应用笔记

ACM32H5 系列芯片

ETH 应用笔记

版本：V1.0

日期：2025-02-15

上海航芯电子科技股份有限公司

1. 概述

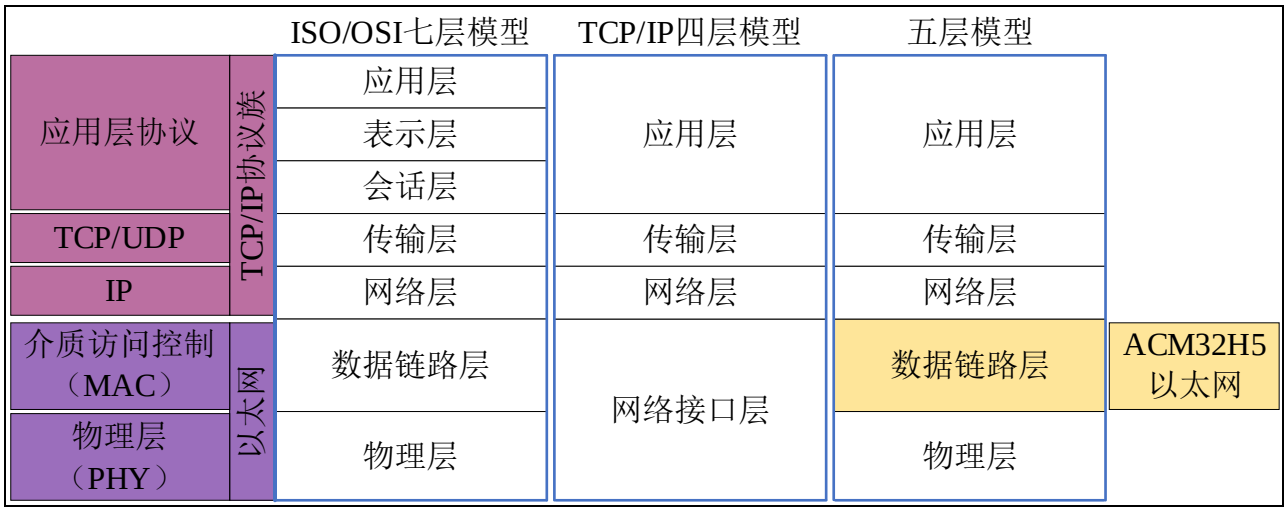
本应用手册适用于 ACM32H5 系列芯片 ETH 模块。它描述了与 ETH 模块相关的设置和功能使用方法，以便在应用程序中进行优化设计。

本应用手册需要结合用户手册中对应的 ETH 模块部分、SDK 中相关的 demo 及 HAL 库驱动一起阅读。

2. 网络分层模型

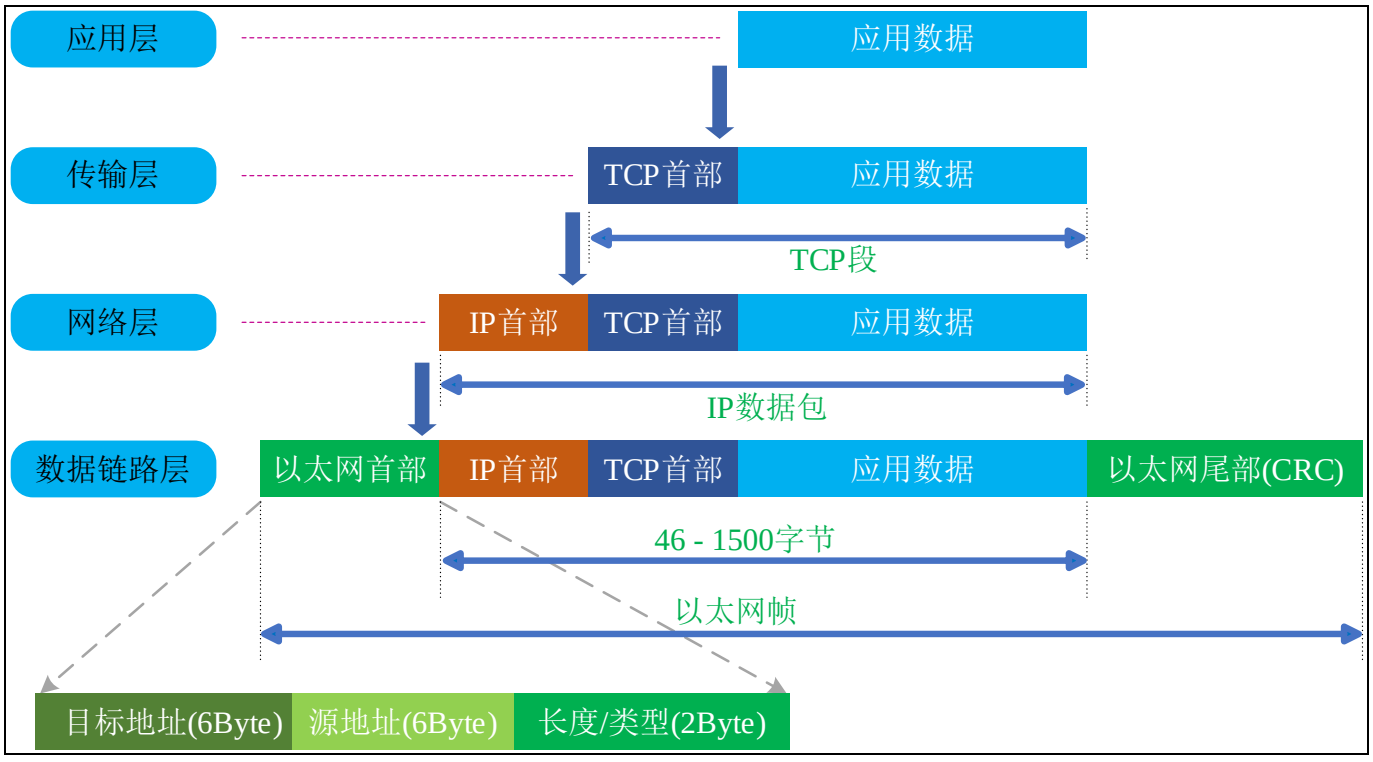
为了使不同体系结构的计算机网络都能互联，国际标准化组织(International Organization for Standardization, ISO)提出了著名的开放系统互连基本参考模型 OSI/RM（Opensystems Interconnection Reference Model），简称 OSI，即广为所知的七层协议体系结构。但 TCP/IP 协议作为互联网的基础协议，其所采用的四层 TCP/IP 体系结构却是实际使用最为广泛的。而本文档为了更加简洁和清楚的阐明以太网概念，将采用折中的五层协议体系结构模型。

图 2-1 网络分层模型



网络分层的体系下，数据发送是一个数据封装的过程，每一层协议在进行数据封装时，都会在上层传来的数据的基础上添加本层的元数据，而数据接收则是封装的逆过程，如图 2-2 为例，说明一个以太网数据包的格式。

图 2-2 数据封包示意图



3. 以太网外设使用说明

ACM32H5xx 可以通过以太网外设，按照 IEEE802.3-2002 标准接收和发送数据。

以太网外设可灵活进行配置，以满足各种应用和客户的需求。它支持两种与外部物理层（PHY）相连的工业标准接口：介质独立接口（MII）和简化介质独立接口（RMII），其中介质独立接口（MII）在 IEEE 802.3 规范中定义，并作为默认使用的接口。以太网外设可应用在诸如交换机和网络接口卡等多种场景。

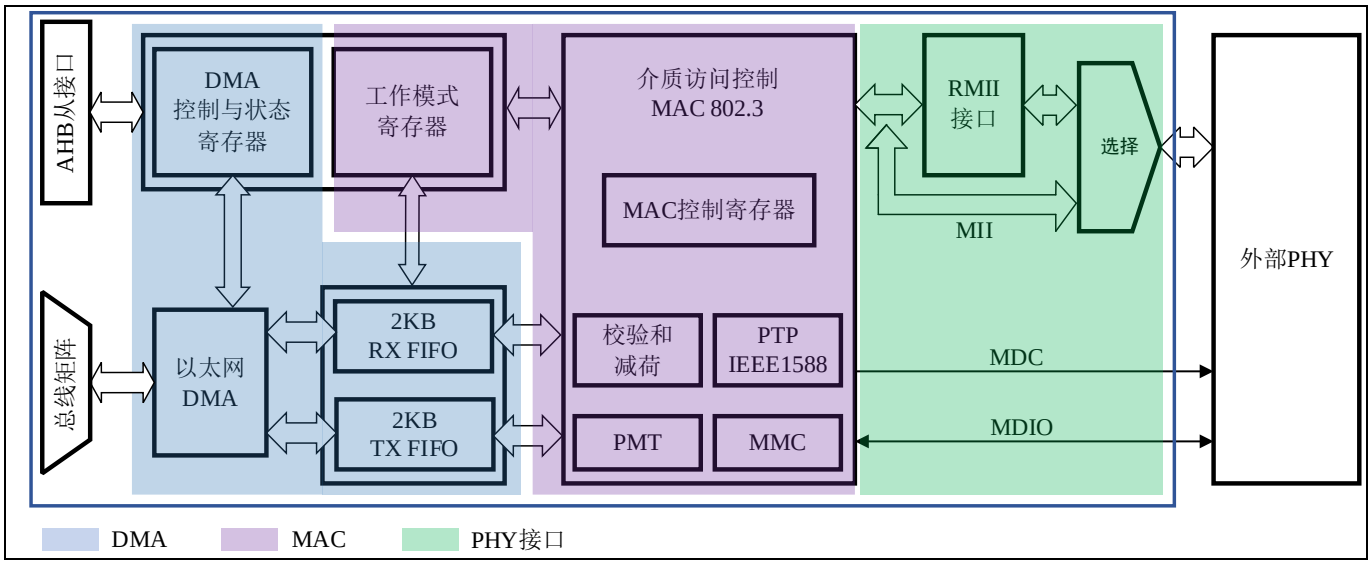
它符合以下标准：

- IEEE 802.3-2008，用于以太网 MAC 和介质独立接口（MII）
- IEEE 1588-2008，用于精密网络时钟同步协议（PTP）
- AMBA 2.0，用于 AHB 主端口和 AHB 从端口
- RMII 联盟的 RMII 规范第 1.2 版

3.1. 以太网外设概览

以太网外设由一个带专用 DMA 控制器的 MAC 802.3（介质访问控制）构成，支持独立介质接口（MII）和简化独立介质接口（RMII），以及一个站点管理接口（SMI）。

图 2-2 以太网模块框图



如图 3-1 所示以太网外设主要包括三个部分：专用的 DMA 控制器，MAC 802.3（介质访问控制器），外部 PHY 接口。

3.2. 以太网外设信号引脚

发送数据时，以太网外设通过相应引脚发送逻辑信号给外部 PHY，外部 PHY 将逻辑信号转换成差分信号发送出去；接收过程相反。表 x 显示了以太网外设和外部 PHY 相连信号所对应的引脚，以及一个 PTP 时基脉冲信号。

表 3-1 以太网外设信号引脚映射

MII 信号	RMII 信号	说明	引脚号
MII_TX_CLK		MII 数据发送时钟，25MHz@100Mbps	PC3
MII_TXD0	RMII_TXD0	MII/RMII 数据发送 0	PB12, PG13

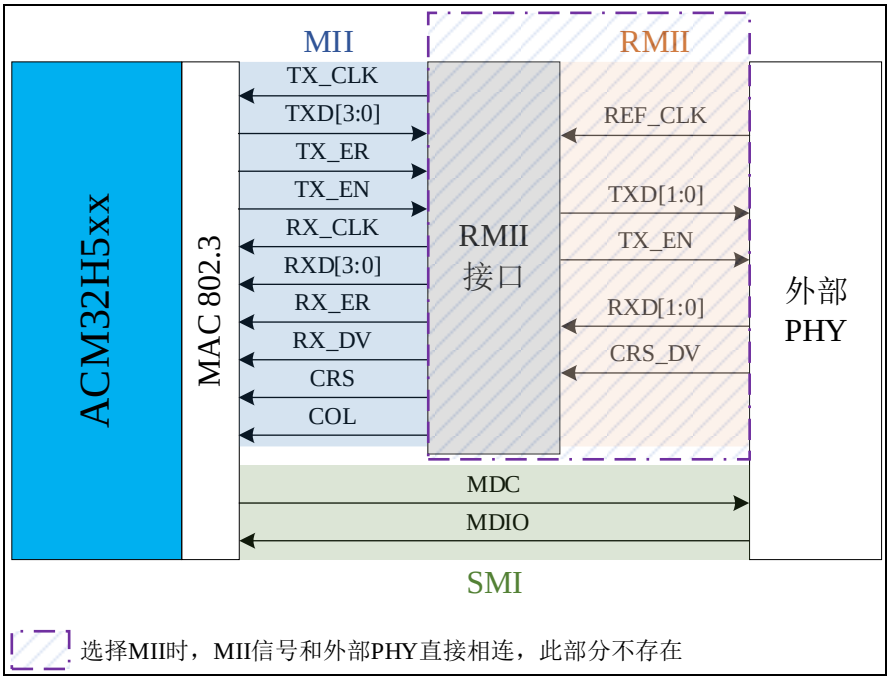
MII_TXD1	RMII_TXD1	MII/RMII 数据发送 1	PB13, PG12, PG14
MII_TXD2		MII 数据发送 2	PC2, PJ1
MII_TXD3		MII 数据发送 3	PB8, PE2, PJ2
MII_TX_ER		MII 数据发送错误	PA9, PA13, PB2
MII_TX_EN	RMII_TX_EN	MII/RMII 数据发送使能	PB11, PG11
MII_RX_CLK	RMII_REF_CLK	MII 数据接收时钟，25MHz@100Mbps RMII 参考时钟，50MHz@100Mbps	PA1
MII_RXD0	RMII_RXD0	MII/RMII 数据接收 0	PC4
MII_RXD1	RMII_RXD1	MII/RMII 数据接收 1	PC5
MII_RXD2		MII 数据接收 2	PB0, PH6, PI15
MII_RXD3		MII 数据接收 3	PB1, PH7, PJ0
MII_RX_ER		MII 数据接收错误	PB10, PI10
MII_RX_DV	RMII_CRS_DV	MII 接收数据有效 RMII 载波侦听/接收数据有效	PA7
MII_CRS		MII 载波侦听	PA0, PH2, PJ3
MII_COL		MII 冲突检测	PA3, PH3, PJ4
MDC		SMI 时钟	PC1
MDIO		SMI 数据（双向）	PA2
PPS_OUT		IEEE1588 PTP 时基脉冲输出信号	PB5, PG8

3.3. 外部 PHY 接口

以太网外设通过独立介质接口（MII）或简化独立介质接口（RMII）与外部 PHY 进行通讯，默认使用独立介质接口（MII），可以配置 SYSCFG_SYSCR 寄存器的 EPIS 位来选择接口。

以太网外设还包含一个站点管理接口（SMI），用以配置外部 PHY。

图 3-2 外部 PHY 接口示意图



3.3.1. 介质独立接口（MII）

介质独立接口（MII）定义了 10 Mbit/s 和 100 Mbit/s 的数据传输速率下，MAC 子层与 PHY 之间的互连。其接口信号相见表 3-2 MII 信号说明。

表 3-2 MII 信号说明

MII 信号	信号描述
TX_CLK	发送数据参考时钟。 2.5MHz@10Mbps，25MHz@100Mbps。
TXD[3:0]	发送数据。 由 MAC 子层和有效 TX_EN 信号同步驱动的 4 数据信号组。TXD[0]是最低有效位，TXD[3]是最高有效位。TX_EN 有效时，数据有效；TX_EN 无效时，数据不能对 PHY 产生影响。
TX_ER	发送错误（可选）。 仅节能以太网（EEE）需要。通过对 CRC 取反表明发送错误。远程站点可以通过检测不正确的 CRC，检测到发送错误。
TX_EN	发送使能。 指示 MAC 正在发送 MII 半字节。它必须与前导码的第一个半字节同步有效（TX_CLK），并且必须在 MII 半字节发送过程中保持有效。
RX_CLK	接收数据参考时钟。 2.5MHz@10Mbps，25MHz@100Mbps。
RXD[3:0]	接收数据。 由 PHY 和有效 RX_DV 信号同步驱动的 4 数据信号组。RXD[0]是最低有效位，RXD[3]是最高有效位。当 RX_EN 无效且 RX_ER 有效时，PHY 通过 RXD[3:0]传输特定信息。

RX_ER	接收错误。 该信号必须在一个或多个时钟周期（RX_CLK）内有效，以向 MAC 子层表明在帧的某个位置检测到错误。只有 RX_DV 有效，此错误信号才有效。
RX_DV	接收数据有效。 表示 PHY 正在将接收到的数据恢复并解码后发送给 MII。它必须与帧的第一个（恢复后的）半字节同步有效（RX_CLK），并且必须保持有效，直到最后一个（恢复后）半字节。它必须在最后一个半字节之后的第一个时钟周期之前无效。为了正确接收帧，RX_DV 信号必须涵盖帧，且开始时间不得晚于起始帧定界符（SFD）字段。
CRS	载波侦测。 当发送或接收介质处于非空闲状态时，PHY 使该信号有效；当发送和接收介质都处于空闲状态时，PHY 使该信号无效。PHY 必须确保在整个冲突条件持续期间，CRS 信号一直有效。此信号不需要与发送和接收时钟同步。在全双工模式下， MAC 子层不关注该信号的状态。
COL	冲突检测信号。 PHY 必须在检测到介质上发生冲突时使该信号有效，并且在冲突条件持续期间保持有效。此信号不需要与发送和接收时钟同步。在全双工模式下， MAC 子层不关注该信号的状态。

3.3.2. 简化介质独立接口（RMII）

相对于 IEEE802.3 所定义的 16 信号独立介质接口（MII），简化独立介质接口（RMII）只有 7 个信号。RMII 模块位于 MAC 和外部 PHY 之间，当选择 RMII 时，RMII 模块执行 MII 信号到 RMII 信号的转换。

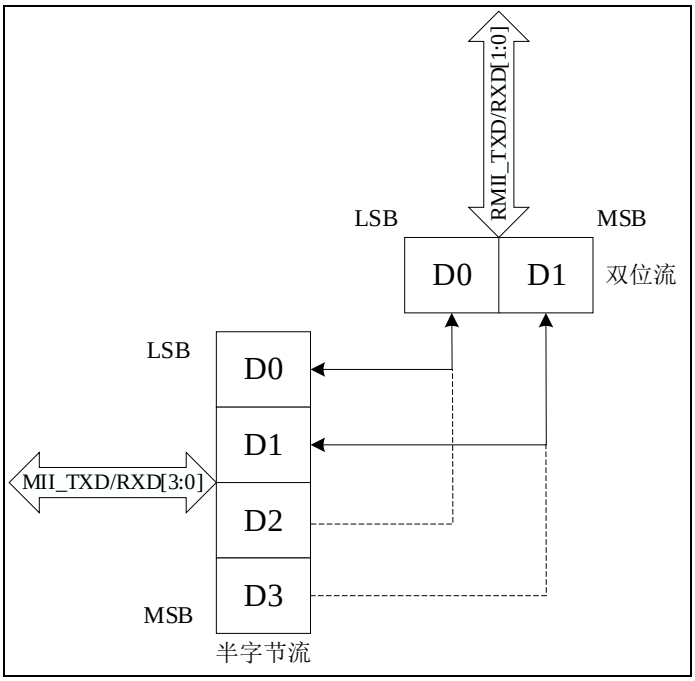
表 3-3 RMII 信号说明

RMII 信号	信号描述
REF_CLK	参考时钟。 50MHz。
TXD[1:0]	发送数据。
TX_EN	发送使能。 为高时表示 TXD[1:0]上正在发送有效数据。
RXD[1:0]	接收数据。
CRS_DV	载波侦测（CRS）和接收数据有效（RX_DV）随时钟周期交替服复用。在 10Mbps 模式下，每 10 个时钟周期交替一次。

3.3.3. 传输位序

MII 是半字节（4 位）传输，RMII 是双位传输，因此每次 MII 传输需要转换成两次 RMII 传输，传输顺序是先传低 2 位，后传高 2 位。

图 3-3 传输位序示意图



3.3.4. 站点管理接口（SMI）

站点管理接口（SMI）允许应用程序通过时钟（MDC）和数据（MDIO）双线访问任意 PHY 寄存器。该接口最多支持访问 32 个 PHY，但同时只能访问一个。

表 3-4 SMI 信号说明

SMI 信号	说明
MDC	参考时钟。 IEEE802.3 规范定义 MDC 最高频率为 2.5MHz，但是 ACM32H5xx 可以支持更高的时钟频率。MDC 的时钟源是 HCLK，分频因子依赖于 ETH_MACMDIOAR 寄存器的 CR[3:0]字段。
MDIO	双向数据信号。空闲时，MDIO 位高阻状态。

MAC 通过 SMI 接口按照固定的帧格式进行交互。SMI 帧格式如表 3-4 所示。

表 3-4 SMI 帧结构

	SMI 帧字段							
	报头 (32 位)	起始 (2 位)	操作码 (2 位)	PHY 地址 (5 位)	寄存器地址 (5 位)	周转 (2 位)	数据 (16 位)	空闲
读	1...1	01	10	ppppp	rrrrr	Z0	d...d	Z
写	1...1	01	01	ppppp	rrrrr	10	d...d	Z

说明 1：Z 表示 MAC 将 MIDO 驱动为高阻态；

说明 2：每个字段的发送位序位 MSB 先发。

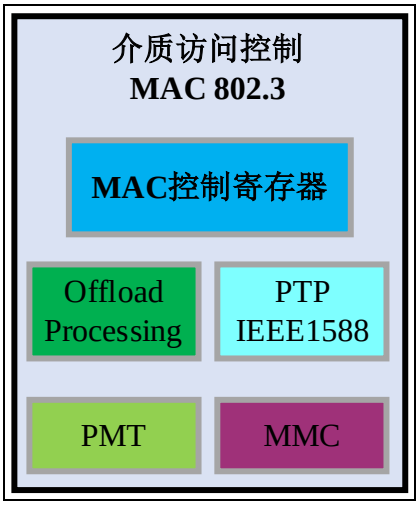
3.4. 介质访问控制：MAC 802.3

MAC 模块可以选择 10/100Mbps 数据速率，以及全双工和半双工工作方式。半双工模式使用带有 CSMA/CD（带冲突检测的载波侦听多路访问）算法来避免冲突，全双工模式无需冲突检测。

从应用角度来看，MAC 模块主要实现的功能为：

- 以太网帧的减荷处理
- 精密时间协议
- 电源管理
- 网络统计

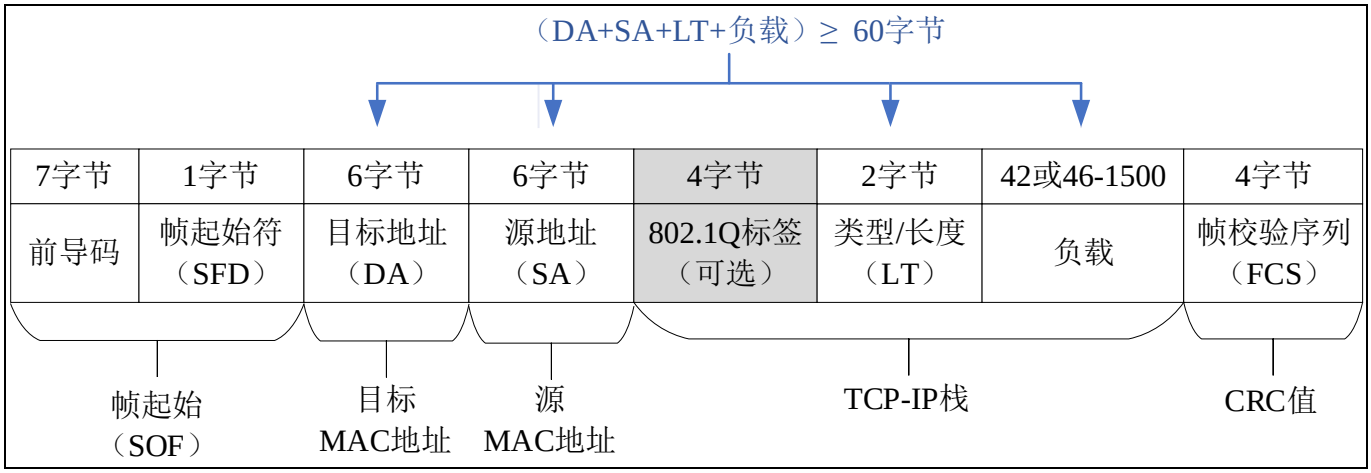
图 3-4 MAC 802.3 示意图



3.4.1. MAC 802.3 帧格式

MAC 802.3 帧格式包括基础帧格式和扩展帧格式。扩展帧格式在源地址和类型/ 长度字段之间增加了 4 字节的 802.1Q 标签字段，用于指示虚拟局域网（VLAN）。

图 3-5 MAC 帧格式



前导码是用于同步的 7 字节 0x55，SFD 是 1 字节 0xD5。

类型/长度是一种兼容的格式，当小于等于 0x5DC 时，表示负载数据长度；当大于等于 0x600 时，表示负载的类型，例如 0x0800 表示 IPv4 协议包。

如果应用提供的数据（DA+SA+LT+负载）总长度低于 60 字节时，MAC 会自动在不足的负载数据后面填充 0（可编程 TDES0 的 DP 位禁止）。

发送时，MAC 会自动计算以太网帧的 CRC 值并添加到 FCS 字段。可以通过编程 TDES0 的 DC 位，禁止发送计算出来的 CRC 值，但是，如果 MAC 使能了自动填充，且数据（DA+SA+LT+负载）小于 60 字节，CRC 仍旧会被追加到填充帧的末尾。

接收时，如果配置了自动去除 CRC 和填充，MAC 会去除 LT 字段小于 0x600 的帧的填充和 FCS 字段，但是对于 LT 字段大于等于 0x600 的帧不执行自动去除操作。

3.4.2. MAC 帧发送和接收

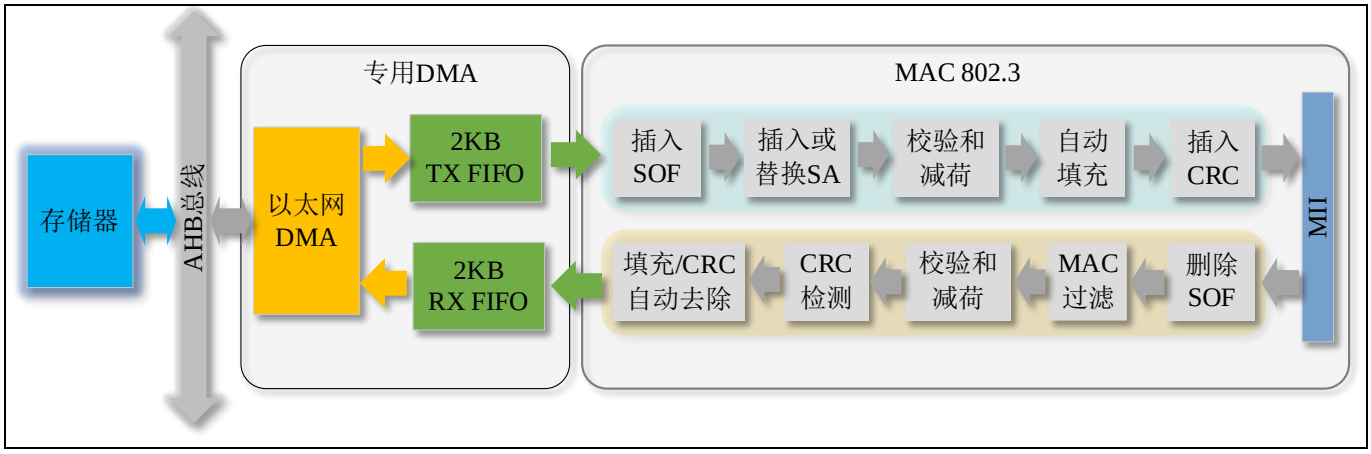
MAC 在接收和发送两个方向上对以太网帧进行处理，除了对帧起始字段的处理，其它都可以通过编程配置。

发送数据时，DMA 将系统存储器中的数据推送大 2KB 的 TX FIFO 中，然后弹出到 MAC，MAC 对 DMA 传输过来的数据进行处理，然后发送到 MII 上；

接收数据时，MAC 对 MII 接口上的数据推入 2KB 的 RX FIFO，并进行错误检测，仅有效帧会被转发给应用程序。

DMA 数据的推入和弹出有两种模式，一种是阈值（Threshold）模式，一种是存储转发（Store-and-forward）模式。只有在存储转发模式下，MAC 才会执行对发送/接收数据的处理。

图 3-6 以太网外设接收和发送数据示意图



3.4.2.1. 检验和减荷

MAC 针对 IPv4、TCP、UDP 和 ICMP 实现了检验和减荷功能。该功能支持校验和计算、发送路径中的校验和插入以及接收路径中的错误检测。只有将 FIFO 配置成存储转发（Store-and-Forward）模式时，校验和减荷才会生效。

设置校验和减荷的相关寄存器和描述付见表 3-5。

表 3-5 检验和减荷相关寄存器/描述符

寄存器/描述符	字段	说明
TDES1	CIC	发送时使能检验和计算及插入
ETH_MACCR	IPCO	接收时使能检验和减荷
ETH_DMAOMR	DTCEFD	是否丢弃校验和错误帧
RDES0	PCE 和 IPHCE	接收过程中出现 IP 首部错误和负载错误标志
TDES0	IHE 和 IPE	发送过程中出现 IP 首部错误和负载错误标志

3.4.2.2. MAC 过滤

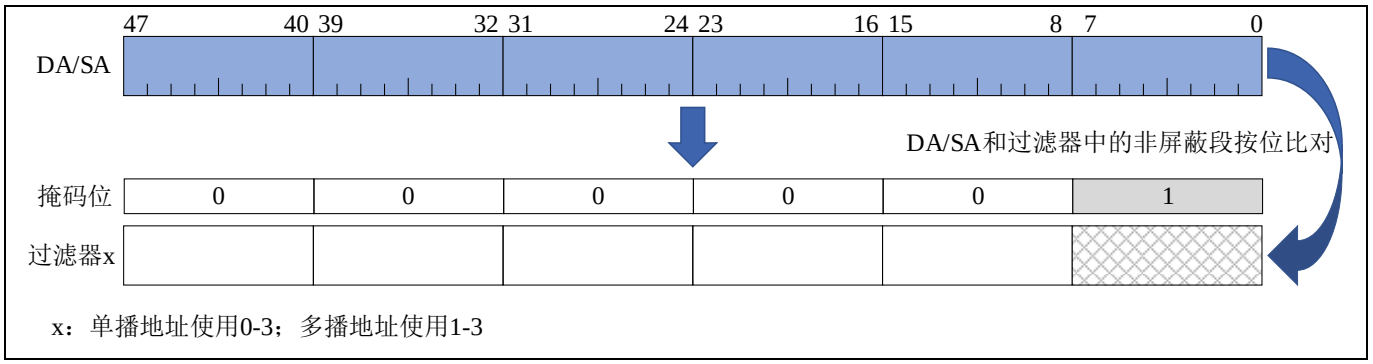
MAC 地址过滤会对所有接收到的帧的目的地址和源地址进行检查，并报告相应地址的过滤状态。

地址检查是基于应用程序选择的不同参数（帧过滤寄存器）来进行的。对于单播和多播目的地址，可以选择完全/组过滤或哈希过滤。广播目的地址默认全部通过，也可以配置 BFD 位丢弃所有广播帧。MAC 还可以对单播源地址进行完全/组过滤。

当相应过滤器的掩码位全部为 0 时，即完全过滤，此时会将源地址或目的地址与过滤器中的地址逐位比对，完全匹配则过滤成功，否则过滤失败。

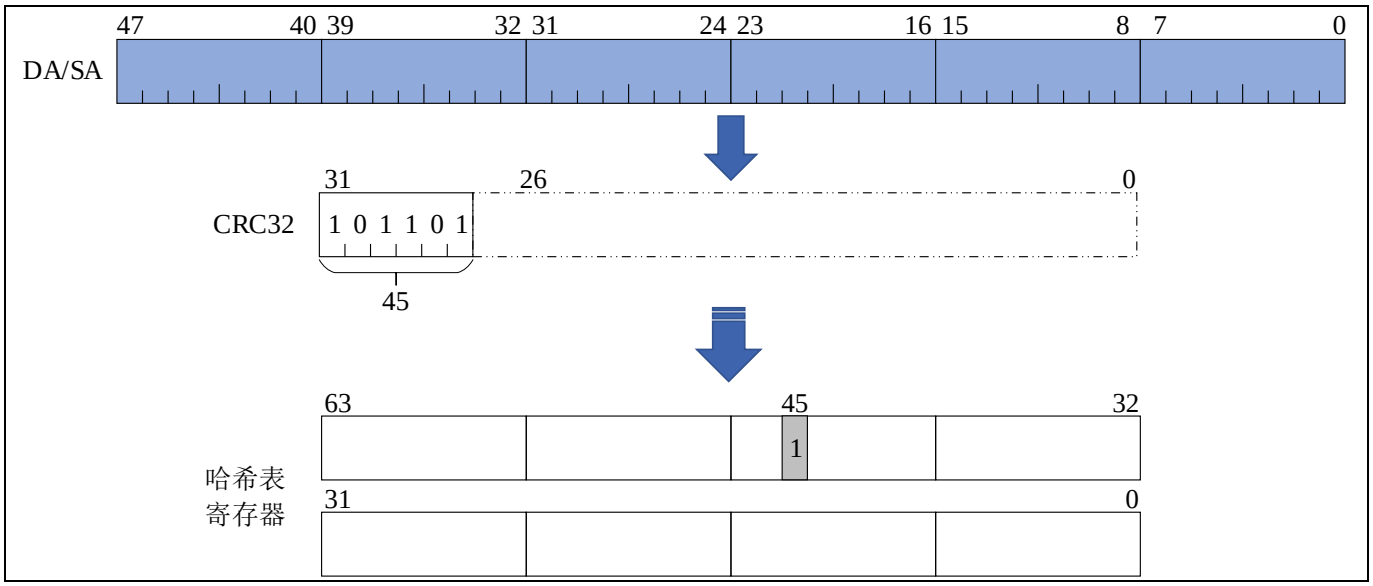
每个过滤器的掩码位对应 8 位过滤器地址，当设置了相应过滤器的掩码位时，则忽略该字段的比对结果，即组过滤。

图 3-7 组过滤示意图



选择哈希过滤时，MAC 使用 64 位哈希表执行非完全过滤。MAC 首先计算所收到帧的目的地址或源地址的 CRC 值，取高 6 位（0-63）作为哈希表的索引。如果索引对应的哈希表寄存器的相应位为 1，该帧通过哈希过滤。

图 3-8 哈希过滤示意图



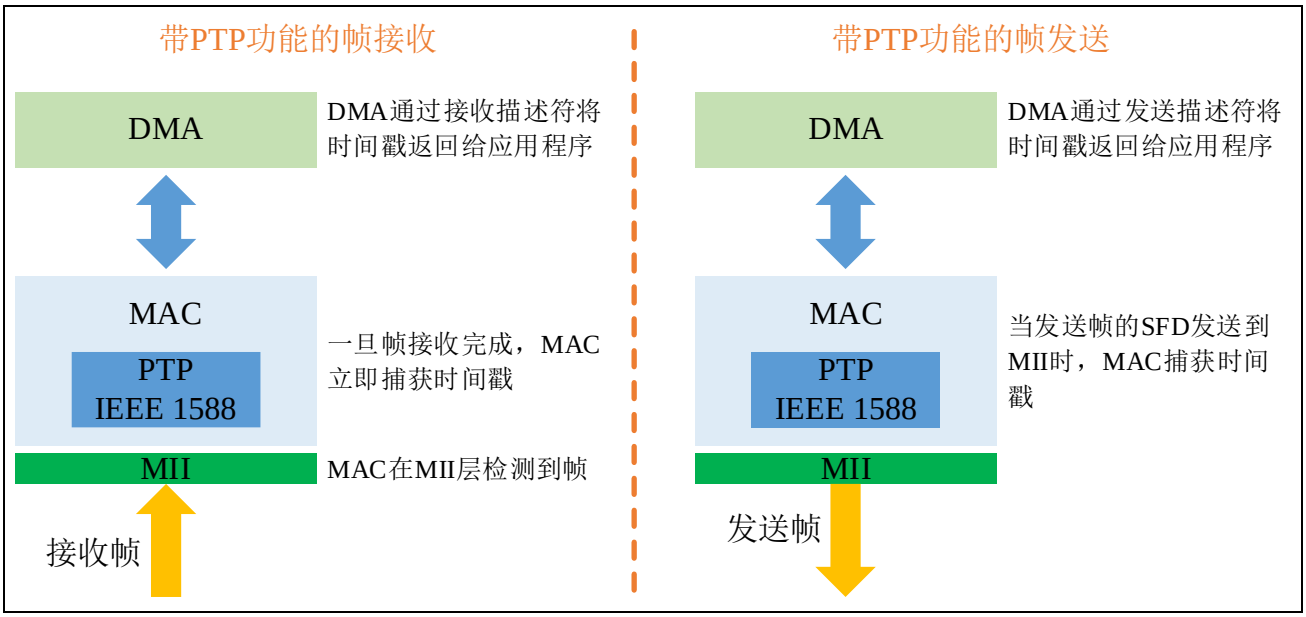
3.4.3. 精密时间协议（PTP IEEE1588）

IEEE 1588 标准定义了一种协议，该协议能够在采用网络通信、本地计算和分布式对象等技术实现的测量与控制系统中实现精确的时钟同步。该协议适用于支持多播消息传递的局域网通信系统，包括（但不限于）以太网。此协议用于对异构系统进行同步，这些系统包含具有不同固有精度、分辨率和稳定性的时钟。该协议利用最少的网络和本地时钟计算资源，支持实现亚微秒级的全系统同步精度。

3.4.3.1. 时间戳

时间戳在接收完成时和帧起始发送到 MII 上时由硬件捕获，DMA 通过接收/发送描述符将时间戳反馈给应用程序。

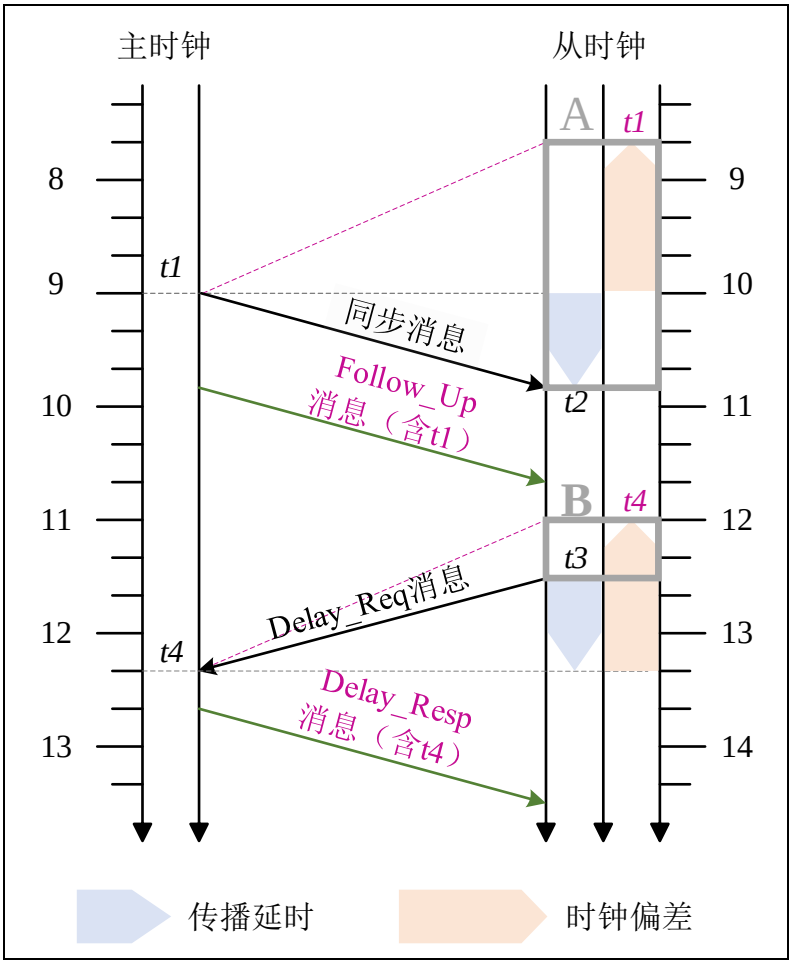
图 3-9 带 PTP 功能的帧收/发



3.4.3.2. 时钟同步

精确时间协议（PTP）是基于消息的协议，它通过用户数据报协议/网际协议（UDP/IP）进行传输。系统或网络被划分为主节点和从节点，用于分配时序/时钟信息。图 372 描述了通过交换 PTP 消息使从节点与主节点同步的协议技术。

图 3-10 网络时间同步示意图



从节点根据主机的时间信息 t1 和 t4，结合自己的时间戳信息（t2 和 t3），通过如下公式即可计算出时钟偏差。

$$A = t2 - t1 = 延时 + 偏差$$

$$B = t4 - t3 = 延时 - 偏差$$

$$延时 = (A + B)/2$$

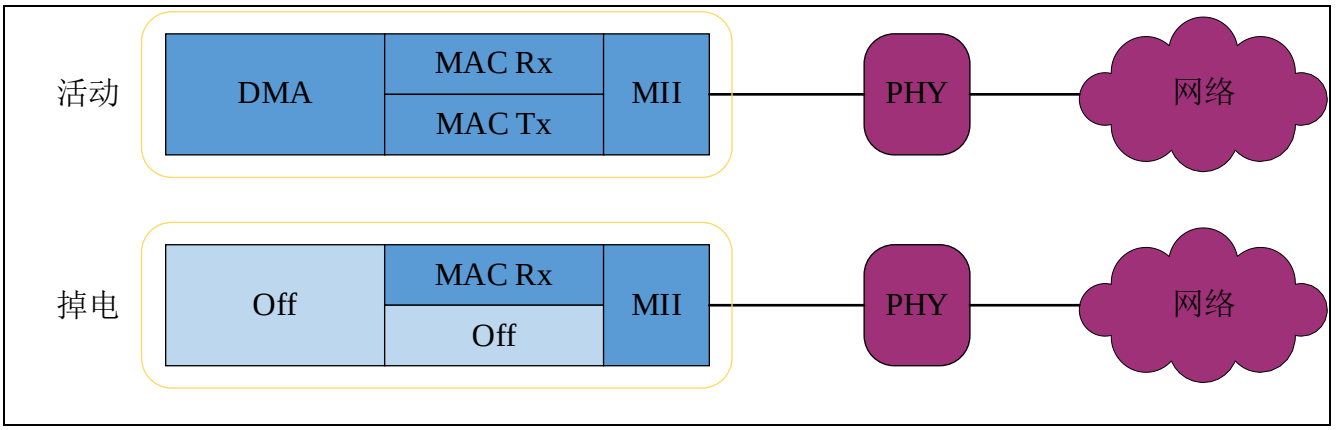
$$\boldsymbol{偏差 = (A - B)/2}$$

3.4.4. 电源管理（PMT）

如果外部中断（EXTI）线 19 使能，电源管理机制允许在系统处于停止（Stop）模式时，继续检测帧。

在掉电模式下， PHY、MII 接口和接收器保持活动，用于检测魔术包和远程唤醒帧，并产生 EXTI 19 唤醒信号。发送器和以太网 DMA 均应关闭。

图 3-11 电源管理示意图



3.5. 以太网专用 DMA

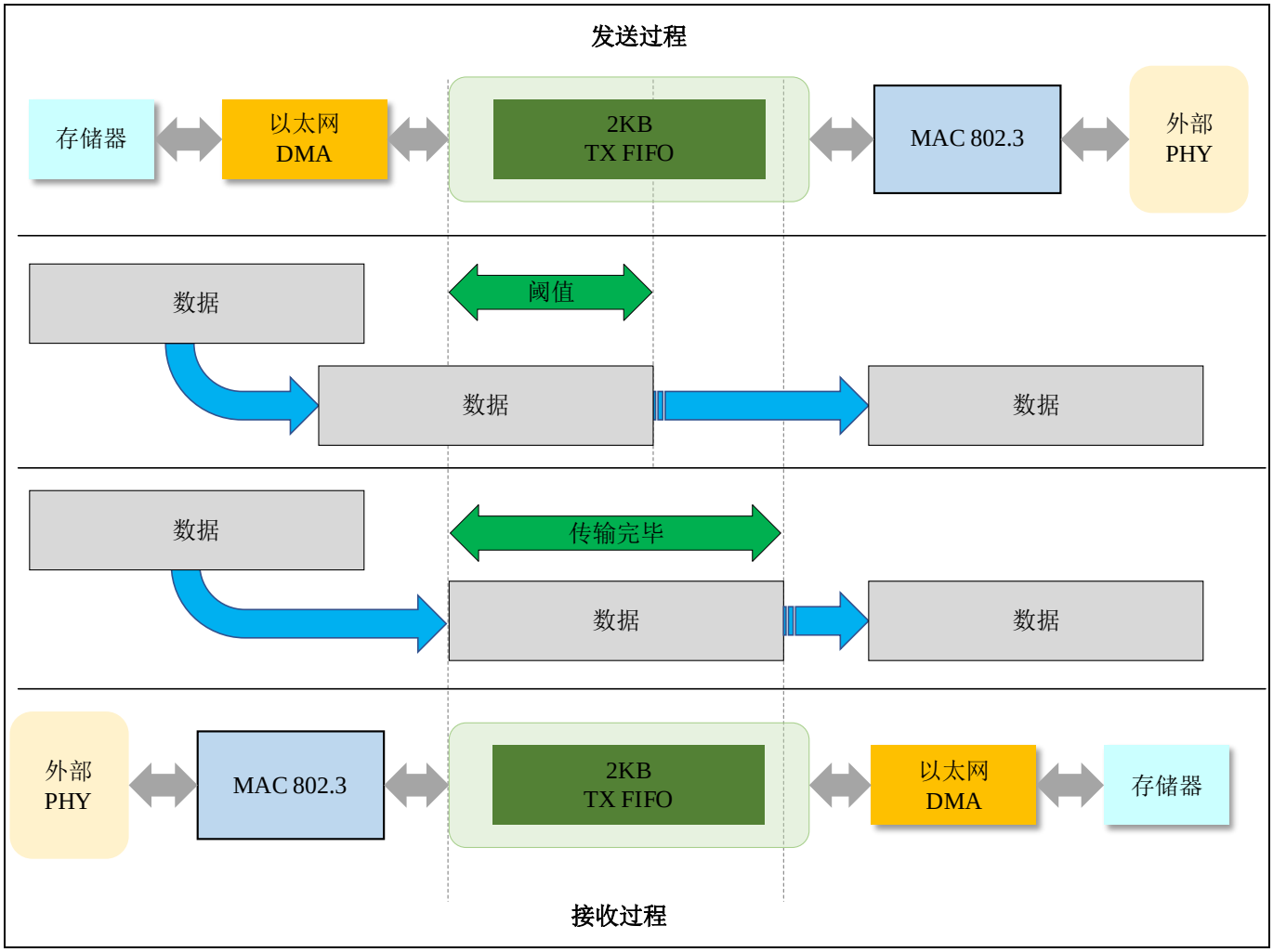
以太网专用 DMA 专为面向包的数据传送（如以太网中的帧）而设计。具有独立的发送和接收引擎以及相应的 CSR（控制和状态寄存器）空间。发送引擎将数据从系统存储器

传送到 Tx FIFO，而接收引擎将数据从 Rx FIFO 传送到系统存储器。 DMA 可以在 CPU 完全不干预的情况下，通过描述符有效地将数据从源传送到目标。

3.5.1. DMA FIFO 工作模式

以太网外设拥有 2KB 的接收 FIFO 和 2KB 的发送 FIFO，可以配置为阈值模式（Threshold）或存储转发（Store-and-Forward）模式。

图 3-12 数据收/发过程中 DMA 工作模式示意图



3.5.2. DMA 描述符和描述符列表

DMA 的传输是由描述符管理的。描述符分为发送描述符合接收描述符，每个描述符都由 4 个（常规描述符）或者 4 个（增强描述符）32 位的字组成，存储在内存中由用户配置。每个描述符最多指向两个缓冲区，DMA 将缓冲区的数据帧传输给 MAC 或者将 MAC 接收到的数据帧传输到缓冲区。单个描述符缓冲区内的数据通常包括整个帧或部分帧，但不能跨帧。

图 3-13 发送描述符

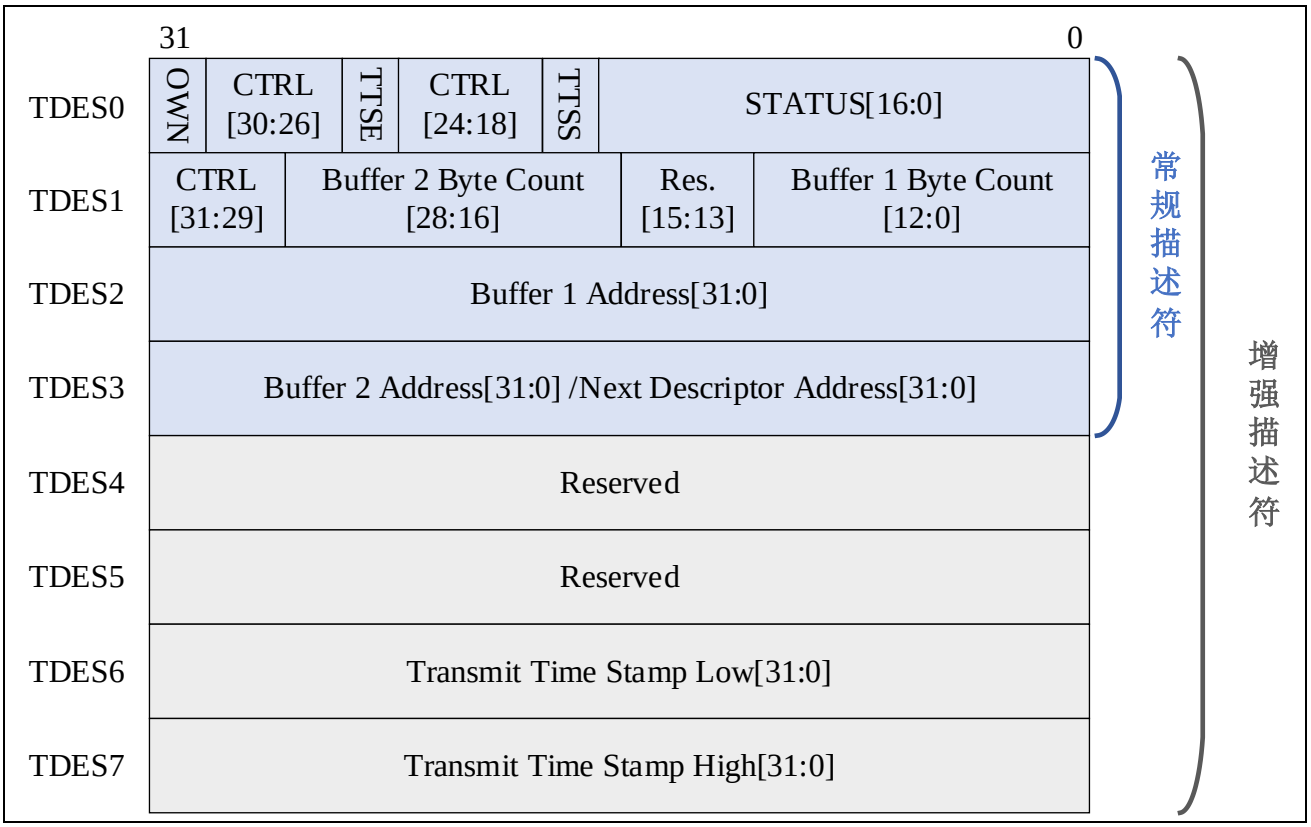
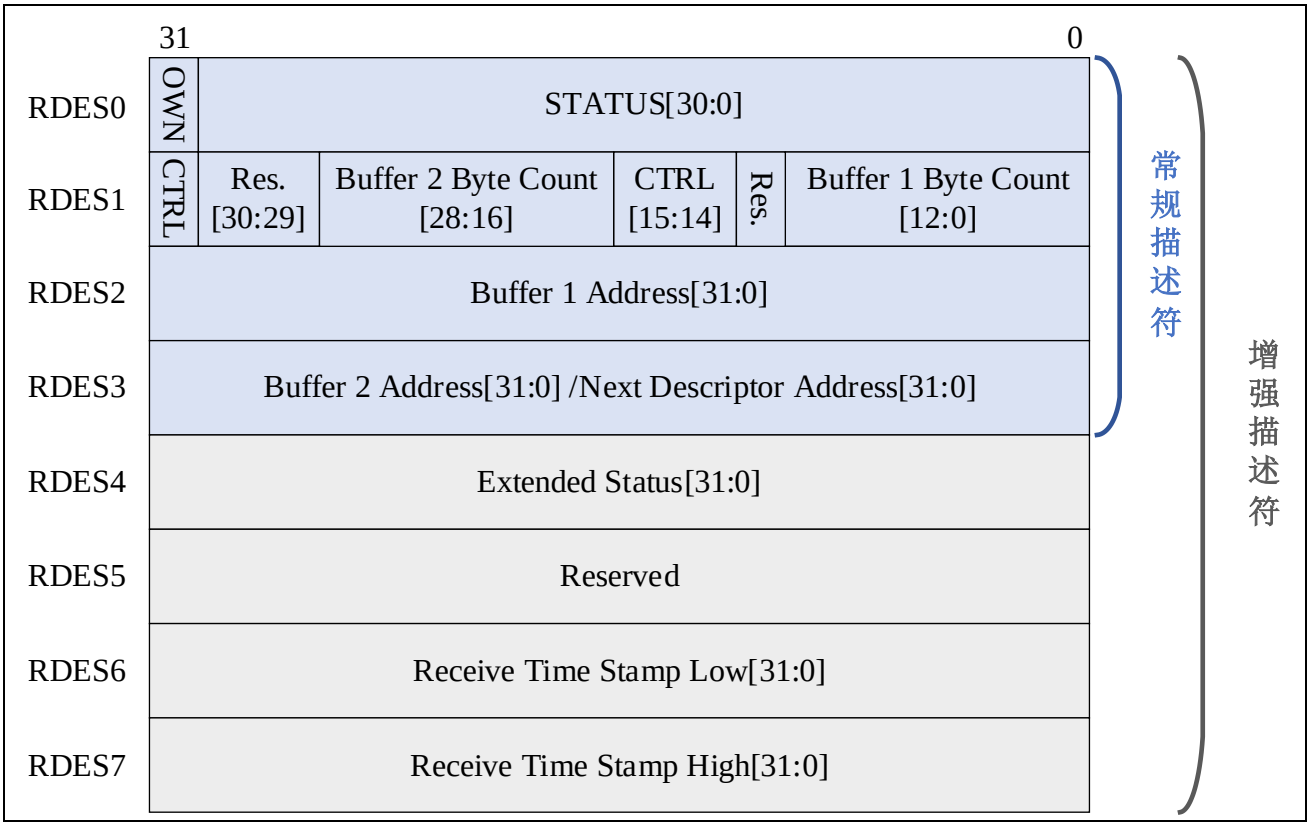


图 3-14 接收描述符



多个描述符按照环形结构或者链式结构组织成发送描述符列表或者接收描述符列表。描述符列表是一种前向环形列表，每个列表的基址（需 32 位对齐）分别写入 ETH_DMATDLAR 寄存器或 ETH_DMARDLAR 寄存器，最后一个描述符会标注为环尾，DMA 重新指向第一个描述符。

图 3-15 描述符列表示意图

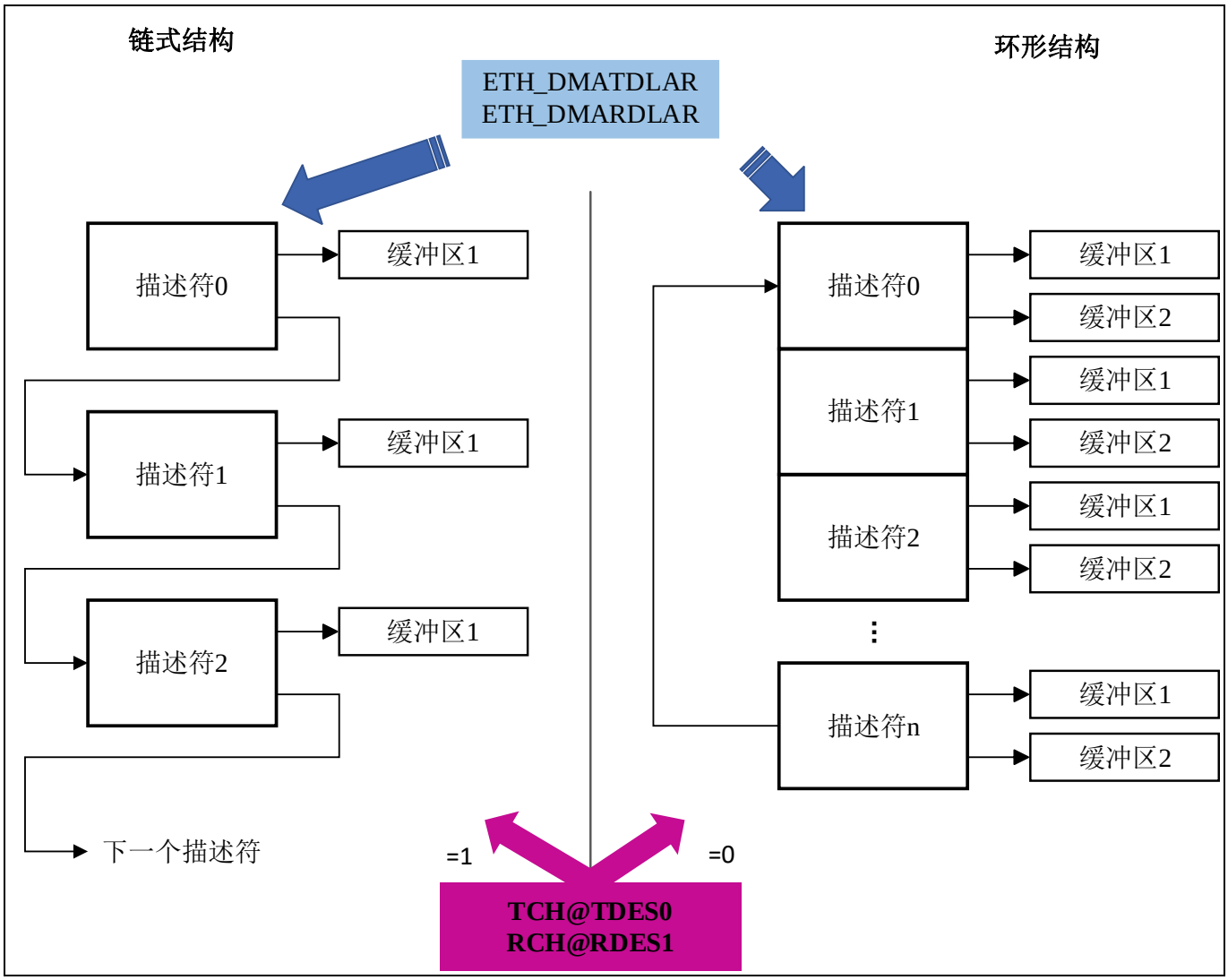


图 3-16 单帧多描述符链式结构发送示意图

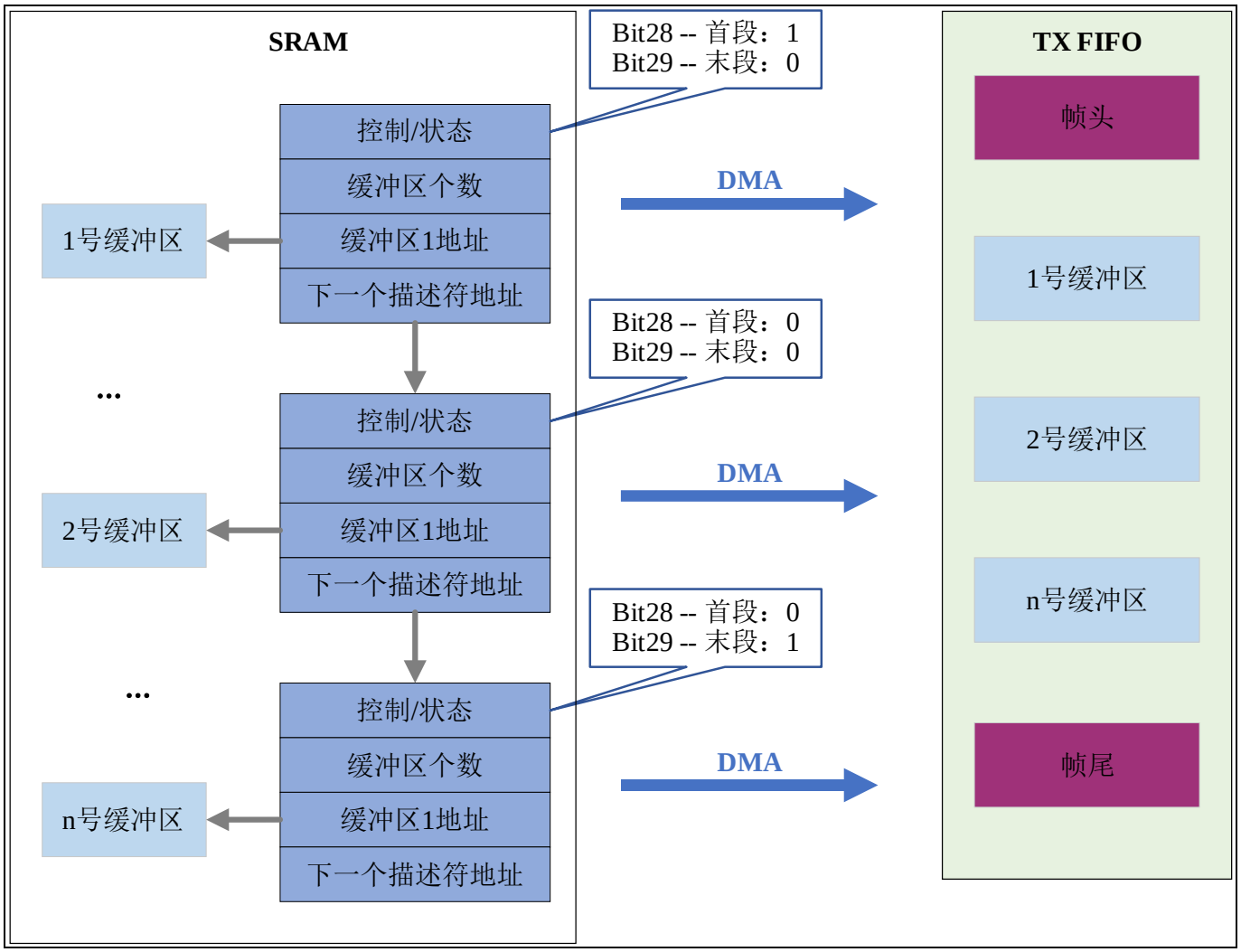
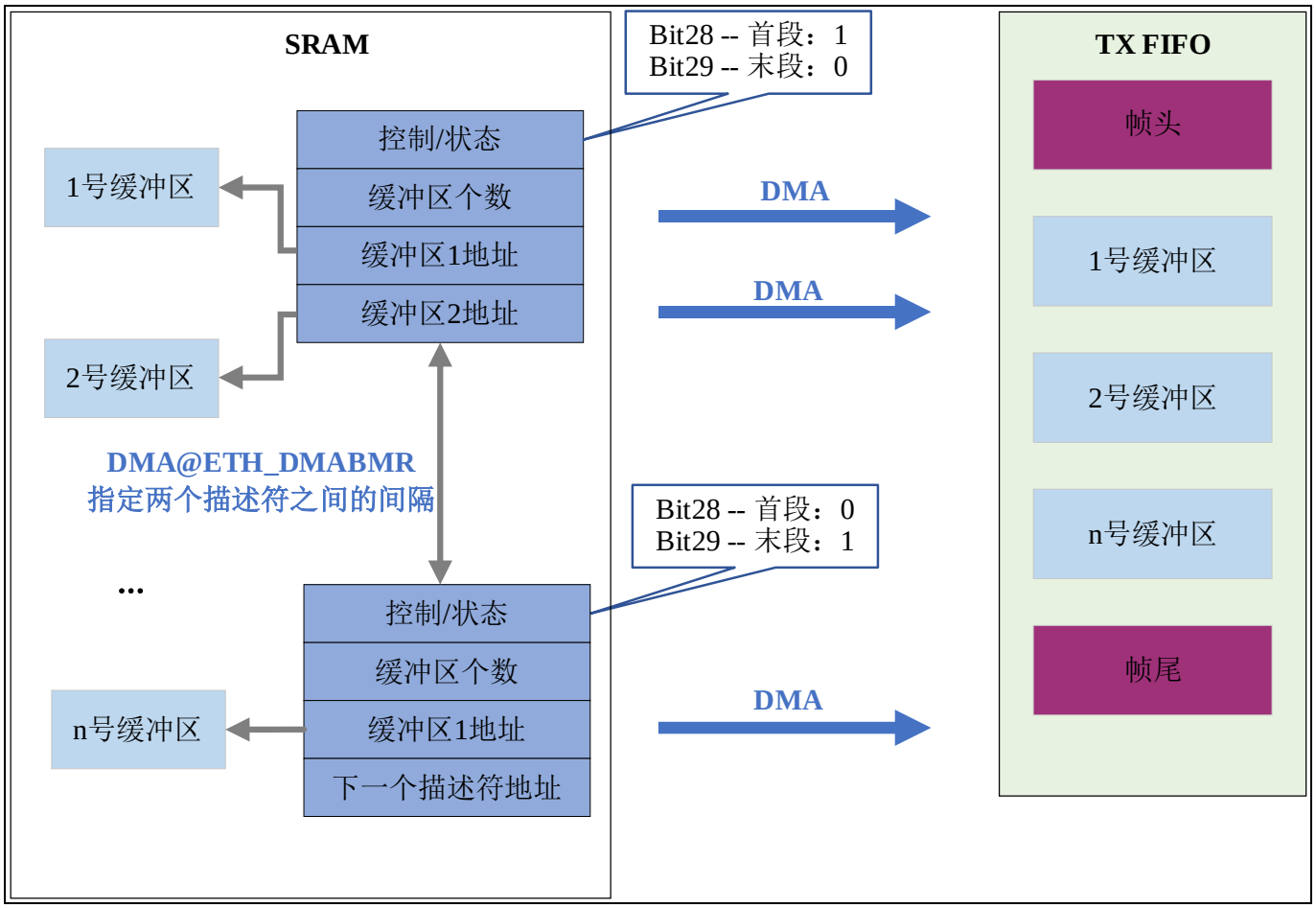


图 3-17 单帧多描述符环形结构发送示意图



4. 应用说明

本节内容主要介绍 ETH 的使用方法。

4.1. ETH 初始化

ACM32H5 HAL 库提供了初始化结构体成员，用于设置 ETH 工作环境参数，并由 ETH 相应初始化配置函数或功能函数调用，这些设定参数将会设置 ETH 相应的寄存器，达到配置 ETH 工作环境的目的。这些内容都定义在库文件“hal_eth.h”及“hal_eth.c”中。

```
HAL_ETH_InitDefaultParamter(&ETH_Handle);
memcpy((void *)ETH_Handle.Init.MACAddr,  (void *) macAddr,  6);
ETH_Handle.Instance                    = ETH;
ETH_Handle.Init.MediaInterface           = ETH_MEDIA_INTERFACE_RMII;
ETH_Handle.Init.AutoNegotiation          = ENABLE;
ETH_Handle.Init.Speed                  = ETH_SPEED_100M;
ETH_Handle.Init.DuplexMode             = ETH_MODE_FULL_DUPLEX;
ETH_Handle.Init.AutoCRC                = ENABLE;
ETH_Handle.Init.AutoPad                = ENABLE;
ETH_Handle.Init.AutoChecksumInsertion = ETH_CHECKSUM_INSERTION_CTRL_IP_HEAD;
ETH_Handle.Init.PhyAddress             = ETH_PHY_ADDR;
ETH_Handle.Init.TxBuff                 = TX_Buff[0];
ETH_Handle.Init.TxBuffNbr              = ETH_TX_DESC_CNT;
ETH_Handle.Init.TxBuffLen              = ETH_TX_BUFF_SIZE;
ETH_Handle.Init.RxBuff                 = Rx_Buff[0];
ETH_Handle.Init.RxBuffNbr              = ETH_RX_DESC_CNT;
ETH_Handle.Init.RxBuffLen              = ETH_RX_BUFF_SIZE;
ETH_Handle.Init.TxDesc                 = DMATxDscrTab;
ETH_Handle.Init.RxDesc                 = DMARxDscrTab;
ETH_Handle.Init.TxDescNbr              = ETH_TX_DESC_CNT;
ETH_Handle.Init.RxDescNbr              = ETH_RX_DESC_CNT;
ETH_Handle.Init.TxDescListMode         = ETH_DESC_LIST_MODE_LIST;
ETH_Handle.Init.RxDescListMode         = ETH_DESC_LIST_MODE_LIST;
ETH_Handle.Init.TxBuffTab              = TxBuffTab;
ETH_Handle.Init.TxBuffNodeNbr          = ETH_TX_DESC_CNT;
ETH_Handle.Init.RxBuffTab              = RxBuffTab;
ETH_Handle.Init.RxBuffNodeNbr          = ETH_RX_DESC_CNT;
ETH_Handle.Init.Delay.Uint               = 2;
ETH_Handle.Init.Delay.Len               = 5;
#if (ETH_VLAN_SUPPORT == ENABLED)
ETH_Handle.MACConfig.VLAN.Enable = ENABLE;
ETH_Handle.MACConfig.VLAN.Mode  =  ETH_VLAN_MODE_REPLACE;
ETH_Handle.MACConfig.VLAN.Tag   =  vlanId;
#endif
ETH_Handle.IT.TxComplete    = ENABLE;
ETH_Handle.IT.RxComplete    = ENABLE;
ETH_Handle.IT.Error        = DISABLE;
ETH_Handle.IT.MMCTx        = DISABLE;
ETH_Handle.IT.MMCRx        = DISABLE;
ETH_Handle.IT.LPI          = DISABLE;
ETH_Handle.IT.PTP          = DISABLE;
ETH_Handle.IT.PMT          = DISABLE;
```

```
HAL_ETH_Init(&ETH_Handle) ;
```

首先调用 HAL_ETH_InitDefaultParamter()函数将所有参数设置为默认值，然后根据应用需求设置相应参数和中断，最终调用 HAL_ETH_Init()函数完成初始化。

其中比较重要的参数说明如下：

- ETH_Handle.Init.MACAddr

  MAC 地址，硬件地址，为网络中唯一标示。MAC 地址长度 6 字节，格式为 XX-XX-XX-XX-XX-XX。

- ETH_Handle.Init.MediaInterface

  ETH 支持 MII 接口（ETH_MEDIA_INTERFACE_MII）、RMII 接口（ETH_MEDIA_INTERFACE_RMII）。

- ETH_Handle.Init.PhyAddress

  5 位 PHY 地址，最大支持 32 个 PHY。

- ETH_Handle.Init.AutoNegotiation

  通过 SMI 接口配置 PHY 工作自从协商模式，根据网络情况自动调整以太网速度和全/半双工模式。

- ETH_Handle.Init. Speed

  ETH 支持 10Mbps（ETH_SPEED_10M）/100Mbps（ETH_SPEED_100M）。

- ETH_Handle.Init. DuplexMode

  ETH 支持全双工（ETH_MODE_FULL_DUPLEX）/半双工（ETH_MODE_HALF_DUPLEX）。

- ETH_Handle.Init.TxBuff / ETH_Handle.Init.RxBuff

  第一个发送/接收缓冲区指针。

- ETH_Handle.Init.TxBuffNbr / ETH_Handle.Init.RxBuffNbr

  发送/接收缓冲区个数。

- ETH_Handle.Init.TxBuffLen / ETH_Handle.Init.RxBuffLen

  发送/接收缓冲区长度。

- ETH_Handle.Init.TxDesc / ETH_Handle.Init.RxDesc

  发送/接收描述符列表基址。

- ETH_Handle.Init.TxDescNbr / ETH_Handle.Init.RxDescNbr

  发送/接收描述符个数。

- ETH_Handle.Init.TxDescListMode / ETH_Handle.Init.RxDescListMode

  发送/接收描述符列表结构：环形（ETH_DESC_LIST_MODE_RING）或链式（ETH_DESC_LIST_MODE_LIST）。

- ETH_Handle.Init.TxBuffTab / ETH_Handle.Init.RxBuffTab

  TxBuffTab / RxBuffTab 中包括发送/接收缓冲区地址、发送/接收数据总长度、下一个发送/接收缓冲区地址。此变量主要为 LWIP 服务。

- ETH_Handle.Init.TxBuffNodeNbr / ETH_Handle.Init.RxBuffNodeNbr

  发送/接收缓冲区列表节点个数。

- ETH_Handle.Init.AutoCRC

  AutoCRC 选择是否自动将硬件计算出来的 CRC 值附加在帧尾。

- ETH_Handle.Init. AutoPad

  选择是否对（DA+SA+LT+负载）小于 60 字节的帧自动填充。

- ETH_Handle.Init. AutoChecksumInsertion

  校验和插入控制，选择硬件是否自动对 IP 报头校验和进行计算和插入、对有效负载（TCP/UDP/ICMP）报头校验和进行计算和插入。

  对于 LWIP，禁止软件对 IP/TCP/UDP/ICMP 报头校验和进行计算和插入，使能硬件自动计算和插入，可节约软件计算的时间。

  Cyclone TCP 强制对 IP/TCP/UDP/ICMP 报头进行计算和插入，因此可禁止校验和插入控制。

  ETH 在对 ICMP 报头进行计算和插入时，校验和初始值必须为 0，而 Cyclone TCP 对 ICMP 报头已强制进行计算和插入，因此，ETH 在 Cyclone TCP 时不能设置对有效负载报头校验和进行计算和插入，否则将会造成 ICMP 报头校验和的错误，导致 PING 协议错误。

- ETH_Handle.Init.Delay.Uint 和 ETH_Handle.Init.Delay.Len

  Uint 指单位延时的延时步长。Len 指延时线，即单位延时个数。

  延时模块（DLYB）对输入 ETH 的 RX 时钟进行相位偏移编程，对接收数据延迟采样采样。DLBY 包括三个时间概念，基本延迟，初始延迟，延迟步长。

  基本延迟，指使用 DLYB 时外围模块消耗的时间，约为 1.7ns；

  初始延迟，指每个单位延迟的基本延迟，典型值为 550ps。

  延迟步长，指每个单位延迟中一个 Uint 的延迟时间，典型值为 50ps。

  DLYB 延迟时间计算公式：

  DLYB = 基本延迟 + （ 初始延迟 + 延迟步长 * Uint ）* Len

4.2. PHY 选择

ACM32H5 EVB 开发板采用的 PHY 芯片为 LAN8710。

PHY 时钟可由独立的外部时钟供给，也可由 ACM32H5 供给。MAC 时钟 TX_CLK，RX_CLK/ REF_CLK 由 PHY 提供。

如果使用 LPI 功能，可选择支持“Compliant with Energy Efficient Ethernet IEEE 802.3az”功能的 PHY 芯片，如 LAN8740。ETH 进入发送 LPI 状态，PHY 可节约 43mA 电流。ETH 进入接收 LPI 状态，PHY 可节约 4mA 电流。

如果使用 WOL 功能，可选择支持“Wake on LAN (Wol) support”功能的芯片，如 LAN8740、LAN8742。

4.3. 发送数据

```
HAL_StatusTypeDef HAL_ETH_Transmit(ETH_HandleTypeDef *heth, ETH_BuffTypeDef *buff,
                                   uint32_t mode, ETH_TxStatusTypeDef *pStatus);
```

- heth：ETH 句柄。
- buff：发送数据缓冲区。这是一个 ETH_BuffTypeDef 类型，可以输入多个缓冲区，可以实现大型数据包的发送。

```
typedef struct __ETH_BufferTypeDef
{
  uint8_t *Buff;
  uint32_t Len;
  struct __ETH_BufferTypeDef *next;
} __attribute__((aligned(4))) ETH_BuffTypeDef;
```

- mode：发送模式。发送模式包括 ETH_TX_MODE_DATA_COPY、ETH_TX_MODE_TIMESTAMP、

ETH_TX_MODE_WAIT_TX_COMPLETE 三个模式。

- ETH_TX_MODE_DATA_COPY：数据复制模式。

  当发送数据存储在不同位置（通过 buff 链接多个数据段），而且除了最后一段数据外，其它数据段长度有小于描述符缓冲区长度的现象，此时，必须设置 ETH_TX_MODE_DATA_COPY，通过复制方式将 buff 指定的多段数据依次复制到描述符缓冲区，从而使所有数据依次填满描述符缓冲区。

  当发送数据只有一段（buff 只有一个数据段），或者除了最后一段数据外，其它数据段长度都等于描述符缓冲区长度时，可以不选择 ETH_TX_MODE_DATA_COPY，此时用 buff 中每个数据段的地址代替描述符缓冲区地址，直接使用 buff 发送数据。

  使用 ETH_TX_MODE_DATA_COPY 发送数据，将多出一个数据复制的时间，发送效率较低。但驱动和应用层之间的耦合性较低。

  不使用 ETH_TX_MODE_DATA_COPY 发送数据，无需数据复制，ETH 直接使用 buff 发送，发送效率较高，但驱动和应用之间的耦合性较高，在驱动发送完成前，应用不能使用 buff 的缓冲区。

- ETH_TX_MODE_TIMESTAMP：时间戳模式。

  设置 ETH_TX_MODE_TIMESTAMP，阻塞式发送，且发送前须在 DESC0 中设置时间戳标志，发送完成后读取时间戳。

- ETH_TX_MODE_WAIT_TX_COMPLETE：等待发送完成模式。

  设置 ETH_TX_MODE_WAIT_TX_COMPLETE，阻塞式发送，发送完成后退出函数。

- pStatus：发送状态。

  返回发送状态，DESC0 及时间戳。

4.4. 接收数据

```
HAL_StatusTypeDef HAL_ETH_Receive(ETH_HandleTypeDef *heth, ETH_BuffTypeDef **buff,
                                  ETH_RxStatusTypeDef *pStatus);
HAL_StatusTypeDef HAL_ETH_ReleaseRxDescriptors(ETH_HandleTypeDef *heth);
```

接收数据分为两个步骤：第一步，接收数据，第二步，释放接收缓冲区。

接收数据，使用 HAL_ETH_Receive()，ETH 将接收缓冲区组成 ETH_BuffTypeDef 类型的链表，并将链表首地址返回到 buff，接收状态（DES0、DES4、时间戳）返回到 pStatus。

用户可直接使用 buff 进行数据处理，也可将数据 copy 到应用层进行处理。

当用户处理完数据，或将所有数据 copy 到应用后，调用 HAL_ETH_ReleaseRxDescriptors()释放接收描述符。只有调用 HAL_ETH_ReleaseRxDescriptors()之后，ETH 才能使用其接收描述符进行继续接收数据。

4.5. 中断

```
void HAL_ETH_TxCpltCallback(ETH_HandleTypeDef *heth);
void HAL_ETH_RxCpltCallback(ETH_HandleTypeDef *heth);
```

使能发送中断，发送完成后自动调用 HAL_ETH_TxCpltCallback()，应用可以在此处理发送事宜。

使能接收中断，接收成功后自动调用 HAL_ETH_RxCpltCallback()，应用可以在此处理接收事宜。

5. 速度测试

测试平台：

- Jperf 2.0.2@Windows11
- 路由器
- Cyclone_TCP_LAN8720 @ACM32H5 EVB 开发板

5.1. 发送速度测试

EVB 开发版配置

测试工程：Socket_UDP_Send

ETH 属性设置：

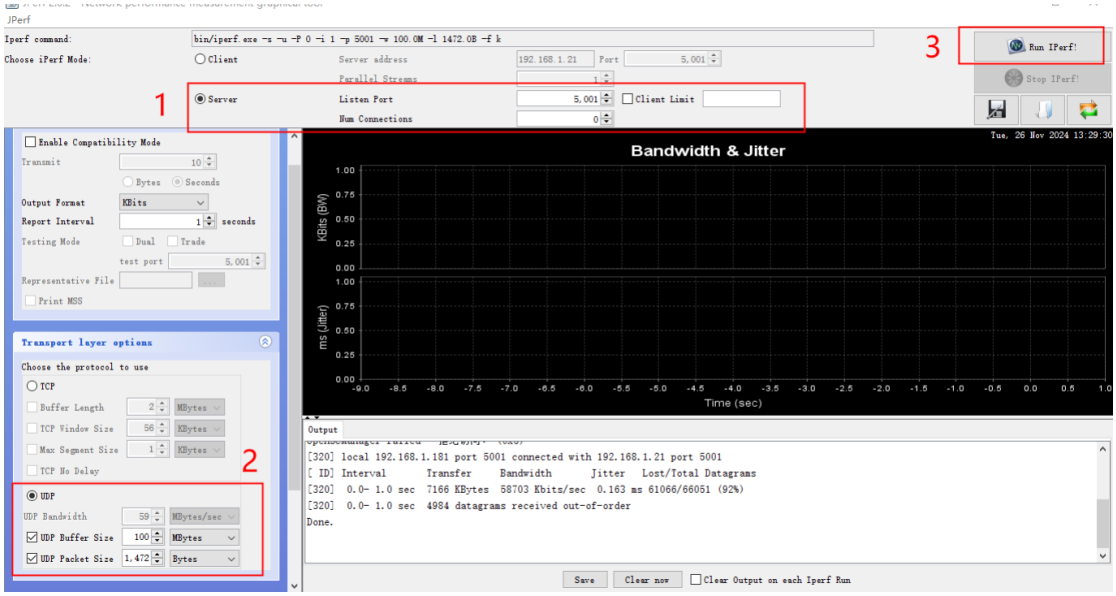
1）IPv4 地址：192.168.1.21

2）IPv4 子网掩码：255.255.255.0

3）IPv4 默认网关：192.168.1.1

4）UDP Port：5001

5.1.1. Jperf 2.0.2 设置



1）选择 Server

- Listen Port：5001（EVB 开发板 PORT）
- 不选择 Client Limit，无需输入 EVB 开发板 IP 地址
- Num Connections：0

2）选择 UDP

- 选择 UDP Buffer Size：100 Mbytes
- 选择 UDP Packet Size：1472 Bytes

5.1.2. 操作流程

1）烧录程序并复位或重新上电；

EVB 开发板输出：

```
************************************************************

system startup
reset source: 10   0
reset source: sysreq.
HCK: 180000000
PCLK1: 180000000
PCLK2: 180000000
PCLK3: 180000000
PCLK4: 180000000

************************************************************

Press the key to start testing UDP sending speed.
```

2）打开 Jperf 软件，按 4.1.2 设置。

3）点击 Jperf 软件的“Run Iperf!”，启动 UDP 接收。

4）EVB 开发板按 KEY，启动 UDP 发送。

UDP 发送完成，EVB 开发板输出：

```
*********************  test start  *********************

Detect network connection status.
Network connection succeeded.

remote: 192.168.1.181, 5001
local: 192.168.1.21, 5001

create socket
Successfully created socket.

Send UDP packets, totally 10000 packets, each packet length: 1518 bytes and payload: 1472 bytes.
data length: 15180000 bytes, time: 1937 ms, speed: 7.000000 MB/s  62.000000 Mbps

close socket

*********************  test end  *********************
```

5.2. 接收速度测试

EVB 开发版配置

测试工程：Socket_UDP_Recv

ETH 属性设置：

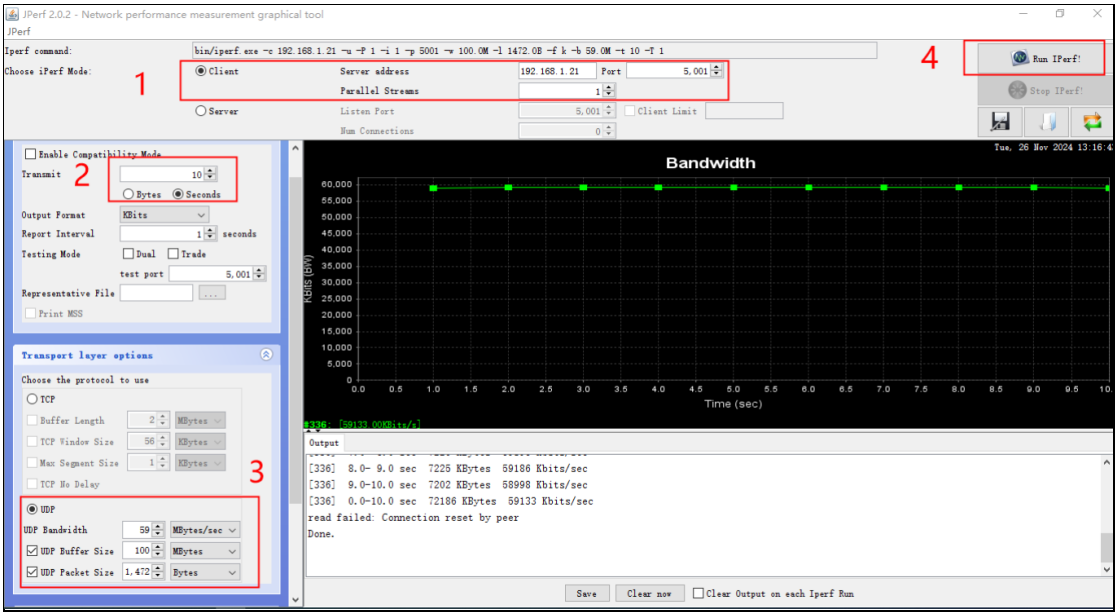
1）IPv4 地址：192.168.1.21

2）IPv4 子网掩码：255.255.255.0

3）IPv4 默认网关：192.168.1.1

4）UDP Port：5001

5.2.1. Jperf 2.0.2 设置



1）选择 Client

- Server address：192.168.1.21（EVB 开发板 IP 地址）
- Port：5001（EVB 开发板 PORT）
- Parallel Streams：1

2）Transmit 选择 10 seconds。

3）选择 UDP

- UDP Bandwidth：59 Mbytes/sec
- 选择 UDP Buffer Size：100 Mbytes
- 选择 UDP Packet Size：1472 Bytes

5.2.2. 操作流程

1）烧录程序并复位或重新上电；

EVB 开发板输出：

```
************************************************************

system startup
reset source: 10   0
reset source: sysreq.
MCK: 180000000
PCLK1: 180000000
PCLK2: 180000000
PCLK3: 180000000
PCLK4: 180000000

************************************************************

Press the key to start testing UDP sending speed.
```

2）打开 Jperf 软件，按 4.2.2 设置。

4）点击 Jperf 软件的“Run Iperf!”，启动 UDP 发送。

5）EVB 开发板按 KEY，启动 UDP 接收。

UDP 接收完成，EVB 开发板输出：

```
************************  test start  ************************

Detect network connection status.
Network connection succeeded.

remote: 192.168.1.181, 5001
local: 192.168.1.21, 5001

create socket
Successfully created socket.

Send UDP packets, totally 10000 packets, each packet length: 1518 bytes and payload: 1472 bytes.
data length: 15180000 bytes, time: 1937 ms, speed: 7.000000 MB/s  62.000000 Mbps

close socket

************************  test end  ************************
```

6. 注意事项

6.1. 发送描述符中校验和插入控制

有效负载报头校验和的计算和插入，当有效负载为 ICMP 时，原始校验和必须为 0，如果外部已经经过软件校验和计算和插入或不为 0，再次进行硬件校验和的计算和插入，会发现校验和计算错误，导致 PING 协议错误。

如果软件对有效负载报头校验和已经进行计算和插入，此时 CIC 不能选择 ETH_CHECKSUM_INSERTION_CTRL_IP_HEAD_PAYLOAD 和 ETH_CHECKSUM_INSERTION_CTRL_FULL。

Cyclone TCP 强制对 IP 报头、有效负载、伪报头进行软件校验和和插入，因此，移植时可选择 ETH_CHECKSUM_INSERTION_CTRL_DISABLE 和 ETH_CHECKSUM_INSERTION_CTRL_IP_HEAD。

LWIP 没有强制，可在头文件中设置是否需要软件进行校验和计算和插入。

6.2. 全双工接收流控

1）接收流控不能自动触发。应用需对触发条件自行判断，并手动启动流控（发送暂停帧）。

2）暂停时间结束前，不能自动重发暂停帧。应用在暂停时间结束前手动发送暂停帧。

3）流控取消时，不能自动发送零时间暂停帧。应用在流控取消时手动发送零时间暂停帧。

7. 功能差异

1）流控

STM32F429 的流控可以自动触发，自动重发，自动停止。

ACM32H5 的流控只能手动触发，手动重发，手动停止。

2）过滤

ACM32H5 过滤功能比 STM32F429 多以下过滤功能：L3/L4 过滤、VLAN 过滤

3）VLAN。

ACM32H5 支持 S-LVAN（0x88A8），STM32F429 不支持 S-LVAN 。

4）LPI

ACM32H5 支持 LPI，STM32F429 不支持 LPI。

5）PTP

ACM32H5 支持 4 个辅助时间戳，对 TIM2/TIM3/TIM23/TIM24 的触发进行快照。STM32F429 不支持辅助时间戳。

ACM32H5 PPS 输出频率，占空比可随意设置， STM32F429 仅支持固定的 16 个频率翻转，占空比固定为 50%。

8. 性能比较

ACM32H5 对标 STM32F429，分别从以下几个方面对以太网收发速度进行性能测试。

1）程序存放于 SPI FLASH、SRAM1、SDRAM 中进行比较。

当程序存放于 SPI FLASH 时，SPI FLASH 接口速度分为 2 分频（XIP-2）、4 分频（XIP-4）。

2）数据存放于 SRAM1、SRAM2、SDRAM。

3）频率分为 180MHz、220MHz。

表 6-1 ACM32H5 和 STM32F429 以太网性能比对表

型号	主频	代码位置	数据位置	接收速度（Mbps）	发送速度（Mbps）
H5	180 MHz	XIP-4	SRAM1	48.15	69.55
			SRAM2	52.34	76.33
		XIP-2	SRAM1	60.84	78.14
			SRAM2	66.91	87.05
		SRAM1	SRAM2	84.33	98.81
		SDRAM	SRAM1	72.85	86.25
			SRAM2	78.7	97.94
			SDRAM	19.94	24.75
	220MHz	XIP-4	SRAM1	59.58	85.1
			SRAM2	65.01	93.41
		XIP-2	SRAM1	72.93	95.62
			SRAM2	75.01	98.81
		SRAM1	SRAM2	98.73	98.81
		SDRAM	SRAM1	84.98	98.49
			SRAM2	94.13	98.81
			SDRAM	24.17	30.45
F4	180 MHz	EFLASH	SRAM	90.63	98.57

9. 版本历史

版本	日期	作者	描述
V0.1	2024-11-22	Aisinochip	初始版
V1.0	2025-01-15	Aisinochip	审阅后更新

联系我们

公司：上海航芯电子科技股份有限公司

地址：上海市闵行区合川路 2570 号科技绿洲三期 2 号楼 702 室

邮编：200241

电话：+86-21-6125 9080

传真：+86-21-6125 9080-830

Email： service@aisinochip.com

Website： www.aisinochip.com