位是否置 1 与相应中断允许位 ETHIE 的状态无关。如果需要，可以用软件查询 ETHIF 位。

ETHIE 位用于定义在相应 ETHIF 位置 1 时，向量中断控制器（Vector Interrupt Controller，VIC）模块的行为。当相应的 ETHIE 位清零时，中断模块不会为事件产生 CPU 中断。如果 ETHIE 位置 1，则中断模块会在 ETHIF 位置 1 时产生 CPU 中断（受以下段落中优先级和子优先级制约）。

处理特定中断的用户软件程序负责在服务程序完成之前清零中断标志位。

以太网控制器中断的优先级可以使用中断控制器的 IPC12 寄存器设置。该优先级定义了中断源将分配到的优先级组。优先级组值的范围从 7 （最高优先级）到 0 （不产生中断）。较高优先级组中的中断会抢占正在处理、但优先级较低的中断。

子优先级位用于设置中断源在优先级组中的优先级。子优先级值的范围从 3 （最高优先级）到 0（最低优先级）。处于相同优先级组，但具有更高子优先级值的中断不会抢占子优先级较低、但正在进行的中断。

优先级组和子优先级位让多个中断源可以共用相同的优先级和子优先级。如果在该配置下同时发生若干个中断，则中断源在优先级 / 子优先级组对中的自然顺序将决定所产生的中断。

自然优先级基于中断源的向量编号。向量编号越小，中断的自然优先级就越高。在当前中断的中断标志清零之后，所有不按照自然顺序执行的中断会基于优先级、子优先级和自然顺序产生相应的中断。

产生允许的中断之后，CPU 将跳转到为该中断分配的向量处。该中断的向量编号与自然顺序编号相同。然后，CPU 将在向量地址处开始执行代码。该向量地址处的用户代码应执行所有特定于应用程序的操作，并清零 ETHIF 中断标志 （如果是可用软件清除的中断，则还要清除 ETHIRQ 寄存器中的相应事件），然后退出。

更多信息，请参见**第 8 章 “中断”**（DS61108）中的向量地址表详细信息。

表 35-9： 各种偏移量的以太网中断向量 （EBASE = 0x8000:0000）

中断	向量 / 自然顺序	IRQ 编号	向量地址 IntCtl.VS = 0x01	向量地址 IntCtl.VS = 0x02	向量地址 IntCtl.VS = 0x04	向量地址 IntCtl.VS = 0x08	向量地址 IntCtl.VS = 0x10
ETH	48	60	8000 0800	8000 0e00	8000 1a00	8000 3200	8000 6200

例 35-6： 允许中断的以太网初始化代码

```
// 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
```

例 35-7： 以太网控制器 ISR 代码

```
/*
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.
}
```

注： 以太网控制器 ISR 代码示例显示的是 MPLAB C32 C 编译器的特定语法。关于对 ISR 的支持，请参见编译器手册。

35.5.2 外部 PHY 中断

一些 PHY 支持在发生特定事件时产生中断信号。通常对于以下类型的事件 / 条件， PHY 中断会置为有效：

- 能量检测
- 退出掉电模式
- 自动协商完成
- 检测到远程故障
- 链路断开
- 自动协商 LP 应答
- 并行检测故障
- 接收到自动协商页面

关于可产生中断的事件的详细信息，请参见特定于供应商的 PHY 数据手册。

如果需要让 PIC32MX 获知并响应该中断，则应将 PHY 中断信号与 PIC32MX 外部中断引脚连接。请注意，该中断不会经过以太网控制器。

软件必须通过 MAC MIIM 寄存器写入 PHY 中的某个寄存器来清除 PHY 产生的中断事件。请注意在这种情况下， PHY 中断只能用软件清除，不能在 ETHIRQ 寄存器中直接清除。

35.6 节能和调试模式下的操作

35.6.1 休眠模式下的以太网操作

当 PIC32MX 器件进入 Sleep （休眠）模式时，系统时钟被禁止。在该模式下不能发生任何以太网传输。对于以太网控制器，如果工作于 RMII 模式，则除外部 MII RX_CLK 和 TX_CLK 信号或 REF_CLK 之外的所有时钟都会被停止。以太网控制器将处于 Sleep （休眠）模式，并允许异步唤醒事件。

如果在以太网控制器正在进行传输时，用户应用程序进入 Sleep （休眠）模式，则以太网控制器会暂挂它的当前状态，直到时钟恢复执行为止。软件应避免这种情况，因为这可能会导致非预期的引脚时序。

软件负责确定链路何时处于可以让以太网控制器安全进入 Sleep （休眠）模式的状态。只有在两种情况下，才建议通过 CPU 执行 WAIT 指令来将器件置为 Sleep （休眠）模式：

- 以太网控制器被禁止。
- 以太网控制器没有任何待处理的发送数据包，并且所有传入的接收数据包都已处理。

在总线正在进行发送事务时将以太网控制器置为 Sleep （休眠）模式可能导致错误的以太网器件行为，这可能导致数据包丢失或链路连接断开。

在器件安全进入 Sleep（休眠）模式之后，以太网控制器会在发生异步使能事件时产生唤醒中断。

35.6.2 空闲模式下的以太网操作

当器件进入 Idle （空闲）模式时，系统时钟源继续保持工作。 SIDL 位 （ETHCON1<13>）用于选择在 Idle （空闲）模式下以太网控制器是停止还是继续工作：

- 如果 SIDL = 0，则在 Idle （空闲）模式下以太网控制器将继续工作，并且会开启时钟。
- 如果 SIDL = 1，则在 Idle （空闲）模式下以太网控制器将停止工作。以太网控制器将会关闭时钟 （RX_CLK、 TX_CLK 或 REF_CLK 除外），从而降低功耗。当以太网控制器在 Idle （空闲）模式下停止时，它的行为与 Sleep （休眠）模式下的行为相同。

注：	对于 SIDL （ETHCON1<13>） = 1 的 Idle （空闲）模式，以太网控制器对待它的方式与对待 Sleep （休眠）模式相同。因此会应用相同的限制。

35.6.3 网络唤醒 （Wake-on-LAN， WOL）操作

以太网控制器中未实现特定的 WOL 功能。WOL 功能通过以下方式来实现：设置相应的接收过滤器，并允许 RXDONE （ETHIRQ<7>）中断在接收到接收数据包时唤醒该系统。通常使用 Magic Packet 过滤器来实现该目的。

如果系统处于 Sleep （休眠）模式或处于低速时钟模式，软件可以允许 RXACT （ETHIRQ<5>）中断。这可以防止在器件处于低功耗模式时，接收缓冲区发生溢出。

使用 RXACT 来从 Sleep （休眠）或 Idle （空闲）（并且 SIDL （ETHCON1<13>）位置 1）中唤醒系统时必须特别小心。

例 35-8 给出了用于将系统置为 Sleep （休眠）或 Idle （空闲）模式的指令序列。

例 35-8：　　使用以太网控制器唤醒系统

```
/*
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
    }
```

在这些情形下，唤醒 ISR 应类似于例 35-9：

例 35-9：　　发生 RXACT 时的以太网控制器唤醒 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.
}
```

注：　以太网控制器 ISR 代码示例显示的是 MPLAB C32 C 编译器的特定语法。关于对 ISR 的支持，请参见编译器手册。

由于以太网控制器产生该中断请求的方式的原因 （只要接收 FIFO 中存在数据，它就有效），在该 ISR 中需要禁止产生进一步的 RXACT 中断。否则，控制权会不断地转回到 ISR，而代码执行则会被锁住。

但是，除了禁止 RXACT 中断之外，还需要进一步的操作：禁止 Sleep （休眠）模式和使能 Idle （空闲）模式，并确保使能以太网控制器 （SIDL = 0） 。需要执行该操作是为了防止这种情况：正好在例 35-9 主循环中使能 RXACT 之后，但恰好在执行 WAIT 指令之前，发生了 XACT 中断。

如果正好在调用 WAIT 指令之前接收到活动，ISR 将会禁止接收活动中断，并且在 ISR 返回到主循环时，将会执行 WAIT 指令。此时，以太网控制器将永远无法唤醒部件。

变通方法是用 ISR 禁止 Sleep （休眠）模式并清零以太网 SIDL 位，从而以太网控制器不会进入休眠模式。

请注意，在禁止 RXACT 中断之后，需要允许另一个以太网中断。通常， RXDONE 位是最佳的候选。

35.6.4 调试模式下的以太网操作

FRZ 位 （ETHCON1<14>）决定 CPU 在 Debug （调试）模式下执行调试异常代码 （即，应用程序暂停）时，以太网控制器是继续运行还是停止。

- 当 FRZ = 0 时，在 Debug （调试）模式下，即使应用程序暂停，以太网控制器仍会继续运行。
- 当 FRZ = 1 并且应用程序在 Debug （调试）模式下暂停时，以太网控制器会完成当前总线事务，然后冻结其操作，并且不会对以太网控制器的状态进行任何更改。所有以太网控制器寄存器都是可读写的；但是，读操作是非破坏性操作，不会改变外设的状态。除 RX_CLK /TX_CLK （MII）或 REF_CLK （RMII）时钟外，送到其他外设的系统时钟都会被停止。在 CPU 继续开始执行代码之后，外设将继续工作。

注：	只有 CPU 在调试异常模式下执行时， FRZ 位才可读写。在所有其他模式下， FRZ 位读为 0。如果 FRZ 位在 Debug（调试）模式期间发生改变，则只有退出当前调试异常模式并重新进入该模式之后，新值才会生效。在调试异常模式期间，在进入 Debug（调试）模式时 FRZ 位会读取外设状态。当以太网外设被冻结时，它将无法执行正常的以太网协议。这意味着发送帧可能会被中止，接收帧可能会被丢弃。

35.7 各种复位的影响

35.7.1 器件复位

在发生器件复位时，所有以太网寄存器会被强制设为它们的复位状态。当异步复位输入变为有效时，以太网逻辑会执行以下操作：

- 复位 SFR （ETHCON1、 ETHCON2 和 ETHSTAT 等）中的所有字段。
- 装入 EMACxSA0、 EMACxSA1 和 EMACxSA2 寄存器，使之包含出厂时预先设定的站点地址。
- 复位发送和接收 DMA 引擎，并将相应 FIFO 置为空状态。
- 中止所有正在进行的数据传输。

35.7.2 上电复位

在发生上电复位时，所有以太网控制器寄存器会被强制设为它们的复位状态。

35.7.3 看门狗定时器复位

在发生看门狗定时器复位时，所有以太网控制器寄存器会被强制设为它们的复位状态。

35.8 I/O 引脚控制

使能以太网控制器时，将按照以太网控制器控制位的定义配置 I/O 引脚方向 （见表 35-10）。端口 TRIS 和 LATCH 寄存器会被改写。

表 35-10： 用于以太网控制器的 I/O 引脚配置

I/O 引脚名称	MII 必需	RMII 必需	模块控制	TRIS(1)	引脚类型	说明
EMDC	是	是	ON	X	O	以太网 MII 管理时钟
EMDIO	是	是	ON	X	I/O	以太网 MII 管理 I/O
ETXCLK	是	否	ON	X	I	以太网 MII 发送时钟
ETXEN	是	是	ON	X	O	以太网发送使能
ETXD0	是	是	ON	X	O	以太网数据发送 0
ETXD1	是	是	ON	X	O	以太网数据发送 1
ETXD2	是	否	ON	X	O	以太网数据发送 2
ETXD3	是	否	ON	X	O	以太网数据发送 3
ETXERR	是	否	ON	X	O	以太网发送错误
ERXCLK	是	否	ON	X	I	以太网 MII 接收时钟
EREF_CLK	否	是	ON	X	I	以太网 RMII 参考时钟
ERXDV	是	否	ON	X	I	以太网 MII 接收数据有效
ECRS_DV	否	是	ON	X	I	以太网 RMII 载波检测 / 接收数据有效
ERXD0	是	是	ON	X	I	以太网数据接收 0
ERXD1	是	是	ON	X	I	以太网数据接收 1
ERXD2	是	否	ON	X	I	以太网数据接收 2
ERXD3	是	否	ON	X	I	以太网数据接收 3
ERXERR	是	是	ON	X	I	以太网接收错误
ECRS	是	否	ON	X	I	以太网载波检测
ECOL	是	否	ON	X	I	检测到以太网冲突

注 1： TRIS 位的设置无关。但是，如果引脚与某个模拟输入共用，则必须正确设置 AD1PCFG 和相应的 TRIS 寄存器，将该引脚配置为数字引脚。

35.9 相关应用笔记

本节列出了与手册本章内容相关的应用笔记。这些应用笔记可能并不是专为 PIC32MX 器件系列而编写的，但其概念是相近的，通过适当修改并受到一定限制即可使用。当前与以太网控制器模块相关的应用笔记有：

标题	应用笔记编号
Ethernet Theory of Operation	AN1120

注： 如需获取更多 PIC32MX 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

35.10 版本历史

版本 A（2009 年 8 月）

这是本文档的初始版本。

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “牢不可破”的。

代码保护功能处于持续发展中。 Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了 《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

ISBN：978-1-60932-572-5

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2002 认证。公司在 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品方面的质量体系流程均符合 ISO/TS-16949:2002。此外，Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

全球销售及服务网点

美洲

公司总部 Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://support.microchip.com
网址：www.microchip.com

亚特兰大 Atlanta
Duluth, GA
Tel: 1-678-957-9614
Fax: 1-678-957-1455

波士顿 Boston
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 Chicago
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

克里夫兰 Cleveland
Independence, OH
Tel: 1-216-447-0464
Fax: 1-216-447-0643

达拉斯 Dallas
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 Detroit
Farmington Hills, MI
Tel: 1-248-538-2250
Fax: 1-248-538-2260

科科莫 Kokomo
Kokomo, IN
Tel: 1-765-864-8360
Fax: 1-765-864-8387

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

圣克拉拉 Santa Clara
Santa Clara, CA
Tel: 1-408-961-6444
Fax: 1-408-961-6445

加拿大多伦多 Toronto
Mississauga, Ontario,
Canada
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

亚太总部 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

中国 – 北京
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

中国 – 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 – 重庆
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

中国 – 香港特别行政区
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 – 南京
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

中国 – 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 – 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 – 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 – 深圳
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

中国 – 武汉
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 – 西安
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

中国 – 厦门
Tel: 86-592-238-8138
Fax: 86-592-238-8130

中国 – 珠海
Tel: 86-756-321-0040
Fax: 86-756-321-0049

台湾地区 – 高雄
Tel: 886-7-213-7830
Fax: 886-7-330-9305

台湾地区 – 台北
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

亚太地区

台湾地区 – 新竹
Tel: 886-3-6578-300
Fax: 886-3-6578-370

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

印度 India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

印度 India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

印度 India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

日本 Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

韩国 Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

韩国 Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

菲律宾 Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

新加坡 Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

泰国 Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

欧洲

奥地利 Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

丹麦 Denmark-Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

法国 France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

德国 Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

意大利 Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

荷兰 Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

西班牙 Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

英国 UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

07/15/10