


硬核处理器系统 (HPS) 提供两种以太网介质访问控制器 (EMAC) 外设。每个 EMAC 均能够通过符合 IEEE 802.3 规范的以太网连接，以 10/100/1000 Mbps 发送和接收数据。EMAC 是 Synopsys® DesignWare® 3504-0 Universal 10/100/1000 Ethernet MAC (DWC_gmac) 的实例。

EMAC 有一个扩展的存储器映射控制和状态寄存器 (CSR) 组，可以被 ARM Cortex™-A9® MPCore™访问。

为更好地了解本章内容，您应该对 IEEE 802.3 介质访问控制 (MAC) 的基础有所了解。

 关于 IEEE 802.3 MAC 的完整信息，请参考 IEEE 网站 (standards.ieee.org/findstds/) 上的 *IEEE Std 802.3-2008 Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*。

Ethernet MAC 的特性

以下是 EMAC 外设支持的特性。

MAC

- IEEE 802.3-2008 兼容
- 10/100/1000 Mbps 数据速率
- 全双工和半双工模式
- 流程控制输入置低时的零量子暂停帧的 IEEE 802.3x 流程控制自动传输
- 接收暂停控制帧到用户的可选转发
- 1000 Mbps 半双工下的数据包突发和帧扩展
- 全双工下的 IEEE 802.3x 流程控制
- 对半双工的反压支持
- IEEE 1588-2002 和 IEEE 1588-2008 精密网络时钟同步
- 用于节能以太网 (EEE) 的 IEEE 802.3-az 版本 D2.0。
- 支持探测在接收帧 IEEE 802.1Q 虚拟局域网 (VLAN) 标记
- 发送通路中的前导码和帧起始数据 (SFD) 插入和接收通路中的删除
- 每帧上的自动循环冗余检查 (CRC) 和帧生成可控制

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Portions © 2011 Synopsys, Inc. Used with permission. All rights reserved. Synopsys and DesignWare are registered trademarks of Synopsys, Inc. All documentation is provided "as is" and without any warranty. Synopsys expressly disclaims any and all warranties, express, implied, or otherwise, including the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, and any warranties arising out of a course of dealing or usage of trade.

†Paragraphs marked with the dagger (†) symbol are Synopsys Proprietary. Used with permission.



- 接收帧上的自动垫 /CRC 撤除的选项
- 可编程帧长度，支持标准和大型 Ethernet 帧（容量高达 16 KB）
- 可编程帧间空隙 (IFG)，40 到 96 之间 8 的倍数的位时间 (bit times)

PHY 接口

- 10/100/1000 的简化吉比特介质独立接口 (RGMII)
- 管理数据输入 / 输出 (MDIO) (IEEE 802.3) 或者 I²C PHY 管理接口

DMA 接口

- 32-bit 接口
- 可编程突发大小，实现最佳总线利用率
- 单通道模式发送和接收引擎
- 字节对齐的寻址模式用于数据缓存支持
- 双缓存（环状）或者链接列表（链式）描述符链接
- 描述符能够每次传递高达 8 KB 数据

管理接口

- 与 CSR 组连接的 32-bit 主机接口
- 正常操作和有错传输的全面状态报告
- 用于不同操作条件的可配置中断选项
- 每帧发送 / 接收完成中断控制
- 发送和接收数据包的返回分离状态

加速

- 发送和接收校验和卸载，用于传输控制协议 (TCP)，用户数据报协议 (UDP)，或者通过因特网协议 (IP) 的互联网控制消息协议 (ICMP)

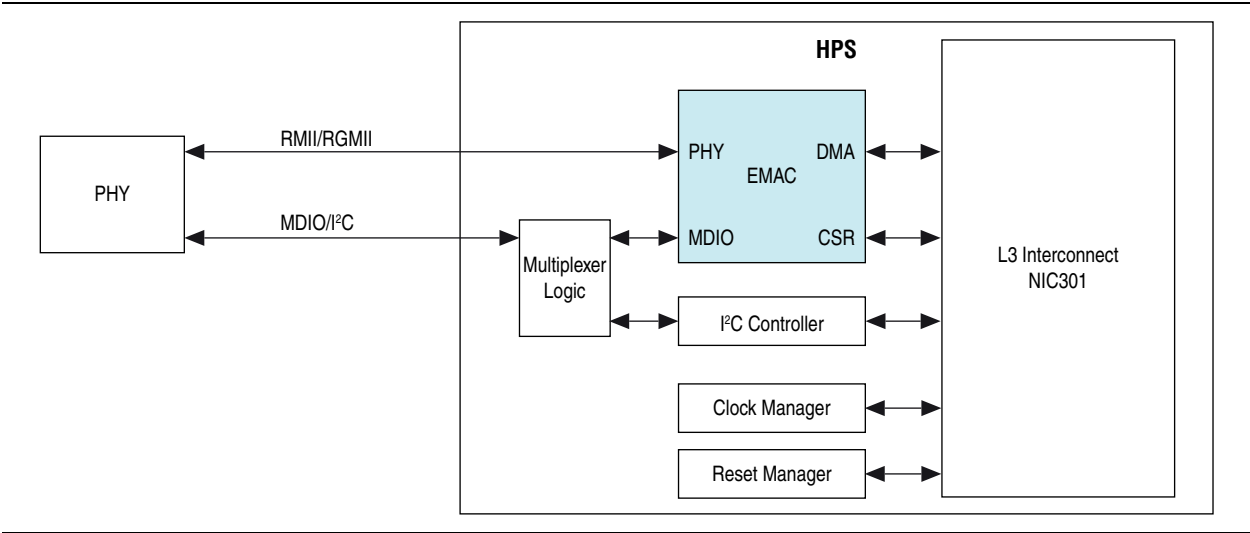
其它特性

- 支持多种灵活的地址过滤模式
- 总共有 31 个可以按字节屏蔽的 48-bit 目的地地址 (DA) 过滤器
- 总共有 31 个可以按字节屏蔽的 48-bit 源地址 (SA) 比较检查
- 256-bit 散列过滤器（可选的）用于多播和单播 DA
- 传递所有多播寻址帧的选项
- 用于传递所有帧的混杂模式支持，无需任何网络监控的过滤
- 传递全部输入数据包（根据过滤器）和状态报告

EMAC 结构图和系统集成

EMAC 集成在片上系统 (SoC)FPGA 器件的 HPS 部分，与 I/O 管脚进行通信。第 17 - 3 页的图 17 - 1 以高级视角展示了 EMAC 集成。

图 17 - 1. EMAC 系统集成



EMAC 到 RGMII 接口

PHY 数据通路 I/O 在表 17 - 1 中有所描述。

表 17 - 1. 外部 PHY 数据接口 (1/2)

EMAC 端口	I/O	宽度	说明
clk_tx_i	In	1	发送时钟。这是用于 RGMII 的发送时钟 (1G/100M/10Mbps 下的 125/25/2.5 MHz)。由 EMAC 生成的所有 PHY 发送信号与此时钟同步。
phy_txd_o	Out	8	PHY 发送数据。这是由 MAC 驱动的一组 8 个发送数据信号。RGMII 接口配置中未使用的比特被拉低。 RGMII: Bits [3:0] 提供 RGMII 发送数据。数据总线随着发送时钟 (clk_tx_i) 的上升沿和下降沿的变化而变化。数据的有效性由 phy_txen_o 认证。 同步到: clk_tx_i, clk_tx_180_i
phy_txen_o	Out	1	PHY 发送数据使能。此信号由 EMAC 组件驱动。 RGMII: 此信号是发送数据的控制信号 (rgmii_tctl), 在时钟的上升沿以及下降沿驱动。 同步到: clk_tx_i, clk_tx_180_i
rst_clk_tx_n_o	Out	1	发送时钟复位输出。
clk_rx_i	In	1	接收时钟。时钟频率在 1G/100M/10Mbps 模式下为 125/25/2.5 MHz。它由外部 PHY 提供。由 EMAC 接收到的所有 PHY 信号与此时钟同步。
phy_rxd_i	In	8	PHY 接收数据。这是从 PHY 接收到的一组 8 个数据信号。 RGMII: Bits [3:0] 提供 RGMII 接收数据。数据总线在接收时钟 (clk_rx_i) 的上升沿以及下降沿采样。数据的有效性由 phy_rxdv_i 认证。 同步到: clk_rx_i, clk_rx_180_i
phy_rxdv_i	In	1	PHY 接收数据有效。此信号由 PHY 驱动。 RGMII: 这是接收控制信号, 用于认证在 phy_rxd 上接收到的数据。此信号在时钟上升沿以及下降沿采样。 同步到: clk_rx_i, clk_rx_180_i
rst_clk_rx_n_o	Out	1	接收时钟复位输出。

表 17-1. 外部 PHY 数据接口 (2/2)

EMAC 端口	I/O	宽度	说明
phy_intf_sel_i[1:0]	In	2	PHY 接口选择：这些管脚选择一个 EMAC 的 PHY 接口。这只在复位位置期间采样，然后被忽略。 ■ 01：RGMII ■ 00, 10 和 11：无效
clk_ref_i	In	1	这是 EMAC 的参考时钟。此时钟是有时钟管理器提供的 emac0_clk 或者 emac1_clk。 系统管理器驱动 phy_intf_sel 信号来控制使用哪一个时钟。 时钟速率为 250 MHz。

PHY 管理接口

HPS 提供对 MDIO 或者 I²C PHY 管理接口的支持。

MDIO 接口

MDIO 接口信号同步于所有支持模式下的 14_mp_clk。

表 17-2. PHY MDIO 管理接口

信号	I/O	宽度	说明
gmii_mdi_i	In	1	Management Data In。PHY 生成此信号以在读操作期间传递寄存器数据。此信号由 gmii_mdc_o 时钟同步驱动。
gmii_mdo_o	Out	1	Management Data Out。EMAC 使用此信号传递控制和数据信息到 PHY。
gmii_mdo_o_e	Out	1	管理数据输出使能。此使能信号驱动来自外部三态 I/O 缓存的 gmii_mdo_o 信号。只要有效数据在 gmii_mdo_o 信号上被驱动，此信号就被置位。此信号的有效状态为高电平有效。
gmii_mdc_o	Out	1	管理数据时钟。EMAC 通过该非周期时钟对 MII 上的 gmii_mdi_i 和 gmii_mdo_o 信号提供时序参考。此时钟的最大频率为 2.5 MHz。此时钟通过时钟分频器从应用时钟生成。

I²C 外部 PHY 管理接口

某些 PHY 器件对其控制接口使用 I²C，而不是 MDIO。小型化可热插拔 (SFP) 光模块或者可热插拔模块通常用于此接口。

HPS 能够使用 4 个通用 I²C 外设中的 2 个来控制 PHY 器件。

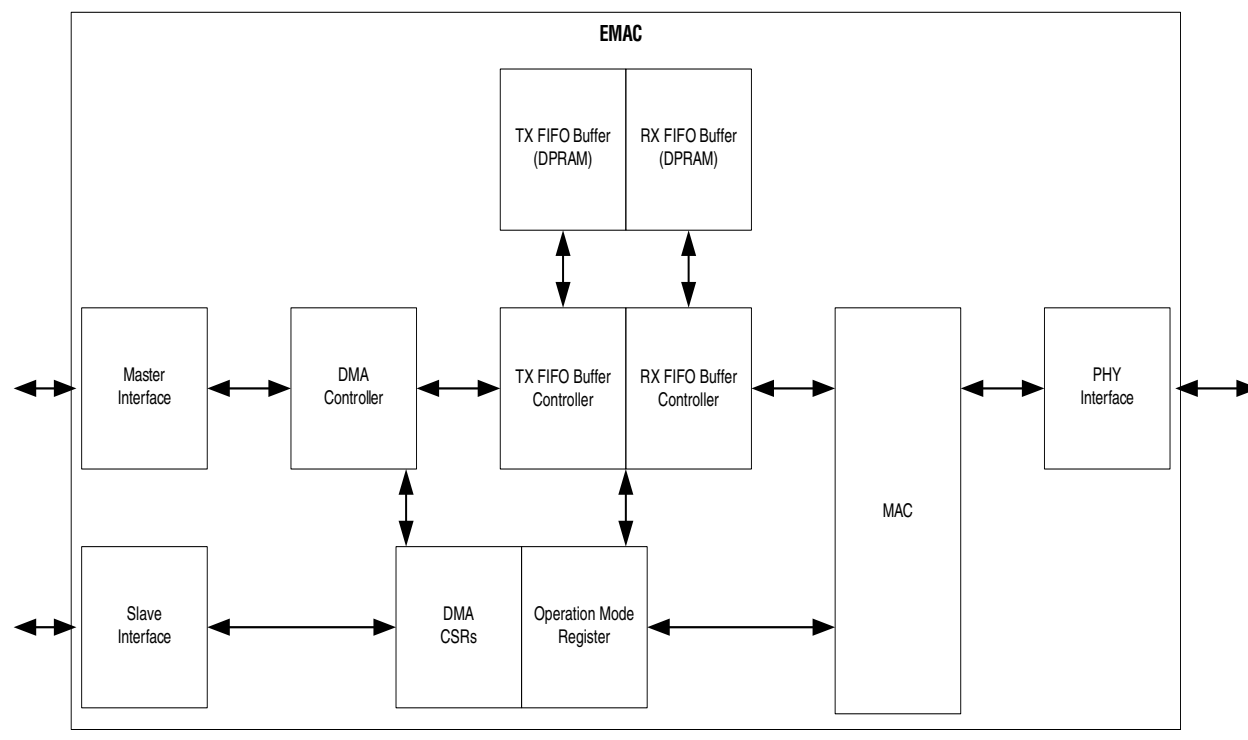
IEEE 1588

EMAC 支持 1μs 解析度所有模式下的 IEEE 1588 操作。ARM® Cortex™-A9 微处理器单元 (MPU) 子系统使用它来维持两个 MAC 内部时间计数器之间的同步。

EMAC 的功能描述

图 17-2 显示了 EMAC 及其接口的高级结构图。

图 17-2. EMAC 结构图



连接到 MAC 的主机接口有两种。管理主机接口是一个 32-bit 从接口，提供对 CSR 组的访问。数据接口是一个 32-bit 接口，通过 NIC-301 L3 互联来控制直接存储器访问 (DMA) 与 HPS 系统其它部分之间的数据传输。

有一个内置的 DMA 控制器，被优化用于 MAC 控制器与系统存储器之间的数据传输。DMA 控制器具有独立的发送和接收引擎，以及独立用于控制状态的寄存器 (CSR) 组。发送引擎将数据从系统存储器传递到器件端口，而接收引擎将数据从器件端口传递到系统存储器。控制器使用描述符在最小的主机干预下有效地将数据从源移到目的地。

EMAC 也包含 FIFO 缓存，用于缓冲并调节应用系统存储器与 EMAC 控制器之间的 Ethernet 帧。发送时，Ethernet 帧读入发送 FIFO 缓存 (1024 x 42 bits)，并最终触发 MAC 执行传递。接收到的 Ethernet 帧存储在接收 FIFO 缓存中，也表明了对 DMA 控制器的 FIFO 缓存填充水平。DMA 控制器然后启动已配置的突发传递。接收以及发送状态从 MAC 中获取，并传递到 DMA 中。

主机接口

EMAC 中有两个主机接口：一个从接口和一个主接口。主接口连接到 L3 互联模块中的 L3 主外设交换接口。

从接口

EMAC CSR 组访问由从接口提供。从接口连接到 level 4 (L4) 总线。

主接口

DMA 接口由主接口提供。在此接口上传递两种类型的数据：数据描述符和实际数据包。此接口在传输全双工 Ethernet 数据包流量上非常高效。在此端口上可以同时执行不同 DMA 通道上的读写数据传输。唯一的例外是发送描述符读操作和回写操作，这两个操作不能同时发生。

DMA 传递分成了软件在接口上可控数量的突发传输。dmagrp 组中的 AXI_Bus_Mode 寄存器用于配置突发行为。

此接口对每个 DMA 通道分配一个不同的 ID，并且对通道中的每个读 DMA 或写 DMA 请求也分配一个不同的 ID。不同 ID 的数据传输能够被重新排序和交错。

互联一旦接收到数据突发的最后差拍 (beat)，在 OK 响应返回时通常执行发布的写数据传输。为防止传输完成中断逻辑争用情况的发生，描述符（状态或时间戳）要始终作为非发布写操作来传输。

从接口可以发出一个错误响应。在此情况下，EMAC 会禁止 DMA 通道，该 DMA 通道生成了原始请求并置位中断信号。主机需要通过硬复位或软复位对 EMAC 进行复位以重新开始 DMA 从此条件中恢复。

EMAC 支持此接口上高达 16 个待执行传输。缓存待执行传输可以消除反压行为，这在高系统加载条件下出现资源争用瓶颈时很重要。

高速缓存控制接口

系统管理器通过此接口对主高速缓存输出提供相应值。这些输入用作 L3 互联的输出，就主传递的高速缓存特性而言，扩展了此模块的性能。

要配置 EMAC DMA 控制器以执行高速缓存的访问，需要在系统管理器中配置高速缓存比特。高速缓存比特只能在 EMAC 控制器退出复位状态前的引导期间访问。



关于更多信息，请参考 *Cyclone V 器件手册卷 3* 的 *System Manager* 章节。

外部 PHY

HPS 支持下面的 PHY 接口：

- 10/100/1000 的 RGMII

EMAC 也有一个控制接口，用于 PHY 的配置和状态监控。在此情况下，PHY 是从器件。有两个可选控制接口：

- MDIO
- I²C 接口

MDIO 接口内置在 EMAC 中，而 I²C 接口使用位于 HPS 上的不同 I²C 外设。从 EMAC 外部多路复用这两个接口。

发送和接收数据 FIFO 缓存

每个 EMAC 组件都有相关的发送和接收数据 FIFO 缓存实例。两个 FIFO 缓存实例都是 1024 x 42 bits。FIFO 缓存字包括：

- 数据：32 比特

- 边带：
 - 帧结束 (EOF)：1 比特
 - 字节使能 (BE)：2 比特
 - 错误纠正代码 (ECC)：7 比特

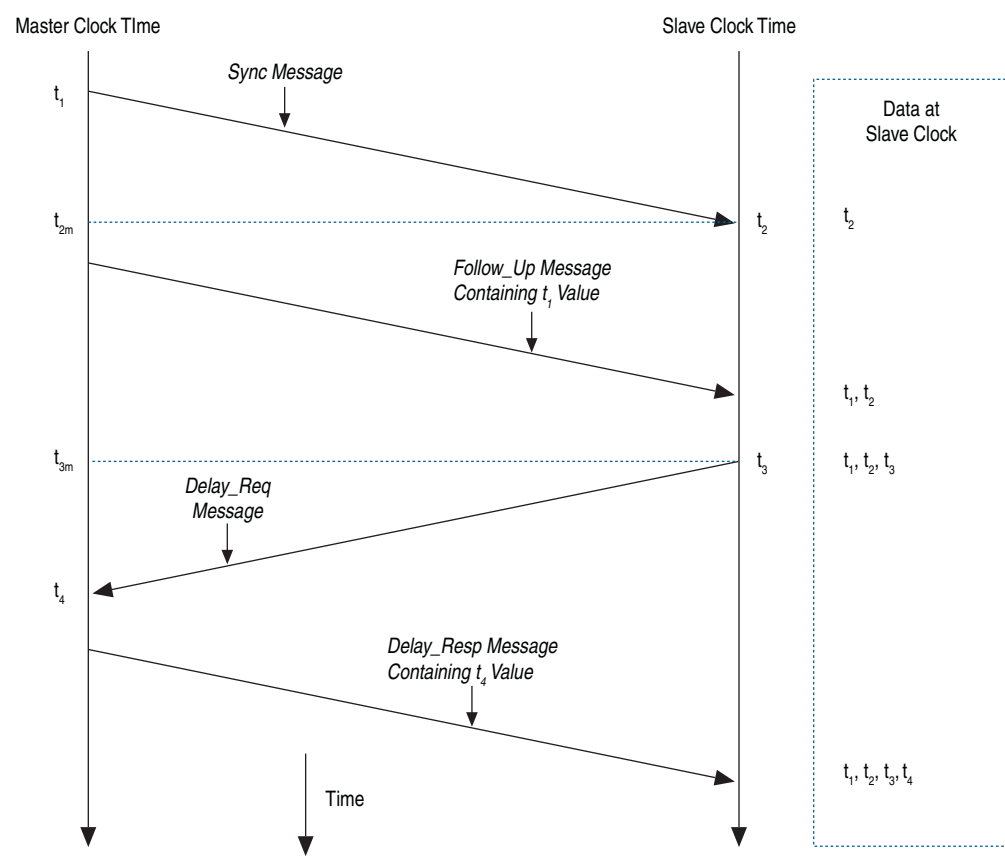
数据和边带受 7 比特单一错误校正，双错误检验 (SEC-DED) 代码字保护。这些 FIFO 缓存 RAM 也包含 ECC 使能，错误注入和状态管脚。使能和错误注入管脚是由系统管理器驱动的输入，状态管脚是驱动到 MPU 子系统的输出。

IEEE 1588-2002 时间戳

IEEE 1588-2002 标准定义了精确时间协议 (PTP)，使能了器件分布网络中的时钟精确同步。PTP 通过局域网支持组播消息而能够应用于系统通信。此协议可以实现异构系统，其中包括各种固有精度，解析度和同步稳定性的时钟。它通常用于自动化系统中，在此系统中，一组通信设备（例如机器人）必须同步，从而在共同的时基上运行。

PTP 通过 UDP/IP 传输。系统或网络被分为 Master 和 Slave 节点，用于分发时序和时钟信息。图 17-3 显示了通过交换 PTP 消息来同步主节点到从节点时 PTP 所采用的过程。

图 17-3. 网络化的时间同步



如图 17-3 所示，PTP 使用下面过程：

1. 主接口广播 PTP Sync 消息到其所有节点。Sync 消息包含主接口的参考时间信息。此消息离开主接口的系统的时间是 t_1 。一定要在 PHY 接口（Ethernet 端口）上采集到此时间。
2. 从接口接收 Sync 消息，并使用其时序参考采集确切时间 t_2 。
3. 主接口发送一条 Follow_up 消息到从接口，其中包含 t_1 信息以便稍后使用。
4. 从接口发送一条 Delay_Req 消息到主接口，指明此帧离开 PHY 接口的确切时间 t_3 。
5. 主接口接收消息，采集进入其系统的准确时间 t_4 。
6. 主接口发送 t_4 信息到 Delay_Resp 消息中的从接口。
7. 从接口使用 t_1 , t_2 , t_3 和 t_4 值来同步其本地时序参考到主接口的时序参考。

大多数的 PTP 实现在 UDP 层以上的软件中完成。然而，需要硬件支持来采集指定 PTP 数据包进入或离开 PHY 接口上的 Ethernet 端口的准确时间。此时序信息必须被采集并返回到软件中，以正确实现高精度的 PTP。

EMAC 旨在支持 $1\mu\text{s}$ 解析度所有模式下的 IEEE 1588 操作。当两个 EMAC 同时运行在 IEEE 1588 环境中时，MPU 子系统负责维持两个 MAC 内部时间计数器之间的同步。

与 FPGA 连接的 IEEE 1588 接口使 FPGA 能够对 `emac_ptp_ref_clk` 输入提供一个备用源，以及监控从每个 EMAC 控制器输出的每秒脉冲。

EMAC 组件提供一个 IEEE 1588 协议的硬件辅助实现。硬件支持用于时间戳维持。当在 PHY 接口上接收到帧时，时间戳被更新，接收描述符也被该值更新。当一个帧的 SFD 被发送时，时间戳同样被更新，并相应地更新发送描述符。



关于 IEEE 1588-2002 标准的详细信息，请参考 IEEE Standards Association 网站 (standards.ieee.org) 上的 *IEEE Standard 1588-2002 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*。

参考时序源

要获得时间快照，EMAC 需要获取参考时钟输入，并用它从内部生成参考时间 (64-bit)，然后采集时间戳。

系统时间寄存器模块

在此模块中维持 64-bit 时间，并使用输入参考时钟 `osc1_clk` 进行更新。`osc1_clk` 时钟来自时钟管理器，`emac_ptp_ref_clk` 时钟来自 FPGA 架构。此时间是用于获取 PHY 接口上发送和接收的 Ethernet 帧快照（时间戳）的源。

使用粗糙纠正方法能够初始化或者纠正系统时间计数器。在此方法中，初始值或者偏移值被写入到 Timestamp Update 寄存器。对于初始化，使用 Timestamp Update 寄存器中的值写入每个 EMAC 的系统时间计数器，而对于系统时间纠正，偏移值被加到系统时间上，或者从系统时间中减去偏移值。

在精细纠正方法中，相对于主时钟，一个从时钟的频漂通过一个周期时间来纠正，而不是通过（粗糙方法中）一个时钟来纠正。这有助于维持线性时间，而且不会导致 PTP 同步消息间隔之间的参考时间的显著变化（或者较大毛刺）。

在此方法中，累加器将 Timestamp_Addend 寄存器中的数据相加，如图 17-4 所示。累加器生成的算术数列用作一个脉冲，递增系统时间计数器。累加器和加数是 32-bit 寄存器。这里，累加器用作高精度频率乘法器或除法器。


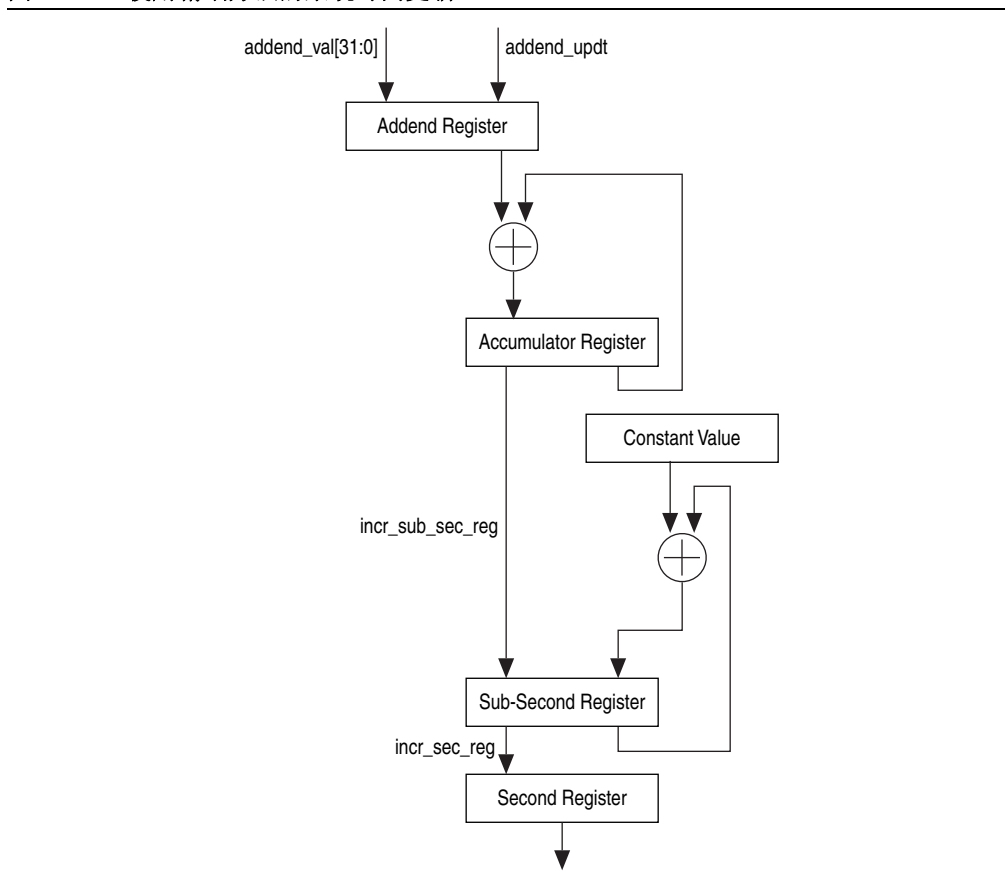
 您必须连接一个高于指定准确度要求频率的 PTP 时钟。

图 17-4 显示了此算法：

图 17-4. 使用精细方法的系统时间更新



System Time Update 逻辑需要一个 50-MHz 时钟频率来达到 20-ns 准确度。频率分频比率 (FreqDivisionRatio) 是参考时钟频率与所需时钟频率的比率。举例说明，如果参考时钟 (`clk_ptp_ref_i`) 是 66 MHz，那么比率为 66 MHz / 50 MHz = 1.32。因此，在寄存器中设置的默认加数值是 232 / 1.32，0xC1F07C1F。

如果参考时钟漂移较低，例如到 65 MHz，那么比率为 65 / 50 或者 1.3，加数寄存器中的设置值为 232 / 1.30 或者 0xC4EC4EC4。如果时钟漂移较高，例如到 67 MHz，那么加数寄存器必须设置成 0xBF0B7672。当无时钟漂移时，必须编程 0xC1F07C1F (232 / 1.32) 的默认加数值。

在图 17-4 中，用于累加亚秒寄存器的常量值为十进制 43，实现了系统时间中的 20 ns 的准确度（或者说，它以 20-ns 的倍数递增）。

软件必须根据 Sync 消息计算频率中的漂移，并且相应地更新 Addend 寄存器。

起初使用 Addend 寄存器中的 `FreqCompensationValue0` 设置从时钟，此值如下：

$$\text{FreqCompensationValue}_0 = 232 / \text{FreqDivisionRatio} \uparrow$$

如果最初假设 `MasterToSlaveDelay` 对于连续的 Sync 消息是相同的，那么必须应用下面描述的算法。几个 Sync 周期过后会出现频率锁。从时钟然后能够确定一个精确的 `MasterToSlaveDelay` 值并使用该新值与主时钟重新同步。†

算法如下：

- 在时间 MasterSyncTime_n 上，主时钟发送一条 Sync 消息到从时钟。当从时钟的本地时钟是 SlaveClockTime_n 时，从时钟接收此消息，并计算 MasterClockTime_n ：

$$\text{MasterClockTime}_n = \text{MasterSyncTime}_n + \text{MasterToSlaveDelay}_n \uparrow$$

- 当前 Sync 周期的主时钟数 $\text{MasterClockCount}_n$ ：

$$\text{MasterClockCount}_n = \text{MasterClockTime}_n - \text{MasterClockTime}_{n-1}$$

（假设 $\text{MasterToSlaveDelay}$ 对于 Sync 周期 n 和 $n-1$ 是相同的）

- 当前 Sync 周期的从时钟数 SlaveClockCount_n ：

$$\text{SlaveClockCount}_n = \text{SlaveClockTime}_n - \text{SlaveClockTime}_{n-1} \uparrow$$

- 当前 Sync 周期的主从时钟数的差 ClockDiffCount_n ：

$$\text{ClockDiffCount}_n = \text{MasterClockCount}_n - \text{SlaveClockCount}_n \uparrow$$

- 从时钟的频率缩放因子 FreqScaleFactor_n ：

$$\text{FreqScaleFactor}_n = (\text{MasterClockCount}_n + \text{ClockDiffCount}_n) / \text{SlaveClockCount}_n \uparrow$$

- Addend 寄存器的频率补偿值 $\text{FreqCompensationValue}_n$ ：

$$\text{FreqCompensationValue}_n = \text{FreqScaleFactor}_n \times \text{FreqCompensationValue}_{n-1} \uparrow$$

理论上，此算法在一个 Sync 周期内达到锁定；然而由于改变网络传播延迟和操作条件，也可能用几个周期达到锁定。

此算法是自纠正的：如果由于某种原因起初对从时钟设置一个不正确的值，那么此算法将以更多的 Sync 周期为代价来纠正此错误值。

发送路径功能

当一个帧的 SFD 在 PHY 接口上被发送时，MAC 采集时间戳。那些您想要采集时间戳的帧在每帧的基础上是可控的。换句话说，每个发送帧能够被标记来表明是否对该帧采集一个时间戳。MAC 不通过处理发送的帧来识别 PTP 帧。您需要指定想要采集时间戳的那些帧。MAC 将时间戳以及帧的 Transmit 状态返回到 FPGA 中实现的硬件中。您可以使用发送描述符中的控制比特。MAC 将时间戳返回到相应发送描述符中的软件中，从而自动将时间戳连接到指定的 PTP 帧。

接收路径功能

MAC 采集在 PHY 接口上接收到的所有帧的时间戳。DMA 将时间戳返回到相应接收描述符中的软件中。时间戳仅写入到最后的接收描述符中。

时间戳误差容限

根据 IEEE 1588 规范，时间戳必须在 PHY 接口上的发送和接收帧的 SFD 上采集。由于 PHY 接口接收和发送时钟不同步于参考时间戳时钟 (osc1_clk)，因此当时间戳在异步时钟域之间移动时会导致少量的漂移。在发送路径中，被采集和报告的时间戳有两个 PTP 时钟的最大误差容限，意味着被采集的时间戳具有参考时序源值，该值是 SFD 在 PHY 接口上发送后的两个时钟内给出的。

同样，在接收路径上，误差容限是三个 PHY 接口时钟，加上两个 PTP 时钟。通过假设 SFD 数据达到 MAC 的 PHY 接口之前该常量延迟出现在系统（或者链路）中，由于 PHY 接口时钟您可以忽略误差容限。

参考时序时钟的频率范围

时间戳信息在异步时钟域之间传递，也就是，从 MAC 时钟域到 FPGA 时钟域。因此，在两个连续的时间戳采集之间要求一个最小延迟。此延迟是 PHY 接口的 4 个时钟周期和 PTP 时钟的 3 个时钟周期。如果两个时间戳采集之间的延迟短于此延迟，那么 MAC 不对第二个帧进拍时间戳快照。

最大 PTP 时钟频率受限于参考时间的最大解析度 (20 ns 产生 50 MHz) 和 PTP 时钟上逻辑操作的可达到的时序约束。此外，参考时间源的解析度或者间隔大小决定了同步的准确性。因此，更高的 PTP 时钟频率会促成更高的系统性能。

最小 PTP 时钟频率取决于两个连续 SFD 字节之间所需的时间。由于 PHY 接口时钟频率由 IEEE 1588 规范决定，因此正确操作所要求的最小 PTP 时钟频率取决于表 17-3 中所示的 MAC 操作模式和操作速度。

表 17-3. 最小 PTP 时钟频率示例

模式	两个 SFD 之间的最小间隙	最小 PTP 频率
100-Mbps 全双工操作	168 MII 时钟 (128 for 64-byte 帧 + 24 min IFG + 16 preamble)	$(3 * PTP) + (4 * MII) \leq 168 * MII$, 也就是, $\sim 0.5 \text{ MHz} ((168 - 4) * 40 \text{ ns} \div 3 = 2180 \text{ ns period})$
1000-Mbps 半双工操作	24 GMII 时钟 (4 由于冲突在 SFD 之后发送的 jam 码型 + 12 IFG + 8 preamble) (1)	$3 * PTP + 4 * GMII \leq 24 * GMII$, 也就是, 18.75 MHz
表 17-3 注释: (1) 关于 jam 码型的详细信息，请参考 IEEE 网站 (standards.ieee.org/findstds/) 上的 <i>IEEE Std 802.3-2008 Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications</i> .		

IEEE 1588-2008 高级时间戳

- 除了 IEEE 1588-2002 Timestamps 中提及的基本时间戳特性，EMAC 也支持在 IEEE 1588-2008 标准中定义的下面高级时间戳特性。
- 支持 IEEE 1588-2008（版本 2）时间戳格式。
 - 提供一个选项获取所有帧或者只有 PTP 类型帧的时间戳。
 - 提供一个选项获取仅事件消息的时间戳。
 - 提供一个选项根据时钟类型（普通，边界，端到端或者点到点）获得相应的时间戳。
 - 提供一个选项将 EMAC 配置成主或从普通和边界时钟。
 - 识别直接通过 Ethernet 发送的帧中 PTP 消息类型，版本和 PTP 净荷，并发送状态。
 - 提供一个选项以数字或二进制格式测量次秒级时间。

关于 IEEE 1588-2008 标准的详细信息，请参考 IEEE Standards Association 主页上的 *IEEE Standard 1588-2008 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*。†

点到点 (Peer-to-Peer) PTP 透明时钟 (P2P TC) 消息支持

除了 SYNC, Delay Request, Follow-up 和 Delay Response 消息，IEEE 1588-2008 版本也支持 Peer-to-Peer PTP (Pdelay) 消息。

时钟类型

EMAC 支持 IEEE 1588-2008 标准中定义的下面时钟类型：

- 普通时钟 (Ordinary clock)
- 边界时钟 (Boundary clock)
- 端到端透明时钟 (End-to-End transparent clock)
- 点到点透明时钟 (Peer-to-Peer transparent clock)

参考时序源

EMAC 支持 IEEE 1588-2008 标准中定义的下面参考时序源特性：

- 48-Bit 秒域
- 固定的每秒脉冲输出
- 灵活的每秒脉冲输出
- 带外部事件的辅助快照（时间戳）

发送路径功能

仅通过交替（增强）描述符格式支持高级时间戳特性。

接收路径功能

MAC 处理接收的帧来识别有效的 PTP 帧。您可以通过使用下面选项来控制发送到应用程序的时间快照：

- 对所有帧使能时间戳。
- IEEE 1588 版本 2 或版本 1 时间戳使能时间戳。
- 对通过 Ethernet 或者 UDP/IP Ethernet 直接发送的 PTP 帧使能时间戳。
- 对 IPv4 或者 IPv6 的接收帧使能时间戳快照。
- 仅对 EVENT 消息 (SYNC, DELAY_REQ, PDELAY_REQ 或者 PDELAY_RESP) 使能时间戳快照。
- 使节点成为主或从节点，并选择时间戳类型。这控制获取的时间戳用于的消息类型。

DMA 将时间戳返回到发送或接收描述符中的软件。只有 32-byte 交替（加强）描述符支持高级时间戳特性。

辅助快照 (Auxiliary Snapshot)

辅助快照特性使您能够基于外部事件来存储系统的快照（时间戳）。事件应该为边带信号 `ptp_aux_ts_trig_i` 的上升沿。一个 Auxiliary 快照输入是可用的。Auxiliary 快照 FIFO 缓存的深度是 16。

用于所有输入的时间戳都存储在通用 FIFO 缓存中。主机通过读取寄存器来获悉哪一个输入的时间戳可用于 FIFO 缓存顶层的读取。MAC 将这些时间戳存储在 FIFO 缓存中。只有 64 比特的时间戳存储在 FIFO 缓存中。存储时间戳时，MAC 通过一个中断来通知主机。通过 FIFO 缓存寄存器访问来读取时间戳的值。

IEEE 802.3az 节能以太网

EMAC 支持 IEEE 802.3-az，版本 D2.0 标准化的节能以太网 (Energy Efficient Ethernet (EEE))，运行在 10/100/1000 Mbps 速率的 MAC 支持 Energy Efficient Ethernet (EEE)。只有当 EMAC 配置成与全双工模式的 RGMII PHY 接口运行时，才支持 EEE。EEE 不支持半双工模式。



关于 IEEE 802.3az Energy Efficient Ethernet 标准的详细信息，请参考 IEEE 802.3 Ethernet Working Group website(www.ieee802.org/3/)。

EEE 使 MAC 能够运行在低功耗空闲 (Low-Power Idle (LPI)) 模式。Ethernet 链路的两个端点都能够在低链路使用期间禁用功能来降低功耗。MAC 控制系统是否进出 LPI 模式与 PHY 进行通信。

LPI 计时器

EMAC 内部的两个计时器与 LPI 模式相关联：

- LPI Link Status (LS) 计时器
- LPI TW 计时器

LPI LS 计时器以 ms 计算出现链路状态后的超时时间。每次链路关闭时，计数器清零，再次开启时，计数器递增，并到达软件设置的终端计数 (terminal count)。PHY 接口不置位 LPI 码型，除非达到终端计数。这确保了通过远程站建立链路后置位 LPI 码型需要的最短时间。该时间长在 IEEE standard 802.3-az，版本 D2.0 中定义为 1 秒。LPI LS 计时器为 10 比特宽。因此，软件最多可以编程 1023 ms。

LPI TW 计时器以 μ s 计算 LPI 置低后的超时时间。计数器的终端计数是解决传输 TW 的值，该值是 MAC 能够继续正常发送操作前的自动协商时间。MAC 支持以 μ s 为单位的 LPI TW 计时器。LPI TW 计时器是 16 比特宽。因此，软件最多可以编程 65535 μ s。

当发送或接收通道进入或者退出 LPI 状态时，EMAC 生成 LPI 中断。

校验和卸载 (Checksum Offload)

像 TCP 和 UDP 这样的通信协议可以实现校验和域，帮助确定通过网络发送的数据的完整性。由于 Ethernet 最广泛的应用是通过 IP 数据报封装 TCP 和 UDP，因此 EMAC 有一个校验和卸载引擎 (Checksum Offload Engine (COE)) 来支持检验和计算和发送路径中的插入，以及接收路径中的错误检测。所支持的卸载类型如下：

- 发送 IP 头校验和
- 发送 TCP/UDP/ICMP 校验和
- 接收 IP 头校验和

- 接收完全校验和

帧过滤 (Frame Filtering)

EMAC 实现下面类型的接收帧过滤。

源地址或者目的地址过滤

地址过滤模块 (Address Filtering Module) 检查每个输入数据包的目的和源地址域。

- “单播目的地址过滤”
- “多播目的地址过滤器”
- “散列或完美地址过滤器”
- “广播地址过滤器”
- “单播源地址过滤器”
- “倒转过滤选项（倒转最后输出上的过滤器匹配结果）”

单播目的地址过滤

支持高达 128 个 MAC 地址用于单播完美过滤。过滤器对接收单播地址的全部 48 个比特与编程 MAC 地址进行比较，检查是否有任何匹配。默认的 MacAddr0 是始终使能的，其它地址 MacAddr1 - MacAddr127 是通过单独的使能比特进行选择的。对于 MacAddr1 - MacAddr31 地址，您可以在比较相应接收 DA 字节期间屏蔽每个字节。这使能了 DA 的组地址过滤。MacAddr32 - MacAddr127 地址没有掩码控制，MAC 地址的全部 6 个字节都与接收到的 DA 6 个字节进行比较。

在散列过滤模式中，过滤器使用 64-bit 散列表来执行单播地址的有缺陷过滤。它使用接收目的地址 CRC 的 6 个高位比特来索引散列表的内容。0 值选择所选寄存器的 Bit 0，二进制 111111 选择 Hash Table 寄存器的 Bit 63。如果相应比特设为 1，那么多播帧就已经通过散列过滤器；否则，该帧就没有通过散列过滤器。

多播目的地址过滤器

通过编程 MAC 可以使其通过所有多播帧。在 Perfect Filtering 模式中，多播地址与编程的 MAC Destination Address 寄存器 (1 - 31) 进行比较。组地址过滤也被支持。在散列过滤模式中，过滤器使用 64-bit 散列表来执行有缺陷的过滤。对于散列过滤，它使用接收多播地址 CRC 的 6 个高位比特来索引散列表的内容。0 值选择所选寄存器的 Bit 0，二进制 111111 选择 Hash Table 寄存器的 Bit 63。如果相应比特设为 1，那么多播帧就已经通过散列过滤器；否则，该帧就没有通过散列过滤器。

散列或完美地址过滤器

通过配置过滤器可以在它的 DA 匹配散列过滤器或者完美过滤器时通过一个帧。此配置适用于单播以及多播帧。

广播地址过滤器

过滤器不过滤默认模式下的任何广播帧。然而，如果 MAC 被编程为拒绝所有广播帧，那么过滤器丢弃任何的广播帧。

单播源地址过滤器

MAC 也能够基于接收帧的源地址域来执行完美过滤。SA 的组过滤也是被支持的。通过屏蔽地址的一个或多个字节可以过滤一组地址。

倒转过滤选项（倒转最后输出上的过滤器匹配结果）

对于目的和源地址过滤有一个选项可以倒转最后输出上的过滤器匹配结果。单播或多播目的地址过滤器的结果在此模式下倒转。

VLAN 过滤

EMAC 支持两种类型的 VLAN 过滤：

- VLAN 置标过滤（VLAN tag-based filtering）
- VLAN 散列过滤（VLAN hash filtering）

VLAN 置标过滤

在 VLAN 置标帧过滤中，MAC 比较接收帧的 VLAN 标签，并对应用提供 VLAN 帧状态。根据编程的模式，MAC 比较接收 VLAN 标签的 12 个低位比特或者全部 16 个比特来确定完美匹配。如果 VLAN 标签过滤是使能的，那么 MAC 转送 VLAN 置标帧以及 VLAN 标签匹配状态，并丢弃不匹配的 VLAN 帧。您也能够对 VLAN 帧使能倒转匹配。此外，您也能够使能 SVLAN 置标帧以及默认用户性能局域网（Customer Virtual Local Area Network (C-VLAN)）置标帧的匹配。

带 16-bit 散列表的 VLAN 散列过滤

MAC 提供具有 16-bit 散列表的 VLAN 散列过滤。MAC 也支持 VLAN 帧的倒转匹配。在倒转匹配模式中，当一个帧的 VLAN 标签匹配完美或散列过滤器时，应该丢弃数据包。当 VLAN 完美和 VLAN 散列匹配被使能时，如果 VLAN 散列或 VLAN 完美过滤器匹配，那么帧就是匹配的。当倒转匹配被设置时，仅在完美和散列过滤器不匹配时转送数据包。

Layer 3 和 Layer 4 过滤器

Layer 3 过滤是指源地址和目的地址过滤。Layer 4 过滤是指源端口和目的端口过滤。通过以下方式过滤帧：

- 匹配的帧
- 不匹配的帧
- Non-TCP 或者 UDP IP 帧

匹配的帧

MAC 将与所有使能域匹配的帧及其状态转送至应用程序。只有在下面其中一个条件为真时，MAC 才给出匹配的域状态：

- 所有使能的 Layer 3 和 Layer 4 域匹配。
- 至少一个使能域匹配，所有其它域被旁路或禁用。

使用 CSR 组可以最多定义 4 个过滤器，标识为过滤器 0 到 3。当多个 Layer 3 和 Layer 4 过滤器使能时，任何过滤器匹配都被认作是一个匹配。如果 1 个以上的过滤器匹配，那么 MAC 提供最低过滤器的状态，过滤器 0 为最低，过滤器 3 为最高。例如，如果过滤器 0 和过滤器 1 匹配，那么 MAC 给出对应于过滤器 0 的状态。

未匹配的帧

MAC 丢弃那些与使能域不匹配的帧。您可以使用倒转匹配功能来阻止或丢弃一个基于 IP 域的指定 TCP 或者 UDP 的帧，并转送所有其它帧。通过配置 EMAC 可以在帧被丢弃时 EMAC 可以接收具有相应终止状态的部分帧，或者彻底将其丢弃。

Non-TCP 或者 UDP IP 帧

默认情况下，所有 non-TCP 或 UDP IP 帧都从 Layer 3 和 Layer 4 过滤器旁路。您可以有选择性地编程 MAC 来丢弃所有基于 IP 的 non-TCP 或 UDP 的帧。

时钟和复位

Ethernet MAC 控制器使用表 17-4 中所示的时钟。

表 17-4. 时钟

名称	额定频率	功能使用	注释
clk_ref_i	250 Mhz	EMAC 的参考时钟	如果从时钟接口提供，那么时钟是 emac0_clk 或者 emac1_clk
clk_tx_i	125/25/2.5 Mhz	自动协商速度降到 10/100Mbps	
clk_rx_i		PHY 提供对 MAC 的参考	MAC 接收到的全部 PHY 信号同步于此时钟

EEE 的时钟选通 (Clock Gating for EEE)

对于 RGMII PHY 接口，您可以对节能以太网 (EEE) 应用选通 (gate) 发送时钟。关于详细信息，请参考第 17-53 页的“节能以太网的编程指南”。

复位

Ethernet MAC 控制器使用表 17-5 中所示的复位信号。

表 17-5. 复位

名称	额定频率	功能使用	注释
rst_clk_tx_n_o		发送时钟复位输出	用于复位外部 PHY 发送时钟域逻辑
rst_clk_rx_n_o		接收时钟复位输出	用于复位外部 PHY 接收时钟域逻辑

中断

中断的生成是由 EMAC 和外部 PHY 器件中的特定事件导致的。中断状态寄存器指示所有可能触发中断的情况，中断使能寄存器决定哪些中断可以传播。

Ethernet MAC 编程模型

DMA 控制器

DMA 具有独立的发送和接收引擎以及 CSR 空间。发送引擎将数据从系统存储器传递到器件端口或者 MAC 传输层 (MTL)，而接收引擎将数据从器件端口传递到系统存储器。控制器使用描述符在最小的 Host CPU 干预下高效地将数据从源移到目的。DMA 被设计用于面向数据包的数据传输，例如：Ethernet 中的帧。通过编程控制器，可以使控制器对诸如帧发送和接收传输完成的情况和其它正常 / 错误情况中断 Host CPU。

DMA 和 Host 驱动器通过两种数据结构进行通信：

- 控制和状态寄存器 (CSR)
- 描述符列表和数据缓存

关于控制和状态寄存器的详细信息，请参考第 17 - 56 页的“Ethernet MAC 地址映射和寄存器定义”。描述符的相关信息在第 17 - 30 页的“普通描述符”和第 17 - 39 页的“替换或增强的描述符”中有所介绍。



在 RTL 配置过程中，您可以选择替换的描述符结构。此描述符结构中的控制比特被重新分配，以便应用程序能够使用更大容量的缓存 (8 KB)。关于该描述符结构的详细位图信息，请参考第 17 - 39 页的“替换或增强的描述符”。第 17 - 16 页的“DMA 控制器”中的所有描述符都是指默认的描述符结构，而不是指该替换描述符结构。如果使用替换描述符结构，那么要忽略第 17 - 16 页的“DMA 控制器”中的描述符特定映射，而参考替换的描述符特定位图。

DMA 将 MAC 接收的数据帧传递到 Host 存储器中的接收缓存中，并发送 Host 存储器中发送缓存中的数据帧。位于 Host 存储器中的描述符用作对这些缓存的指针。

有两个描述符列表：一个用于接收，另一个用于发送。每个列表的基地址被分别写入 Register 3（接收描述符列表地址寄存器）和 Register 4（发送描述符列表地址寄存器）中。描述符列表是正向链接的（显式或隐式）。最后的描述符可以指回到第一个入口来创建一个环状结构。通过设置链接在接收和发送描述符 (RDES1[24] 和 TDES1[24]) 中的第二个地址来完成描述符的显式链接。描述符列表位于 Host 物理存储器地址空间中。每个描述符最多可以指向两个缓存，这样就能够使用这两个缓存，并对它们进行物理寻址，而不是使用存储器中的连续缓存。

数据缓存位于 Host 物理存储器空间中，包含一个整帧或者帧的一部分，但不能多于一个帧。缓存只包含数据，缓存状态在描述符中维护。数据链 (data chaining) 是指跨越多个数据缓存的帧。当检测到帧结束 (end-of-frame) 时，DMA 跳到下一个帧缓存。数据链能够被使能或禁用。

图 17 - 5 和图 17 - 6 显示了描述符环状和链式结构。

图 17 - 5. 描述符环状结构

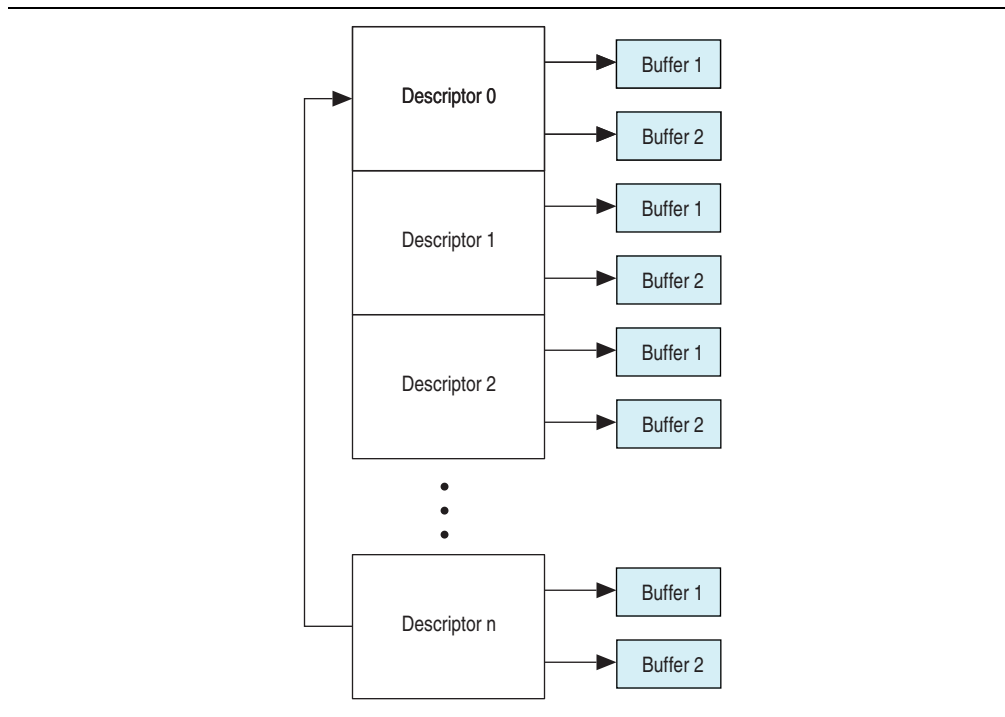
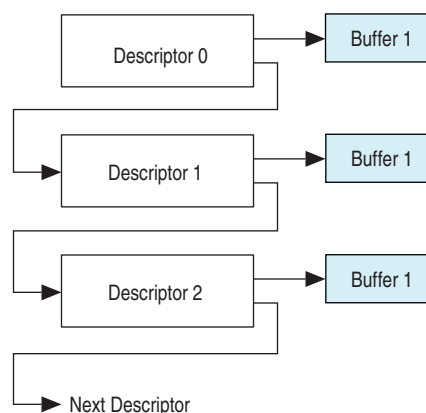


图 17-6. 描述符链式结构



初始化

EMAC 的初始化如下：

1. 写入 Register 0(总线模式寄存器)来设置 Host 总线访问参数。
2. 写入 Register 7(中断使能寄存器)来屏蔽不必要的中断原因。
3. 创建发送和接收描述符列表，然后写入 DMA Register 3(接收描述符列表地址寄存器)和 Register 4(发送描述符列表地址寄存器)，对 DMA 提供每个列表的起始地址。
4. 对所需的过滤选项写入 Register 1(MAC 帧过滤器)，Register 2(散列表高寄存器)和 Register 3(散列表低寄存器)。
5. 写入 Register 1(MAC 帧管理器)来配置操作模式并使能发送操作(Bit 3: 发送器使能)。根据(从 PHY 读取的)自动协商结果来设置 PS 和 DM 比特。
6. 写入 Register 6(操作模式寄存器)设置 Bits 13 和 1 来开始发送和接收。
7. 写入 Register 0(MAC 配置寄存器)来使能接收操作(Bit 2: 接收器使能)。

发送和接收引擎进入 Running 状态，并试图从相应的描述符列表中获取描述符。接收和发送引擎然后开始处理接收和发送操作。发送和接收过程互不依赖，可分别开始或停止。

主机总线突发访问 (Host Bus Burst Access)

通过 Register 0(总线模式寄存器)的 FB 比特可以将 DMA 配置成试图在主接口上执行固定长度突发传递。Register 0(总线模式寄存器)的 PBL 字段(Bits [13:8])指示并限制最大突发长度。始终以最大可能的突发长度(受限于 PBL 或 $16 * 8 / \text{总线宽度}$)访问接收和发送描述符来读取 16 字节。

仅当 MTL 发送 FIFO 缓存中有足够的空间用于存储配置的突发时(或者帧结束前的字节数少于配置的突发长度时)，发送 DMA 才启动数据传递。DMA 指示起始地址和到主接口所需要的传递数量。当接口配置成固定长度突发时，它使用 INCR4, 8 或 16 与 SINGLE 传输的最佳组合来传递数据。否则(非固定长度突发)，使用 INCR(未定义长度)和 SINGLE 传输传递数据。

仅当 MTL 接收 FIFO 缓存中有足够的数据用于配置的突发时，或者在接收 FIFO 缓存中检测到帧结束(当它短于配置的突发长度时)时，接收 DMA 才启动数据传递。DMA 指示起始地址和到主接口所需要的传递数量。当接口配置成固定长度突发时，它使用 INCR4, 8 或 16 与 SINGLE 传输的最佳组合来传递数据。如果固定突发在接口上结束前达到帧结束，那么会执行伪传递来完成固定突发。否则(Register 0(总线模式寄存器)的 FB 比特被复位)，它将使用 INCR(未定义长度)和 SINGLE 传输传递数据。

当接口配置为地址对齐的差拍 (beat) 时，两个 DMA 引擎都确保了启动的第一个突发传递少于或等于配置的 PBL 大小。因此，所有接下来的差拍都开始于与配置的 PBL 对齐的地址。由于接口不支持大于 INCR16 的尺寸，因此 DMA 仅对齐最大尺寸为 16 (PBL > 16) 差拍的地址。

主机数据缓存对齐 (Host Data Buffer Alignment)

发送和接收数据缓存对起始地址对齐没有任何的限制。例如，在具有 32-bit 存储器的系统中，缓存的起始地址能够对齐于任意四个字节。然而，DMA 始终通过对齐于总线宽度的地址和不需要的字节通道的伪数据来启动传递。这通常发生在 Ethernet 帧的开始或结束传递期间。软件驱动器应该根据缓存的起始地址和帧大小来丢弃伪字节。

示例：缓存读

如果发送缓存地址是 0x0000FF2 (对于 32-bit 数据总线)，并且需要传输 15 个字节，那么 DMA 从地址 0x0000FF0 读取 5 个全字，但当传递数据到 MTL 发送 FIFO 缓存时，多余的字节 (头两个字节) 被丢弃或被忽略。同样地，最后一个传递的最后 3 个字节也被忽略。DMA 始终确保它传递一个完全 32-bit 数据到 MTL 发送 FIFO 缓存，除非它是帧结束 (end-of-frame)。

示例：缓存写

如果接收缓存地址是 0x0000FF2 (对于 64-bit 数据总线)，并且需要传输接收帧的 16 个字节，那么 DMA 从地址 0x0000FF0 写入 3 个全字。但是第一个传输的头 2 个字节和第三个传输的最后 6 个字节有伪数据。

缓存大小计算

DMA 不更新发送和接收描述符中的大小域。DMA 只更新描述符的状态域 (RDES 和 TDES)。驱动器必须执行大小计算。

发送 DMA 传递确切数量的字节 (由 TDES1 的缓存大小域表明) 到 MAC。如果一个描述符标记为第一个 (first) (TDES1 的 FS 比特被设置)，那么 DMA 将缓存的第一个传递标记为帧的开始。如果一个描述符标记为最后 (last) (TDES1 的 LS 比特)，那么 DMA 将此数据缓存的最后传递标记为到 MTL 的帧结束。

接收 DMA 传递数据到缓存直到缓存变满，或者接收到来自 MTL 的帧结束。如果一个描述符没有标记为最后 (last) (RDES0 的 LS 比特)，那么描述符的相应缓存变满，并且当此描述符的 FS 比特被设置时，缓存中的有效数据由其缓存大小域减去数据缓存指针偏移准确计算得出。当数据缓存指针对齐于数据总线宽度时，偏移为 0。如果一个描述符标记为最后，那么缓存可能未满 (如 RDES1 中的缓存大小所指示的)。要计算该最后缓存中的有效数据量，驱动器必须读取帧长 (RDES0[29:16] 的 FL 比特) 并减去此帧中之前缓存的大小总和。接收 DMA 总是使用一个新的描述符来传递下一个帧的起始。



甚至当接收缓存的起始地址没有与系统总线的数据宽度对齐时，系统也应该分配一个与系统总线宽度对齐的接收缓存。例如，如果系统从地址 0x1000 开始分配一个 1,024-byte (1 KB) 接收缓存，那么软件能够编程接收描述符中的缓存起始地址，使其有一个 0x1002 偏移。接收 DMA 将帧及头两个位置 (0x1000 和 0x1001) 中的伪数据写入到此缓存中。实际的帧从位置 0x1002 写入。因此，尽管缓存容量编程为 1,024 字节，但由于起始地址偏移，此缓存中的实际有用空间为 1,022 字节。

发送

发送功能使用发送描述符，第 17 - 30 页的“发送描述符”中有详细说明。

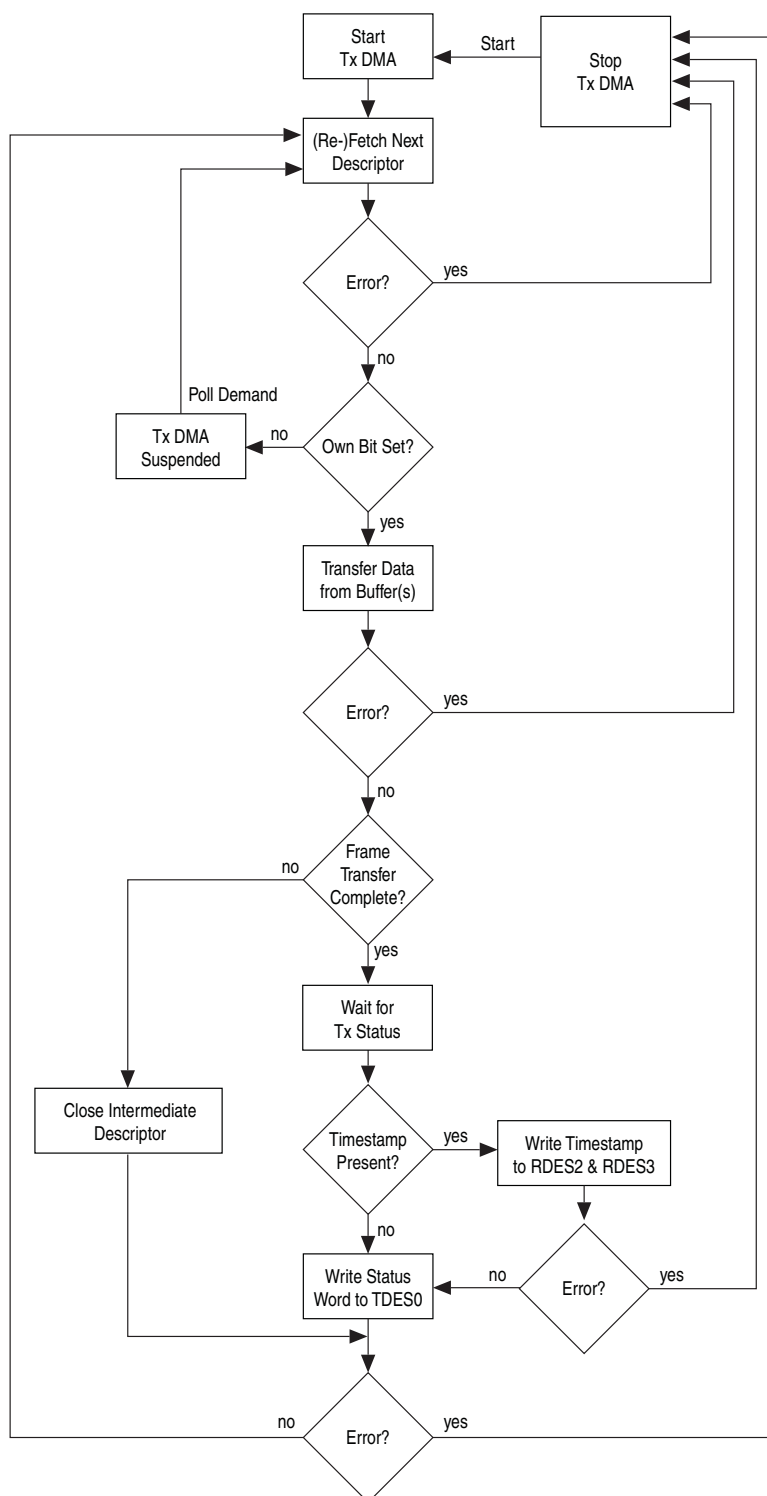
TX DMA 操作：默认 (Non-OSF) 模式

默认模式下的发送 DMA 引擎按如下运行：

1. 主机建立发送描述符 (TDES0-TDES3)，通过 Ethernet 帧数据建立相应数据缓存后设置 Own 比特 (TDES0[31])。
2. 当 Register 6 (操作模式寄存器) 的 Bit 13 (ST) 被设置时，DMA 进入 Run 状态。
3. 当置于 Run 状态时，DMA 轮询发送描述符列表来获得需要发送的帧。轮询开始后，它以序列描述符环状顺序或者链式顺序继续运行。如果 DMA 检测到一个记为 Host (TDES0[31] = 0) 拥有的描述符，或者出现一个错误情况，那么发送暂停，Register 5 (状态寄存器) 的 Bit 2 (发送缓存不可用) 以及 Bit 16 (正常中断汇总) 被设置。发送引擎执行步骤 9。
4. 如果获得的描述符标记为 DMA (TDES0[31] = 1) 拥有，那么 DMA 对获取描述符的发送数据缓存地址进行解码。
5. DMA 从 Host 存储器获取发送数据，并将这些数据传输到 MTL 用于发送。
6. 如果一个 Ethernet 帧通过多个描述符中数据缓存存储，那么 DMA 将关闭中间描述符并获取下一个描述符。重复步骤 3，4 和 5，直到 Ethernet 帧结束数据传递到 MTL。
7. 帧发送完成后，如果 IEEE 1588 时间戳被使能用于帧 (如发送状态指示的)，那么从 MTL 获得的时间戳值将被写入到包含帧结束缓存的发送描述符 (TDES2 和 TDES3)。接下来，状态信息被写入到该发送描述符 (TDES0) 中。因为 Own 比特在此步骤被清零，所以 Host 现在拥有该描述符。如果对此帧没有使能时间戳，那么 DMA 不改变 TDES2 和 TDES3 的内容。
8. 帧 (在此帧最后描述符中设置了 Interrupt on Completion (TDES1[31])) 的发送完成后，Register 5 (状态寄存器) 的 Bit 0 (发送中断) 被设置。DMA 引擎然后返回到步骤 3。
9. 在 Suspend 状态中，当 DMA 接收到 Transmit Poll 命令，并且 Underflow Interrupt Status 比特清零时，DMA 尝试重新获得描述符 (因此返回到步骤 3)。

默认模式下的 TX DMA 发送流程如图 17-7 所示。

图 17 - 7. 默认模式下的 TX DMA 操作



TX DMA 操作：OSF 模式

置于 Run 状态时，发送过程能够同时获取两个帧，而不需要关闭第一个 Status 描述符 [如果 Register 6 (操作模式寄存器) 中的 Bit 2(OSF) 被设置]。由于发送过程完成传递第一个帧，因此它立即对第二个帧轮询发送描述符列表。如果第二个帧是有效的，那么发送过程在写入第一个帧的状态信息前传递此帧。

在 OSF 模式中，Run 状态发送 DMA 按下面顺序运行：

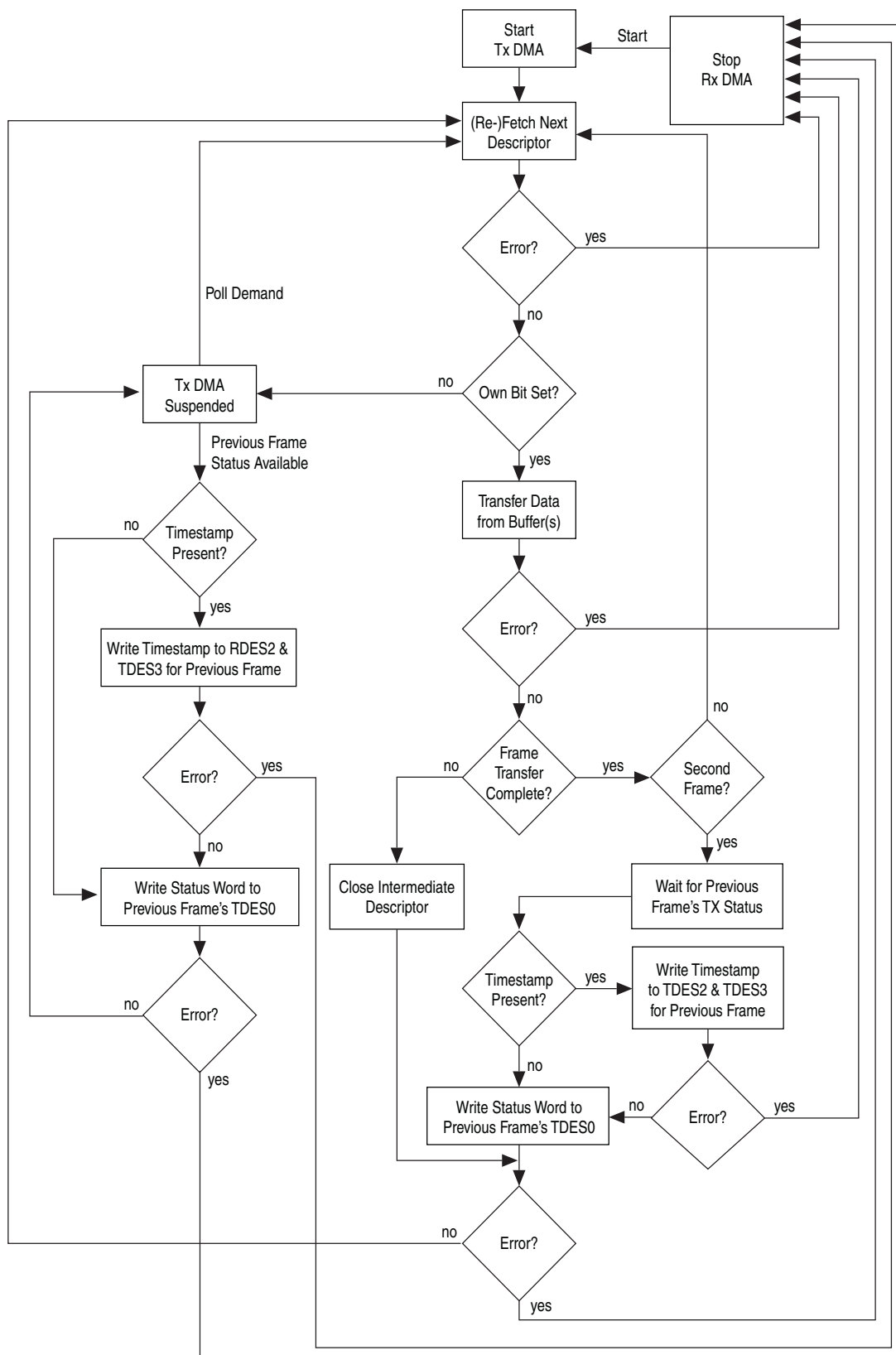
1. DMA 按照第 17 - 20 页的 “TX DMA 操作：默认 (Non-OSF) 模式” 中的步骤 1 - 6 运行。
2. DMA 获取下一个描述符，而不关闭前一个帧的最后描述符。
3. 如果 DMA 具有所获取的描述符，那么 DMA 对此描述符中的发送缓存地址进行解码。如果 DMA 不具有此描述符，那么 DMA 将进入 Suspend 模式，并跳到步骤 7。
4. DMA 从 Host 存储器获取发送帧，并将此帧传输到 MTL，直到传递了帧结束数据。如果此帧在多个描述符之间被分离，则关闭中间描述符。
5. DMA 等待前一个帧的帧发送状态和时间戳。一旦状态可用，DMA 就将时间戳写入到 TDES2 和 TDES3 (如果采集到了这样的时间戳 (由状态比特指示))。DMA 然后将状态 (一个清零的 Own 比特) 写入到相应的 TDES0，从而关闭描述符。如果时间戳对于前一个帧没有使能，那么 DMA 不改变 TDES2 和 TDES3 的内容。
6. 如果使能，发送中断被设置，DMA 获取下一个描述符，那么执行步骤 3 (当 Status 正常时)。如果前一个发送状态显示一个下溢错误，那么 DMA 进入 Suspend 模式 (步骤 7)。
7. 在 Suspend 模式中，如果从 MTL 接收到暂挂状态和时间戳，那么 DMA 将时间戳 (如果对当前帧使能) 写入到 TDES2 和 TDES3 中，那么写入状态到相应的 TDES0 中，然后设置相关中断并返回到 Suspend 模式。
8. 只有在接收到 Transmit Poll 命令 (Register 1 (Transmit Poll Demand Register)) 之后，DMA 才能够退出 Suspend 模式，并进入 Run 状态 (根据暂挂状态来执行步骤 1 或步骤 2)。



由于 DMA 在关闭当前描述符之前提前获取下一个描述符，因此描述符链中应该有两个以上不同的描述符用于正确的操作。

图 17-8 显示了基本流程。

图 17 - 8. OSF 模式的 TX DMA 操作



发送帧处理

发送 DMA 期望数据缓存包含完整的 Ethernet 帧，不包括前同步 (preamble)，垫 (pad) 字节和 FCS 域。DA，SA 和 Type/Len 域包含有效数据。如果发送描述符指示 MAC 必须禁用 CRC 或 PAD 插入，那么缓存必须有完整的 Ethernet 帧（不包括 preamble），包括 CRC 字节。

帧可以是数据链式的并能够跨越几个缓存。帧必须分别被 First Descriptor (TDES1[29]) 和 Last Descriptor (TDES1[30]) 界定。

发送开始时，First Descriptor 必须有 (TDES1[29]) 组。此情况发生时，帧数据从 Host 缓存传输到 MTL 发送 FIFO 缓存。与此同时，如果当前帧的 Last Descriptor (TDES1[30]) 被清零，那么发送过程会试图获取下一个描述符。发送过程期望此描述符的 TDES1[29] 被清零。如果 TDES1[30] 被清零，那么它表明一个中间缓存。如果 TDES1[30] 被设置，那么它表明帧的最后缓存。

帧的最后缓存发送后，DMA 将最终状态信息写回到描述符（其最后段在 Transmit Descriptor 1 (TDES1[30] 中设置) 的 Transmit Descriptor 0 (TDES0) 字中。此时，如果 Interrupt on Completion (TDES1[31]) 被设置，那么 Register 5 (状态寄存器) 的 Bit 0 (发送中断) 被设置，获取下一个描述符并重复过程。

MTL 发送 FIFO 缓存已经达到可编程发送阈值 (Register 6 (操作模式寄存器) 的 Bits [16:14])，或者 FIFO 缓存中包含一个全名之后开始实际帧发送。此外也有一个 Store and Forward Mode (Register 6 (操作模式寄存器) 的 Bit 21) 选项。当 DMA 完成传递帧时，描述符被释放 (Own 比特 TDES0[31] 清零)。



为确保一个帧和它下一个帧的正确发送，您必须制定一个非零缓存容量，用于发送描述符，其 Last Descriptor (TDES1[30]) 已被设置。

发送轮询暂停

发送轮询可以被下面其中一个种情况暂停：

- DMA 检测到一个 Host (TDES0[31]=0) 拥有的描述符。为继续，驱动器必须将描述符的拥有权给予 DMA，然而发出一个 Poll Demand 命令。
- 由于下溢而检测到发送错误时，帧发送会被终止。相应的 Transmit Descriptor 0 (TDES0) 比特被设置。

如果因为第一种情况 DMA 进入 SUSPEND 状态，那么 Register 5 (状态寄存器) 的 Bit 16 (正常中断汇总) 和 Bit 2 (发送缓存不可用) 都被设置。如果出现第二种情况，那么 Register 5 (状态寄存器) 的 Bit 15 (异常中断汇总) 和 Bit 5 (发送下溢) 都被设置，信息被写入到 Transmit Descriptor 0，从而导致暂挂 (suspension)。

在两种情况下，发送列表中的位置被保留。被保留的位置是被 DMA 关闭的最后描述符后面的描述符位置。

驱动器必须在纠正暂挂原因后显式地发出 Transmit Poll Demand 命令。

接收

接收功能使用接收描述符，在第 17 - 33 页的“接收描述符”中有详细介绍。

接收 DMA 引擎的接收序列如第 17 - 27 页的图 17 - 9 中所示，执行如下：

1. 主机建立接收描述符 (RDES0-RDES3) 并设置 Own 比特 (RDES0[31])。
2. 当 Register 6 (操作模式寄存器) 的 Bit 1 (SR) 被设置时, DMA 进入 Run 状态。当置于 Run 状态时, DMA 轮询接收描述符列表, 尝试获取可用描述符。如果获取的描述符不可用 (被主机所有), 那么 DMA 进入 Suspend 状态, 跳到步骤 9。
3. DMA 对获取描述符的接收数据缓存地址进行解码。
4. 输入帧被处理并布置在获取描述符的数据缓存中。
5. 当缓存变满, 或者帧传输完成时, 接收引擎获取下一个描述符。
6. 如果当前帧传输完成, 那么 DMA 执行步骤 7。如果 DMA 不具有下一个获取的描述符, 并且帧传输没有完成 (还没有传输 EOF), 那么 DMA 设置 RDES0 中的 Descriptor Error 比特 (除非在 Register 6 (操作模式寄存器) 的 Bit 24 上禁用 flushing)。DMA 关闭当前描述符 (清零 Own 比特), 并将其标记为中间 (intermediate), 通过清零 RDES0 中的 Last Segment (LS) 比特 (如果没有禁用 flushing, 那么将其标记为 Last Descriptor), 然后执行步骤 8。如果 DMA 具有下一个描述符, 但当前帧传输没有完成, 那么 DMA 关闭标记为 intermediate 的当前描述符, 返回到步骤 4。
7. 如果使能了 IEEE 1588 时间戳, 那么 DMA 将时间戳 (如果可用) 写入到当前描述符的 RDES2 和 RDES3。DMA 然后从 MTL 获取接收帧的状态, 并将状态字写入到当前描述符的 RDES0 中, 清零 Own 比特并设置 Last Segment 比特。
8. 接收引擎检查最新描述符的 Own 比特。如果主机具有该描述符 (Own 比特为 0), 那么 Register 5 (状态寄存器) 的 Bit 7 (接收缓存不可用) 会被设置, DMA 接收引擎进入 Suspended 状态 (步骤 9)。如果 DMA 具有此描述符, 那么引擎返回到步骤 4 并等待下一个帧。
9. 接收引擎进入 Suspend 状态之前, 接收 FIFO 缓存中的部分帧被刷新。您可以使用 Register 6 (操作模式寄存器) 的 Bit 24 来控制刷新 (flushing)。
10. 当发出 Receive Poll 命令或者 MTL 的接收 FIFO 缓存中有下一个帧的起始时, 接收 DMA 退出 Suspend 状态。引擎执行步骤 2 并获取下一个描述符。

接收描述符采集

接收引擎始终试图从输入帧中采集一个额外的描述符。如果下面的任意条件得到满足，就会尝试描述符采集：

- Register 6(操作模式寄存器)的 Bit 1(开始或结束接收)置于 Run 状态后被立即设置。
- 帧的当前传输结束前，当前描述符的数据缓存已满。
- 控制器已经完成帧接收，但当前接收描述符还没有关闭。
- 由于主机拥有 (host-owned) 缓存 (RDES0[31] = 0)，接收过程已被暂挂，并接收到一个新的帧。
- 已经发出接收轮询命令。

接收帧处理

仅当帧通过地址过滤器，并且帧大小大于或等于对 MTL 的接收 FIFO 缓存设置的可配置阈值字节，或者在 Store-and-Forward 模式下将完整的帧写入到 FIFO 缓存中时，MAC 才将接收的帧传输到 Host 存储器。

如果帧无法通过地址过滤，那么它在其本身的 MAC 模块中丢弃（除非 Register 1 (MAC 帧过滤器) 的 Bit 31 (接收全部) 被设置）。少于 64 字节的帧（由于冲突或提取终止）能够从 MTL 接收 FIFO 缓存中移除。

接收到 64 个（可配置阈值）字节之后，MTL 模块请求 DMA 模块开始传递帧数据到当前描述符指定的接收缓存中。DMA Host 接口准备好接收数据传递后（如果 DMA 没有从主机获取发送数据），DMA 设置 First Descriptor (RDES0[9]) 来对帧分界。当 Own(RDES[31]) 比特复位成 0 时释放描述符，要么 Data 缓存填满，要么帧的最后段传输到接收缓存。如果帧包含在单一描述符中，那么 Last Descriptor (RDES[8]) 以及 First Descriptor (RDES[9]) 被设置。

DMA 获取下一个描述符，设置 Last Descriptor(RDES[8]) 比特，并释放前一个帧描述符中的 RDES0 状态比特。DMA 然后设置 Register 5（状态寄存器）的 Bit 6（接收中断）。重复同样的过程，直到 DMA 遇到一个标记为主机拥有的描述符。如果出现此情况，接收过程设置 Register 5（状态寄存器）的 Bit 7(接收缓存不可用)，然后进入 Suspend 状态。接收列表中的位置被保留。

接收过程暂挂

如果接收 Process 处于 Suspend 状态时一个新的接收帧到达，那么 DMA 重新获取 Host 存储器中的当前描述符。如果现在 DMA 具有描述符，那么接收过程重新进入 Run 状态，并开始帧接收。如果主机仍然具有描述符，默认情况下，DMA 丢弃 MTL RX FIFO 缓存顶层的当前帧，并增加错过的帧计数器。如果一个以上的帧存储在 MTL EX FIFO 缓存中，那么重复此过程。

通过禁用 Flushing(Register 6(操作模式寄存器)的 Bit 24) 可以避免 MTL EX FIFO 缓存顶层帧的丢弃或刷新。在这样的条件下，接收过程设置 Receive Buffer Unavailable 状态，并返回到 Suspend 状态。

中断

中断作为各种事件的结果而生成。DMA Register 5(状态寄存器)包含可能导致中断的全部比特。Register 7(中断使能寄存器)包含每个能导致中断的事件的使能比特。

正如 Register 5（状态寄存器）中所描述的，有两种中断：正常（Normal）和异常（Abnormal）。通过写入一个 1 到相应比特位置来对中断清零。当一个组中的所有使能中断清零时，相应的汇总比特（summary bit）也被清零。当两个汇总比特都清零时，sbd_intr_o 中断信号被置低。如果 MAC 是中断置位的原因，那么 Register 5（状态寄存器）的所有 GLI, GMI, GPI, TTI 或者 GLPII 都被置高，如图 17-10 所示。

图 17-10. sbd_intr_o 生成 (1)

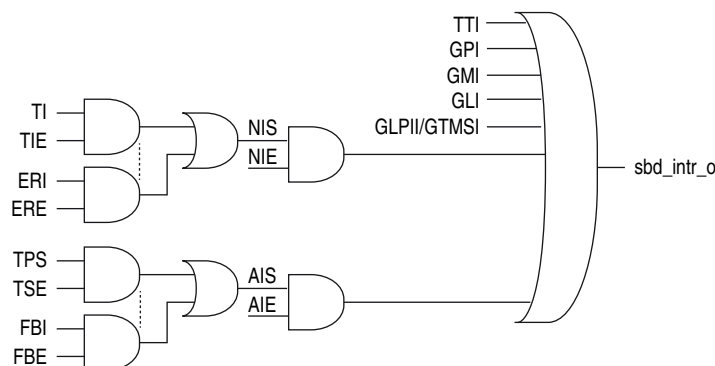


图 17-10 注释：

(1) NIS 和 AIS 信号被寄存。



Register 5（状态寄存器）是中断状态寄存器。仅当 Register 7（中断使能寄存器）中的相应中断使能比特被设置时，中断管脚（sbd_intr_o）才会由于该状态寄存器中的任意事件而被置位。

中断不排序，如果中断事件出现在驱动器响应之前，那么不会生成其它中断。例如，Register 5（状态寄存器）的 Bit 6（接收中断）表明一个或多个帧被传输到 Host 缓存中。驱动器必须扫描所有的描述符，从最后一个记录的位置到 DMA 具有的第一个位置。

对于同步的多个事件，一个中断只生成一次。驱动器必须扫描 Register 5（状态寄存器）来查找中断的原因。驱动器清零 Register 5（状态寄存器）中的相应比特后，除非出现新的中断事件，否则不会再次生成中断。例如，控制器生成 Register 5（状态寄存器）的 Bit 6（接收中断），驱动器开始读取 Register 5（状态寄存器）。接下来，Register 5（状态寄存器）的 Bit 7（接收缓存不可用）出现。驱动器清除接收中断。即使那时，由于活动或未决的 Receive Buffer Unavailable 中断，sbd_intr_o 信号也没有置低。

Register 9（接收中断看门狗计时器寄存器）的 Bits 7:0（中断计时器）用于接收中断的灵活控制。当使用非零值对该中断计时器编程时，在没有置位接收中断的情况下（因为没有使能第 17-35 页的表 17-12 中相应 Receive Descriptor (RDES1[31]) 中的接收中断），RX DMA 一旦完成接收帧到系统存储器的传输，该中断计时器就被激活。当计时器依照编程的值运行完毕时，如果 Register 7（中断使能寄存器）中的相应 RI 被使能，那么 RI 比特被设置，并且中断被置位。当一个帧被传递到存储器中时，计时器在到期前被禁用，RI 被设置，因为它对于该描述符是使能的。

DMA 的错误响应

对于由 DMA 通道启动的任何数据传输，如果从接口以错误响应回复，那么该 DMA 将停止所有操作并更新 Register 5（状态寄存器）中的错误比特和 Fatal Bus Error 比特。DMA 控制器只能在软复位或硬复位 EMAC 并重新初始化 DMA 之后才能继续操作。

描述符概述

此部分描述了 HPS EMAC DMA 描述符。

Ethernet 子系统上的 DMA 根据第 17 - 16 页的“DMA 控制器”中介绍的描述符链接列表来传输数据。描述符在系统存储器中创建。EMAC DMA 支持下面两种类型描述符：

- 普通描述符 — 这是默认的 4 DWORDS 描述符格式。关于此描述符类型的信息，请参考第 17 - 30 页的“普通描述符”。
- 加强描述符 — 这是可选的 8 DWORDS (32 bytes) 描述符格式。关于此描述符类型的信息，请参考第 17 - 39 页的“替换或增强的描述符”。

每个描述符包含两个缓存，两个字节数 (byte-count) 缓存和两个地址指针，使适配器端口与各种类型的存储器管理方案兼容。一旦配置，就不能改变描述符结构。

描述符字节顺序 (Descriptor Endianness)

描述符地址必须对齐于使用的总线宽度 (Word, DWord 或者分别为 32-bit, 64-bit 或 128-bit 总线的 LWord)。数据总线配置成 little-endian。



本小节中的图显示了普通发送和接收描述符。如果使用增强描述符，那么必须确保在系统存储器中相应地创建描述符。关于增强描述符的详细信息，请参考第 17 - 39 页的“替换或增强的描述符”。

普通描述符

这是默认的 4 DWORDS 描述符格式。当没有使能高级功能（例如：IEEE 1588 高级时间戳）使能时，您可以使用该描述符格式。

发送描述符

EMAC 中的 DMA 需要至少一个描述符用于发送帧。除了两个缓存，两个字节数缓存和两个地址指针，发送描述符还有控制域，在每个发送帧基础上用于控制 MAC 操作。

发送描述符 0 (TDES0)

TDES0 包含发送的帧状态和描述符拥有权的信息。

表 17 - 6. 发送描述符 0 (1/2)

比特	说明
31	OWN: Own 比特 (Own Bit) 该比特设置时表明 DMA 拥有描述符。此比特被复位时，表明 Host 拥有描述符。当 DMA 完成帧发送，或者当分配在描述符中的缓存为空时，DMA 对该比特清零。帧的 First 描述符的 ownership 比特应该在属于同一帧的所有接下来的描述符被设置之后设置。这避免了获取描述符与驱动器设置 ownership 比特之间可能的争用情况。
30:18	保留。
17	TTSS: TX 时间戳状态 (TX Timestamp Status) 该状态比特表明已经采集到相应发送帧的时间戳。当该比特被设置时，TDES2 和 TDES3 具有采集的发送帧的时间戳值。仅当描述符中的 Last Segment 控制比特 (TDES1[30]) 被设置时，该域才有效。
16	IHE: IP 头错误 (IP Header Error) 该比特设置时表明 Checksum Offload 引擎检测到一个 IP 头错误，因此没有对校验和插入修改已发送的帧。

表 17 - 6. 发送描述符 0 (2/2)

比特	说明
15	<p>ES: 错误汇总</p> <p>表明下面比特的逻辑 OR:</p> <ul style="list-style-type: none"> ■ TDES0[16]: IP 头错误 ■ TDES0[14]: Jabber 超时 ■ TDES0[13]: 帧刷新 ■ TDES0[12]: 净荷校验和错误 ■ TDES0[11]: 载体丢失 ■ TDES0[10]: 无载体 ■ TDES0[9]: 迟冲突 ■ TDES0[8]: 过度冲突 ■ TDES0[2]: 过度延迟 ■ TDES0[1]: 下溢错误
14	<p>JT: Jabber 超时 (Jabber Timeout)</p> <p>该比特设置时表明 MAC 发送器已经经历了 jabber 超时。仅当 Register 0 (MAC 配置寄存器) 的 JD 比特没有设置时, 才设置该比特。</p>
13	<p>FF: 帧刷新 (Frame Flushed)</p> <p>该比特设置时表明由于 CPU 发出的软件刷新命令而使 DMA 或 MAC 传输层 (MTL) 对帧进行刷新。</p>
12	<p>12 PCE: 净荷校验和错误 (Payload Checksum Error)</p> <p>该比特设置时表明 Checksum Offload 引擎有故障, 没有插入校验和到封装的 TCP, UDP 或 ICMP 净荷。此故障可以由欠缺的字节 (由 IP Header 的 Payload Length 域指明) 导致, 或者在还没有计算校验和的情况下 MTL 就开始传递帧到 Store-and-Forward 模式的 MAC 发送器。第二种错误情况仅出现在发送 FIFO 缓存深度少于 Ethernet 帧 (被发送以避免死锁) 的长度, 当 FIFO 缓存变满时, 即便在 store-and-forward 模式下, MTL 开始传输帧。</p>
11	<p>LC: 载体丢失 (Loss of Carrier)</p> <p>该比特设置时表明在帧发送期间出现了载体丢失 (Loss of Carrier) (也就是, gmii_crs_i 信号在帧发送期间在一个或多个发送时钟周期内无效)。这只对无冲突情况下的发送帧有效, 和当 MAC 运行在半双工模式下时有效。</p>
10	<p>NC: 无载体 (No Carrier)</p> <p>该比特被设置时表明发送期间 PHY 的载波监听信号没有被置位。</p>
9	保留
8	<p>EC: 过度冲突 (Excessive Collision)</p> <p>设置时, 当试图发送当前帧时, 在 16 个连续冲突后终止该比特发送。如果 Register 0 (MAC 配置寄存器) 中的 Bit 9 (禁用入口) 被设置, 那么该比特在第一个冲突后被设置, 帧的发送被终止。</p>
7	<p>VF: VLAN 帧 (VLAN Frame)</p> <p>该比特设置时表明发送的帧是一个 VLAN 类型的帧。</p>
6:3	<p>CC: 冲突计数 (Collision Count)</p> <p>此 4-bit 计数器值表明帧发送前出现的冲突数量。Excessive Collisions 比特 (TDES0[8]) 被设置前, 该数无效。</p>
2	<p>ED: 过度延迟 (Excessive Deferral)</p> <p>设置时, 如果 Register 0 (MAC 配置寄存器) 中的 Bit 4 (延迟检查) 被设置成高电平, 那么该比特表明由于超过 24,288 比特时间的过度延迟 (1000-Mbps 模式或者 Jumbo 帧使能模式下的 155,680 比特时间) 发送已经结束。</p>
1	<p>UF: 下溢错误 (Underflow Error)</p> <p>设置时, 该比特表明由于 Host 存储器中的数据迟到到达, MAC 终止了帧。Underflow Error 表明 DMA 发送数据时碰到一个空的发送缓存。发送过程进入暂挂状态, 并设置 Transmit Underflow (Register 5[5]) 以及 Transmit Interrupt (Register 5[0])。</p>
0	<p>DB: 延迟比特 (Deferred Bit)</p> <p>设置时, 该比特表明由于出现载体, MAC 推迟发送。该比特仅在半双工模式下有效。</p>

发送描述符 1 (TDES1)

TDES1 包含缓存大小和其它比特，用于控制描述符链或者环，以及正在传输的帧。



关于计算缓存大小的详细信息，请参考第 17 - 19 页的“缓存大小计算”。

表 17 - 7. 发送描述符 1

Bit	Description
31	IC: 完成时中断 (Interrupt on Completion) 设置时，该比特在当前帧发送后设置 Transmit Interrupt (Register 5[0])。
30	LS: 最后区段 (Last Segment) 设置时，该比特表明缓存包含帧的最后区段。当该比特设置时，TBS1 或者 TBS2 域应该有一个非零值。
29	FS: 第一区段 (First Segment) 设置时，该比特表明缓存包含帧的第一个区段。
28:27	CIC: 校验和插入控制 (Checksum Insertion Control) 这些比特控制 Ethernet 帧中的校验和的插入，Ethernet 帧封装 TCP, UDP 或者基于 IPv4 或 IPv6 的 ICMP。 <ul style="list-style-type: none"> 0: 无操作。校验和引擎被旁路。 1: 插入 IPv4 头校验和。当帧封装一个 IPv4 数据报时，使用该值插入 IPv4 头校验和。 2: 插入 TCP/UDP/ICMP 校验和。只能通过 TCP, UDP 或者 ICMP 计算校验和，TCP, UDP 或者 ICMP 伪头校验和应该在相应的输入帧的 Checksum 域。如果封装的数据报符合 IPv4，也将插入一个 IPv4 头校验和。 3: 插入一个在引擎中完整计算出的 TCP/UDP/ICMP 校验和。换句话说，TCP, UDP 或者 ICMP 伪头包括在校验和计算中，输入帧的相应 Checksum 域有一个非零值。如果封装的数据报符合 IPv4，也将插入一个 IPv4 头校验和。 校验和引擎检测 TCP, UDP 或者 ICMP 区段是否在 IPv4 或者 IPv6 中封装，并相应地处理器数据。
26	DC: 禁止 CRC (Disable CRC) 设置时，MAC 不将 CRC 添加到发送帧的结尾。该比特仅在 first segment (TDES1[29]) 设置时有效。
25	TER: 发送环结束 (Transmit End of Ring) 设置时，该比特表明描述符列表到达其最后的描述符。这返回到列表的基地址，创建一个描述符环。
24	TCH: 第二个地址链接 (Second Address Chained) 设置时，该比特表明描述符中的第二个地址是下一个描述符地址，而不是第二个缓存地址。当 TDES1[24] 设置时，TBS2 (TDES1[21 - 11]) 的值为 “don’t care”。TDES1[25] 优先于 TDES1[24]。
23	DP: 禁止 Padding (Disable Padding) 设置时，MAC 不会自动添加 padding 到短于 64 字节的帧。当复位该比特时，DMA 自动添加 padding 和 CRC 到短于 64 字节的帧，即使 DC (TDES1[26]) 比特的状态，CRC 域也被添加。该比特仅在 first segment (TDES1[29]) 设置时有效。
22	TTSE: 发送时间戳使能 (Transmit Timestamp Enable) 设置时，该比特对于描述符参考的发送帧使能 IEEE1588 硬件时间戳。该比特仅在 First Segment 控制比特 (TDES1[29]) 设置时有效。
21:11	TBS2: 发送缓存 2 大小 (Transmit Buffer 2 Size) 这些比特以字节表明第二个数据缓存。如果 TDES1[24] 被设置，此域无效。
10:0	TBS1: 发送缓存 1 大小 (Transmit Buffer 1 Size) 这些比特表明第一个数据缓存字节大小。如果该域是 0，那么 DMA 将忽略该缓存并根据 TCH (Bit 24) 的值使用 Buffer 2 或者下一个描述符。

发送描述符 2 (TDES2)

TDES2 包含描述符的第一个缓存的地址指针。

表 17-8. 发送描述符 2

比特	说明
31:0	缓存 1 地址指针 这些比特代表缓存 1 的物理地址。对缓存地址对齐没有限制。关于缓存地址对齐的详细信息，请参考第 17-19 页的“主机数据缓存对齐 (Host Data Buffer Alignment)”。

发送描述符 3 (TDES3)

TDES3 包含描述符的第二个缓存或者下一个描述符的地址指针。

表 17-9. 发送描述符 3

比特	说明
31:0	缓存 2 地址指针（下一个描述符地址） 当使用描述符环状结构时表明缓存 2 的物理地址。如果 Second Address Chained (TDES1[24]) 比特被设置，那么此地址包含 Next 描述符所位于的物理地址的指针。仅当 TDES1[24] 被设置时，缓存地址指针才必须对齐于总线宽度。（从内部忽略 LSB。）†

接收描述符

EMAC 中的 DMA 在接收一个帧时至少需要两个描述符。DMA 的接收状态机始终试图从输入帧中获得一个额外的描述符。（输入帧的大小是未知的）。在 RX DMA 关闭一个描述符之前，RX DMA 试图获得下一个描述符，即便没有接收到任何的帧。

在单一描述符（接收）系统中，如果接收缓存不能存储输入帧，并且下一个描述符不属于 DMA，那么子系统会生成一个描述符错误。因此，Host 被迫增加它的描述符池或者缓存大小。否则，子系统开始丢弃全部输入帧。

接收描述符 0 (RDES0)

RDES0 包含关于接收的帧状态，帧长度和描述符所有权的信息。

表 17-10. 接收描述符 0 (1/3)

比特	说明
31	OWN: Own Bit 该比特设置时表明 EMAC 的 DMA 拥有描述符。此比特被复位时，表明 Host 拥有描述符。当 DMA 完成帧接收，或者当与该描述符关联的缓存变满时，DMA 清零该比特。
30	AFM: 目的地址过滤器失败 (Destination Address Filter Fail) 设置时，该比特表明一个帧在 MAC 的 DA Filter 中失败。
29:16	FL: 帧长 (Frame Length) 这些比特表明传输到主机存储器（包括 CRC）中的接收帧的字节长。当 Last Descriptor (RDES0[8]) 被设置并且 Descriptor Error (RDES0[14]) 或者 Overflow Error 比特复位时，该域有效。当 IP 校验和计算 (Type 1) 被使能并且接收帧不是一个 MAC 控制帧时，帧长度也包括附加在 Ethernet 帧上的两个字节。 当 Last Descriptor (RDES0[8]) 被设置时，该域有效。当 Last Descriptor 和 Error Summary 比特没被设置时，该域表明对当前帧传输的字节累加数。

表 17 - 10. 接收描述符 0 (2/3)

比特	说明
15	<p>ES: 错误汇总 (Error Summary)</p> <p>表明下面比特的逻辑 OR:</p> <ul style="list-style-type: none"> ■ RDES0[0]: 净荷校验和错误 ■ RDES0[1]: CRC 错误 ■ RDES0[3]: 接收错误 ■ RDES0[4]: 看门狗超时 ■ RDES0[6]: 迟冲突 ■ RDES0[7]: IPC 校验和 (Type 2) ■ RDES0[11]: 上溢错误 ■ RDES0[14]: 描述符错误 <p>该域仅在 Last Descriptor (RDES0[8]) 设置时有效。</p>
14	<p>DE: 描述符错误 (Descriptor Error)</p> <p>设置时, 该比特表明一个不适合当前描述符缓存的帧导致的帧截短, 和 DMA 不具有下一个描述符。帧被截短。该域仅在 Last Descriptor (RDES0[8]) 设置时有效。</p>
13	<p>SAF: 源地址过滤器失败 (Source Address Filter Fail)</p> <p>设置时, 该比特表明帧的 SA 域在 MAC 中的 SA Filter 中失败。</p>
12	<p>LE: 长度错误 (Length Error)</p> <p>设置时, 该比特表明接收帧的实际长度和 Length/ Type 域不匹配。该比特仅在 Frame Type (RDES0[5]) 比特复位时有效。当出现 CRC 错误时, 长度错误状态无效。</p>
11	<p>OE: 上溢错误 (Overflow Error)</p> <p>设置时, 该比特表明由于 RX FIFO 缓存上溢而使接收的帧损坏。</p> <p>注意: 该比特仅在 DMA 传输部分帧到应用中时被设置。这只发生在 RX FIFO 缓存运行在 threshold 模式下。在 store-and-forward 模式中, RX FIFO 缓存中的全部部分帧被完全丢弃。</p>
10	<p>VLAN: VLAN 标记 (VLAN Tag)</p> <p>设置时, 该比特表明有该描述符指向的帧是一个被 MAC 标记了的 VLAN 帧。VLAN 标记取决于接收帧的 VLAN 域, 根据 Register 7 (VLAN Tag Register) 设置。</p>
9	<p>FS: 第一个描述符 (First Descriptor)</p> <p>设置时, 该比特表明此描述符包含帧的第一个缓存。如果第一个缓存的大小是 0, 那么第二个缓存包含帧的开始部分。如果第二个缓存的大小也是 0, 那么下一个描述符包含帧的开始部分。</p>
8	<p>LS: 最后描述符 (Last Descriptor)</p> <p>设置时, 该比特表明由此描述符指向的缓存是帧的最后缓存。</p>
7	<p>IPC 校验和错误</p> <p>当 IP Checksum Engine (Type 1) 使能时, 该比特设置时表明 MAC 计算出的 16-bit IPv4 Header 校验和与接收到的校验和字节不匹配。当该比特在此模式下设置时, Bit 15 (ES) 不被设置。如果当 Full Checksum Offload Engine (Type 2) 使能时该比特被设置, 那么它表明 IPv4 或 IPv6 中的错误。该错误可以由不一致的 Ethernet Type 域和 IP header Version 域值, IPv4 中的头校验和失配, 或者 Ethernet 帧缺少所需的 IP 头字节数导致。关于详细信息, 请参考第 17 - 35 页的表 17 - 11。</p>
6	<p>LC: 迟延迟 (Late Collision)</p> <p>设置时, 该比特表明在半双工模式下接收帧时出现了延迟冲突。</p>
5	<p>FT: 帧类型 (Frame Type)</p> <p>设置时, 该比特表明接收帧是一个 Ethernet 类型的帧 (LT 域大于或等于 0x0600)。当此比特复位时, 它表明接收帧是一个 IEEE802.3 帧。此比特对少于 14 字节的 Runt 帧无效。关于详细信息, 请参考第 17 - 35 页的表 17 - 11。</p>
4	<p>RWT: 接收看门狗超时 (Receive Watchdog Timeout)</p> <p>设置时, 该比特表明接收当前帧时接收 Watchdog Timer 已经过期, 当前帧在 Watchdog Timeout 后被截短。</p>
3	<p>RE: 接收错误 (Receive Error)</p> <p>设置时, 该比特表明帧接收期间 gmii_rxdv_i 被置位时, gmii_rxer_i 信号被置位。错误可以是更少扩展或者没有扩展, 或者扩展期间的错误 (rxd != 0xf)。</p>

表 17 - 10. 接收描述符 0 (3/3)

比特	说明
2	DE: Dribble 比特错误 (Dribble Bit Error) 设置时, 该比特表明接收帧有一个非整型倍数的字节 (奇数的字节)。该比特仅在 MII 模式中有效。
1	CE: CRC 错误 (CRC Error) 设置时, 该比特表明接收帧上出现 CRC 错误。该域仅在 Last Descriptor (RDES0[8]) 设置时有效。
0	RX MAC 地址或者净荷校验和错误 设置时, 该比特表明 RX MAC Address 寄存器值 (1 到 15) 匹配帧的 DA 域。复位时, 该比特表明 RX MAC Address Register 0 值匹配 DA 域。如果 Full Checksum Offload Engine 使能, 那么该比特设置时表明 EMAC 计算出的 TCP, UDP 或者 ICMP 校验和与接收的封装 TCP, UDP 或者 ICMP 区段的 Checksum 域不匹配。当接收的净荷字节数量不匹配接收 Ethernet 帧中封装 IPv4 或 IPv6 数据包的 Length 域中指定的值时, 该比特也被设置。关于详细信息, 请参考第 17 - 35 页的表 17 - 11。

当 Full Checksum Offload Engine (Type 2) 使能时, Bits 5, 7 和 0 的置换反映了表 17 - 11 中描述的情况。

表 17 - 11. COE (Type 2) 使能时的接收描述符 0

Bit 5: 帧类型	Bit 7: IPC 校验 和错误	Bit 0: 净荷校验 和错误	帧状态
0	0	0	IEEE 802.3 类型帧 (Length 域值少于 0x0600)
1	0	0	IPv4/IPv6 类型帧, 没有检测到校验和错误
1	0	1	检测到净荷校验和错误的 IPv4/IPv6 类型帧 (正如对 PCE 的描述)
1	1	0	检测到 IP 头校验和错误的 IPv4/IPv6 类型帧 (正如对 IPC CE 的描述)
1	1	1	检测到 IP 头和净荷校验和错误的 IPv4/IPv6 类型帧
0	0	1	由于不支持的净荷导致的无 IP 头校验和错误, 和净荷检查旁路的 IPv4/IPv6 类型帧
0	1	1	一个既不是 IPv4 也不是 IPv6 的帧类型 (Checksum Offload 引擎完全旁路校验和)
0	1	0	保留

接收描述符 1 (RDES1)

RDES1 包含缓存大小和用于控制描述符链或环的其它比特。



关于计算缓存大小的详细信息, 请参考第 17 - 19 页的 “缓存大小计算”。

表 17 - 12. 接收描述符 1 (1/2)

比特	说明
31	禁止完成时中断 (Disable Interrupt on Completion) 设置时, 对于结束于该描述符指向的缓存的接收帧, 该比特防止 Register 5 (Status Register) 的 bit 6 (RI) 的设置, 从而禁止了由于该帧 RI 而导致的对 Host 中断的置位。
30:26	保留
25	RER: 接收环结束 (Receive End of Ring) 设置时, 该比特表明描述符列表达到其最后的描述符。DMA 返回列表的基地址, 创建一个描述符环。
24	RCH: 第二个地址链接 (Second Address Chained) 设置时, 该比特表明描述符中的第二个地址是下一个描述符地址, 而不是第二个缓存地址。当 RDES1[24] 被设置时, RBS2 (RDES1[21-11]) 的值为 “don't care”。RDES1[25] 优先于 RDES1[24]。

表 17 - 12. 接收描述符 1 (2/2)

比特	说明
23:2 2	保留
21:1 1	RBS2: 接收缓存 2 大小 (Receive Buffer 2 Size) 这些比特表明第二个数据缓存大小 (以字节表示)。缓存大小必须是 4 的倍数, 即使 RDES3 (缓存 2 地址指针) 的值与总线宽不对齐。在缓存大小不是 4 的倍数的情况下, 产生的行为是未定义的。如果 RDES1[24] 被设置, 那么该域无效。
10:0	RBS1: 接收缓存 1 大小 (Receive Buffer 1 Size) 表明第一个数据缓存大小 (以字节表示)。缓存大小必须是 4 的倍数, 即使 RDES2 (缓存 1 地址指针) 的值是不对齐的。在缓存大小不是 4 的倍数的情况下, 产生的行为是未定义的。如果该域是 0, 那么 DMA 将忽略此缓存并根据 RCH (Bit 24) 的值使用缓存 2 或者下一个描述符。

接收描述符 2 (RDES2)

RDES2 包含指向描述符中第一个数据缓存的地址指针。


 关于缓存地址对齐的详细信息, 请参考第 17 - 19 页的“主机数据缓存对齐 (Host Data Buffer Alignment)”。

表 17 - 13. 接收描述符 2 (默认操作)

比特	说明
31:0	缓存 1 地址指针 (Buffer 1 Address Pointer) 这些比特指示缓存 1 的物理地址。对缓存地址对齐没有限制, 但除了下面情况: 当 RDES2 值用于存储帧的起始部分时 DMA 对其地址生成使用 RDES2[1:0] 中编程的值。帧的起始部分传输期间 DMA 执行 RDES2[1:0] bits 为 0 的写操作, 但帧数据根据实际缓存地址指针移位。如果地址指针指向一个存储帧的中间或最后部分的缓存, 那么 DMA 将忽略 RDES2[1:0]。

接收描述符 3 (RDES3)

RDES3 包含地址指针, 指向描述符中的第二个数据缓存或者下一个描述符。

表 17 - 14. 接收描述符 3

比特	说明
31:0	缓存 2 地址指针 (Buffer 2 Address Pointer) (Next Descriptor Address) 这些比特表明当使用描述符环状结构时的 Buffer 2 的物理地址。如果 Second Address Chained (RDES1[24]) 比特被设置, 那么该地址包含一个指向存储下一个描述符的物理存储器的指针。 如果 RDES1[24] 设置, 那么缓存 (下一个描述符) 地址指针必须是总线宽对齐的 (RDES3[1:0] = 0。从内部忽略 LSBs。)然而, 当 RDES1[24] 被复位时, 对 RDES3 值没有限制, 除了下面的情况: 当 RDES3 值用于存储帧的起始部分时 DMA 对其地址生成使用 RDES3[1:0] 中编程的值。如果地址指针指向一个存储帧的中间或最后部分的缓存, 那么 DMA 将忽略 RDES3[1:0]。

IEEE 1588-2005 时间戳使能的描述符格式

默认的描述符格式 (如第 17 - 30 页的“发送描述符”和第 17 - 33 页的“接收描述符”中所描述) 和域描述保持不变, 当由软件创建时 (DES0 中的 Own 比特被设置)。

然而, 如果软件已使能了 IEEE 1588-2005 功能, 那么当 DMA 关闭描述符 (DES0 中的 Own 比特被清零) 时, DES2 和 DES3 描述符域 (请参考表 17 - 15) 代表不同的意思。

DMA 在清零 DES0 中 Own 比特前使用时间戳值来更新 DES2 和 DES3。当 EMAC 操作在 32-bit 模式下时，使用较低的 32 个时间戳比特更新 DES2（Sub-second 域，在接下来的部分中称作 TSL），并使用较高的 32 个时间戳比特更新 DES3（seconds 域，在接下来的部分中称作 TSH）。

表 17 - 15. 当 DMA 清零 Own 比特时的描述符域

	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
DES0																																
DES1																																
DES2	Timestamp Low [31:0]																															
DES3	Timestamp High [31:0]																															

接下来的部分详细介绍了此模式下的接收和发送描述符。

发送描述符

除了第 17 - 36 页的“IEEE 1588-2005 时间戳使能的描述符格式”中描述的变更，对于时间戳，发送描述符也有其它的控制和状态比特（分别是 TTSE 和 TTSS），如表 17 - 16 所示。软件设置 TTSE 比特（当 Own 比特被设置时），指示 EMAC 生成一个时间戳，用于正在发送的相应 Ethernet 帧。当描述符关闭时（Own 比特清零），如果 TDES2 和 TDES3 域中的时间戳被更新，那么 DMA 设置 TTSS 比特。

表 17 - 16. 发送描述符域 - 一般格式

	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
TDES0	O W N	RES														T T S S	状态 [16:0]															
TDES1	其它控制域										T T S E	缓存 2 字节数 [21:11]										缓存 1 字节数 [10:0]										
TDES2	TTSL																															
TDES3	TTSH																															

发送时间戳控制和状态域

对于一般发送描述符和增强格式发送描述符而言，这些域的位置是不同的。在这两种情况下，此域值在描述符关闭时被 DMA 保留。

第 17 - 36 页的表 17 - 14 和第 17 - 30 页的表 17 - 6 中对默认（一般）描述符格式的更新如下所述。

表 17 - 17. 发送时间戳状态 - 一般描述符格式情况 (TDES0)

比特	说明
17	TTSS: 发送时间戳状态 此域是一个状态比特，表明对于相应发送帧采集到了时间戳。当此比特被设置时，TDES2 和 TDES3 都有一个对发送帧采集到的时间戳值。 仅当 Last Segment 控制比特（描述符中的 TDES1[30]）被设置时，此域才有效。

表 17 - 18. 发送时间戳控制 - 一般描述符格式情况 (TDES1)

比特	说明
22	TTSE: 发送时间戳使能 设置时，此域对描述符所描述的发送帧使能 IEEE1588 硬件时间戳。仅当 First Segment 控制比特（描述符中的 TDES1[29]）被设置时，此域才有效。

发送时间戳域

当发送描述符格式和域描述由软件创建时（当 Own 比特被设置时），两者都保持不变。然而，当 DMA 关闭最后一个描述符（由 TDES1 或 TDES0 中的 LS 比特以备选描述符格式标记的）和 IEEE 1588 功能使能 (Own 比特清零) 时，使用该帧的时间戳（如果获得）对 TDES2 和 TDES3 描述符域进行更新。

表 17 - 19 和表 17 - 20 描述了当描述符关闭时具有不同含义的域。

表 17 - 19. 发送描述符域 (TDES2)

比特	说明
31:0	TTSL: 发送帧时间戳低 DMA 使用对相应发送帧采集的时间戳的最低有效 32 比特对该域进行更新。仅当描述符中的 Last Segment 控制比特 (LS) 被设置时，此域才有时间戳。

表 17 - 20. 发送描述符域 (TDES3)

比特	说明
31:0	TTSH: 发送帧时间戳高 DMA 使用对相应发送帧采集的时间戳的最高有效 32 比特对该域进行更新。仅当描述符中的 Last Segment 控制比特 (LS) 被设置时，此域才有时间戳。

接收描述符

接收时间戳

表 17 - 21 显示了时间戳使能时的接收描述符的格式。

表 17 - 21. 接收时间戳域

	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
RDES0																																
RDES1																																
RDES2	接收帧时间戳低 [31:0]																															
RDES3	接收帧时间戳高 [31:0]																															

第 17 - 39 页的表 17 - 22 和第 17 - 39 页的表 17 - 23 描述了当接收描述符关闭和时间戳使能时对于 RDES2 和 RDES3 具有不同含义的域。

 当软件禁止 Timestamp 功能时 (Register 448 中的 TSENA 为低)，在关闭 RDES0 前 DMA 不更新描述符的 RDES2 或 RDES3 域。

表 17 - 22. 接收描述符域 (RDES2)

比特	说明
31:0	RTSL: 接收帧时间戳低 DMA 使用对相应发送帧采集的时间戳的最低有效 32 比特对该域进行更新。DMA 仅对由 Last Descriptor 状态比特 (RDES0[8]) 指示的接收帧更新该域。当该域和 RDES3 的 RTSH 域显示一个全 1 的值时，此时间戳一定被当作损坏的。

表 17 - 23. 接收描述符域 (RDES3)

比特	说明
31:0	RTSH: 接收帧时间戳高 DMA 使用对相应发送帧采集的时间戳的最高有效 32 比特对该域进行更新。DMA 仅对由 Last Descriptor 状态比特 (RDES0[8]) 指示的接收帧更新该域。 当该域和 RDES2 的 RTSL 域显示一个全 1 的值时，此时间戳一定被当作损坏的。

替换或增强的描述符

替换（或增强的）描述符结构能够具有 8 DWORDS（32-bytes），而不是普通描述符格式中的 4 DWORDS。该替换描述符结构的特性如下：

- 普通描述符结构支持高达 2,048 字节的数据缓存。替换描述符结构被实现以支持高达 8 KB 的缓存（可用于 Jumbo 帧）。
- 在 TDES0, TDES1, RDES0（高级时间戳或者 IPC 全卸载配置）和 RDES1 中有控制和状态比特的重约束 (re-assignment)。

- 当您选择高级时间戳时，发送描述符将时间戳存储在 TDES6 和 TDES7 中。
- 当选择高级时间戳，IPC Full Checksum Offload Engine 或者 Layer 3 和 Layer 4 过滤功能时，该接收描述符结构也用于存储扩展状态 (RDES4) 和时间戳 (RDES6 和 RDES7)。
- 您可以选择下面其中一个描述符结构的选项：
 - 如果 Register 448(时间戳控制寄存器)中的时间戳使能或者 Register 0 (MAC 配置寄存器) 中的 Checksum Offload 使能，那么软件需要对每个描述符分配 32 字节 (8 DWORDS) 的存储器。这样，软件应该设置 Register 0 (总线模式寄存器) 的 Bit 7 (替换描述符大小)。
 - 如果描述符或者 Checksum Offload 没有使能，那么就不需要扩展描述符 (DES4 到 DES7)。因此，软件能够使用默认大小为 16 字节 (4 DWORDS) 的替换描述符。这样，软件应该将 Register 0 (总线模式寄存器) 的 Bit 7 (替换描述符大小) 复位成 0。
- 当选择的替换描述符没有 Timestamp 或者 Receive IPC Full Checksum Offload Engine (Type 2) 功能时，描述符的大小始终为 4 DWORDs (DES0-DES3)。因此，软件能够使用默认大小为 16 字节的替换描述符。

发送描述符

发送描述符结构如第 17 - 41 页的表 17 - 24 所示。应用软件在描述符初始化期间必须编程控制比特 TDES0[31:18]。DMA 更新描述符时写回所有的控制比特，除了 OWN 比特 (被 DMA 清零)，并更新状态比特 [7:0]。第 17 - 42 页的表 17 - 26，第 17 - 44 页的表 17 - 27，第 17 - 44 页的表 17 - 28 和第 17 - 44 页的表 17 - 29 分别给出了发送器描述符字 0 (TDES0) 到字 3 (TDES3) 的内容。

在高级时间戳支持下，通过设置 TDES0 的 Bit 25 (TTSE) 能够对指定帧使能要拍摄的时间戳快照。当描述符关闭时 (也就是当 OWN 比特清零时)，时间戳被写入到 TDES6 和 TDES7。这由第 17 - 41 页的表 17 - 24 中所示的 TDES0 的状态 Bit 17 (TTSS) 表明。TDES6 和 TDES7 的内容在第 17 - 44 页的表 17 - 30 和第 17 - 45 页的表 17 - 31 中所介绍。


 当高级时间戳功能使能时，软件应该设置 Register 0（总线模式寄存器）的 Bit 7，以便 DMA 在扩展描述符大小下运行。当该控制比特复位时，TDES4-TDES7 描述符空间无效。

表 17 - 24. 发送描述符域- 替换（增强）格式

	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
TDES0	O W N	Ctrl [30:26]					T T S E	Ctrl [24:18]					T T S S	Status [16:7]										Ctrl/Stat us [6:3]				Status [2:0]				
TDES1	Ctrl [31:29]		缓存 2 字节数 [28:16]												RES			缓存 1 字节数 [12:0]														
TDES2	缓存 1 地址 [31:0]																															
TDES3	缓存 2 地址 [31:0] 或者下一个描述符地址 [31:0]																															
TDES4	保留																															
TDES5	保留																															
TDES6	发送时间戳低 [31:0]																															
TDES7	发送时间戳高 [31:0]																															

DMA 从系统存储器始终读取或者获取描述符的 4 个 DWORDS 来获得表 17 - 25 中的缓存和控制信息。

表 17 - 25. 替换（增强）格式的发送描述符获取（读）

	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
TDES0	OWN	Ctrl [30:26]					TTSE	Ctrl [24:18]					为状态保留 [17:7]							SL0T Number [6:3]			为状态保留 [2:0]									
TDES1	Ctrl [31:29]		缓存 2 字节数 [28:16]										RES			缓存 1 字节数 [12:0]																
TDES2	缓存 1 地址 [31:0]																															
TDES3	缓存 2 地址 [31:0] 或者下一个描述符地址 [31:0]																															

表 17 - 26. 发送描述符字 0 (T(1/3)DES0) (1/3)

比特	说明
31	OWN: Own 比特 设置时，该比特表明描述符被 DMA 具有。当该比特复位时，它表明描述符被 Host 具有。DMA 在完成帧传输时或者完成读取位于描述符中的缓存时对该比特清零。在属于同一帧的全部随后描述符被设置后，帧的第一个描述符的 ownership 比特必须被设置。这可以避免获在取描述符与驱动器设置 ownership 比特之间的潜在竞争情况。
30	IC: 完成时中断 (Interrupt on Completion) 设置时，预发送帧被发送后，此比特设置 Transmit Interrupt (Register 5[0])。
29	LS: 最后部分 (Last Segment) 设置时，此比特表明缓存包含帧的最后部分。该比特设置时，TDES1 中的 TBS1 或者 TBS2 应该有一个非零值。
28	FS: 第一部分 (First Segment) 设置时，该比特表明缓存包含帧的第一部分。
27	DC: 禁止 CRC (Disable CRC) 该比特设置时，MAC 不在发送帧的结尾附加 CRC，仅当第一部分 (TDES0[28]) 设置时这才有效。
26	DP: 禁止 Pad (Disable Pad) 设置时，MAC 不自动添加 padding 到少于 64 字节的帧。当该比特复位时，DMA 自动添加 padding 和 CRC 到少于 64 字节的帧，并且 CRC 域被添加而不管 DC (TDES0[27]) 比特的状态。仅当第一部分 (TDES0[28]) 设置时这才有效。
25	TTSE: 发送时间戳使能 (Transmit Timestamp Enable) 设置时，该比特对描述符引用的发送帧使能 IEEE1588 硬件时间戳。仅当 First Segment 控制比特 (TDES0[28]) 设置时这才有效。
24	保留
23:22	CIC: 校验和插入控制 (Checksum Insertion Control) 该比特控制校验和计算和插入。下面介绍了比特编码： ■ 0: 校验和插入禁用。 ■ 1: 仅使能 IP 头校验和计算和插入。 ■ 2: 使能 IP 头校验和和净荷校验和计算和插入，但在硬件中不计算伪头。 ■ 3: 使能 IP 头校验和和净荷校验和计算和插入，在硬件中计算伪头校验和。 当 First Segment 控制比特 (TDES0[28]) 设置时，该域有效。

表 17 - 26. 发送描述符字 0 (T(2/3)DES0)(2/3)

比特	说明
21	TER: 发送环结束 (Transmit End of Ring) 设置时, 该比特表明描述符列达到其最后的描述符。DMA 返回到列的基地址, 创建一个描述符环。
20	TCH: 第二个地址链接 (Second Address Chained) 设置时, 该比特表明描述符中的第二个地址是下一个描述符地址, 而不是第二个缓存地址。当 TDES0[20] 设置时, TBS2 (TDES1[28:16]) 的值为 “don’t care”。 TDES0[21] 优先于 TDES0[20]。
19:18	保留
17	TTSS: 发送时间戳状态 (Transmit Timestamp Status) 该域用作一个状态比特, 表明采集到描述发送帧的时间戳。该比特设置时, TDES2 和 TDES3 有一个对发送帧采集的时间戳值。仅当描述符的 Last Segment 控制比特 (TDES0[29]) 被设置时, 该域才有效。
16	IHE: IP 头错误 (IP Header Error) 设置时, 该比特表明 MAC 发送器在 IP 数据报头中检测到错误。发送器根据从应用接收到的头字节数来检查 IPv4 数据包中的头长, 并在出现失配时指示一个错误状态。对于 IPv6 帧, 如果主头长 (main header length) 不是 40 字节, 则会报告头错误。此外, IPv4 或 IPv6 帧的 Ethernet Length/Type 域值必须匹配同数据包一起接收到的 IP 头版本。对于 IPv4 帧, 如果 Header Length 域的值小于 0x5, 那么也会指示一个错误状态。
15	ES: 错误汇总 (Error Summary) 指示下面比特的逻辑 OR: <ul style="list-style-type: none"> ■ TDES0[14]: Jabber 超时 (Jabber Timeout) ■ TDES0[13]: 帧刷新 (Frame Flush) ■ TDES0[11]: 载体丢失 (Loss of Carrier) ■ TDES0[10]: 无载体 (No Carrier) ■ TDES0[9]: 迟冲突 (Late Collision) ■ TDES0[8]: 过度冲突 (Excessive Collision) ■ TDES0[2]: 过度延迟 (Excessive Deferral) ■ TDES0[1]: 下溢错误 (Underflow Error) ■ TDES0[16]: IP 头错误 (IP Header Error) ■ TDES0[12]: IP 净荷错误 (IP Payload Error)
14	JT: Jabber 超时 设置时, 该比特表明 MAC 发送器经历了一个 jabber time-out。仅当 Register 0 (MAC Configuration Register) 的 Bit 22 (Jabber Disable) 没有设置时, 该比特才被设置。
13	FF: 帧刷新 设置时, 该比特表明由于 CPU 发出一个软件 Flush 命令, DMA 或 MTL 刷新了帧。
12	IPE: IP 净荷错误 设置时, 该比特表明 MAC 发送器在 TCP, UDP 或者 ICMP IP 数据报净荷中检测到错误。 发送器根据从应用接收到的 TCP, UDP 或者 ICMP 数据包字节的实际数来检查 IPv4 或 IPv6 头中的净荷长, 并在出现失配时指示一个错误状态。
11	LC: 载体丢失 设置时, 该比特表明帧发送期间出现载体丢失 (也就是帧发送期间在一个或多个发送时钟周期中 gmii_crs_i 信号是无效的。当 MAC 运行在半双工模式下, 这只对无冲突的发送帧有效。
10	NC: 无载体 设置时, 该比特表明来自 PHY 的 Carrier Sense 信号在发送期间没有置位。
9	保留
8	EC: 过度延迟 设置时, 该比特表明当试图发送当前帧时, 在 16 个连续冲突过后发送终止。如果 Register 0 (MAC Configuration Register) 中的 Bit 9 (Disable Retry) 比特被设置, 那么该比特在第一个冲突后设置, 并终止帧发送。
7	VF: VLAN 帧 设置时, 该比特表明发送的帧是一个 VLAN 类型的帧。

表 17 - 26. 发送描述符 0 (T(3/3)DES0) (3/3)

比特	说明
6:3	CC: 冲突数量 (Status 域) 这些状态比特表明帧被发送前出现的冲突数量。当 Excessive Collisions 比特 (TDES0[8]) 设置时, 此数无效。EMAC 仅在半双工模式下更新该状态域。
2	ED: 过度延迟 设置时, 该比特表明当 Register 0 (MAC Configuration Register) 中的 Bit 4 (Deferral Check) 比特设为高电平时, 由于超过 24,288 比特时间的过度延迟 (1,000-Mbps 模式下的 155,680 比特时间或者如果 Jumbo 帧使能) 而终止发送。
1	UF: 下溢错误 设置时, 该比特表明由于数据从 Host 存储器到达, MAC 终止了帧。Underflow Error 表明 DMA 在发送帧期间碰到一个空的发送缓存。发送过程进入 Suspended 状态, 并设置 Transmit Underflow (Register 5[5]) 以及 Transmit Interrupt (Register 5[0])。
0	DB: 延迟比特 设置时, 该比特表明由于出现载体 MAC 在发送前延迟。该比特仅在半双工模式下有效。

表 17 - 27. 发送描述符 1 (TDES1)

比特	说明
31:29	保留
28:16	TBS2: 发送缓存 2 大小 该域以字节表示第二个数据缓存大小。当 TDES0[20] 设置时, 该域无效。关于计算缓存大小的详细信息, 请参考第 17 - 19 页的“缓存大小计算”。
15:13	保留
12:0	TBS1: 发送缓存 1 大小 这些比特以字节表示第一个数据缓存大小。如果该域为 0, 那么 DMA 忽略该缓存并根据 TCH (TDES0[20]) 的值使用 Buffer 2 或者下一个描述符。

表 17 - 28. 发送描述符 2 (TDES2)

比特	说明
31:0	缓存 1 地址指针 这些比特表明 Buffer 1 的物理地址。对缓存地址对齐没有限制。关于缓存地址对齐的更多信息, 请参考第 17 - 19 页的“主机数据缓存对齐 (Host Data Buffer Alignment)”。

表 17 - 29. 发送描述符 3 (TDES3)

比特	说明
31:0	缓存 2 地址指针 (下一个描述符地址) 当使用描述符环状结构时, 这些比特表示 Buffer 2 的物理地址。如果 Second Address Chained (TDES1[24]) 比特被设置, 那么该地址包括下一个描述符出现的物理存储器的指针。仅当 TDES1[24] 设置时, 缓存地址指针才必须对于与总线宽。(内部忽略 LSBs) †

表 17 - 30. 发送描述符 6 (TDES6)

比特	说明
31:0	TTSL: 发送帧时间戳低 DMA 使用对相应发送帧采集的时间戳的最低有效 32 比特对该域进行更新。仅当描述符中的 Last Segment 比特 (LS) 被设置和 Timestamp status (TTSS) 比特被设置时, 此域才有时间戳。

表 17 - 31. 发送描述符 7 (TDES7)

比特	说明
31:0	TTS: 发送帧时间戳低 DMA 使用对相应发送帧采集的时间戳的最高有效 32 比特对该域进行更新。仅当描述符中的 Last Segment 比特 (LS) 被设置和 Timestamp status (TTSS) 比特被设置时，此域才有时间戳。

接收描述符

接收描述符的结构如表 17 - 32 所示。当选择了高级时间戳或者 IPC Full Offload 功能时，此接收描述符能够有 32 字节的描述符数据 (8 DWORDs)。当其中的一个功能使能时，软件应该设置 Register 0 (Bus Mode Register) 的 Bit 7，以便 DMA 使用扩展的描述符大小进行操作。当该控制比特复位时，RDES0[0] 始终被清零，并且 RDES4- RDES7 描述符空间是无效的。

表 17 - 32. 接收描述符域- 替换（增强）格式

	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
TDES0	O W N	状态 [30:0]																														
TDES1	C T R L	RES [30:29]	缓存 2 字节数 [28:16]														Ctrl [15:14]	R E S	缓存 1 字节数 [12:0]													
TDES2	缓存 1 地址 [31:0]																															
TDES3	缓存 2 地址 [31:0] 或者下一个描述符地址 [31:0]																															
TDES4	扩展状态 [31:0]																															
TDES5	保留																															
TDES6	发送时间戳低 [31:0]																															
TDES7	发送时间戳高 [31:0]																															

表 17 - 33 描述符了 RDES0 的内容。第 17 - 47 页的表 17 - 34，第 17 - 48 页的表 17 - 35 和第 17 - 48 页的表 17 - 36 分别描述了 RDES1 到 RDES3 的内容。

表 17 - 33. 接收描述符域 (RDES0) (1/2)

比特	说明
31	OWN: Own Bit 设置时，该比特表明描述符被 EMAC 的 DMA 具有。当该比特复位时，它表明描述符被 Host 具有。当 DMA 完成帧接收时，后者当与此描述符相关的缓存变满时，DMA 清零该比特。
30	AFM: 目标地址过滤器错误 (Destination Address Filter Fail) 设置时，该比特表明一个帧在 MAC 中的 DA Filter 中失败。
29:16	FL: 帧长 (Frame Length) 这些比特表明传递到主机存储器 (包括 CRC) 的接收帧的字节长。当 Last Descriptor (RDES0[8]) 设置时和 Descriptor Error (RDES0[14]) 或者 Overflow Error 比特复位时，此域有效。当 IP 校验和计算 (Type 1) 使能和接收帧不是 MAC 控制帧时，帧长也包括附加到 Ethernet 帧的两个字节。 当 Last Descriptor (RDES0[8]) 设置时，该域有效。当 Last Descriptor 和 Error Summary 比特没有设置时，该域表明对当前帧传递的累加字节数。
15	ES: 错误汇总 (Error Summary) 指示下面比特的逻辑 OR: <ul style="list-style-type: none"> ■ RDES0[1]: CRC 错误 (CRC Error) ■ RDES0[3]: 接收错误 (Receive Error) ■ RDES0[4]: 看门狗超时 (Watchdog Timeout) ■ RDES0[6]: 迟冲突 (Late Collision) ■ RDES0[7]: 巨帧 (Giant Frame) ■ RDES4[4:3]: IP 头或净荷错误 (IP Header or Payload Error) ■ RDES0[11]: 上溢错误 (Overflow Error) ■ RDES0[14]: 描述符错误 (Descriptor Error) 仅当 Last Descriptor (RDES0[8]) 设置时，该域才有效。
14	DE: 描述符错误 (Descriptor Error) 设置时，该比特表明由于帧不适合当前描述符缓存而导致的帧截短和 DMA 不具有 Next 描述符。此帧是截短的。此域仅在 Last Descriptor (RDES0[8]) 设置时有效。
13	SAF: 源地址过滤器失败 (Source Address Filter Fail) 设置时，该比特表明帧的 SA 域在 MAC 的 SA Filter 中失败。
12	LE: 长度错误 (Length Error) 设置时，该比特表明接收帧的实际长度和 Length/Type 域不匹配。此比特仅在 Frame Type (RDES0[5]) 比特复位时有效。
11	OE: 上溢错误 (Overflow Error) 设置时，此比特表明由于 MTL 中的缓存上溢而导致了接收帧损坏。 注意：该比特仅在 DMA 传递部分帧到应用时设置。这仅发生在 RX FIFO 缓存运行在阈值模式下。在存储转发 (store-and-forward) 模式下，RX FIFO 缓存中的所有部分帧被彻底丢弃。
10	VLAN: VLAN 标记 (VLAN Tag) 设置时，该比特表明此描述符指向的帧是一个被 MAC 标记的 VLAN 帧。VLAN 标记是根据 Register 7 (VLAN Tag Register) 设置来检查接收帧的 VLAN 域来决定的。
9	FS: 第一个描述符 (First Descriptor) 当设置时，该比特表明描述符包含帧的第一个缓存。如果第一个缓存的大小为 0，那么第二个缓存就包含帧的起始部分。如果第二个缓存的大小也为 0，那么下一个描述符就包含帧的起始部分。
8	LS: 最后一个描述符 (Last Descriptor) 设置时，该比特表明此描述符指向的缓存是帧的最后缓存。

表 17 - 33. 接收描述符域 (RDES0) (2/2)

比特	说明
7	Timestamp Available, IP Checksum Error (Type1), or Giant Frame 当高级时间戳功能使能时，该比特设置时指示时间戳的块照写入到描述符 6 (RDES6) 和 7 (RDES7) 中。这仅在 Last Descriptor 比特 (RDES0[8]) 设置时有效。 当选择 IP Checksum Engine (Type 1) 时，该比特设置时指示 EMAC 计算出的 16-bit IPv4 Header 校验和不匹配接收的校验和字节。 否则，该比特设置时表明 Giant 帧状态。Giant 帧大于 1,518-byte (或者 VLAN 的 1,522-byte, 或者 2,000-byte, 当 MAC Configuration 寄存器的 Bit 27 (2KPE) 设置时) 普通帧，并且大于 9,018-byte (VLAN 的 9,022-byte) 帧，当 Jumbo 帧处理使能时。
6	LC: 迟冲突 (Late Collision) 该比特设置时表明在半双工模式下接收帧时出现的迟冲突。
5	FT: 帧类型 (Frame Type) 该比特设置时表明接收帧是 Ethernet 类型帧 (LT 域大于或等于 0x0600)。该比特复位时表明接收的帧是一个 IEEE802.3 帧。该比特对于少于 14 字节的 Runt 帧是无效的。
4	RWT: 接收看门狗超时 (Receive Watchdog Timeout) 该比特设置时表明在接收当前帧期间接收 Watchdog Timer 已经过期和在 Watchdog Timeout 之后当前帧被截短。
3	RE: 接收错误 (Receive Error) 该比特设置时表明在帧接收期间 gmii_rxdv_i 被置位时 gmii_rxer_i 信号被置位。错误能够是更少或没有扩展，或者是扩展期间的错误 (rxd !=0xf)。
2	DE: Dribble Bit Error 该比特设置时表明接收的帧有一个非整型倍数的字节 (奇数半字节)。该比特仅在 MII 模式下有效。
1	CE: CRC 错误 (CRC Error) 该比特设置时表明接收帧上出现了 CRC 错误。该域仅在 Last Descriptor (RDES0[8]) 设置时有效。
0	Extended Status Available/RX MAC Address 当出现高级时间戳或者 IP Checksum Offload (Type 2) 时，该比特设置时表明扩展的状态在描述符 4 (RDES4) 中可用。这仅在 Last Descriptor 比特 (RDES0[8]) 设置时有效。 当选择 Advance Timestamp Feature 或者 IPC Full Offload 时，该比特表明 RX MAC Address 状态。该比特设置时表明 RX MAC Address 寄存器值 (1 到 15) 匹配帧的 DA 域。复位时，该比特表明 RX MAC Address Register 0 值匹配 DA 域。

表 17 - 34. 接收描述符域 1 (RDES1)

比特	说明
31	DIC: 完成时禁止中断 (Disable Interrupt on Completion) 该比特设置时防止设置 Status Register 的 RI 比特 (CSR5[6])，从而禁止了由于该帧 RI 而导致的对 Host 中断的置位。†
30:29	保留
28:16	RBS2: 接收缓存 2 大小 这些比特以字节表示第二个数据缓存的大小。该缓存大小必须是 4 的倍数，即便 RDES3 (缓存 2 地址指针) 的值没有对齐于总线宽。如果缓存大小不是 4 的倍数，那么产生的行为是未定义的。如果 RDES1[14] 设置，那么该域是无效的。关于计算缓存大小的更多信息，请参考第 17 - 19 页的“缓存大小计算”。
15	RER: Receive End of Ring 该比特设置时表明描述符列到达了它的最后描述符。DMA 返回列的基地址，创建一个描述符环。
14	RCH: Second Address Chained 该比特设置时表明描述符中的第二个地址是下一个描述符地址，而不是第二个缓存地址。当该比特设置时，RBS2 (RDES1[28:16]) 的值是 “don't care”。RDES1[15] 优先于 RDES1[14]。
13	保留
12:0	RBS1: Receive Buffer 1 Size 以字节表示第一个数据缓存大小。缓存大小必须是 4 的倍数，即便 RDES2 (缓存 1 地址指) 的值没有对齐。当缓存大小不是 4 的倍数时，产生的结果是未定义的。如果该域为 0，那么 DMA 会忽略此缓存而根据 RCH (Bit 14) 的值来使用 Buffer 2 或者下一个描述符。关于计算缓存大小的更多信息，请参考第 17 - 19 页的“缓存大小计算”。

表 17 - 35. 接收描述符域 2 (RDES2)

比特	说明
31:0	Buffer 1 Address Pointer 这些比特表明 Buffer 1 的物理地址。除了下面情况，对缓存地址对齐没有限制：当 RDES2 值用于存储帧的起始部分时，DMA 对其地址生成使用 RDES2[1:0] 中的编程值。帧起始部分传递期间，DMA 使用为 0 的 RDES2[1:0] 比特执行写操作，但是帧根据实际的缓存地址指针移位了。如果地址指针指向一个存储帧的最后部分中间的缓存，那么 DMA 将忽略 RDES2[1:0]。关于缓存地址对齐的更多信息，请参考第 17 - 19 页的“主机数据缓存对齐 (Host Data Buffer Alignment)”。

表 17 - 36. 接收描述符域 3 (RDES3)

比特	说明
31:0	Buffer 2 Address Pointer (Next Descriptor Address) 当使用描述符环状结构时，这些比特表示 Buffer 2 的物理地址。如果 Second Address Chained (TDES1[24]) 比特被设置，那么该地址包括下一个描述符出现的物理存储器的指针。仅当 TDES1[24] 设置时，缓存地址指针才必须对于与总线宽。(内部忽略 LSBs) 如果 RDES1[24] 设置，那么缓存（下一个描述符）地址指针必须是总线宽对齐的 (RDES3[1:0] = 0。从内部忽略 LSBs。)然而，当 RDES1[24] 被复位时，对 RDES3 值没有限制，除了下面的情况：当 RDES3 值用于存储帧的起始部分时 DMA 对其地址生成使用 RDES3[1:0] 中编程的值。如果地址指针指向一个存储帧的中间或最后部分的缓存，那么 DMA 将忽略 RDES3[1:0]。

表 17 - 37 显示了写入的扩展状态。仅当存在 IPC 相关状态或者时间戳时写入扩展状态。扩展状态的可用性由 RDES0 的 Bit 0 指示。仅当选择了 Advance Timestamp 或者 IPC Full Offload 功能时此状态才可用。

表 17 - 37. 接收描述符域 4 (RDES4) (1/2)

比特	说明
31:28	保留
27:26	Layer 3 和 Layer 4 过滤器数量匹配 这些比特表明与接收帧匹配的 Layer 3 和 Layer 4 Filter 的数量。 <ul style="list-style-type: none"> 00: Filter 0 01: Filter 1 10: Filter 2 11: Filter 3 该域仅在 Bit 24 或 Bit 25 设为高时有效。当多于一个过滤器匹配时，这些比特仅给出最低的过滤器数。
25	Layer 4 过滤器匹配 设置时，该比特表明接收帧匹配其中的一个使能的 Layer 4 Port Number 域。仅当下面其中一个条件为真时，才给出此状态。 <ul style="list-style-type: none"> Layer 3 域没有使能，所有使能的 Layer 4 域都匹配。 所有使能的 Layer 3 和 Layer 4 过滤器域匹配。 当多于一个过滤器匹配时，该比特给出由 Bits [27:26] 指示的过滤器的 layer 4 过滤器状态。
24	Layer 3 过滤器匹配 该比特设置时表明接收帧匹配其中的一个使能的 Layer 3 IP Address 域。 仅当下面其中一个条件为真时此状态才有效： <ul style="list-style-type: none"> 所有使能的 Layer 3 域都匹配，所有使能的 Layer 4 域都被旁路。 所有使能的过滤器域都匹配。 当多个过滤器匹配时，该比特给出由 Bits [27:26] 指示的过滤器的 layer 3 过滤器状态。
23:15	保留
14	时间戳丢弃 该比特设置时表明对该帧采集的时间戳，但由于上溢在 MTL RX FIFO 缓存中丢弃。

表 17 - 37. 接收描述符域 4 (RDES4) (2/2)

比特	说明
13	PTP 版本 该比特设置时表明接收的 PTP 消息是一个 IEEE 1588 版本 2 格式，复位时是一个版本 1 格式。
12	PTP 帧类型 该比特设置时表明 PTP 消息直接通过 Ethernet 发出。该比特没有设置并且消息类型是非零时，它表明 PTP 消息通过 UDP-IPv4 或者 UDP-IPv6 发送。关于 IPv4 或 IPv6 的信息能够从 Bits 6 和 7 获得。
11:8	消息类型 通过对这些比特进行编码，给出接收消息的类型。 <ul style="list-style-type: none"> ■ 0000: 没有接收到的 PTP 消息 ■ 0001: SYNC (所有时钟类型) ■ 0010: Follow_Up (所有时钟类型) ■ 0011: Delay_Req (所有时钟类型) ■ 0100: Delay_Resp (所有时钟类型) ■ 0101: Pdelay_Req (点到点透明时钟) ■ 0110: Pdelay_Resp (点到点透明时钟) ■ 0111: Pdelay_Resp_Follow_Up (点到点透明时钟) ■ 1000: 宣称 ■ 1001: 管理 ■ 1010: 信号 ■ 1011-1110: 保留 ■ 1111: 带保留消息类型的 PTP 数据包
7	接收到 IPv6 数据包 该比特设置时表明接收的数据包是一个 IPv6 数据包。此比特仅在 Register 0 (MAC Configuration Register) 的 Bit 10 (IPC) 设置时更新。
6	接收到 IPv4 数据包 该比特设置时表明接收的数据包是一个 IPv4 数据包。此比特仅在 Register 0 (MAC Configuration Register) 的 Bit 10 (IPC) 设置时更新。
5	旁路的 IP 校验和 该比特设置时表明校验和卸载引擎被旁路。
4	IP 净荷错误 该比特设置时表明 EMAC 计算出的 16-bit IP 净荷校验和 (也就是 TCP, UDP 或者 ICMP 校验和) 不匹配接收部分 (received segment) 中的相应的校验和域。当 TCP, UDP 或者 ICMP segment 长度不匹配 IP Header 域中的净荷长度值时，该比特也被设置。当 Bit 7 或者 Bit 6 设置时，该比特有效。
3	IP 头错误 设置时，该比特表明由 EMAC 计算出的 16-bit IPv4 头校验和不匹配接收的校验和字节，或者 IP 数据报版本与 Ethernet Type 值不一致。该比特 Bit 7 或者 Bit 6 设置时有效。
2:0	IP 净荷类型 这些比特表明封装在 IP 数据报中的净荷类型，该 IP 数据报由接收校验和下载引擎 (COE) 处理。如果由于 IP 头错误或者分段的 IP 它不处理 IP 数据报的净荷，那么 COE 也将这些比特设为 0。 <ul style="list-style-type: none"> ■ 0: 未知或没有处理 IP 净荷 ■ 1: UDP ■ 2: TCP ■ 3: ICMP ■ 4-7: 保留 该比特 Bit 7 或者 Bit 6 设置时有效。

RDES6 和 RDES7 包含时间戳的快照。RDES6 和 RDES7 中的时间戳快照的可用性由 RDES0 描述符中的 Bit 7 表明。RDES6 和 RDES7 的内容分别在第 17 - 50 页的表 17 - 38 和表 17 - 39 中识别。

表 17 - 38. 接收描述符域 6 (RDES6)

比特	说明
31:0	RTSL: 接收帧时间戳低 (Receive Frame Timestamp Low) DMA 使用对相应接收帧采集的时间戳的最低有效 32 比特对该域进行更新。DMA 只对接收帧的最后描述符 (由 Last Descriptor 状态比特 (RDES0[8]) 指示) 更新该域。


表 17 - 39. 接收描述符域 7 (RDES7)

比特	说明
31:0	RTSH: 接收帧时间戳高 (Receive Frame Timestamp High) DMA 使用对相应接收帧采集的时间戳的最高有效 32 比特对该域进行更新。DMA 只对接收帧的最后描述符 (由 Last Descriptor 状态比特 (RDES0[8]) 指示) 更新该域。

初始化 DMA

本章提供了使用正确顺序初始化 DMA/MAC 寄存器的说明。执行下面的步骤来初始化 DMA:

- 1. 提供一个软件复位。这会复位所有的 EMAC 内部寄存器和逻辑。(DMA Register 0 (Bus Mode Register) - bit 0)。
- 2. 等待复位进程结束 (轮询 DMA Register 0 (Bus Mode Register) 的比特 0, 该比特仅在复位操作完成时清零)。
- 3. 轮询 Register 11 (AHB 或 AXI Status) 的比特来确认所有之前启动的传输 (软件复位前) 或者正在进行的传输都已经完成。

 如果软复位后应用不能轮询寄存器 (由于性能原因), 那么建议您继续执行接下来的步骤并在触发 DMA 操作前再次检查该寄存器 (如步骤 12 中所提及的)。

- 4. 编程下面的域通过设置 DMA Register 0 (Bus Mode Register) 中的值来初始化 Bus Mode Register:
 - a. Mixed Burst and AAL (混合突发和 AAL)
 - b. Fixed burst or undefined burst (固定突发或者未定义突发)
 - c. Burst length values and burst mode values (突发长度值和突发模式值)
 - d. Descriptor Length (only valid if Ring Mode is used) (描述符长度 (仅在使用环形模式时有效))
 - e. TX and RX DMA Arbitration scheme (TX 和 RX DMA 仲裁方案)

5. 编程 Register 10 (AXI Bus Mode Register) 中的接口选项。如果固定突发长度使能，那么选择可能在总线 (bits[7:1]) 上的最大突发长度。
6. 对发送和接收创建一个正确的描述符链。此外，要确保 DMA 具有接收描述符（描述符的 bit 31 应该被设置）。当使用 OSF 模式时，至少需要两个描述符。关于描述符的更多信息，请参考第 17 - 30 页的“普通描述符”和第 17 - 39 页的“替换或增强的描述符”。
7. 重用描述符前，要确保您的软件在链中创建三个或更多不同的发送或接收描述符。
8. 使用发送和接收描述符的基地址初始化接收和发送描述符列表地址（分别是 Register 3 (接收描述符列表地址寄存器) Register 4 (发送描述符列表地址寄存器)）。
9. 编程下面的域来初始化（操作模式寄存器）中的操作模式
 - a. 接收和发送存储和转发
 - b. 接收和发送阈值控制 (RTC 和 TTC)
 - c. 硬件流程控制使能
 - d. MTL 接收和发送 FIFO 缓存 (RFA 和 RFD) 的流程控制激活和失效阈值
 - e. 错误帧和较小的好帧转发使能
 - f. OSF 模式
10. 通过写入状态寄存器的被设置比特（仅中断比特）来清除中断请求。例如，通过 bit 16 写入 1，普通中断汇总清零该比特 (DMA Register 5 (状态寄存器))。
11. 通过编程 Register 7 (中断使能寄存器) 来使能中断。



仅当没有执行步骤 3 时，才执行步骤 12。

12. 读取 Register 11 (AHB 或 AXI 状态) 来确认所有之前的传输都已完成。



当您读取 Register 11 (AHB 或 AXI 状态) 时，如果之前的传输仍在进行，那么强烈建议您检查由主接口寻址的从组件。

13. 通过设置控制寄存器 (DMA Register 6 (操作模式寄存器)) 的 SR (bit 1) 和 ST (bit 13) 开始接收和发送 DMA。

初始化 MAC

下面的 MAC 初始化操作能够在 DMA 初始化后执行。如果 MAC 初始化在 DMA 设置前完成，那么仅在 DMA 有效后使能 MAC 接收器（下面最后一步）。否则，接收帧填充 RX FIFO 缓存并且上溢。

1. 编程 EMAC Register 4 (GMII 地址寄存器)，控制外部 PHY 的管理周期。例如，Physical Layer Address PA (bits 15-11)。此外，设置 bit 0 (GMII Busy) 来读取并写入 PHY。
2. 通过指定 Register 4 (GMII Address Register) 的 bits 15-11 中的相应地址值，从 PHY 读取 Register 5 (GMII Data Register) 的 16-bit 数据，用于链路开启，操作速度和操作模式。
3. 提供 MAC 地址寄存器 (Register 16 (MAC Address0 High Register) 和 Register 17 (MAC Address0 Low Register))。由于支持 128 MAC 地址，因此您需要编程相应地编程 MAC 地址。
4. 编程 Register 2 (Hash Table High Register) 和 Register 3 (Hash Table Low Register)。
5. 编程下面的域，对 Register 1 (MAC Frame Filter) 中的输入帧设置相应的过滤器：
 - a. 接收所有
 - b. 混杂模式
 - c. Hash 或 Perfect 过滤器
 - d. 单播，多播，广播和控制帧过滤器设置
6. 编程 Register 6 (流程控制寄存器) 中的下面域来实现正确的流程控制：
 - a. 暂停时间和其它暂停帧控制比特
 - b. 接收和发送流程控制比特
 - c. 流程控制忙 / 反压激活
7. 根据需要对您的配置编程 Interrupt Mask 寄存器比特。
8. 编程 Register 0 (MAC 配置寄存器) 中的相应域。例如，发送和 jabber 禁用期间的帧间隔。您可以根据自动协商选择 Duplex 模式 (bit 11) 或者端口选 (bit 15)。
9. 设置 Register 0 (MAC 配置寄存器) 中的 Bit 3 (TE) 和 Bit 2 (RE)。



当 EMAC DMA 有效地发送或接收时，不要修改配置（例如：双工模式，速度，端口或者环回）。仅当 EMAC DMA 发送器和接收器无效时，软件才应该修改这些参数。

执行普通接收和发送操作

对于普通操作，执行下面的步骤：

1. 对于普通的发送和接收中断，读取中断状态。然后，轮询描述符，读取 Host 拥有的描述符状态（发送或者接收）。
2. 对描述符设置相应的值，确保 DMA 具有发送和接收描述符来继续数据的发送和接收。
3. 如果 DMA 不具有描述符（或者没有描述符可用），那么 DMA 进入 SUSPEND 状态。通过释放描述符和通过写入 0 到 TX/RX 轮询需求寄存器 (Register 1 (Transmit Poll Demand Register) 和 Register 2 (Receive Poll Demand Register)) 来发出一个轮询请求可以继续发送和接收。
4. 对于调试进程，可以读取当前主机发送器或接收器描述符地址指针的值 (Register 18 (Current Host Transmit Descriptor Register) 和 Register 19 (Current Host Receive Descriptor Register))。
5. 对于调试进程，可以读取当前主机发送缓存地址指针和接收缓存地址指针的值 (Register 20 (Current Host Transmit Buffer Address Register) 和 Register 21 (Current Host Receive Buffer Address Register))。

停止和开始发送

执行下面的步骤来暂停发送：


1. 通过清零 Register 6（操作模式寄存器）的 bit 13（开始或停止发送命令）来禁止 transmit DMA（如果适用）。
2. 等待之前的帧发送完成，通过读取 Register 9（调试寄存器）的相应比特来检查。
3. 通过清零 Register 0（MAC Configuration Register）中的 Bit 3（TE）和 Bit 2（RE）来禁止 MAC 发送器和 MAC 接收器。
4. 在确保 RX FIFO 缓存中的数据被传递到系统存储器中后（通过读取 Register 9（调试寄存器）），禁止 receive DMA（如果适用）。
5. 确保 TX FIFO 缓存和 RX FIFO 缓存都是空的。
6. 要重新开始操作，首先要开始 DMAs，然后使能 MAC 发送器和接收器。


节能以太网的编程指南

进入和退出 TX LPI 模式

EMAC 中具有 Energy Efficient Ethernet (EEE) 特性。要使用这一特性，需要在 EMAC 初始化期间执行下面步骤：

1. 通过 MDIO 接口读取 PHY 寄存器，检查远端是否有 EEE 功能，然后协商计时器值。
2. 通过 MDIO 接口编程 PHY 寄存器（包括 RX_CLK_stoppable 比特，指示 PHY 是否停止 LPI 模式下的 RX 时钟）†
3. 编程 Register 13 (LPI Timers Control Register) 中的 Bits[16:5]，LST 和 Bits[15:0]，TWT。
4. 通过使用 MDIO 接口来读取 PHY 芯片的链路状态，并相应地更新 Register 12 (LPI Control and Status Register) 的 Bit 17 (PLS)。只要 PHY 中的链路状态发生变化，就要进行更新。
5. 设置 Register 12 (LPI Control and Status Register) 的 Set Bit 16 (LPIEN)，使 MAC 进入 LPI 状态。在完成正在进行的传输并设置 Bit 0 (TLPIEN) 后，MAC 进入 LPI 模式。

 要使 MAC 仅在完成 TX FIFO 缓存中所有排列的帧传输后进入 LPI 状态，您应该设置 Register 12 (LPI Control and Status Register) 的 Bit 19 (LPITXA)。

 要在 LPI 状态期间关闭发送时钟，需要使用 sbd_tx_clk_gating_ctrl_o 信号来选通 (gating) 时钟输入。

 要在 LPI 状态对系统的其它部分关闭 CSR 时钟或者电源，您应该等待 Register 12 (LPI Control and Status Register) 的 TLPIEN 中断的生成。如要离开 LPI 状态，则需要在执行步骤 6 之前还原时钟。

6. 复位 Register 12 (LPI Control and Status Register) 的 Reset Bit 16 (LPIEN)，使 MAC 离开 LPI 状态。

在设置 TLPIEX 中断状态比特和继续传输之前，MAC 等待 Bits [15:0]，TWT 中配置的时间。

断开 (gating off) LPI 模式下的 CSR 时钟

当 MAC 处于低功耗空闲 (LPI) 模式中时，通过断开 CSR 时钟可以降低功耗。

断开 (gating off) RX LPI 模式下的 CSR 时钟

当 MAC 从 PHY 接收到 LPI 码型时执行下面的操作。

1. MAC RX 进入 LPI 模式，并设置 RX LPI 入口中断状态 [Register 12 (LPI_Control_Status) 的 RLPIEN 中断]。
2. 中断管脚 (sbd_intr_o) 被置位。当主机读取 Register 12 (LPI_Control_Status) 时，sbd_intr_o 中断被清零。

sbd_intr_o 中断被置位并且 MAC TX 也处于 LPI 模式后，您能够断开 CSR 时钟。当断开 CSR 时钟时，如果 MAC TX 没有处于 LPI 模式，那么在 CSR 中将不会报告并更新 MAC 发送器上的事件。

要还原 CSR 时钟，需要等待来自 PHY 的 LPI 退出指示，然后 MAC 置位 lpi_intr_o (同步于 clk_rx_i) 上的 LPI 退出中断。当读取 Register 12 时，lpi_intr_o 中断被清零。

断开 (gating off) TX LPI 模式下的 CSR 时钟

当 Register 12 (LPI Control and Status Register) 的 Bit 16 (LPIEN) 被设置时执行下面的操作：

1. Transmit LPI Entry 中断 (Register 12 的 TLPIEN 比特) 被设置。
2. 中断管脚 (sbd_intr_o) 被置位。当主机读取 Register 12 时，sbd_intr_o 中断被清零。

sbd_intr_o 中断被置位并且 MAC RX 也处于 LPI 模式后，您能够断开 CSR 时钟。当断开 CSR 时钟时，如果 MAC RX 没有处于 LPI 模式，那么在 CSR 中将不会报告并更新 MAC 接收器上的事件。

对于还原 CSR 时钟，当 MAC 必须离开 TX LPI 模式时要开启 CSR 时钟。

CSR 时钟继续后，复位 Register 12 (LPI Control and Status Register) 的 Bit 16 (LPIEN)，使 MAC 离开 LPI 模式。

灵活的脉冲每秒 (Pulse-Per-Second (PPS)) 输出编程指南

在 PPS 上生成单脉冲

要在 PPS 上生成单脉冲：

1. 编程 Register 459 (PPS Control Register) 的 Bits [6:5]，TRGTMODESEL 中的 11 或 10 (中断)。这会指导 MAC 对 PPS 信号输出的起始时间使用 Target Time 寄存器 (寄存器 455 和 456)。
2. 编程 Target Time 寄存器 (寄存器 455 和 456) 中的起始时间值。
3. 编程 Register 473 (PPS0 Width Register) 中的 PPS 信号输出的宽度。
4. 将 Register 459 (PPS Control Register) 的 Bits [3:0]，PPSCMD 编程为 0001。这会指导 MAC 在 Target Time 寄存器 (寄存器 455 和 456) 中编程的时间上生成 PPS 信号输出上的单脉冲。

一旦执行 PPSCMD (PPSCMD bits = 0)，您就能够在编程的起始时间逝去前通过 Cancel Start Command (PPSCMD=0011) 取消脉冲生成。您也可以提前编程下一个脉冲的行为。要编程下一个脉冲：

1. 编程 Target Time 寄存器 (寄存器 455 和 456) 中下一个脉冲的起始时间。该时间应该晚于前一个脉冲下降沿出现的时间。
2. 编程 Register 473 (PPS0 Width Register) 中的下一个 PPS 信号输出的宽度。
3. 编程 Register 459 (PPS Control Register) 的 Bits [3:0]，PPSCMD，在前一个脉冲置低时间后生成一个单脉冲。这会指导 MAC 在 Target Time 寄存器中编程的时间上生成 PPS 信号输出上的单脉冲。如果在前一个脉冲变低前发出此命令，那么新命令会覆盖前一个命令，并且 EMAC 可能仅生成 1 个扩展脉冲。

在 PPS 上生成一个脉冲序列 (Pulse Train)

要在 PPS 上生成一个脉冲序列：

1. 编程 Register 459 (PPS Control Register) 的 Bits [6:5], TRGTMODESEL 中的 11 或 10 (中断)。这会指导 MAC 对 PPS 信号输出的起始时间使用 Target Time 寄存器 (寄存器 455 和 456)。
2. 编程 Target Time 寄存器 (寄存器 455 和 456) 中的起始时间值。
3. 编程 Register 473 (PPS0 Width Register) 中 PPS 信号输出上的脉冲序列之间的间隔值。
4. 编程 Register 473 (PPS0 Width Register) 中 PPS 信号输出的宽度。
5. 将 Register 459 (PPS Control Register) 的 Bits [3:0], PPSCMD 编程为 0010。这会指导 MAC 在 Target Time 寄存器 (寄存器 455 和 456) 中编程的起始时间上生成 PPS 信号输出上的脉冲序列。默认情况下, PPS 脉冲序列是自由运行的, 除非被 'STOP Pulse train at time' 或者 'STOP Pulse Train immediately' 命令停止。
6. 编程 Target Time 寄存器 (寄存器 455 和 456) 中的停止值。确保在再次编程 Target Time 寄存器 (寄存器 455 和 456) 之前复位 Register 456 (Target Time Nanoseconds Register) 的 Bit 31 (TSTRBUSY)。
7. 将 Register 459 (PPS Control Register) PPSCMD 的域 (bit 3:0) 编程为 0100。这会在步骤 6 中编程的停止时间过后停止 PPS 信号输出上的脉冲序列。

通过在 PPSCMD 域中编程 0101, 您可以随时停止脉冲序列。类似地, 您可以通过编程 PPSCMD 域中的 0110, 在 (步骤 6 中编程) 的时间消逝前取消 Stop Pulse train 命令 (步骤 7)。您可以通过在 PPSCMD 域中编程 0011, 在 (步骤 2 中) 编程的起始时间 (步骤 2) 消逝之前取消脉冲序列生成。

在不影响 PPS 的情况下生成中断

Register 459 (PPS Control Register) 的 Bits [6:5], TRGTMODESEL 使您能够通过编程 Target Time 寄存器 (寄存器 455 和 456) 来完成下面任何一个操作：

- 仅生成中断。
- 生成中断和 PPS 起始和停止时间。
- 仅生成 PPS 起始和停止时间。

通过编程 Target Time 寄存器 (寄存器 455 和 456) 仅生成中断事件：

1. 编程 Register 459 (PPS Control Register) 的 Bits [6:5], TRGTMODESEL 中的 00 (中断)。这会指导 MAC 对目标时间中断使用 Target Time 寄存器 (寄存器 455 和 456)。
2. 在 Target Time 寄存器 (寄存器 455 和 456) 编程一个目标时间值。这会指导 MAC 在目标时间消逝后生成一个中断。如果 Bits [6:5], TRGTMODESEL 改变 (例如：控制 PPS), 那么中断生成被新模式和新编程的 Target Time 寄存器值覆盖。

Ethernet MAC 地址映射和寄存器定义




地址映射和寄存器定义位于本卷中的 [hps.html](#) 文件中。点击下面链接打开文件。

要查看模块定义和基地址, 点击下面模块实例的链接：

- `emac0`
- `emac1`

然后，点击寄存器名来查看寄存器和域描述。寄存器地址是相对于每个模块实例地址的偏移。

 所有模块的基地址也列于 *Cyclone V 器件手册* 卷 3 中的 *Introduction to the Hard Processor System* 章节中。

文档修订历史

表 17 - 40 显示了本文档的修订历史。

表 17 - 40. 文档修订历史

日期	版本	修订内容
2012 年 11 月	1.2	次要更新。
2012 年 5 月	1.1	添加了编程模型部分。
2012 年 1 月	1.0	首次发布。

