LPC1700 系列微控制器

第 11 章 USB 设备控制器

用户手册 Rev00.04

广州周立功单片机发展有限公司

地址：广州市天河北路 689 号光大银行大厦 12 楼 F4

网址：http://www.zlgmcu.com

销售与服务网络

广州周立功单片机发展有限公司

地址：广州市天河北路 689 号光大银行大厦 12 楼 F4 邮编：510630

电话：(020)38730972 38730976 38730916 38730917 38730977

传真：(020)38730925

网址：http://www.zlgmcu.com

广州专卖店

地址：广州市天河区新赛格电子城 203-204 室

电话：(020)87578634 87569917

传真：(020)87578842

南京周立功

地址：南京市珠江路 280 号珠江大厦 2006 室

电话：(025)83613221 83613271 83603500

传真：(025)83613271

北京周立功

地址：北京市海淀区知春路 113 号银网中心 A 座 1207-1208 室（中发电子市场斜对面）

电话：(010)62536178 62536179 82628073

传真：(010)82614433

重庆周立功

地址：重庆市石桥铺科园一路二号大西洋国际大厦（赛格电子市场）1611 室

电话：(023)68796438 68796439

传真：(023)68796439

杭州周立功

地址：杭州市天目山路 217 号江南电子大厦 502 室

电话：(0571)89719480 89719481 89719482
89719483 89719448 89719485

传真：(0571) 89719494

成都周立功

地址：成都市一环路南二段 1 号数码同人港 401 室（磨子桥立交西北角）

电话：(028) 85439836 85437446

传真：(028) 85437896

深圳周立功

地址：深圳市深南中路 2070 号电子科技大厦 C 座 4 楼 D 室

电话：(0755)83781788（5 线）

传真：(0755)83793285

武汉周立功

地址：武汉市洪山区广埠屯珞瑜路 158 号 12128 室（华中电脑数码市场）

电话：(027)87168497 87168297 87168397

传真：(027)87163755

上海周立功

地址：上海市北京东路 668 号科技京城东座 7E 室

电话：(021)53083452 53083453 53083496

传真：(021)53083491

西安办事处

地址：西安市长安北路 54 号太平洋大厦 1201 室

电话：(029)87881296 83063000 87881295

传真：(029)87880865

目录

第11章 USB设备控制器

11.1 如何阅读本章

本章描述了所有 LPC1700 系列 Cortex-M3 微控制器器件上的 USB 设备控制器。

11.2 基本配置

使用下列寄存器来配置 USB 控制器：

a）功率：在 PCONP 寄存器中置位 PCUSB。

注：复位时，USB 模块被禁止（PCUSB=0）。

b）时钟：USB 模块与特定的 PLL（或主 PLL）一起使用来获得 USB 时钟（见“PLL1 寄存器描述”）。

c）管脚：使用 PINSEL0～PINSEL5 和 PINMODE0～PINMODE5 来选择 USB 管脚及其模式，见“引脚连接模块”章节的“寄存器描述”小节。

d）唤醒：USB 总线端口上的活动可将微控制器从掉电模式中唤醒，见“计时和功率控制”章节的“从低功耗模式中唤醒”小节。

e）中断：使用相关的中断设置使能寄存器在 NVIC 中将中断使能。

f）初始化：见本章的“USB 设备控制器初始化”描述。

11.3 简介

通用串行总线（USB）为 4 线总线，支持一个主机与一个或多个外设（高达 127 个）之间的通信。主控制器通过一个基于令牌的协议为连接的设备分配 USB 带宽。USB 总线支持设备的热插拔与动态配置。主控制器启动所有的事务处理。

主机将事务安排在 1ms 的帧中。每帧都包含一个帧开始（SOF）标记和与设备端点进行往返数据传输的事务。每一个设备最多可以具有 16 个逻辑端点或 32 个物理端点。针对端点定义了 4 种传输类型。控制传输可用来对设备进行配置。中断传输则用于周期数据传输。批量传输在对传输速率没有严格要求时使用。同步传输保证了传输时间，但没有纠错功能。

通用串行总线的详细信息请参考 USB 规范制订者论坛（USB Implementers Forum）的网站。

LPC1700 系列 Cortex-M3 微控制器的 USB 设备控制器使能与 USB 主控制器之间的全速（12Mb/s）数据交换。

表 11.1 本章用到的与 USB 相关的首字母缩写词、简写以及定义

缩写词	定义
AHB	先进的高性能总线
ATLE	自动传输长度提取
ATX	模拟收发器
DD	DMA 描述符
DDP	DMA 描述指针
DMA	直接存储器访问
EOP	信息包结束
EP	端点

续上表

缩写词	定义
EP_RAM	端点 RAM
FS	全速
LED	发光二极管
LS	低速
MPS	最大信息包容量
NAK	否定应答
PLL	锁相环
RAM	随机访问存储器
SOF	帧开始
SIE	串行接口引擎
SRAM	同步 RAM
UDCA	USB 设备通信区域
USB	通用串行总线

11.4 特性

- 完全兼容 USB 2.0 全速规范；
- 支持 32 个物理（16 个逻辑）端点；
- 支持控制、批量、中断和同步端点；
- 运行时，可调整使用的端点；
- 运行时，可通过软件选择端点最大包长度（可达到 USB 规范规定的最大长度）；
- 支持 SoftConnect 和 GoodLink 特性；
- 所有非控制端点支持 DMA 传输；
- 允许 CPU 控制和 DMA 模式之间的动态切换；
- 实现了批量端点和同步端点上的双缓冲。

11.5 固定的端点配置

表 11.2列出了所支持的端点配置。端点在运行时利用端点使用寄存器来使用端点并进行配置，见“端点使用寄存器”的描述。

表 11.2 固定的端点配置

逻辑端点	物理端点	端点类型	方向	数据包长度（字节）	双缓冲
0	0	控制	Out	8,16,32,64	无
0	1	控制	In	8,16,32,64	无
1	2	中断	Out	1~64	无
1	3	中断	In	1~64	无
2	4	批量	Out	8,16,32,64	有
2	5	批量	In	8,16,32,64	有
3	6	同步	Out	1~1023	有
3	7	同步	In	1~1023	有
4	8	中断	Out	1~64	无
4	9	中断	In	1~64	无
5	10	批量	Out	8,16,32,64	有

续上表

逻辑端点	物理端点	端点类型	方向	数据包长度（字节）	双缓冲
5	11	批量	In	8,16,32,64	有
6	12	同步	Out	1~1023	有
6	13	同步	In	1~1023	有
7	14	中断	Out	1~64	无
7	15	中断	In	1~64	无
8	16	批量	Out	8,16,32,64	有
8	17	批量	In	8,16,32,64	有
9	18	同步	Out	1~1023	有
9	19	同步	In	1~1023	有
10	20	中断	Out	1~64	无
10	21	中断	In	1~64	无
11	22	批量	Out	8,16,32,64	有
11	23	批量	In	8,16,32,64	有
12	24	同步	Out	1~1023	有
12	25	同步	In	1~1023	有
13	26	中断	Out	1~64	无
13	27	中断	In	1~64	无
14	28	批量	Out	8,16,32,64	有
14	29	批量	In	8,16,32,64	有
15	30	批量	Out	8,16,32,64	有
15	31	批量	In	8,16,32,64	有

11.6 功能描述

USB设备控制器的结构如图 11.1所示。

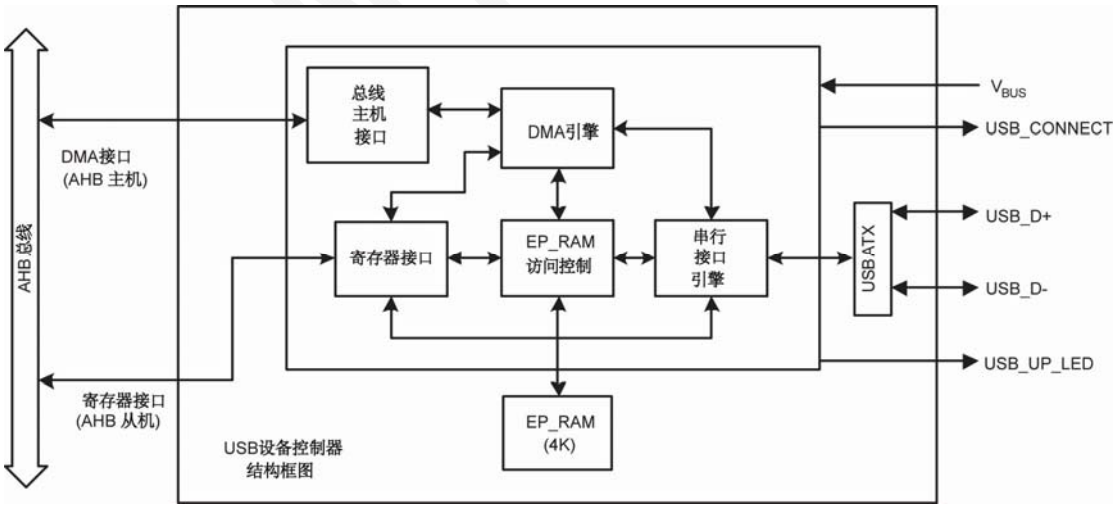


图 11.1 USB 设备控制器方框图

11.6.1 模拟收发器

USB 设备控制器有一个内置的模拟收发器（ATX）。USB ATX 发送/接收 USB 总线的双向 D+和 D-信号。

11.6.2 串行接口引擎（SIE）

SIE 实现全速 USB 协议层。从速度角度来看，它完全是线缆硬连接的（hardwired），不需要任何固件干涉。SIE 对 EP_RAM 中的端点缓冲区与 USB 总线之间的数据传输进行处理。该模块的功能包括：同步模式识别，并行/串行转换，位填充/解除填充、CRC 校验/产生、PID 验证/生成、地址识别和握手信号的评估/生成。

11.6.3 端点RAM（EP_RAM）

每个端点缓冲区都以基于 FIFO 的 SRAM 形式实现。专用于此用途的 SRAM 称作 EP_RAM。每个已实现的端点在 EP_RAM 中都有一个保留空间。所需的总的 EP_RAM 空间由使用的端点数、端点的最大包长度以及端点是否支持双缓冲来决定。

11.6.4 EP_RAM访问控制

EP_RAM 访问控制逻辑对 EP_RAM 和能够访问 EP_RAM 的 3 个源之间的数据传输进行处理。这 3 个源指的是 CPU（通过寄存器接口进行访问）、SIE 和 DMA 引擎。

11.6.5 DMA引擎和总线主机接口

当某个端点的 DMA 引擎使能时，它在 AHB 总线上的 RAM 与 EP_RAM 中的端点缓冲区之间传输数据。所有端点共用一个 DMA 通道。在传输数据时，DMA 引擎通过总线主机接口作为 AHB 总线上的主机运行。

11.6.6 寄存器接口

寄存器接口允许 CPU 来控制 USB 设备控制器的操作。而且在将发送数据写入控制器以及从控制器中读取接收数据时也会用到寄存器接口。

11.6.7 SoftConnect

USB 连接可通过一个 1.5kΩ 上拉电阻将 D+（对于全速设备）拉为高电平来实现。在确立与 USB 连接之前，软件可以使用 SoftConnect 特性来完成其初始化序列。该特性还可以在无需拔下电缆的情况下执行 USB 总线连接的重新初始化。

在使用 SoftConnect 特性时，CONNECT 信号应控制一个外部开关，这个开关与 D+和+3.3V 之间的 1.5kΩ 电阻相连。然后，通过使用 SIE 设置设备状态命令来对 CON 位执行写操作，从而实现软件对 CONNECT 信号的控制。

11.6.8 GoodLink

GoodLink 技术可用于指示 USB 连接是否良好。在成功地对设备进行清点和配置之后，LED 指示器将永久接通。在挂起期间，LED 是关闭的。

该特性对 USB 设备的状态提供一个用户友好的指示。它是一个非常有用的区域诊断工具，可以将故障设备分离出来。

在使用 GoodLink 特性时，UP_LED 信号用于控制 LED。UP_LED 信号使用 SIE 配置设备命令来控制。

11.7 操作概述

USB 总线事务在设备端点和主机之间传输数据。事务的方向由主机一方定义。OUT 事务指的是将数据从主机传送到设备。IN 事务指的是将数据从设备传送主机。所有的事务都由主控制器启动。

对于 OUT 事务，USB ATX 接收 USB 总线的双向 D+和 D-信号。串行接口引擎（SIE）从 ATX 中接收串行数据并将它转换为并行数据流。并行数据会被写入 EP_RAM 的对应的端点缓冲区中。

对于 IN 事务，SIE 从 EP_RAM 的端点缓冲区中读取并行数据，将它转换为串行数据，并使用 USB ATX 将它传输到 USB 总线上。

一旦数据接收或发送完成，我们就可以对端点缓冲区进行读和写操作。这一点该如何实现由端点的类型和工作模式决定。每个端点有两种工作模式：从（CPU 控制的）模式和 DMA 模式。

在从模式中，CPU 使用寄存器接口在 RAM 和端点缓冲区之间传输数据。从模式的详细描述请参考“从模式操作”章节。

在 DMA 模式中，DMA 在 RAM 和端点缓冲区之间传输数据。DMA 模式的详细描述请参考“DMA 操作”章节。

11.8 引脚描述

表 11.3 USB 外部接口

名称	方向	描述
V_{BUS}	I	V_{BUS} 状态输入。在没有通过对应的 PINSEL 寄存器将功能使能时，它在内部驱动为高电平
USB_CONNECT	O	SoftConnect 控制信号
USB_UP_LED	O	GoodLink LED 控制信号
USB_D+	I/O	正向差分数据
USB_D-	I/O	反向差分数据

11.9 时钟和功率管理

本小节描述了 USB 设备控制器的时钟和功率管理特性。

11.9.1 功率要求

USB 协议坚决要求对 USB 设备进行功率管理。如果 USB 设备从总线中汲取功率（总线供电设备），则功率管理将变得非常重要。一个由总线供电的设备应满足下面的约束条件：

a）一个处于未配置状态的设备从总线中汲取的最大电流应为 100mA。

b）一个已配置好的设备能够汲取的电流最高只能达到配置描述符的 Max Power 字段中规定的值。这个最大值为 500mA。

c）处于挂起状态的设备能够汲取的最大电流为 500μA。

11.9.2 时钟

表 11.4所示为USB设备控制器的时钟。

表 11.4 USB 设备控制器时钟源

时钟源	描述
AHB 主机时钟	AHB 主机总线接口和 DMA 的时钟
AHB 从机时钟	AHB 从机接口的时钟
usbclk	来自 USB 特定 PLL 或主 PLL 的 48MHz 时钟，用于恢复来自 USB 总线的 12MHz 时钟

11.9.3 功率管理支持

为了节省功率，USB 设备控制器在没有使用时自动禁止 AHB 主时钟和 usbclk。

当 USB 设备控制器进入挂起状态时（总线在 3ms 内保持空闲），设备控制器的 usbclk 输入将自动禁能，有助于节省功率。但如果软件希望访问设备控制器的寄存器，则 usbclk 必须有效。若想在设备处于挂起状态时对设备控制器的寄存器进行访问，则必须使用 USBClkCtrl 寄存器和 USBClkSt 寄存器。

当软件希望访问设备控制器的寄存器时，首先，它应该通过置位 USBClkCtrl 寄存器中的 DEV_CLK_EN 以保证 usbclk 使能，然后查询 USBClkSt 寄存器中对应的 DEV_CLK_ON 位，直到该位置位。一旦 DEV_CLK_ON 置位，在软件将 DEV_CLK_EN 清零之前，usbclk 会一直保持使能。

在进行 DMA 传输时，设备控制器自动开启 AHB 主时钟。一旦 AHB 主时钟启动，它将在至少 2ms（2 帧）内保持有效，这样有助于确保 DMA 吞吐量不受到 AHB 主时钟关闭的影响。在上一次 DMA 访问之后 2ms，AHB 主时钟自动禁能以节省功率。如有需要，软件还可以使用 USBClkCtrl 寄存器强制该时钟保持使能。

需要注意的是只要 PCONP 的 PCUSB 位置位，AHB 从时钟将始终是使能的。如果没有使用设备控制器，则将 PCUSB 清零即可以将该控制器的所有时钟禁能。

使用 USB_NEED_CLK 信号可使芯片进入掉电模式或从掉电模式中唤醒变得非常容易。如果 USBClkSt 寄存器中的任何一个位有效，则 USB_NEED_CLK 有效。

当设备在 DEV_CLK_EN 和 AHB_CLK_EN 均为清零的情况下进入挂起状态之后，DEV_CLK_ON 和 AHB_CLK_ON 在对应的时钟关闭时将清零。当这两个位都为 0 时，USB_NEED_CLK 为低电平，表示可通过写 PCON 寄存器将芯片置于掉电模式。USB_NEED_CLK 的状态可以从 USBIntSt 寄存器中读取。

挂起状态下的任何总线活动都将使 USB_NEED_CLK 信号有效。当芯片处于掉电模式，USB 中断使能时，USB_NEED_CLK 有效将会使芯片从掉电模式中唤醒。

11.9.4 远程唤醒

USB 设备控制器支持软件启动的远程唤醒功能。远程唤醒会恢复 USB 总线上由设备启动的信号。远程唤醒通过将 SIE 设置设备状态寄存器中的 SUS 位清零来实现。在写寄存器之前，必须使用 USBClkCtrl 寄存器将设备控制器的所有时钟使能。

11.10 寄存器描述

表 11.5显示了CPU可直接访问的USB设备控制器寄存器。串行接口引擎（SIE）含有通过SIE命令寄存器进行间接访问的其它寄存器。详细信息请参考“串行接口引擎命令描述”。

表 11.5 USB 设备寄存器映射

名称	描述	访问	复位值[1]	地址
时钟控制寄存器				
USBClkCtrl	USB 时钟控制	R/W	0x0000 0000	0x5000 CFF4
USBClkSt	USB 时钟状态	RO	0x0000 0000	0x5000 CFF8
设备中断寄存器				
USBIntSt	USB 中断状态	R/W	0x8000 0000	0x400F C1C0
USBDevIntSt	USB 设备中断状态	RO	0x0000 0010	0x5000 C200
USBDevIntEn	USB 设备中断使能	R/W	0x0000 0000	0x5000 C204
USBDevIntClr	USB 设备中断清零	WO	0x0000 0000	0x5000 C208
USBDevIntSet	USB 设备中断设置	WO	0x0000 0000	0x5000 C20C
USBDevIntPri	USB 设备中断优先级	WO	0x00	0x5000 C22C
端点中断寄存器				
USBEpIntSt	USB 端点中断状态	RO	0x0000 0000	0x5000 C230
USBEpIntEn	USB 端点中断使能	R/W	0x0000 0000	0x5000 C234
USBEpIntClr	USB 端点中断清零	WO	0x0000 0000	0x5000 C238
USBEpIntSet	USB 端点中断设置	WO	0x0000 0000	0x5000 C23C
USBEpIntPri	USB 端点优先级	WO[2]	0x0000 0000	0x5000 C240
端点使用寄存器				
USBReEp	USB 使用端点	R/W	0x0000 0003	0x5000 C244
USBEpInd	USB 端点索引	WO[2]	0x0000 0000	0x5000 C248
USBMaxPSize	USB 最大包长度	R/W	0x0000 0008	0x5000 C24C
USB 传输寄存器				
USBRxData	USB 接收数据	RO	0x0000 0000	0x5000 C218
USBRxPLen	USB 接收包长度	RO	0x0000 0000	0x5000 C220
USBTxData	USB 发送数据	WO[2]	0x0000 0000	0x5000 C21C
USBTxPLen	USB 发送包长度	WO[2]	0x0000 0000	0x5000 C224
USBCtrl	USB 控制	R/W	0x0000 0000	0x5000 C228
SIE 命令寄存器				
USBCmdCode	USB 命令代码	WO[2]	0x0000 0000	0x5000 C210
USBCmdData	USB 命令数据	RO	0x0000 0000	0x5000 C214
DMA 寄存器				
USBDMARSt	USB DMA 请求状态	RO	0x0000 0000	0x5000 C250
USBDMARClr	USB DMA 请求清零	WO[2]	0x0000 0000	0x5000 C254
USBDMARSet	USB DMA 请求设置	WO[2]	0x0000 0000	0x5000 C258
USBUDCAH	USB UDCA Head	R/W	0x0000 0000	0x5000 C280
USBEpDMASt	USB 端点 DMA 状态	RO	0x0000 0000	0x5000 C284
USBEpDMAEn	USB 端点 DMA 使能	WO[2]	0x0000 0000	0x5000 C288
USBEpDMADis	USB 端点 DMA 禁能	WO[2]	0x0000 0000	0x5000 C28C
USBDMAIntSt	USB DMA 中断状态	RO	0x0000 0000	0x5000 C290
USBDMAIntEn	USB DMA 中断使能	R/W	0x0000 0000	0x5000 C294
USBEoTIntSt	USB 传输结束中断状态	RO	0x0000 0000	0x5000 C2A0

续上表

名称	描述	访问	复位值[1]	地址
USBEoTIntClr	USB 传输结束中断清除	WO[2]	0x0000 0000	0x5000 C2A4
USBEoTIntSet	USB 传输结束中断设置	WO[2]	0x0000 0000	0x5000 C2A8
USBNDDRIntSt	USB 新 DD 请求中断状态	RO	0x0000 0000	0x5000 C2AC
USBNDDRIntClr	USB 新 DD 请求中断清除	WO[2]	0x0000 0000	0x5000 C2B0
USBNDDRIntSet	USB 新 DD 请求中断设置	WO[2]	0x0000 0000	0x5000 C2B4
USBSysErrIntSt	USB 系统错误中断状态	RO	0x0000 0000	0x5000 C2B8
USBSysErrIntClr	USB 系统错误中断清除	WO[2]	0x0000 0000	0x5000 C2BC
USBSysErrIntSet	USB 系统错误中断设置	WO[2]	0x0000 0000	0x5000 C2C0

[1] 复位值只反映使用到的位中的数据。它不包括保留位的内容。

[2] 读取 WO 寄存器将返回一个无效值。

11.10.1 时钟控制寄存器

1. USB时钟控制寄存器（USBClkCtrl - 0x5000 CFF4）

该寄存器用于控制 USB 设备控制器的时钟。任何时候只要软件想访问设备控制器的寄存器，DEV_CLK_EN 和 AHB_CLK_EN 必须置位。只有在访问 USBPortSel 寄存器时才需要将 PORTSEL_CLK_EN 位置位。

假设 USBClkCtrl 的对应位已经置位，软件不需要在每次寄存器访问时重复执行置位操作。需要注意的是该寄存器只有在 PCONP 中的 PCUSB 位置位时才起作用；当 PCUSB 清零时，不管该寄存器的内容如何，设备控制器的所有时钟都是禁止的。USBClkCtrl 是一个读/写寄存器。

表 11.6 USBClkCtrl 寄存器位描述

位	符号	描述	复位值
0	-	保留，用户软件不应该向保留位写入 1。从保留位读取的值是未定义的	NA
1	DEV_CLK_EN	设备时钟使能。使能设备控制器的 usbclk 输入	0
2	-	保留，用户软件不应该向保留位写入 1。从保留位读取的值是未定义的	NA
3	PORTSEL_CLK_EN	端口选择寄存器的时钟使能	NA
4	AHB_CLK_EN	AHB 时钟使能	0
31:5	-	保留，用户软件不应向保留位写入 1。从保留位读出的值未被定义	NA

2. USB时钟状态寄存器（USBClkSt - 0x5000 CFF8）

该寄存器用于保持时钟可用的状态。将该寄存器的位相或可以形成 USB_NEED_CLK 信号。当通过 USBClkCtrl 将一个时钟使能时，软件应该查询 USBClkSt 中的对应位。如果该位是置位的，则软件能够进一步进行寄存器访问。假如 USBClkCtrl 寄存器中的位没受到打扰，则软件不需要在每次访问时重复查询操作。USBClkSt 为只读寄存器。

表 11.7 USB 时钟状态寄存器位描述

位	符号	描述	复位值
0	-	保留，用户软件不应该向保留位写入 1。从保留位读取的值是未定义的	NA
1	DEV_CLK_ON	设备时钟开启。设备控制器的 usbclk 输入有效	0
2	-	保留，用户软件不应该向保留位写入 1。从保留位读取的值是未定义的	NA
3	PORTSEL_CLK_ON	端口选择寄存器的时钟开启	NA
4	AHB_CLK_ON	AHB 时钟开启	0
31:5	-	保留，用户软件不应向保留位写入 1。从保留位读出的值未被定义	NA

11.10.2 设备中断寄存器

1. USB中断状态寄存器（USBIntSt - 0xE01F C1C0）

USB 设备控制器有 3 条中断线。USB 中断状态寄存器允许软件对其执行一次读操作来确定这 3 条中断线的状态。所有 3 条中断线相“或”，连接到向量中断控制器的一个通道。该寄存器还包含 USB_NEED_CLK 状态位和 EN_USB_INTS 控制位。USBIntSt 为读/写寄存器。

表 11.8 USB 中断状态寄存器位描述

位	符号	描述	复位值
0	USB_INT_REQ_LP	低优先级中断线的状态。该位为只读位	0
1	USB_INT_REQ_HP	高优先级中断线的状态。该位为只读位	0
2	USB_INT_REQ_DMA	DMA 中断线的状态。该位为只读位	0
7:3	-	保留，用户软件不应该向保留位写入 1。从保留位读取的值是未定义的	NA
8	USB_NEED_CLK	USB 需要时钟指示器。当检测到 USB 活动或检测到 USB 数据管脚上的状态发生改变时，该位被设置为 1，而且它表示需要一个 48MHz 的 PLL 支持时钟。一旦该位为 1，它就在接收到/发送最后包的 5ms 后或在挂起改变（SUS_CH）中断发生的 2ms 后复位为 0。如果选择 USB 总线上的活动从掉电模式中唤醒器件，那么该位从 0 到 1 的变化可唤醒微控制器。（详细信息请参考“时钟和功率控制”章节中的“从低功耗模式中唤醒”小节）。有关 PLL 和请求掉电模式的描述也可参考“时钟和功率控制”章节中的“PLL0 和掉电模式”和“外设功率控制寄存器（PCONP-0x400F C0C4）”。该位为只读位	1
30:9	-	保留，用户软件不应向保留位写入 1。从保留位读出的值未被定义	NA
31	EN_USB_INTS	使能所有 USB 中断。当该位清零时，向量中断控制器将无法检测到 USB 中断线相或后的输出	1

2. USB设备中断状态寄存器（USBDevIntSt - 0x5000 C200）

USBDevIntSt 寄存器记录着各个中断的状态。‘0’表示未产生中断，‘1’表示中断已经出现。USBDevIntSt 为只读寄存器。

表 11.9 USB 设备中断状态寄存器位分配

复位值：0x0000 0000

位	31	30	29	28	27	26	25	24
符号	-	-	-	-	-	-	-	-
位	23	22	21	20	19	18	17	16
符号	-	-	-	-	-	-	-	-
位	15	14	13	12	11	10	9	8
符号	-	-	-	-	-	-	ERR_INT	EP_RLZED
位	7	6	5	4	3	2	1	0
符号	TxENDPKT	RxENDPKT	CDFULL	CCEMPTY	DEV_STAT	EP_SLOW	EP_FAST	FRAME

表 11.10 USB 设备中断状态寄存器位描述

位	符号	描述	复位值
0	FRAME	每隔 1ms 产生一次帧中断。该位用在同步包的传输中	0
1	EP_FAST	端点的快速中断传输。如果端点中断在端点中断优先级寄存器（USBEpIntPri）中相应的位被置位，则该端点中断与 EP_FAST 位相关	0
2	EP_SLOW	端点的慢速中断。如果端点中断在端点中断优先级寄存器（USBEpIntPri）中相应的位未置位，则该端点中断与 EP_SLOW 位相关	0
3	DEV_STAT	该位在 USB 总线复位、USB 挂起改变或连接改变时置位。请参考“设置设备状态（命令：0xFE，数据：写 1 字节）”小节	0
4	CCEMPTY	命令代码寄存器（USBCmdCode）为空（可写入新的命令）	1
5	CDFULL	命令数据寄存器（USBCmdData）已满（现在可以读取数据）	0
6	RxENDPKT	在端点缓冲区中的当前数据包已传送给 CPU	0
7	TxENDPKT	传输到端点缓冲区的数据字节数与 TxPacket 长度寄存器（USBTxPLen）中编程设定的字节数相等	0
8	EP_RLZED	端点被使用。该位在使用端点寄存器（USBReEp）或 MaxPacketSize 寄存器（USBMaxPSize）更新且相应的操作完成时置位	0
9	ERR_INT	错误中断。USB 设备的任何总线错误中断。请参考“读错误状态（命令：0xFB，数据：读 1 字节）”	0
31:10	-	保留，用户软件不应该向保留位写入 1。从保留位读取的值未定义	NA

3. USB设备中断使能寄存器（USBDevIntEn - 0x5000 C204）

向该寄存器中的某个位写入 1 时，如果 USBDevIntSt 中的对应位置位，则可以在其中一条中断线上产生中断。默认是将中断发送到 USB_INT_REQ_LP 中断线。另外，通过改变 USBDevIntPri 的值，也可以将 EP_FAST 或 FRAME 中断发送到 USB_INT_REQ_HP 中断线。USBDevIntEn 为读/写寄存器。

表 11.11 USB 设备中断使能寄存器位分配

复位值：0x0000 0000

位	31	30	29	28	27	26	25	24
符号	-	-	-	-	-	-	-	-
位	23	22	21	20	19	18	17	16
符号	-	-	-	-	-	-	-	-
位	15	14	13	12	11	10	9	8
符号	-	-	-	-	-	-	ERR_INT	EP_RLZED
位	7	6	5	4	3	2	1	0
符号	TxENDPKT	RxENDPKT	CDFULL	CCEMPTY	DEV_STAT	EP_SLOW	EP_FAST	FRAME

表 11.12 USB 设备中断使能寄存器位描述

位	符号	值	描述	复位值
31:0	见上表 USBDevIntEn 的位分配	0	没有中断产生	0
		1	当设备中断状态（USBDevIntSt）寄存器（表 11.9）中的对应位置位时，中断产生。默认是将中断发送到 USB_INT_REQ_LP中断线。另外，通过改变USBDevIntPri 的值，也可以将 EP_FAST 或 FRAME 中断发送到 USB_INT_REQ_HP中断线	

4. USB设备中断清除寄存器（USBDevIntClr - 0x5000 C208）

向该寄存器的某个位写入 1 可将 USBDevIntSt 中的对应位清零；写入 0 无效。

注：在将 EP_SLOW 或 EP_FAST 中断位清零之前，USBEPIntSt 中对应的端点中断也应该清零。

USBDevIntClr 为只写寄存器。

表 11.13 USB 设备中断清除寄存器位分配

复位值：0x0000 0000

位	31	30	29	28	27	26	25	24
符号	-	-	-	-	-	-	-	-
位	23	22	21	20	19	18	17	16
符号	-	-	-	-	-	-	-	-
位	15	14	13	12	11	10	9	8
符号	-	-	-	-	-	-	ERR_INT	EP_RLZED
位	7	6	5	4	3	2	1	0
符号	TxENDPKT	RxENDPKT	CDFULL	CCEMPTY	DEV_STAT	EP_SLOW	EP_FAST	FRAME

表 11.14 USB 设备中断清零寄存器位描述

位	符号	值	描述	复位值
31:0	见上表 USBDevIntClr 的位分配	0	无效	0
		1	USBDevIntSt 寄存器（见“USB 设备中断状态寄存器”）中的相应位被清零	

5. USB设备中断设置寄存器（USBDevIntSet - 0x5000 C20C）

向该寄存器的某个位写入 1 可将 USBDevIntSt 中的对应位置位；写入 0 无效。

USBDevIntSet 为只写寄存器。

表 11.15 USB 设备中断设置寄存器位分配

复位值：0x0000 0000

位	31	30	29	28	27	26	25	24
符号	-	-	-	-	-	-	-	-
位	23	22	21	20	19	18	17	16
符号	-	-	-	-	-	-	-	-
位	15	14	13	12	11	10	9	8
符号	-	-	-	-	-	-	ERR_INT	EP_RLZED
位	7	6	5	4	3	2	1	0
符号	TxENDPKT	RxENDPKT	CDFULL	CCEMPTY	DEV_STAT	EP_SLOW	EP_FAST	FRAME

表 11.16 USB 设备中断设置寄存器位描述

位	符号	值	描述	复位值
31:0	见上表 USBDevIntSet 的位分配	0	无效	0
		1	USBDevIntSt 寄存器（见“USB 设备中断状态寄存器”）中的相应位被置位	

6. USB设备中断优先级寄存器（USBDevIntPri – 0x5000 C22C）

向该寄存器的某个位写入 1 将使对应的中断发送到 USB_INT_REQ_HP 中断线；写入 0 时将中断发送到 USB_INT_REQ_LP 中断线。EP_FAST 或 FRAME 中断中的任何一个都可以发送到 USB_INT_REQ_HP，但不能同时进行。如果软件试图将这两个位置位，则不向 USB_INT_REQ_HP 发送任何中断。USBDevIntPri 为只写寄存器。

表 11.17 USB 设备中断优先级寄存器位描述

位	符号	值	描述	复位值
0	FRAME	0	FRAME 中断将进入 USB_INT_REQ_LP	0
		1	FRAME 中断将进入 USB_INT_REQ_HP	
1	EP_FAST	0	EP_FAST 中断将进入 USB_INT_REQ_LP	0
		1	EP_FAST 中断将进入 USB_INT_REQ_HP	
7:2	-	-	保留，用户软件不应向保留位写入 1。从保留位读出的值未被定义	NA

11.10.3 端点中断寄存器

该组寄存器可以简化端点中断的处理。端点中断在从模式操作中使用。

1. USB端点中断状态寄存器（USBEpIntSt - 0x5000 C230）

每个物理非同步端点在 USB 端点中断状态寄存器都对应一个位，用来指示端点产生的中断。当准确接收到一个数据包时，所有异步 OUT 端点产生一个中断。当成功发送一个数据包或 NAK 握手信号发送到总线上（如果 NAK 中断已使能）时，所有异步 IN 端点也产生一个中断（参考“设置模式”（命令：0xF3，数据：写 1 字节）。该寄存器中的某个位为 1 时会导致 USBDevIntSt 寄存器中的 EP_FAST 或 EP_SLOW 位置位，具体是哪个位置位将由 USBEpDevIntPri 寄存器的

对应位的值决定。USBEpIntSt 为只读寄存器。需要注意的是对于同步端点，在出现 FRAME 中断时对数据包进行处理。

表 11.18　USB 端点中断状态寄存器位分配

复位值：0x0000 0000

位	31	30	29	28	27	26	25	24
符号	EP15TX	EP15RX	EP14TX	EP14RX	EP13TX	EP13RX	EP12TX	EP12RX
位	23	22	21	20	19	18	17	16
符号	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
位	15	14	13	12	11	10	9	8
符号	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX
位	7	6	5	4	3	2	1	0
符号	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

表 11.19　USB 端点中断状态寄存器位描述

位	符号	描述	复位值
0	EP0RX	端点 0，接收完数据中断位	0
1	EP0TX	端点 0，发送完数据中断位或发送一个 NAK	0
2	EP1RX	端点 1，接收完数据中断位	0
3	EP1TX	端点 1，发送完数据中断位或发送一个 NAK	0
4	EP2RX	端点 2，接收完数据中断位	0
5	EP2TX	端点 2，发送完数据中断位或发送一个 NAK	0
6	EP3RX	端点 3，同步端点	NA
7	EP3TX	端点 3，同步端点	NA
8	EP4RX	端点 4，接收完数据中断位	0
9	EP4TX	端点 4，发送完数据中断位或发送一个 NAK	0
10	EP5RX	端点 5，接收完数据中断位	0
11	EP5TX	端点 5，发送完数据中断位或发送一个 NAK	0
12	EP6RX	端点 6，同步端点	NA
13	EP6TX	端点 6，同步端点	NA
14	EP7RX	端点 7，接收完数据中断位	0
15	EP7TX	端点 7，发送完数据中断位或发送一个 NAK	0
16	EP8RX	端点 8，接收完数据中断位	0
17	EP8TX	端点 8，发送完数据中断位或发送一个 NAK	0
18	EP9RX	端点 9，同步端点	NA
19	EP9TX	端点 9，同步端点	NA
20	EP10RX	端点 10，接收完数据中断位	0
21	EP10TX	端点 10，发送完数据中断位或发送一个 NAK	0
22	EP11RX	端点 11，接收完数据中断位	0
23	EP11TX	端点 11，发送完数据中断位或发送一个 NAK	0
24	EP12RX	端点 12，同步端点	NA
25	EP12TX	端点 12，同步端点	NA

续上表

位	符号	描述	复位值
26	EP13RX	端点 13，接收完数据中断位	0
27	EP13TX	端点 13，发送完数据中断位或发送一个 NAK	0
28	EP14RX	端点 14，接收完数据中断位	0
29	EP14TX	端点 14，发送完数据中断位或发送一个 NAK	0
30	EP15RX	端点 15，接收完数据中断位	0
31	EP15TX	端点 15，发送完数据中断位或发送一个 NAK	0

2. USB端点中断使能寄存器（USBEpIntEn - 0x5000 C234）

该寄存器通过置位某个位来使 USBEpIntSt 中的对应位在出现与该端点相关的中断时置位；写入 0 将使 USBDMARSt 中的对应位在出现与该端点相关的中断时置位。USBEpIntEn 为读/写寄存器。

表 11.20 USB 端点中断使能寄存器位分配

复位值：0x0000 0000

位	31	30	29	28	27	26	25	24
符号	EP15TX	EP15RX	EP14TX	EP14RX	EP13TX	EP13RX	EP12TX	EP12RX
位	23	22	21	20	19	18	17	16
符号	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
位	15	14	13	12	11	10	9	8
符号	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX
位	7	6	5	4	3	2	1	0
符号	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

表 11.21 USB 端点中断使能寄存器位描述

位	符号	值	描述	复位值
31:0	见上表 USBEpIntEn 的位分配	0	USBDMARSt 中的对应位在出现该端点的中断时置位	0
		1	USBEpIntSt 中的对应位在出现该端点的中断时置位。暗示该端点处于从模式	

3. USB端点中断清除寄存器（USBEpIntClr - 0x5000 C238）

向该寄存器的某个位写入‘1’时，将针对对应的物理端点执行SIE选择端点/清除中断命令（表 11.65）；写入 0 无效。在执行选择端点/清除中断命令之前，硬件会把USBDevIntSt寄存器中的CDFULL位清零。在执行完命令时，CDFULL位置位，USBCmdData包含端点的状态，并且USBEpIntSt中的对应位清零。

注意事项：

- 在使用 USBEpIntClr 寄存器将中断清除时，软件必须等待 CDFULL 位置位，从而确保对应的中断在能够继续发生之前已清零；
- 虽然可以同时将 USBEpIntClr 寄存器中的多个位置位，但建议不要这样做；当操作结束时，在所有清零的多个位中，只有最低有效位对应的端点状态是可用的；
- 另外，SIE 选择端点/清除中断命令可以使用 SIE 命令寄存器直接调用，但建议使用 USBEpIntClr 寄存器，因为它的使用更加方便。

在该寄存器中，每个物理端点都有自己的保留位。该寄存器的位字段定义与表 11.18中显示的USBEpIntSt的位定义是相同的。USBEpIntClr为只写寄存器。

表 11.22 USB 端点中断清除寄存器位分配

复位值：0x0000 0000

位	31	30	29	28	27	26	25	24
符号	EP15TX	EP15RX	EP14TX	EP14RX	EP13TX	EP13RX	EP12TX	EP12RX
位	23	22	21	20	19	18	17	16
符号	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
位	15	14	13	12	11	10	9	8
符号	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX
位	7	6	5	4	3	2	1	0
符号	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

表 11.23 USB 端点中断清除寄存器位描述

位	符号	值	描述	复位值
31:0	见上表 USBEpIntClr 的位分配	0	无效	0
		1	通过执行对应端点的 SIE 选择端点/清除中断命令，可令 USBEpIntSt 中的对应位清零	

4. USB端点中断设置寄存器（USBEpIntSet - 0x5000 C23C）

向该寄存器中的某位写入‘1’会置位 USBEpIntSt 寄存器中的对应位；写入‘0’无影响。每个端点在该寄存器中都有相应的位。USBEpIntSet 为只写寄存器。

表 11.24 USB 端点中断设置寄存器位分配

复位值：0x0000 0000

位	31	30	29	28	27	26	25	24
符号	EP15TX	EP15RX	EP14TX	EP14RX	EP13TX	EP13RX	EP12TX	EP12RX
位	23	22	21	20	19	18	17	16
符号	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
位	15	14	13	12	11	10	9	8
符号	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX
位	7	6	5	4	3	2	1	0
符号	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

表 11.25 USB 端点中断设置寄存器位描述

位	符号	值	描述	复位值
31:0	见上表 USBEpIntSet 的位分配	0	无效	0
		1	将 USBEpIntSt 中的对应位置位	

5. USB端点中断优先级寄存器（USBEpIntPri - 0x5000 C240）

该寄存器决定一个端点中断是进入 USBDevIntSt 寄存器的 EP_FAST 还是 EP_SLOW 位。如果将该寄存器中的某位置位，相应的端点中断进入 EP_FAST；反之，则进入 EP_SLOW。也可以有多个端点的中断进入 EP_FAST 或 EP_SLOW。

需要注意的是 USBDevIntPri 寄存器确定是让 EP_FAST 中断进入 USB_INT_REQ_HP 还是 USB_INT_REQ_LP 中断线。

USBEpIntPri 为只写寄存器。

表 11.26　USB 端点中断优先级寄存器位分配

复位值：0x0000 0000

位	31	30	29	28	27	26	25	24
符号	EP15TX	EP15RX	EP14TX	EP14RX	EP13TX	EP13RX	EP12TX	EP12RX
位	23	22	21	20	19	18	17	16
符号	EP11TX	EP11RX	EP10TX	EP10RX	EP9TX	EP9RX	EP8TX	EP8RX
位	15	14	13	12	11	10	9	8
符号	EP7TX	EP7RX	EP6TX	EP6RX	EP5TX	EP5RX	EP4TX	EP4RX
位	7	6	5	4	3	2	1	0
符号	EP3TX	EP3RX	EP2TX	EP2RX	EP1TX	EP1RX	EP0TX	EP0RX

表 11.27　USB 端点中断优先级寄存器位描述

位	符号	值	描述	复位值
31:0	见上表 USBEpIntPri 的位分配	0	对应中断进入 USBDevIntSt 寄存器的 EP_SLOW 位	0
		1	对应中断进入 USBDevIntSt 寄存器的 EP_FAST 位	

11.10.4 端点使用寄存器

端点使用寄存器组主要用于在运行时使用端点以及对端点进行配置。

1. EP RAM要求

USB 设备控制器使用一个基于 FIFO 的 RAM 作为端点缓冲区。专用于此用途的 RAM 称作端点 RAM（EP_RAM）。每个端点在 EP_RAM 中都有保留空间。每个端点所需的 EP_RAM 空间由 MaxPacketSize 的值以及是否具有双缓冲来决定。USB 设备使用 32 个字的 EP_RAM 来存放端点缓冲区的指针。EP_RAM 为字对齐，但 MaxPacketSize 是按字节定义的，因此，必须将 RAM 深度调整到下一个字边界。并且，每个缓冲区都有一个字的头信息（header），用于显示接收到的包的长度。

物理端点所需的 EP_RAM 空间（以字为单位）如下所示：

$$\mathrm{EPRAMspace} = \left(\frac{(\mathrm{MaxPacketSize} + 3)}{4} + 1\right) \times \mathrm{dbstatus}$$

这里，单缓冲端点的 dbstatus 为 1，双缓冲端点的 dbstatus 为 2。

所有已使用的端点都占据 EP_RAM 空间，因此，总的 EP_RAM 要求为：

$$\mathrm{TotalEPRAMspace} = 32 + \sum_{n=0}^{N} \mathrm{EPRAMspace}(n)$$

这里的 N 为已使用的端点的数目。总的 EP_RAM 空间不应该超出 4096 个字节（4kB，1k 字）。

2. USB使用端点寄存器（USBReEp - 0x5000 C244）

向该寄存器的某个位写入 1 可使用对应的端点；写入 0 时不能使用对应的端点。当总线复位时，该寄存器回到其复位状态。USBReEp 为读/写寄存器。

表 11.28 USB 使用端点寄存器位分配

复位值：0x0000 0003

位	31	30	29	28	27	26	25	24
符号	EP31	EP30	EP29	EP28	EP27	EP26	EP25	EP24
位	23	22	21	20	19	18	17	16
符号	EP23	EP22	EP21	EP20	EP19	EP18	EP17	EP16
位	15	14	13	12	11	10	9	8
符号	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8
位	7	6	5	4	3	2	1	0
符号	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0

表 11.29 USB 使用端点寄存器位描述

位	符号	值	描述	复位值
0	EP0	0	不使用控制端点 EP0	1
		1	使用控制端点 EP0	
1	EP1	0	不使用控制端点 EP1	1
		1	使用控制端点 EP1	
31:2	EPxx	0	不使用端点 EPxx	0
		1	使用端点 EPxx	

复位时只使用了控制端点。其它端点如有必要可通过对 USBReEp 寄存器中的相应位进行设置来使用。如果想计算使用端点所需的 EP_RAM 空间，请参考“EP RAM 要求”小节。

端点使用是多周期操作。其伪代码如下所示：

```
将 USBDevIntSt 中的 EP_RLZED 位清零;
对于即将被使用的每个端点,
{
    /* 将使用端点寄存器中已有的值进行“或”操作 */
    USBReEp |= (UInt32) ((0x1 << endpt));
    /*将物理端点编号加载到端点索引寄存器中.*/
    USBEpIn = (UInt32) endpointnumber;
    /* 加载最大包长度寄存器 */
    USBEpMaxPSize = MPS;
    /* 检验设备中断状态寄存器中的 EP_RLZED 位是否置位 */
    while (!(USBDevIntSt & EP_RLZED))
    {
        /*等待，直到端点使用完成 */
    }
    /* 清除 EP_RLZED 位 */
    将 USBDevIntSt 中的 EP_RLZED 位清零;
}
```

USB设备不会响应任何未使用端点的事务。SIE配置设备命令只会令使用的且使能的端点响应事务。详细信息请参考表 11.60。

3. USB端点索引寄存器（USBEpIn - 0x5000 C248）

每个端点都含有一个寄存器，用来记录端点的最大包长度值。这个寄存器实际上是一个寄存器数组。因此，在执行写操作之前，应通过 USBEpIn 寄存器来‘寻址’该寄存器。

USBEpIn 寄存器保存着物理端点的编号。写 USBMaxPSize 寄存器就是通过端点索引寄存器来设置寄存器阵列组成元素的值。USBEpIn 为只写寄存器。

表 11.30 USB 端点索引寄存器位描述

位	符号	描述	复位值
4:0	PHY_EP	物理端点的编号（0～31）	0
31:5	-	保留，用户软件不应该向保留位写入 1。从保留位读取的值是未定义的	NA

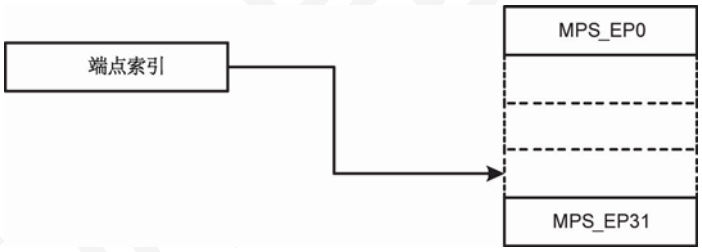
4. USB MaxPacketSize寄存器（USBMaxPSize - 0x5000 C24C）

复位时，控制端点的最大包长度指定为 8 字节；其它端点指定为 0。若修改USBMaxPSize的值，则需要重新计算EP_RAM中的端点缓冲区地址。这是一个多周期处理操作。在结束时，USBDevIntSt（表 11.9）中的EP_RLZED位将置位。USMaxPSize数组索引如图 11.2所示。USBMaxPSize为读/写寄存器。

表 11.31 USB MaxPacketSize 寄存器位描述

位	符号	描述	复位值
9:0	MPS	最大包长度值	0x008[1]
31:10	-	保留，用户软件不应该向保留位写入 1。从保留位读取的值是未定义的	NA

[1] 该值为 EP0 和 EP1 的复位值。其它所有端点的复位值为 0x0。



端点索引通过 USBEpIn 寄存器进行设置。MPS_EP0~MPS_EP31 通过 USBMaxPSize 寄存器进行访问。

图 11.2 USB MaxPacketSize 寄存器数组索引

11.10.5 USB传输寄存器

在从模式中操作时，该组寄存器用于在端点缓冲区和 RAM 之间传输数据。详见“从模式操作”小节。

1. USB接收数据寄存器（USBRxData - 0x5000 C218）

在 OUT 数据传输中，CPU 从该寄存器中读出端点缓冲区的数据。在读取该寄存器之前，应该对 USBCtrl 寄存器中的 RD_EN 位和 LOG_ENDPOINT 字段进行适当的设置。该寄存器的读操作将提取来自所选的端点缓冲区的数据。数据采用小端格式：从 USB 总线上接收到的第一个字节将是 USBRxData 寄存器的最低有效字节。USBRxData 为只读寄存器。

表 11.32 USB 接收数据寄存器位描述

位	符号	描述	复位值
31:0	RX_DATA	接收数据	0x0000 0000

2. USB接收包长度寄存器（USBRxPLen - 0x5000 C220）

该寄存器给出了正在传输的当前数据包剩余在端点缓冲区（当前通过 USBRxData 寄存器读取的数据包所在的端点缓冲区）的字节数，并判定数据包是否有效。在读取该寄存器之前，应该对 USBCtrl 寄存器中的 RD_EN 位和 LOG_ENDPOINT 字段进行适当的设置。每一次读取 USBRxData 寄存器时都会对该寄存器进行更新。USBRxPLen 为只读寄存器。

表 11.33 USB 接收包长度寄存器位描述

位	符号	值	描述	复位值
9:0	PKT_LNGTH	-	可以从当前所选的缓冲区中读出的剩余字节数。当该字段的值减少到 0 时，在 USBDevIntSt 寄存器中的 RxENDPKT 位将置位	0
10	DV	-	数据有效。该位对于同步端点非常有用。非同步端点在接收到错误的数据包时不会产生中断。但是无效数据包可能和总线复位一同产生。对于同步端点，即使接收到错误的数据包，数据传输仍可继续进行。这种情况下数据包的 DV 位不能置位	0
		0	数据无效	
		1	数据有效	
11	PKT_RDY	-	PKT_LNGTH 字段有效，并且数据包准备就绪可以读出	0
31:12	-	-	保留，用户软件不应该向保留位写入 1。从保留位读取的值是未定义的	NA

3. USB发送数据寄存器（USBTxData - 0x5000 C21C）

在 IN 数据传输中，CPU 将端点数据写入该寄存器。在对该寄存器执行写操作之前，应该对 USBCtrl 寄存器中的 WR_EN 位和 LOG_ENDPOINT 字段进行适当的设置，并且应将数据包长度写入 USBTxPlen 寄存器。在对该寄存器执行写操作时，数据被写入所选的端点缓冲区。数据采用小端格式：USB 总线上发送的第一个字节为 USBTxData 的最低有效字节。USBTxData 为只写寄存器。

表 11.34 USB 发送数据寄存器位描述

位	符号	描述	复位值
31:0	TX_DATA	发送数据	0x0000 0000

4. USB发送包长度寄存器（USBTxPLen - 0x5000 C224）

该寄存器包含从 CPU 传输到所选的端点缓冲区的数据的字节数。在将数据写入 USBTxData 之前，软件应先将数据包长度（≤MaxPacketSize）写入该寄存器。在每次对 USBTxData 进行写操作之后，硬件将 USBTxPLen 寄存器中的值减去 4。在开始处理之前，应该将 USBCtrl 寄存器中的 WR_EN 位和 LOG_ENDPOINT 字段设置以选择所需的端点缓冲区。

当数据缓冲区的长度大于端点的 MaxPacketSize 时，软件应提交数据包长度为 MaxPacketSize 的数据，并在最后一个包中发送剩余附加的字节。例如，如果 MaxPacketSize 为 64 字节，即将发送的数据缓冲区长度为 130 字节，则软件将发送两个 64 位数据包并在最后一个包中发送剩余的两个字节。因此，总共在 USB 上发送了 3 个数据包。USBTxPLen 为只写寄存器。

表 11.35 USB 发送包长度寄存器位描述

位	符号	值	描述	复位值
9:0	PKT_LNGTH	-	可以写入所选的端点缓冲区的剩余字节数。每次写 USBTxData 寄存器，该域的值由硬件减 4。当该字域减到 0 时，在 USBDevIntSt 寄存器中的 TxENDPKT 位被置位	0x000
31:10	-	-	保留，用户软件不应向保留位写入 1。从保留位读出的值未被定义	NA

5. USB控制寄存器（USBCtrl - 0x5000 C228）

该寄存器控制 USB 设备的数据传输操作。它选择在执行 USBRxData 和 USBTxData 寄存器访问时的端点缓冲区，并使能缓冲区的读和写操作。USBCtrl 为读/写寄存器。

表 11.36 USB 控制寄存器位描述

位	符号	值	描述	复位值
0	RD_EN	0 1	读模式控制。使能使用 USBRxData 寄存器从 OUT 端点缓冲区中读取数据，端点由本寄存器的 LOG_ENDPOINT 域指定。在从 USBRxData 寄存器中读取当前数据包的最后一个字时，硬件将该位清零 读模式禁能 读模式使能	0
1	WR_EN	0 1	写模式控制。使能使用 USBTxData 寄存器将数据写入 IN 端点缓冲区。端点由本寄存器的 LOG_ENDPOINT 指定。当 USBTxLen 中的字节数已发送完时，硬件将该位清零 写模式禁能 写模式使能	0
5:2	LOG_ENDPOINT	-	逻辑端点编号	0x0
31:6	-	-	保留，用户软件不应向保留位写入 1。从保留位读出的值未被定义	NA

11.10.6 SIE命令代码寄存器

SIE 命令代码寄存器用于与串行接口引擎进行通信。详细信息请参考“串行接口引擎命令描述”小节。

1. USB命令代码寄存器（USBCmdCode - 0x5000 C210）

该寄存器用于将命令和写数据发送到 SIE。此处写入的命令将传输到 SIE 并在 SIE 中执行。在执行了命令之后，该寄存器变成空，USBDevIntSt 寄存器中的 CCEMPTY 位置位。详细信息请参考“串行接口引擎命令寄存器”小节。USBCmdCode 为只写寄存器。

表 11.37 USB 命令代码寄存器位描述

位	符号	值	描述	复位值
7:0	-	-	保留，用户软件不应向保留位写入 1。从保留位读出的值未被定义	NA
15:8	CMD_PHASE	0x01 0x02 0x05	命令阶段： 读 写 命令	0x00
23:16	CMD_CODE/ CMD_WDATA		这是一个多用途域。当 CMD_PHASE 为命令或读状态时，该域包含着命令代码（CMD_CODE）。当 CMD_PHASE 为写状态时，该域包含着命令写数据（CMD_WDATA）	0x00
31:24	-	-	保留，用户软件不应向保留位写入 1。从保留位读出的值未被定义	NA

2. USB命令数据寄存器（USBCmdData - 0x5000 C214）

该寄存器包含执行SIE命令之后获得的数据。当准备读取该数据时，USBDevIntSt寄存器中的CD_FULL位是置位的，详细信息见表 11.9。USBCmdData为只读寄存器。

表 11.38 USB 命令数据寄存器位描述

位	符号	描述	复位值
7:0	CMD_RDATA	命令读数据	0x00
31:8	-	保留，用户软件不应该向保留位写入 1。从保留位读取的值是未定义的	NA

11.10.7 DMA寄存器

该组寄存器用于 DMA 模式操作（详见“DMA 操作”）。

1. USB DMA请求状态寄存器（USBDMARSt - 0x5000 C250）

当出现端点中断（见 USBEpIntSt 的描述）并且 USBEpIntEn 中的对应位为 0 时，该寄存器中与非同步端点相关的位由硬件置位。当 USBEpIntEn 中的对应位为 0 并且出现 FRAME 中断时，与同步端点相关的位置位。如果 DMA 对于 USBEpDMASt 寄存器中的对应端点来说是使能的，则该寄存器中为 1 的位可用作 DMA 引擎开始数据传输的标志。对于控制端点（EP0 和 EP1），不可以使能 DMA。USBDMARSt 为只读寄存器。

表 11.39 USB DMA 请求状态寄存器位分配

复位值：0x0000 0000

位	31	30	29	28	27	26	25	24
符号	EP31	EP30	EP29	EP28	EP27	EP26	EP25	EP24
位	23	22	21	20	19	18	17	16
符号	EP23	EP22	EP21	EP20	EP19	EP18	EP17	EP16
位	15	14	13	12	11	10	9	8
符号	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8
位	7	6	5	4	3	2	1	0
符号	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0

表 11.40 USB DMA 请求状态寄存器位描述

位	符号	值	描述	复位值
0	EP0	0	控制端点 OUT（对于这个端点，DMA 不可以使能，EP0 位必须为 0）	0
1	EP1	0	控制端点 IN（对于这个端点，DMA 不可以使能，EP1 位必须为 0）	0
31:2	EPxx	 0 1	端点 xx（2≤ xx ≤31）的 DMA 请求 端点 xx 没有请求 DMA 端点 xx 请求 DMA	0

[1] 对于这个端点，DMA 不可以使能，USBDMARSt 中的对应位必须为 0。

2. USB DMA请求清除寄存器（USBDMARClr - 0x5000 C254）

向该寄存器的某个位写入 1 将把 USBDMARSt 寄存器中的对应位清零；写入 0 无效。

该寄存器用于在使能端点的 DMA 操作之前进行初始化。当端点的 DMA 操作使能时，硬件在完成一次数据包传输时将 USBDMARSt 寄存器中的对应位清零。因此，当端点的 DMA 操作使能时，软件不应该使用该寄存器进行清零操作。

USBDMARClr 为只写寄存器。

USBDMARClr寄存器的位分配与USBDMARSt寄存器相同（表 11.39）。

表 11.41 USB DMA 请求清除寄存器位描述

位	符号	值	描述	复位值
0	EP0	0	控制端点 OUT（对于这个端点，DMA 不可以使能，EP0 位必须为 0）	0
1	EP1	0	控制端点 IN（对于这个端点，DMA 不可以使能，EP1 位必须为 0）	0
31:2	EPxx	 0 1	清除端点 xx（2≤ xx ≤31）的 DMA 请求 无影响 清除 USBDMARSt 中的相应位	0

3. USB DMA请求设置寄存器（USBDMARSet - 0x5000 C258）

向该寄存器的某位写入 1 将把 USBDMARSt 寄存器中的对应位置位；写入 0 无效。

该寄存器允许软件提出 DMA 请求。这一特性在端点的操作由从模式切换到 DMA 模式时非常有用：如果即将在 DMA 模式中处理的数据包在 USBEpIntEn 中的对应位清零之前到达，则硬件不会提出 DMA 请求，此时软件可以使用该寄存器手动地启动 DMA 传输。

软件也可以使用该寄存器来启动 DMA 传输，以便在从主机接收到 IN 令牌包之前，预先积极地填充 IN 端点缓冲区。

USBDMARSet为只写寄存器。USBDMARSet寄存器的位分配与USBDMARSt寄存器是相同的（表 11.39）。

表 11.42 USB DMA 请求设置寄存器位描述

位	符号	值	描述	复位值
0	EP0	0	控制端点 OUT（对于这个端点，DMA 不可以使能，EP0 位必须为 0）	0
1	EP1	0	控制端点 IN（对于这个端点，DMA 不可以使能，EP1 位必须为 0）	0
31:2	EPxx	 0 1	设置端点 xx（2≤ xx ≤31）的 DMA 请求 无影响 设置 USBDMARSt 中的相应位	0

4. USB UDCA Head寄存器（USBUDCAH- 0x5000 C280）

UDCA（USB 设备通信区域）Head 寄存器保留 UDCA 在 USB RAM 中的地址。UDCA 和 DMA 描述符的详细信息请参考“USB 设备通信区域”和“DMA 描述符”。USBUDCAH 为读/写寄存器。

表 11.43 USB UDCA Head 寄存器位描述

位	符号	描述	复位值
6:0	-	保留，用户软件不应向保留位写入 1。UDCA 以 128 字节为边界对齐	0x00
31:7	UDCA_ADDR	UDCA 的起始地址	0

5. USB EP DMA状态寄存器（USBEpDMASt- 0x5000 C284）

该寄存器中的位表示对应端点的 DMA 操作是否使能。只有当该寄存器中的相应位置位时，端点的 DMA 传输才能启动。USBEpDMASt 为只读寄存器。

表 11.44 USB EP DMA 状态寄存器位描述

位	符号	值	描述	复位值
0	EP0_DMA_ENABLE	0	控制端点 OUT（对于这个端点，DMA 不可以使能，EP0_DMA_ENABLE 位必须为 0）	0
1	EP1_DMA_ENABLE	0	控制端点 IN（对于这个端点，DMA 不可以使能，EP1_DMA_ENABLE 位必须为 0）	0
31:2	EPxx_DMA_ENABLE		表示端点 xx（2≤xx≤31）的 DMA 是否使能	0
		0	禁止端点 EPxx 的 DMA 操作	
		1	使能端点 EPxx 的 DMA 操作	

6. USB EP DMA使能寄存器（USBEpDMAEn- 0x5000 C288）

向该寄存器的某个位写入 1 将使能对应端点的 DMA 操作；写入 0 无效。控制端点 EP0 和 EP1 的 DMA 操作不可以使能。USBEpDMAEn 为只写寄存器。

表 11.45 USB EP DMA 使能寄存器位描述

位	符号	值	描述	复位值
0	EP0_DMA_ENABLE	0	控制端点 OUT（对于这个端点，DMA 不可以使能，EP0_DMA_ENABLE 位必须为 0）	0
1	EP1_DMA_ENABLE	0	控制端点 IN（对于这个端点，DMA 不可以使能，EP1_DMA_ENABLE 位必须为 0）	0
31:2	EPxx_DMA_ENABLE		端点 xx（2≤xx≤31）的 DMA 使能控制位	0
		0	无效	
		1	使能端点 EPxx 的 DMA 操作	

7. USB EP DMA禁能寄存器（USBEpDMADis- 0x5000 C28C）

向该寄存器的某个位写入 1 将把USBEpDMASt中的对应位清零；写入 0 对于USBEpDMASt中的对应位将无效。而只要对该寄存器进行写操作，不管写入的值如何，内部DMA_PROCEED标志都会清零。有关DMA_PROCEED标志的详细信息请参考“优化的描述符读取操作”。如果当寄存器中的对应位清零时正在处理端点的DMA传输，那么在DMA禁能之前该传输完成。当在DMA传输过程中检测到错误条件时，对应位由硬件清零。USBEpDMADis为只写寄存器。

表 11.46 USB EP DMA 禁能寄存器位描述

位	符号	值	描述	复位值
0	EP0_DMA_DISABLE	0	控制端点 OUT（对于这个端点，DMA 不可以使能，EP0_DMA_DISABLE 位必须为 0）	0
1	EP1_DMA_DISABLE	0	控制端点 IN（对于这个端点，DMA 不可以使能，EP1_DMA_DISABLE 位必须为 0）	0
31:2	EPxx_DMA_DISABLE		端点 xx（2≤xx≤31）的 DMA 禁能控制位	0
		0	无效	
		1	禁能端点 EPxx 的 DMA 操作	

8. USB DMA中断状态寄存器（USBDMAIntSt- 0x5000 C290）

该寄存器中的每个位反映对应的中断状态寄存器中的每个位是否置位。USBDMAIntSt 为只读寄存器。

表 11.47 USB DMA 中断状态寄存器位描述

位	符号	值	描述	复位值
0	EOT		传输结束中断位	0
		0	USBEoTIntSt 寄存器中的所有位都为 0	
		1	USBEoTIntSt 寄存器中至少有一个位为 1	
1	NDDR		新的 DD 请求中断位	0
		0	USBNDDRIntSt 寄存器中的所有位都为 0	
		1	USBNDDRIntSt 寄存器中至少有一个位为 1	
2	ERR		系统错误中断位	0
		0	USBSysErrIntSt 寄存器中的所有位都为 0	
		1	USBSysErrIntSt 寄存器中至少有一个位为 1	
31:3	-		保留，用户软件不应向其写入 1。从保留位读出的值未被定义	NA

9. USB DMA中断使能寄存器（USBDMAIntEn- 0x5000 C294）

向该寄存器中的某个位写入 1 时，如果 USBDMAIntSt 中的对应位置位，则将在 USB_INT_REQ_DMA 中断线上产生一个中断。USBDMAIntEn 为读/写寄存器。

表 11.48 USB DMA 中断使能寄存器位描述

位	符号	值	描述	复位值
0	EOT		传输结束中断使能位	0
		0	传输结束中断禁能	
		1	传输结束中断使能	
1	NDDR		新的 DD 请求中断使能位	0
		0	新的 DD 请求中断禁能	
		1	新的 DD 请求中断使能	
2	ERR		系统错误中断使能位	0
		0	系统错误中断禁能	
		1	系统错误中断使能	
31:3	-		保留，用户软件不应向其写入 1。从保留位读出的值未被定义	NA

10. USB传输结束中断状态寄存器（USBEoTIntSt- 0x5000 C2A0）

如果当前 DMA 描述符的 DMA 传输完成，则该寄存器中与端点对应的位将置位（可以是正常结束也可以是因为一个错误而置位）。产生中断的原因在描述符的 DD_status 区域记录。USBEoTIntSt 为只读寄存器。

表 11.49 USB 传输结束中断状态寄存器位描述

位	符号	值	描述	复位值
31:0	EPxx		端点 xx（2≤xx≤31）的传输结束中断请求	0
		0	没有端点 xx 的传输结束中断请求	
		1	有端点 xx 的传输结束中断请求	

11. USB传输结束中断清零寄存器（USBEoTIntClr- 0x5000 C2A4）

向该寄存器的某个位写入 1 可清除 USBEoTIntSt 寄存器中的对应位；写入 0 无效。USBEoTIntClr 为只写寄存器。

表 11.50 USB 传输结束中断清零寄存器位描述

位	符号	值	描述	复位值
31:0	EPxx		清除端点 xx（2≤xx≤31）的传输结束中断请求	0
		0	无效	
		1	清除 USBEoTIntSt 寄存器中 EPxx 传输结束的中断请求	

12. USB传输结束中断置位寄存器（USBEoTIntSet- 0x5000 C2A8）

向该寄存器的某个位写入 1 可将 USBEoTIntSt 寄存器中的对应位置位；写入 0 无效。USBEoTIntSet 为只写寄存器。

表 11.51 USB 传输结束中断置位寄存器位描述

位	符号	值	描述	复位值
31:0	EPxx		设置端点 xx（2≤xx≤31）的传输结束中断请求	0
		0	无效	
		1	将 USBEoTIntSt 寄存器中 EPxx 传输结束的中断请求置位	

13. USB新DD请求中断状态寄存器（USBNDDRIntSt- 0x5000 C2AC）

当从 USB 设备中请求一次传输并且对应的端点没有检测到有效的 DD 时，该寄存器中的位置位。USBNDDRIntSt 为只读寄存器。

表 11.52 USB 新 DD 请求中断状态寄存器位描述

位	符号	值	描述	复位值
31:0	EPxx		端点 xx（2≤xx≤31）的新 DD 中断请求	0
		0	没有端点 xx 的新 DD 中断请求	
		1	有端点 xx 的新 DD 中断请求	

14. USB新DD请求中断清零寄存器（USBNDDRIntClr- 0x5000 C2B0）

向该寄存器中的某个位写入 1 可清除 USBNDDRIntSt 寄存器中的对应位；写入 0 无效。USBNDDRIntClr 为只写寄存器。

表 11.53 USB 新 DD 请求中断清零寄存器位描述

位	符号	值	描述	复位值
31:0	EPxx		清除端点 xx（2≤xx≤31）的新 DD 中断请求	0
		0	无效	
		1	清除 USBNDDRIntSt 寄存器中的端点 xx 的新 DD 中断请求	

15. USB新DD请求中断置位寄存器（USBNDDRIntSet- 0x5000 C2B4）

向该寄存器中的某个位写入 1 可置位 USBNDDRIntSt 寄存器中的对应位；写入 0 无效。USBNDDRIntSet 为只写寄存器。

表 11.54 USB 新 DD 请求中断置位寄存器位描述

位	符号	值	描述	复位值
31:0	EPxx		设置端点 xx（2≤xx≤31）的新 DD 中断请求	0
		0	无效	
		1	置位 USBNDDRIntSt 寄存器中的端点 xx 的新 DD 中断请求	

16. USB系统错误中断状态寄存器（USBSysErrIntSt- 0x5000 C2B8）

如果在传输数据或者在获取或更新 DD 时出现系统错误（AHB 总线错误），则该寄存器中的对应位置位。USBSysErrIntSt 为只读寄存器。

表 11.55 USB 系统错误中断状态寄存器位描述

位	符号	值	描述	复位值
31:0	EPxx		端点 xx（2≤xx≤31）的系统错误中断请求	0
		0	没有端点 xx 的系统错误中断请求	
		1	有端点 xx 的系统错误中断请求	

17. USB系统错误中断清零寄存器（USBSysErrIntClr- 0x5000 C2BC）

向该寄存器中的某个位写入 1 可清除 USBSysErrIntSt 寄存器中的对应位；写入 0 无效。USBSysErrIntClr 为只写寄存器。

表 11.56 USB 系统错误中断清零寄存器位描述

位	符号	值	描述	复位值
31:0	EPxx		清除端点 xx（2≤xx≤31）的系统错误中断请求	0
		0	无效	
		1	清除 USBSysErrIntSt 寄存器中的端点 xx 的系统错误中断请求	

18. USB系统错误中断置位寄存器（USBSysErrIntSet- 0x5000 C2C0）

向该寄存器中的某个位写入 1 可置位 USBSysErrIntSt 寄存器中的对应位；写入 0 无效。USBSysErrIntSet 为只写寄存器。

表 11.57 USB 系统错误中断置位寄存器位描述

位	符号	值	描述	复位值
31:0	EPxx		设置端点 xx（2≤xx≤31）的系统错误中断请求	0
		0	无效	
		1	将 USBSysErrIntSt 寄存器中的端点 xx 的系统错误中断请求置位	

11.11 中断处理

本节描述了如何将任意端点上的一个中断事件发送到嵌套的向量中断控制器（NVIC）。中断事件处理如图 11.3所示。

所有的非同步 OUT 端点（控制、批量和中断端点）在成功地接收到一个信息包时产生中断。所有的非同步 IN 端点在成功地发送一个信息包时，或者在发送一个 NAK 信号并且通过 SIE 设置模式命令将 NAK 上的中断使能时产生中断，详见“设置模式”的描述。对于同步端点，每 1ms 产生一个帧中断。

从模式和 DMA 模式的中断处理是不同的。

（1）从模式

如果在端点上出现一个中断事件并且该端点中断在 USBEpIntEn 寄存器中是使能的，则 USBEpIntSt 寄存器中的对应状态位将置位。对于非同步端点，根据对应的 USBEpIntPri[n]寄存器，我们将所有的端点中断事件划分为两种类型：快速端点中断事件和慢速端点中断事件。所有快速端点中断事件相或并发送到 USBDevIntSt 寄存器中的 EP_FAST 位。所有慢速端点中断事件相或并发送到 USBDevIntSt 中的 EP_SLOW 位。

对于同步端点，USBDevIntSt 寄存器中的 FRAME 位每 1ms 置位一次。

USBDevIntSt 寄存器保存所有端点中断事件的状态以及其它各种中断的状态（见“USB 设备中断状态寄存器”）。默认情况下，将所有中断（如果已在 USBDevIntEn 寄存器中使能）发送到 USBIntSt 寄存器中的 USB_INT_REQ_LP 位，请求低优先级中断处理。而利用 USBDevIntPri 寄存器，也可以将 FRAME 或 EP_FAST 位发送到 USBIntSt 寄存器中的 USB_INT_REQ_HP 位。

EP_FAST 和 FRAME 中断中只有一个能够发送到 USB_INT_REQ_HP 位。如果试图将这两个位都发送到 USB_INT_REQ_HP，则这两个中断事件会都发送到 USB_INT_REQ_LP。

慢速端点中断事件始终直接发送到USB_INT_REQ_LP位，通过软件请求低优先级中断处理。

发送到 NVIC 的最后一个中断信号由 USBIntSt 寄存器中的 EN_USB_INTS 位控制。只有当 EN_USB_INTS 位置位时，USB 中断才会发送到 NVIC。

（2）DMA 模式

如果在非控制端点上出现中断事件并且该端点中断在 USBEpIntEn 寄存器中没有使能，则 USBDMARSt 寄存器中对应的状态位由硬件置位。如果 DMA 传输对于 USBEpDMASt 寄存器中的对应端点是使能的，则 USBDMARSt 中置位的位可作为 DMA 引擎传输数据的标志。

每个端点在 DMA 模式中传输数据时可产生 3 种类型的中断：传输结束中断、新 DD 请求中断和系统错误中断。这些中断事件会分别把 USBEoTIntSt、USBNDDRIntSt 和 USBSysErrIntSt 寄存器中与各个端点对应的位置位。然后，来自所有端点的传输结束中断相或并发送到 USBDMAIntSt 中的 EOT 位。同样，所有的新 DD 请求中断和系统错误中断事件分别发送到 USBDMAStInt 寄存器中的 NDDR 位和 ERR 位。

EOT、NDDR 和 ERR 位（如果已在 USBDMAIntEn 寄存器中使能）相或来将 USBIntSt 寄存器中的 USB_INT_REQ_DMA 位置位。如果 USBIntSt 中的 EN_USB_INTS 位是置位的，则将中断发送到 NVIC。

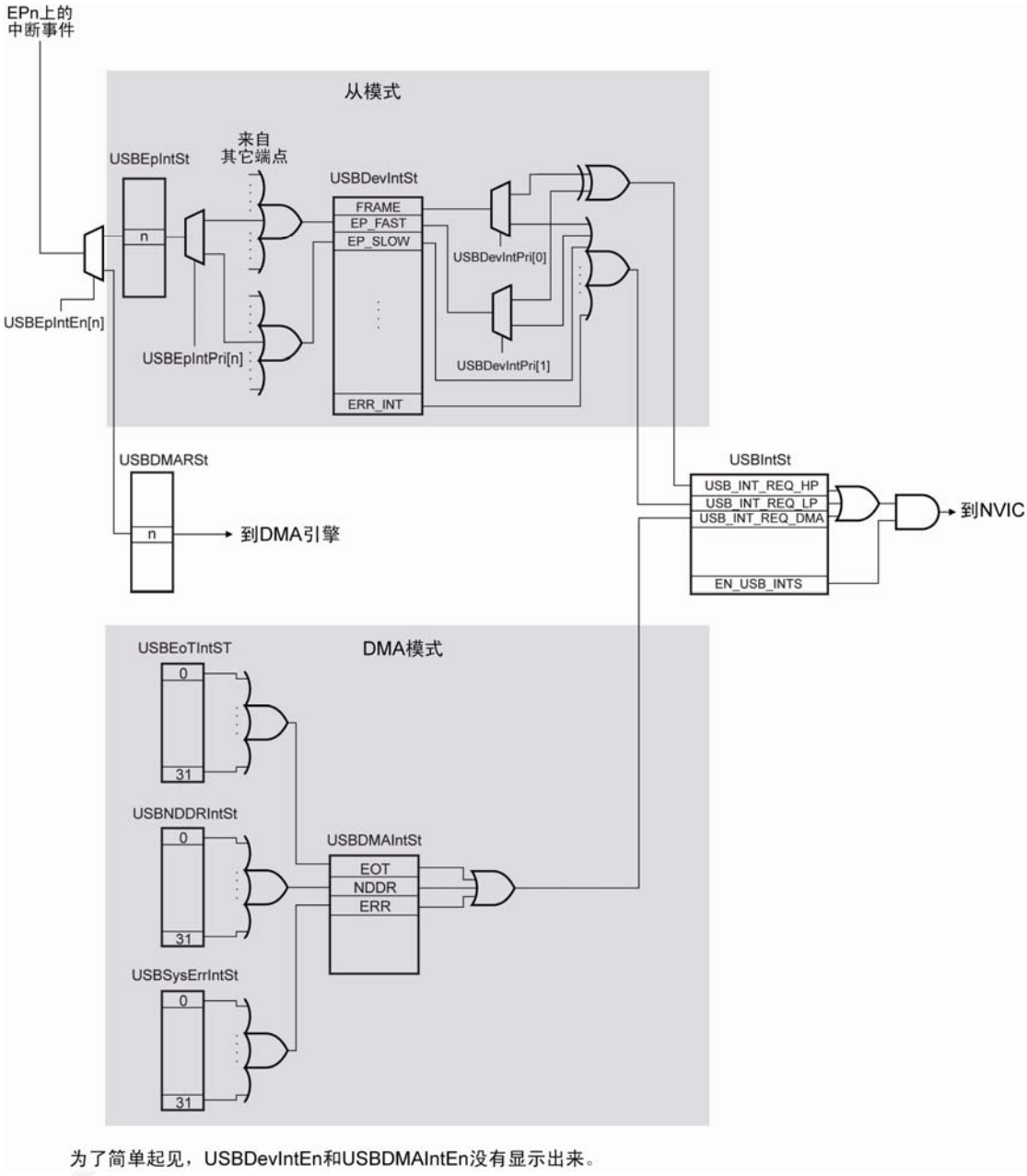


图 11.3 中断事件处理

11.12 串行接口引擎命令描述

串行接口引擎（SIE）的函数和寄存器使用命令来访问。命令由命令代码组成，后面是可选的数据字节（可以是读操作或写操作）。在执行上述访问时将使用 USBCmdCode（见“USB 命令代码寄存器”表）和 USBCmdData（见“USB 命令数据寄存器”表）寄存器。

一次完整的访问包含两个阶段：

- **命令阶段**：对 USBCmdCode 寄存器执行写操作，将 CMD_PHASE 字段设置为 0x05（命令），CMD_CODE 字段设置为所需的命令代码。在命令执行完时，USBDevIntSt 寄存器中的 CCEMPTY 位置位；
- **数据阶段（可选）**：如果执行写操作，则将 USBCmdCode 寄存器中的 CMD_PHASE 字段设置为 0x01（写），CMD_WDATA 字段设置为所需的写数据。写操作完成时，USBDevIntSt 寄存器中的 CCEMPTY 位置位。如果执行读操作，则将 USBCmdCode 寄

存器中的 CMD_PHASE 字段设置为 0x02（读），CMD_CODE 字段利用读对应的命令代码来设置。读操作完成时，USBDevIntSt 寄存器中的 CDFULL 位将置位，表示 USBCmdData 寄存器中的数据在执行读操作时是可用的。在寄存器为多字节的情况下，首先访问的是最低有效字节。

在表 11.58中列出了可用的命令。下面是读取当前帧编号的命令的例子（读 2 个字节）：

```
USBDevIntClr = 0x30;                // 将 CCEMPTY 和 CDFULL 清零
USBCmdCode = 0x00F50500;            // CMD_CODE=0xF5, CMD_PHASE=0x05(命令)
while (!(USBDevIntSt & 0x10));      // 等待 CCEMPTY.
USBDevIntClr = 0x10;                // 清除 CCEMPTY 中断位.
USBCmdCode = 0x00F50200;            // CMD_CODE=0xF5, CMD_PHASE=0x02(读)
while (!(USBDevIntSt & 0x20));      // 等待 CDFULL.
USBDevIntClr = 0x20;                // 清除 CDFULL.
CurFrameNum = USBCmdData;           // 读帧编号的 LSB 字节
USBCmdCode = 0x00F50200;            // CMD_CODE=0xF5, CMD_PHASE=0x02(读)
while (!(USBDevIntSt & 0x20));      // 等待 CDFULL.
Temp = USBCmdData;                  // 读帧编号的 MSB 字节
USBDevIntClr = 0x20;                // 清除 CDFULL 中断位
CurFrameNum = CurFrameNum | (Temp << 8);
```

下面是设置地址命令的例子（写 1 个字节）：

```
USBDevIntClr = 0x10;                // 清除 CCEMPTY
USBCmdCode = 0x00D00500;            // CMD_CODE=0xD0, CMD_PHASE=0x05(命令)
while (!(USBDevIntSt & 0x10));      // 等待 CCEMPTY
USBDevIntClr = 0x10;                // 清除 CCEMPTY
USBCmdCode = 0x008A0100;            // CMD_WDATA=0x8A(DEV_EN=1, DEV_ADDR=0xA),
                                    // CMD_PHASE=0x01(写)
while (!(USBDevIntSt & 0x10));      // 等待 CCEMPTY
USBDevIntClr = 0x10;                // 清除 CCEMPTY
```

表 11.58　SIE 命令代码表

命令名称	接受者	代码（Hex）	数据阶段
设备命令			
设置地址	设备	D0	写 1 个字节
配置设备	设备	D8	写 1 个字节
设置模式	设备	F3	写 1 个字节
读取当前帧号	设备	F5	读 1 或 2 个字节
读测试寄存器	设备	FD	读 2 个字节
设置设备状态	设备	FE	写 1 个字节
获得设备状态	设备	FE	读 1 个字节
获得错误代码	设备	FF	读 1 个字节
读错误状态	设备	FB	读 1 个字节

续上表

命令名称	接受者	代码（Hex）	数据阶段
端点命令			
选择端点	端点 0	00	读 1 个字节（可选）
	端点 1	01	读 1 个字节（可选）
	端点 xx	xx	读 1 个字节（可选）
选择端点/清除中断	端点 0	40	读 1 个字节
	端点 1	41	读 1 个字节
	端点 xx	xx+40	读 1 个字节
设置端点状态	端点 0	40	写 1 个字节
	端点 1	41	写 1 个字节
	端点 xx	xx+40	写 1 个字节
清空缓冲区	所选的端点	F2	读 1 个字节（可选）
确认缓冲区	所选的端点	FA	无

11.12.1 设置地址（命令：0xD0，数据：写 1 个字节）

设置地址命令用于设置 USB 分配的地址并使能（内含的）函数。设备中的地址设置将在控制处理的状态阶段之后生效。总线复位之后，DEV_ADDR 设置为 0x00，DEV_EN 设置为 1。设备将响应函数地址 0x00，端点 0（默认端点）的信息包。

表 11.59 设备设置地址寄存器位描述

位	符号	描述	复位值
6:0	DEV_ADDR	由软件设置的设备地址。总线复位之后，该字段的值为 0x00	0x00
7	DEV_EN	设备使能。总线复位之后，该位为 1 0：设备不会响应任何包 1：设备将响应函数地址为 DEV_ADDR 的信息包	0

11.12.2 配置设备（命令：0xD8，数据：写 1 个字节）

向寄存器写入 1 表示对设备进行配置并且所有已使能的非控制端点将作出响应。在默认状态下，即使设备没有配置，控制端点也始终是使能的并作出响应。

表 11.60 配置设备寄存器位描述

位	符号	描述	复位值
0	CONF_DEVICE	对设备进行配置。所有使能的非控制端点将作出响应。在总线复位时，该位由硬件清零。当该位置位时，如果设备不是在挂起状态（SUS=0），则 UP_LED 信号被驱动为低电平	
7:1	-	保留，用户软件不应向保留位写入 1。从保留位读出的值未被定义	NA

11.12.3 设置模式（命令：0xF3，数据：写 1 个字节）

表 11.61 设置模式寄存器位描述

位	符号	值	描述	复位值
0	AP_CLK	0	始终是 PLL 时钟 USB_NEED_CLK 有效；当设备进入挂起状态时，可以将 48MHz 时钟停止	0
		1	USB_NEED_CLK 固定为 1；当设备进入挂起状态时，不可以将 48MHz 时钟停止	
1	INAK_CI	0	控制 IN 端点的 NAK 中断 只有在成功处理时才产生中断	0
		1	当 IN 处理成功完成以及得到 NAK 应答时都产生中断	
2	INAK_CO	0	控制 OUT 端点的 NAK 中断 只有成功处理时才产生中断	0
		1	当 OUT 处理成功完成以及获得 NAK 应答时都产生中断	
3	INAK_II	0	中断 IN 端点的 NAK 中断 只有成功处理时才产生中断	0
		1	当 IN 处理成功完成以及获得 NAK 应答时都产生中断	
4	INAK_IO[1]	0	中断 OUT 端点的 NAK 中断 只有成功处理时才产生中断	0
		1	当 OUT 处理成功完成以及获得 NAK 应答时都产生中断	
5	INAK_BI	0	批量 IN 端点的 NAK 中断 只有成功处理时才产生中断	0
		1	当 IN 处理成功完成以及获得 NAK 应答时都产生中断	
6	INAK_BO[2]	0	批量 OUT 端点的 NAK 中断 只有成功处理时才产生中断	0
		1	当 OUT 处理成功完成以及获得 NAK 应答时都产生中断	
7	-	-	保留，用户软件不应向保留位写入 1。从保留位读出的值未被定义	NA

[1] 如果 DMA 对于任何的中断 OUT 端点来说都是使能的，则该位应复位为 0。

[2] 如果 DMA 对于任何的批量 OUT 端点来说都是使能的，则该位应复位为 0。

11.12.4 读当前帧编号（命令：0xF5，数据：读 1 个或 2 个字节）

返回上一次成功接收到的 SOF 的帧编号。帧编号为 11 位宽，首先返回最低有效字节。如果用户只需要帧编号的低 8 位，则只需要读第一个字节。

- 如果在一帧信息的开始处没有接收到 SOF，则返回的帧编号为上一次成功接收到 SOF 的帧编号。
- 如果 SOF 帧编号含有 CRC 错误，则在设备接收帧编号时，返回的帧编号是被破坏的。

11.12.5 读测试寄存器（命令：0xFD，数据：读 2 个字节）

测试寄存器为 16 位宽。如果 USB 时钟（usbclk 和 AHB 从机时钟）正在运行，则它将返回 0xA50F 的值。

11.12.6 设置设备状态（命令：0xFE，数据：写 1 个字节）

设置设备状态命令将对设备状态寄存器中的位进行设置。

表 11.62 设置设备状态寄存器位描述

位	符号	值	描述	复位值
0	CON	0	连接位表示设备的当前连接状态。它对用于 SoftConnect 的 CONNECT 输出管脚进行控制。读取连接位时将返回当前的连接状态。当 V_{BUS} 状态输入为低电平并持续 3ms 以上时，该位由硬件清零 向该位写入 0 将使 CONNECT 管脚变为高电平	0
		1	向该位写入 1 将使 CONNECT 管脚变为低电平	
1	CON_CH	0	连接发生改变 该位在读操作时清零	0
		1	当设备的上拉电阻由于 V_{BUS} 消失而断开连接时，该位置位。当该位为 1 时，产生 DEV_STAT 中断	
2	SUS	0	挂起：挂起位表示当前的挂起状态。当设备被挂起（SUS=1）并且 CPU 向 SUS 位写入 0 时，该设备将产生一个远程唤醒。这只有在设备被连接时（CON=1）才发生。当设备没有连接或没有挂起时，向该位写入 0 是无效的。向该位写入 1 也无效 出现任何活动时，该位复位为 0	0
		1	当设备在其上行端口（upstream port）上，持续 3ms 以上都没有看到任何活动时，该位置位	
3	SUS_CH	0	挂起位（SUS）变化指示器。SUS 位在以下情况下会翻转： ● 设备进入挂起状态 ● 设备断开连接 ● 设备在其上行端口上接收到恢复信号 该位在读操作时清零 SUS 位没有改变	0
		1	SUS 位发生改变。同时产生一个 DEV_STAT 中断	
4	RST	0	总线复位位。在总线复位时，设备将自动进入默认状态。在默认状态下： ● 设备没有配置 ● 将对地址 0 作出响应 ● 控制端点将处于暂停状态 ● 除了控制端点 EP0 和 EP1 以外，所有端点都没有实现 ● 所有端点的数据切换（data toggling）均被复位 ● 所有缓冲区被清零 ● 端点中断状态没有发生改变 ● 产生 DEV_STAT 中断 注：当设备没有被连接（CON=0）时将忽略总线复位 该位在读操作时清零	0
		1	在设备接收到总线复位时，该位置位。产生 DEV_STAT 中断	
7:5	-		保留，用户软件不应向保留位写入 1。从保留位读出的值未被定义	NA

11.12.7 获得设备状态（命令：0xFE，数据：读 1 个字节）

获得设备状态命令将返回设备状态寄存器的值。读取设备状态时将返回 1 个字节的数据。这个设备状态寄存器的位字段定义与设置设备状态寄存器是相同的，如表 11.62所示。

注：为确保操作正确，在执行获得设备状态命令之前必须将 USBDevIntSt 寄存器中的 DEV_STAT 位清零。

11.12.8 获得错误代码（命令：0xFF，数据：读 1 个字节）

SIE 中可能会出现不同的错误状态。获得错误代码命令可返回上一个已发生的错误代码。错误代码由 4 个最低有效位构成。

表 11.63 获得错误代码寄存器位描述

位	符号	值	描述	复位值
3:0	EC		错误代码	0x0
		0000	无错误	
		0001	PID 编码错误	
		0010	未知的 PID	
		0011	意外的信息包－任何违反规范的包序列	
		0100	令牌 CRC 中的错误	
		0101	数据 CRC 中的错误	
		0110	超时错误	
		0111	多路串扰（Babble）	
		1000	信息包结束时的错误	
		1001	发送/接收 NAK	
		1010	发送暂停	
		1011	缓冲区溢出错误	
		1100	发送空包（只针对 ISO 端点）	
		1101	位填充错误	
		1110	同步时的错误	
		1111	数据 PID 中的 Toggle 位错误，数据无效	
4	EA	-	一旦读该寄存器，Error Active 位将被复位	
7:5	-		保留，用户软件不应向保留位写入 1。从保留位读出的值未被定义	NA

11.12.9 读错误状态（命令：0xFB，数据：读 1 个字节）

该命令从 USB 设备中读取 8 位错误寄存器。该寄存器记录了最近在 SIE 中发生的错误事件。如果该寄存器中的任何一位置位，则 USBDevIntSt 寄存器中的 ERR_INT 位置位。在对该寄存器执行读操作之后，错误位清零。

表 11.64 读错误状态寄存器位描述

位	符号	描述	复位值
0	PID_ERR	PID 编码错误、未知的 PID 或未知的令牌 CRC	0
1	UEPKT	意外的信息包－任何违反规范的信息包序列	0
2	DCRC	数据 CRC 错误	0
3	TIMEOUT	超时错误	0

续上表

位	符号	描述	复位值
4	EOP	信息包结束错误	0
5	B_OVRN	缓冲区溢出	0
6	BTSTF	位填充错误	0
7	TGL_ERR	数据 PID 中的错误翻转位（toggle bit），数据无效	0

11.12.10 选择端点（命令 0x00- 0x1F，数据：读 1 个字节（可选））

选择端点命令将一个内部指针初始化为指向 EP_RAM 中的所选缓冲区的起始字节。该命令后面可跟着一个读数据操作，从而在端点缓冲区的信息包上返回一些附加信息。选择端点命令的命令代码与物理端点编号是相同的。在单缓冲端点的情况下 B_2_FULL 位无效。

表 11.65 选择端点寄存器位描述

位	符号	值	描述	复位值
0	FE		满/空。该位表示端点缓冲区的满/空状态。对于 IN 端点，FE 位是 B_1_FULL 和 B_2_FULL 位相与的结果。对于 OUT 端点，FE 位是 B_1_FULL 和 B_2_FULL 位相或的结果。对于单缓冲端点，该位只简单地反映 B_1_FULL 的状态	0
		0	对于 IN 端点，至少有一个端点写缓冲区是空的	
		1	对于 OUT 端点，至少有一个端点读缓冲区是满的	
1	ST		暂停的端点指示器	0
		0	所选的端点没有暂停	
		1	所选的端点被暂停	
2	STP		SETUP 位：该位的值在每次成功地接收到信息包（即在所选物理端点上获得 ACK 应答的封包）之后更新	0
		0	在所选端点上执行选择端点/清除中断命令时，STP 位清零	
		1	所选端点上一次接收到的包为 SETUP 包	
3	PO		包覆盖（over-written）位	0
		0	PO 位由“选择端点/清除中断”命令来清零	
		1	之前接收到的包被 SETUP 包覆盖	
4	EPN		EP NAKed 位表示发送一个 NAK。如果主机向已满的 OUT 缓冲区发送一个 OUT 包，则设备返回 NAK。如果主机向空的 IN 缓冲区发送一个 IN 令牌包，则设备返回 NAK	0
		0	当设备在接收到 OUT 包之后发送一个 ACK，或者当设备在发送 IN 包之后看到一个 ACK 时，EPN 位复位	
		1	当设备发送一个 NAK 并且 NAK 特性的中断使能时，EPN 位置位	
5	B_1_FULL		缓冲区 1 的状态	0
		0	缓冲区 1 为空	
		1	缓冲区 1 为满	
6	B_2_FULL		缓冲区 2 的状态	0
		0	缓冲区 2 为空	
		1	缓冲区 2 为满	
7	-		保留，用户软件不应向保留位写入 1。从保留位读出的值未被定义	NA

11.12.11 选择端点/清除中断（命令：0x40-0x5F，数据：读 1 个字节）

命令 0x40-0x5F 对于各个选择端点来说是相同的，不同的地方在于：

- 该命令将 USBEpIntSt 寄存器中与端点对应的位清零。
- 如果是控制 OUT 端点，则该命令把对应的选择端点寄存器中的 STP 和 PO 位清零。
- 必须读取一个字节。

注：可通过使用 USBCmdCode 和 USBCmdData 寄存器，或者将 USBEpIntClr 中对应的位置位来调用该命令。为方便起见，建议使用 USBEpIntClr 寄存器。

11.12.12 设置端点状态（命令：0x40-0x55，数据：写 1 个字节（可选））

设置端点状态命令用于对端点的状态位 7:5 和位 0 进行设置。设置端点状态的命令代码等于 0x40 与十六进制物理端点编号的和。对于各种类型的端点来说，并不是所有的位都是可设置的。

表 11.66 设置端点状态寄存器位描述

位	符号	值	描述	复位值
0	ST	0	暂停的端点位。当一个暂停的端点接收到 SETUP 令牌时，不管接收到的 SETUP 包的内容如何，该端点都会自动退出暂停状态。如果端点应该停留在其暂停状态，则 CPU 可以通过将该位置位使得该端点再次暂停。当一个暂停的端点由于设置端点状态命令或接收到一个 SETUP 令牌而退出暂停状态时，该端点也会被重新初始化。这样可以将缓冲区清空：如果是 OUT 缓冲区，则它将等待 DATA 0 PID；如果是 IN 缓冲区，则它会写 DATA 0 PID。端点的中断状态不会发生改变。若端点已经退出暂停状态，则向该位写入 0 将对端点进行初始化。若通过置位端点状态命令将端点暂停，则该端点也会被重新初始化 端点没有暂停	0
		1	端点被暂停	
4:1	-	-	保留，用户软件不应向保留位写入 1。从保留位读出的值未被定义	NA
5	DA	0	禁能端点位 端点使能	0
		1	端点禁能	
6	RF_MO	0	速率反馈（rate feedback）模式 中断端点位于翻转模式（Toggle mode）中	0
		1	中断端点位于速率反馈模式中。这意味着无需数据翻转位（data toggle bit）就可以进行传输	
7	CND_ST	0	条件暂停位 两个控制端点都没有暂停	0
		1	将两个控制端点暂停，除非选择端点寄存器中的 STP 位是置位的。该位只是针对控制 OUT 端点而定义的	

11.12.13 清空缓冲区（命令：0xF2，数据：读 1 个字节（可选））

在成功接收到主机发出的 OUT 包时，内部硬件 FIFO 状态 Buffer_Full 标志置位。通过返回一个 NAK 应答，设备将拒绝后面的所有信息包。当设备软件读取数据时，它应该通过发出清空缓冲区命令来将缓冲区清空。这样可将内部 Buffer_Full 标志清零。当缓冲区为空时，设备可以

接收新的信息包。

当可选数据字节的位 0 为 1 时，前面接收到的信息包被 SETUP 包覆盖（over-written）。Packet over-written 位只在控制传输中使用。根据 USB 规范，不管缓冲区的状态如何，都应该接受 SETUP 包。在读取 SETUP 数据之后，软件应该一直检查 PO 位的状态。如果该位是置位的，则软件应该丢弃之前读取的数据，通过发出选择端点/清除中断命令将 PO 位清零，读取新的 SETUP 数据并再次检查 PO 位的状态。

该命令使用情况的详细描述请参考“从模式操作”。

表 11.67 清空缓冲区寄存器位描述

位	符号	值	描述	复位值
0	PO	0	Packet over-written 位。该位只适用于控制端点 EP0 之前接收到的信息包保持完好	0
		1	之前接收到的信息包被后面的 SETUP 包覆盖	
7:1	-	-	保留，用户软件不应向保留位写入 1。从保留位读出的值未被定义	NA

11.12.14 确认缓冲区（命令：0xFA，数据：无）

当 CPU 已将数据写入 IN 缓冲区时，软件应发出一个确认缓冲区命令。该命令告知硬件，缓冲区准备在 USB 总线上进行发送操作。当接收到下一个 IN 令牌包时，硬件将会发送缓冲区中的内容。

在内部有一个称作 Buffer_Full 的硬件 FIFO 状态标志。该标志由确认缓冲区命令置位并当数据已在 USB 总线上发送完成时清零，缓冲区为空。

当控制 IN 端点对应的 OUT 缓冲区的 Packet Over-written（PO）位置位，或者包含一个挂起的 SETUP 包时，不能确认控制 IN 缓冲区有效。对于控制端点，已确认的缓冲区将在接收到 SETUP 包时无效。

该命令使用情况的详细描述请参考“从模式操作”。

11.13 USB设备控制器的初始化

LPC1700 系列 Cortex-M3 微控制器 USB 设备控制器的初始化步骤如下所示：

a）通过置位 PCONP 的 PCUSB 位来使能设备控制器；

b）配置和使能 PLL 和时钟分频器以提供 48MHz 的 usbclk 和所需的 cclk 频率。为了使设备控制器中的同步逻辑能够正确操作，最小的 cclk 频率为 18MHz。确定 PLL 设置和配置的步骤请参考“确定 PLL0 设置的过程”；

c）通过置位 USBClkCtrl 寄存器中 DEV_CLK_EN 和 AHB_CLK_EN 位来使能设备控制器时钟。查询 USBClkSt 寄存器中对应的时钟位直到它们被置位；

d）通过向对应的 PINSEL 寄存器执行写操作来使能 USB 管脚功能；

e）使用对应的 PINMODE 寄存器将 V_{BUS} 管脚上的上拉电阻禁能；

f）针对 EP0 和 EP1 设置 USBEpIn 和 USBMaxPSize 寄存器，并等待直到 USBDevIntSt 寄存器中的 EP_RLZED 位置位，表示端点 EP0 和 EP1 已实现；

g）使能端点中断（从模式）：

- 使用 USBEpIntClr 将所有端点中断清零；
- 使用 USBDevIntClr 将所有设备中断清零；
- 通过置位 USBEpIntEn 中的对应位，使能所需端点的从模式操作；

- 使用 USBEpIntPri 设置每个已使能的中断的优先级；
- 使用 SIE 设置模式命令对所需的中断模式进行配置；
- 使用 USBDevIntEn（通常是 DEV_STAT，EP_SLOW，也可能是 EP_FAST）使能设备中断。

h）配置 DMA（DMA 模式）：

- 使用 USBEpDMADis 禁止所有端点的 DMA 操作；
- 使用 USBDMARClr 清除所有挂起的 DMA 请求；
- 使用 USBEoTIntClr、USBNDDRIntClr 和 USBSysErrIntClr 清除所有的 DMA 中断；
- 在系统存储器中准备 UDCA；
- 将所需的 UDCA 地址（例如，0x7FD0 0000）写入 USBUDCAH；
- 使用 USBEpDMAEn 将所需端点的 DMA 操作使能；
- 置位 USBDMAIntEn 中的 EOT、DDR 和 ERR 位。

i）通过将对应的地址写入相关的向量表单元并使能 NVIC 中的 USB 中断，来安装 NVIC 中的 USB 中断处理器；

j）使用 SIE 设置地址命令将默认的 USB 地址设置为 0x0，DEV_EN 设为 1。总线复位也可以实现上述设置；

k）使用 SIE 设置设备状态命令将 CON 位设为 1，以便将 CONNECT 激活。

端点的配置根据软件应用程序进行更改。默认情况下，除了控制端点 EP0 和 EP1 外，所有的端点都是禁能的。其它端点在从主机中接收到 SET_CONFIGURATION 或 SET_INTERFACE 设备请求之后使用软件使能和配置。

11.14 从模式操作

在从模式中，CPU 使用寄存器接口在 RAM 和端点缓冲区之间传输数据。

11.14.1 中断的产生

在从模式中，RAM 和端点缓冲区之间的数据包传输可以在出现端点中断时启动来响应中断。端点中断使用 USBEpIntEn 寄存器来使能，并从 USBEpIntSt 寄存器中进行查询。

所有非同步的 OUT 端点在成功地接收到一个信息包时产生端点中断。所有非同步的 IN 端点在成功地发送一个信息包时，或者在总线上发送了一个 NAK 握手信号并且 NAK 特性的中断使能时产生中断。

对于同步端点，在出现 FRAME 中断（在 USBDevIntSt 中）时进行数据传输。

11.14.2 OUT端点的数据传输

当软件想从端点缓冲区中读取数据时，它应该将USBCtrl寄存器中的RD_EN位置位并将LOG_ENDPOINT字段设置为所需的端点编号。控制逻辑将获取信息包长度输入到USBRxPLen寄存器，并将PKT_RDY位置位（见表 11.33）。

现在，软件可以从USBRxData寄存器（表 11.32）中读取数据。当到达信息包的结尾时，RD_EN位清零，并且USBDevSt寄存器中的RxENDPKT位置位。此时，软件可发出清空缓冲区命令（表 11.67），端点准备接受下一个信息包。对于OUT同步端点来说，不管缓冲区是否已清空，它都会接收下一个包。因此，在帧结束之前没有从缓冲区中读出的数据将丢失。详细信息请参考“双缓冲的端点操作”。

如果软件在读取整个包之前将 RD_EN 清零，则读操作中止，数据保留在端点缓冲区中。当该端点的 RD_EN 位再次置位时，数据将从起始处读起。

11.14.3 IN端点的数据传输

当向端点缓冲区写入数据时，WR_EN 位（“USB 控制寄存器（USBCtrl-0x5000 C228）”置位，并且软件将信息包中即将被发送的字节数写入 USBTxPLen 寄存器（见“USB 发送包长度寄存器”的描述）。然后，它可以向 USBTxData 寄存器连续地写入数据。

如果向 USBTxData 写入的数据量已达到 USBTxPLen 寄存器中设置的字节数，则 WR_EN 位清零，并且 USBDevIntSt 寄存器中的 TxENDPKT 位置位。软件发出一个确认缓冲区（“确认缓冲区（命令：0xFA，数据：无）”）命令。现在，端点准备发送信息包。对于 IN 同步端点来说，只有在下一个 FRAME 中断出现之前对缓冲区进行了确认，缓冲区中的数据才会被发送；否则，下一帧将发送一个空包。如果软件在写入整个包之前将 WR_EN 清零，则在下一次该端点的 WR_EN 位置位时，写操作将再次从缓冲区的起始处执行。

对于同一个逻辑端点，RD_EN 和 WR_EN 可以同时为高电平，因此，可以间隔（interleaved）执行读和写操作。

11.15 DMA操作

在 DMA 模式中，DMA 在 RAM 和端点缓冲区之间传输数据。

下面将对 DMA 模式的操作进行描述。DMA 操作的背景信息在“USB 设备通信区域”和“触发 DMA 引擎”中描述。DMA 描述符的字段在“DMA 描述符”中描述。最后 3 个小节描述了 DMA 操作：“非同步端点的操作”、“同步端点操作”和“自动长度传输提取（ATLE）模式操作”。

11.15.1 传输术语

本小节提到了 3 种传输类型：

a）USB 传输：在 USB 总线上进行的数据传输。USB2.0 规范将 USB 传输简单地称作传输。在本小节中，我们把它称作 USB 传输以便与 DMA 传输相区分。USB 传输由多个事务处理组成，每个事务处理又由信息包组成。

b）DMA 传输：端点缓冲区和系统存储器（RAM）之间的数据传输。

c）信息包传输：在本小节中，信息包传输指的是端点缓冲区和系统存储器（RAM）之间的信息包传输。DMA 传输由一个或多个信息包传输组成。

11.15.2 USB设备通信区域

CPU 和 DMA 控制器通过一个公共的存储器区域进行通信，这个公共区域称作 USB 设备通信区域，即 UDCA。UDCA 为 32 字的 DMA 描述符指针（DDP）数组，每个 DDP 对应一个物理端点。如果针对端点定义了一个这样的数组，数组中的每个 DDP 均指向 DMA 描述符的起始地址。未实现的端点和 DMA 操作禁能的端点的 DDP 将被忽略并且可以设置为 NULL（0x0）。

UDCA 的起始地址存放在 USBUDCAH 寄存器中。UDCA 可以位于 RAM 的任意 128 字节边界，由 CPU 和 DMA 控制器访问。

图 11.4显示了UDCA以及UDCA Head（USBUDCAH）寄存器和DMA描述符之间的关系。

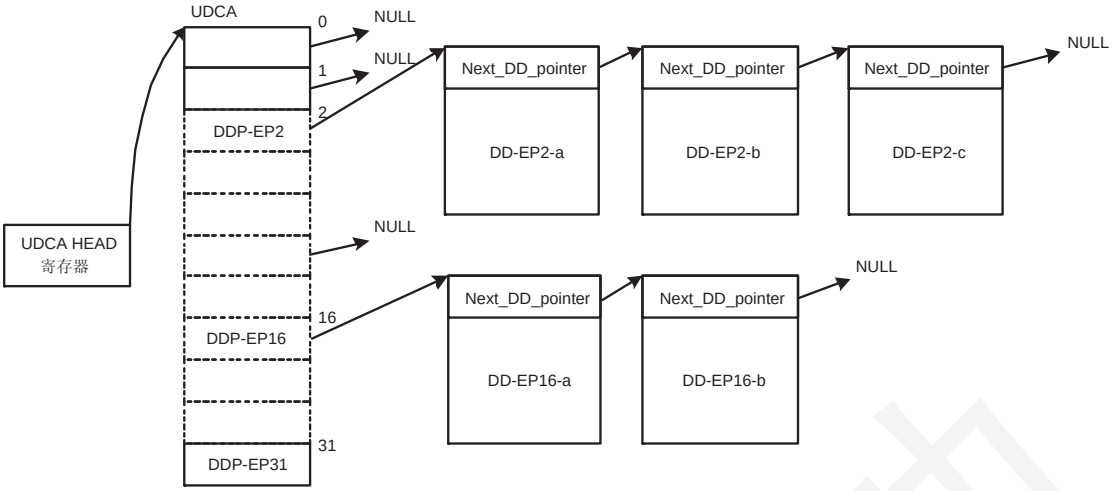


图 11.4 UDCA Head 寄存器和 DMA 描述符

11.15.3 触发DMA引擎

当通过将 USBEpIntEn 寄存器中的某个位设置为 0（见“USB 端点中断使能寄存器”）来禁止从模式操作时，对应的端点发出一个 DMA 请求，然后端点中断产生（见“USB DMA 请求状态寄存器（USBDMARSt- 0xFFE0 C250）”的描述）。

当在 USBEpDMASt 寄存器中使能端点的 DMA 操作时，该端点的 DMA 传输开始，USBDMARSt 中的对应位置位，并且找到了该端点的一个有效的 DD。

所有端点共用一个 DMA 通道以降低硬件开销。如果在 USBDMARSt 中有多个 DMA 请求是有效的，则最先处理物理端点编号最小的端点。

在 DMA 模式中，应该使用 SIE 设置模式命令（见“设置模式”的描述）将与批量 OUT 和中断 OUT 端点的 NAK 中断对应的位（INAK_BO 和 INAK_IO）设置为 0。

11.15.4 DMA描述符

DMA 传输通过一个称作 DMA 描述符（DD）的数据结构来描述。

DMA 描述符（DD）存放在 USB RAM 中。这些描述符可以位于 USB RAM 的任何字对齐的地址中。USB RAM 作为 USB 用途的系统存储器的一部分。它从地址 0x7FD0 0000 开始，大小为 8KB。

用于非同步端点的 DMA 描述符为 4 个字长，用于同步端点的 DMA 描述符为 5 个字长。

与 DMA 传输相关的参数如下所示：

- DMA 缓冲区的起始地址；
- DMA 缓冲区的长度；
- 下一个 DMA 描述符的起始地址；
- 控制信息；
- 计数信息（已传输的字节数）；
- 状态信息。

表 11.68列出了DMA描述符的各个字段。

表 11.68 DMA 描述符

字的位置	访问（H/W）	访问（S/W）	位的位置	描述
0	R	R/W	31:0	Next_DD_pointer（USB RAM 地址）
1	R	R/W	1:0	DMA_mode（00 为正常模式；01 为 ATLE 模式）
	R	R/W	2	Next_DD_valid（1 为有效；0 为无效）
	-	-	3	保留
	R	R/W	4	Isochronous_endpoint（1 为同步端点；0 为非同步端点）
	R	R/W	15:5	Max_packet_size
	R/W[1]	R/W	31:16	DMA_buffer_length 对于非同步端点，该值表示的是字节数；对于同步端点，该值表示的是包的个数
2	R/W	R/W	31:0	DMA_buffer_start_addr
3	R/W	R/I	0	DD_retired（初始化为 0）
	W	R/I	4:1	DD_status（初始化为 0000）： 0000：NotServiced 0001：BeingServiced 0010：NormalCompletion 0011：DataUnderrun（短包） 1000：DataOverrun 1001：SystemError
	W	R/I	5	Packet_valid（初始化为 0）
	W	R/I	6	LS_byte_extracted（ATLE 模式）（初始化为 0）
	W	R/I	7	MS_byte_extracted（ATLE 模式）（初始化为 0）
	R	W	13:8	Message_length_position（ATLE 模式）
	-	-	15:14	保留
	R/W	R/I	31:16	Present_DMA_count（初始化为 0）
4	R/W	R/W	31:0	Isochronous_packetsize_memory_address

[1] 在 ATLE 模式中，该字段的访问属性为只写。

表注：R-读；W-写；I-初始化。

1. Next_DD_pointer

该字段是一个指针，存放的是可以提取下一个 DMA 描述符的存储单元。

2. DMA_mode

该字段指定 DMA 操作的模式。DMA 操作定义了两种模式：正常模式和自动传输长度提取（ATLE）模式。在正常模式中，软件对 OUT 端点的 DMA_buffer_length 进行初始化。在 ATLE 模式中，DMA_buffer_length 从输入的数据中提取。详细信息请参考“自动长度传输提取（ATLE）模式操作”的描述。

3. Next_DD_Valid

该位表示软件是否已准备下一个 DMA 描述符。如果该位置位，则 DMA 引擎会在用完当前描述符时提取下一个描述符。

4. Isochronous_endpoint

该位置位表示描述符属于同步端点，因此，在提取描述符时必须读取 5 个字。

5. Max_packet_size

该字段表示端点的最大信息包容量。在针对 IN 端点从存储器中传输数据时使用该参数。OUT 端点使用它来检测短包。该参数只适用于非同步端点。Max_packet_size 的值应该与使用 USBMaxPSize 寄存器分配给端点的 MPS 值相同。

6. DMA_buffer_length

该字段表示传输数据时分配的 DMA 缓冲区的深度。在达到缓冲区长度限制时，DMA 引擎将停止使用这个描述符并寻找下一个描述符。

在正常模式中操作时，软件针对 IN 端点和 OUT 端点设置该值。在 ATLE 模式中，软件只对 IN 端点设置该值。对于 OUT 端点，硬件使用从数据流中提取的长度来设置该值。

对于同步端点，DMA_buffer_length 指的是包的个数；对于非同步端点，该字段的值用字节数表示。

7. DMA_buffer_start_addr

该字段为执行数据读取操作或写入操作的地址。DMA 引擎每完成一次信息包传输就会对该字段进行一次更新。

8. DD_retired

当 DMA 引擎用完当前描述符时，该位由硬件置位。在到达缓冲区末端、传输一个短包（非同步端点）或检测到一个错误条件时该位置位。

9. DD_status

DMA 传输的状态在该字段中编码。其编码定义如下：

- **NotServiced**：没有传输信息包。
- **BeingServiced**：至少传输一个信息包。
- **NormalCompletion**：DMA 描述符是在达到缓冲区末端并且没有出现错误的情况下退出的。DD_retired 位也置位。
- **DataUnderrun**：在到达 DMA 缓冲区末端之前，USB 传输由于接收到一个短包而结束。DD_retired 位也置位。
- **DataOverrun**：当信息包正在传输时，DMA 缓冲区到达末尾。这是一个错误情况。DD_retired 位也是置位的。当前的 DMA 计数字段的值等于 DMA_buffer_length 的值。这个信息包必须在下一次 DMA 传输时从端点缓冲区重新发送。USBEpDMASt 中对应的 EPxx_DMA_ENABLE 位清零。
- **SystemError**：正在处理的 DMA 传输由于 AHB 总线上的错误而中止。此时，DD_retired 位不置位。USBEpDMASt 中对应的 EPxx_DMA_ENABLE 位清零。由于更新 DD 的同时会发生系统错误，因此，RAM 中的 DD 字段可能是不可靠的。

10. Packet_valid

该位用于同步端点。它表示上一次传输到存储器中的包是否是在无错误的情况下接收到的。如果信息包有效，则该位置位，即它是在无错误的情况下接收的。同步端点的操作详见“同步端点操作”。

该位对于非同步端点是不需要的，因为只有当信息包没有错误时才产生 DMA 请求，因此，在产生 DMA 请求时，Packet_valid 将始终置位。

11. LS_byte_extracted

在 ATLE 模式中使用。该位置位表示提取了传输长度的最低有效字节（LSB）。提取的长度在 DMA_buffer_length 字段的位 23:16 中指示。

12. MS_byte_extracted

在 ATLE 模式中使用。该位置位表示提取了传输长度的最高有效字节（MSB）。提取的长度在 DMA_buffer_length 字段的位 31:24 中指示。当 LS_byte_extracted 和 MS_byte_extracted 位置位时，提取操作停止。

13. Present_DMA_count

该字段表示 DMA 引擎传输的字节数。DMA 引擎在每完成一个信息包传输之后对该字段进行更新。

对于同步端点，Present_DMA_count 为传输的信息包的个数；对于非同步端点，Present_DMA_count 为字节数。

14. Message_length_position

在 ATLE 模式中使用。该字段给出了嵌入到输入数据包中的消息长度位置的偏移量。该字段只适用于 OUT 端点。偏移量 0 表示消息长度从第一个包的第一个字节开始。

15. Isochronous_packetsize_memory_address

该字段为即将进行传输或被提取的信息包容量信息以及帧编号所在的存储器缓冲区的地址，见图 11.5。该字段只适用于同步端点。

11.15.5 非同步端点操作

1. 设置DMA传输

软件为那些即将使能DMA传输的物理端点准备DMA描述符（DD）。这些DD都在USB RAM中。我们将第一个DD的起始地址写入UDCA中对应端点的DMA描述指针（DDP）单元。然后，软件将USBEpDMAEn寄存器（见“USB EP DMA使能寄存器”的描述）中对应端点的EPxx_DMA_ENABLE位置位。描述符中的DMA_mode字段设置为“00”，采用正常模式。描述符的其它所有字段按表 11.68的规定进行初始化。

物理端点 0 和 1（默认的控制端点）不支持 DMA 操作。

2. 查找DMA描述符

当某个端点的 DMA 传输被触发时，DMA 引擎将首先确定是否必须提取一个新的描述符。如果上一次传输的包与这次传输是针对相同的端点并且 DD 没有处于退出状态，则不需要提取一个新的描述符。上述条件可使用一个内部标志 DMA_PROCEED 来识别（见“优化的描述符读取操作”）。

如果必须读取一个新的描述符，则 DMA 引擎将计算该端点的 DDP 的位置并从该位置取出 DD 的起始地址。DD 起始地址如果在位置 0，则该地址看作是无效的。此时将出现 NDDR 中断。所有其它字对齐的地址都看作是有效的。

在提取 DD 时，首先读取 DD 状态字（字 3）并且检查 DD_retired 位的状态。如果该位没有置位，则 DDP 指向一个有效的 DD。如果 DD_retired 置位，则 DMA 引擎将读取 DD 的控制字（字 1）。

如果 Next_DD_valid 位置位，则 DMA 引擎将取出 DD 的 Next_DD_pointer 字段（字 0）并将它加载到 DDP 中，这个新的 DDP 就被写入 UDCA 区域。

然后，从 DDP 中的地址处提取整个 DD（4 个字）。这个 DD 将给出即将进行的 DMA 传输

的详细情况。DMA 引擎把从该 DD 中提取的信息（起始地址、DMA 计数等）加载到它的硬件资源中。

如果 Next_DD_valid 没有置位，而 DD_retired 位置位，则 DMA 引擎唤起该端点的 NDDR 中断并将对应的 EPxx_DMA_ENABLE 位清零。

3. 传输数据

对于 OUT 端点，DMA 引擎从 EP_RAM 中读取当前包并传输到从 DMA_buffer_start_addr 开始的 USB RAM 存储单元中。对于 IN 端点，数据从 DMA_buffer_start_addr 开始的 USB RAM 中提取并写入 EP_RAM 中。DMA_buffer_start_addr 和 Present_DMA_count 字段在每传输一个包之后进行更新。

4. 优化的描述符读取操作

DMA 传输通常都是多包传输。硬件将不会再从存储器中读取新的 DD，除非端点发生改变。硬件将内部标志 DMA_PROCEED 置位来指示多包传输正在进行中。

当 DMA_buffer_length 字段中规定的字节数已经传输完成之后，DMA_PROCEED 标志被清零。当软件对 USBEpDMADis 寄存器执行写操作时，该标志也会被清零。可以清除 DMA_PROCEED 标志的功能允许软件强制再次读取 DD 以进行下个信息包的传输。向 USBEpDMADis 寄存器写入全 0 可以在无需禁止任何端点的 DMA 操作的情况下将 DMA_PROCEED 标志清零。

5. 结束信息包传输

在执行信息包传输时，DMA 引擎把带有更新后的状态信息的 DD 写回到读操作时的相同存储单元。DD 中的 DMA_buffer_start_addr、Present_DMA_count 以及 DD_status 字段被更新。

DMA 描述符在退出时可以有以下几种类型：

正常结束：如果当前包完全传输并且 Present_DMA_count 字段等于 DMA_buffer_length，则 DD 正常结束。DD 将被写回到存储器中，此时的 DD_retired 位置位，DD_status 设置为 NormalCompletion。该端点的 EOT 中断产生。

USB 传输结束：如果当前包已完全传输，包的容量小于 Max_packet_size 字段，并且仍然没有到达 DMA 缓冲区的末尾，则 USB 传输结束。DD 将被写回到存储器中，此时的 DD_retired 位置位，DD_Status 设置为 DataUnderrun。该端点的 EOT 中断产生。

错误结束：如果当前包没有完全传输，即在信息包传输过程中达到 DMA 缓冲区的末尾，则错误情况出现。DD 被写回到存储器中，此时的 DD_retired 位置位，DD_status 设置为 DataOverrun 状态代码。该端点的 EOT 中断产生，并且 USBEpDMASt 寄存器中的对应位清零。当使用 USBEpDMAEn 寄存器再次将对应的 EPxx_DMA_ENABLE 位置位时，没有完全传输的信息包将会从端点缓冲区重新发送。

6. No_Packet DD

对于 IN 传输，如果系统暂时没有任何数据要发送，则通过设置 No_Packet DD，系统可以响应 NDDR 中断。这可以通过将 DD 中的 Max_packet_size 和 DMA_buffer_length 字段设置为 0 来实现。在处理 No_Packet DD 时，DMA 引擎在没有传输信息包的情况下将 USBDMARSt 中对应端点的 DMA 请求清零。DD 退出，此时的状态代码为 NormalCompletion。No_Packet DD 可以根据需要进行多次设置。设备将在 USB 总线上以 NAK 响应 IN 令牌包，直到带有一个数据包的 DD 被设置并且 DMA 将数据包传输到端点缓冲区中。

11.15.6 同步端点操作

对于同步端点，每个包的长度都可以不同。每传输一帧信息，每个同步端点均有一个信息包。

1. 设置DMA传输

软件将 DD 中的同步端点位设置为 1，并设置 Isochronous_packetsize_memory_address 字段的初始值。所有其它字段的初始化与非同步端点相同。

对于同步端点，DMA_buffer_length 和 Present_DMA_count 字段的值用帧数目而不是字节数来表示。

2. 查找DMA描述符

查找描述符的方法与查找非同步端点的方法相同。

DMA 使能的同步端点可以在每个 FRAME 中断时产生 DMA 请求。在处理 DMA 请求时，DMA 引擎将读取描述符，并且如果 Isochronous_endpoint 位置位，则 DMA 引擎将从 DD 的第 5 个字中读取 Isochronous_packetsize_memory_address。

3. 传输数据

数据在存储单元 DMA_buffer_start_addr 中进行传输。在传输完信息包的末尾后，Present_DMA_count 的值加 1。

同步包的容量存放在存储器中。信息包容量存储器中的每个字均被划分为下面几个字段：Frame_number（位 31～17），Packet_valid（位 16）以及 Packet_length（位 15～0）。对于一个给定的 DD，分配给信息包容量的存储器的空间应该是 DMA_buffer_length 个字，一个字可供一个包进行传输。

（1）OUT 端点

在传输完每帧信息之后，软件将信息包容量写入 Isochronous_packet_size_memory_address 指示的地址单元中，并且 Isochronous_packet_size_memory_address 的值加 4。

（2）IN 端点

在同步信息包容量中，我们只需要使用 Packet_length 字段。在传输每帧信息时，USB 设备将传输一个端点数据包到主机，该数据包的容量在 Packet_length 字段中指定，并在数据包传输结束时将 Isochronous_packetsize_memory_address 的值加 4。如果 Packet_length 为 0，则 USB 设备将发送一个空包。

4. DMA描述符结束

用于同步端点的 DD 只能以 NormalCompletion 的状态代码结束。因为同步端点上没有短包，并且 USB 传输在没有出现系统错误时将继续进行，所以不会出现同步端点的 DataOverrun 检测。

5. 同步OUT端点操作举例

假设将同步端点设置为传输 10 帧，当帧编号为 21 时，传输开始。在传输了 4 个信息包长度分别为 10、15、8 和 20 字节的帧信息之后，描述符和存储器映射如图 11.5所示。

The_total_number_of_bytes_transferred=0x0A+0x0F+0x08+0x14=0x35。

信息包长度存储器中的所有字的 Packet_valid 位（位 16）被设置为 1。

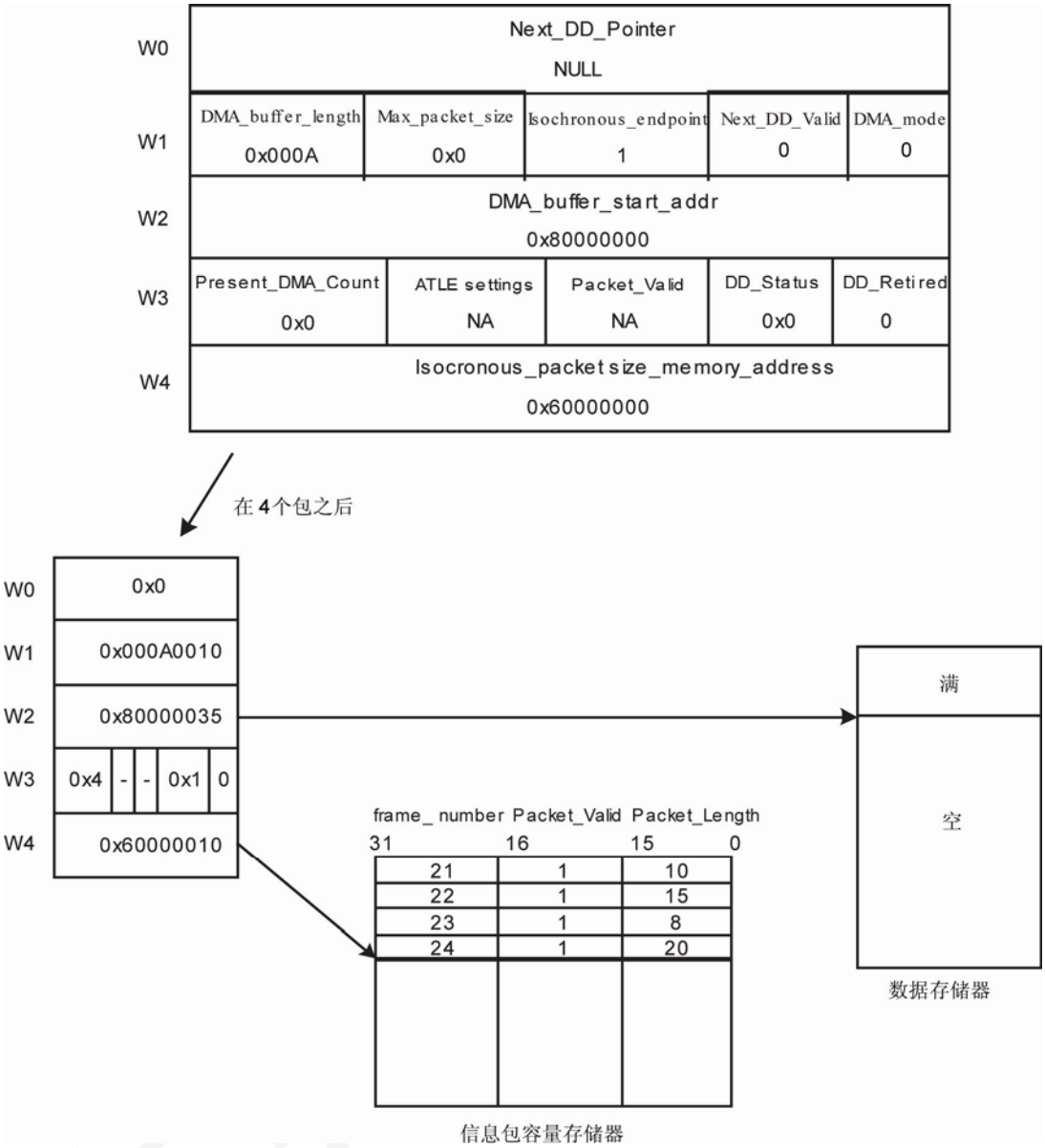


图 11.5 同步 OUT 端点的操作举例

11.15.7 自动长度传输提取（ATLE）模式操作

有些主机驱动程序例如 NDIS（网络驱动程序接口规范）主机驱动程序可以将多个小容量的 USB 传输（delta 传输）连接起来形成一个大容量的 USB 传输。对于 OUT USB 传输，设备硬件必须将这个已连接的传输分解开，返回到原来的多个小容量传输，并将它们传输到独立的 DMA 缓冲区中。通过将 DMA 描述符中的 DMA 模式设置为自动传输长度提取（ATLE）模式来实现上述操作。只有批量端点支持 ATLE 模式。

1. ATLE模式中的OUT传输

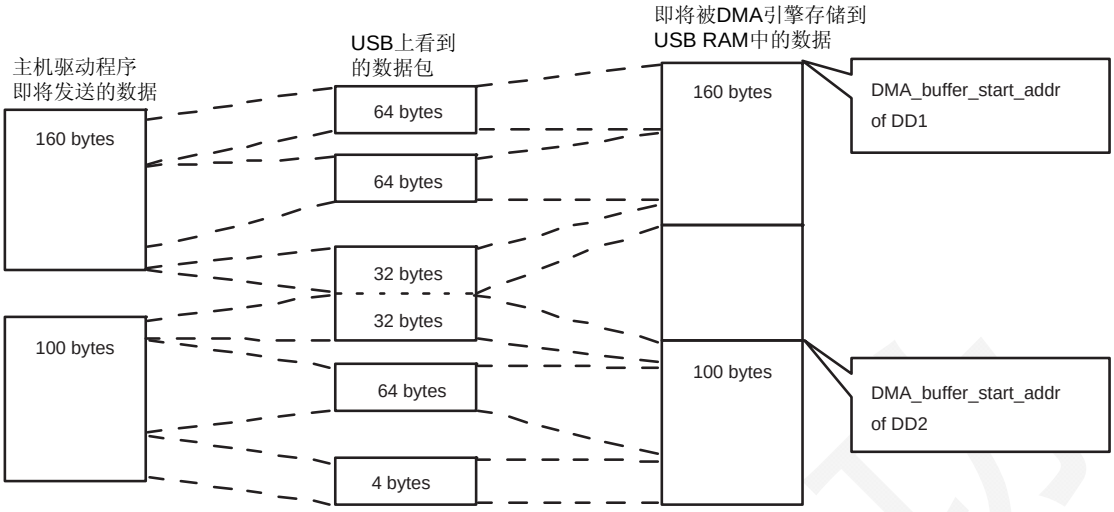


图 11.6 ATLE 模式中的数据传输

图 11.6显示了ATLE模式中的一个典型的OUT USB传输。在图中，主机分别将 160 字节和 100 字节的两个USB传输连接起来。假定MaxPacketSize为 64，则设备硬件将这个USB传输解释为 4 个 64 字节的数据包和一个 4 字节的短包。第 3 和第 4 个数据包被连接起来。注：在正常模式中，该USB传输将被解释为 64、64、32、64 和 36 字节。

现在，DMA 引擎需要将这两个 USB 传输分解开并将它们放在 DMA 描述符 1（DD1）和 DMA 描述符 2（DD2）的 DMA_buffer_start_addr 字段指示的存储单元中。

硬件在 Message_length_position 指定的偏移量（从 USB 传输的起始字节开始）处读取两字节宽的 DMA_buffer_length，并将它写入 DD 的 DMA_buffer_length 字段中。如果 DMA_buffer_length 的两个字节分在两个包中，而为了确保在此情况下提取这两个字节，那么硬件会在提取了各个字节后将 LS_byte_extracted 和 MS_byte_extracted 标志置位。在提取了 MS 字节之后，DMA 传输继续进行，与在正常模式中相同。

在准备一个新的 DD 时，LS_byte_extracted 和 MS_byte_extracted 标志由软件清零。因此，一旦 DD 退出，硬件就必须为下一个 DD 再次提取传输长度。

如果在传输一个连接包（例如图 11.6中的第 3 个包）的过程中DD1 退出，并且没有设置DD2（DD1 中的Next_DD_valid字段为 0），则在DD1 退出时，DD_status被设置为DataOverrun状态代码。这将被看作为一种错误情况，并且USBEpDMASt中对应的EPxx_DMA_ENABLE位由硬件清零。

在 ATLE 模式中，即将传输的最后一个缓冲区长度始终以一个短包或空包结束，表示 USB 传输结束。如果相连的传输长度以 MaxPacketSize 包边界结束，则（NDIS）主机将发送一个空包来标记 USB 传输结束。

2. ATLE模式中的IN传输

对于从设备到主机的 IN USB 传输，DMA_buffer_length 由设备软件设置，与正常模式相同。

在 ATLE 模式中，设备将来自多个 DD 的数据连接起来形成一个 USB 传输。如果在信息包（包容量小于 MaxPacketSize）传输过程中 DD 退出，则提取由 Next_DD_pointer 指示的下一个 DD，并且由剩余的字节所构成的 MaxPacketSize 信息包将从下一个 DD 的缓冲区中传输。

如果没有设置下一个 DD（即 DD 中的 Next_DD_valid 为 0），并且当前 DD 的 DMA 缓冲区长度在到达 MaxPacketSize 包边界之前没有结束，则当前 DD 中的可用字节作为短包在 USB 上

发送，这标志着主机的 USB 传输结束。

如果最后一个缓冲区长度以 MaxPacketSize 包边界结束，则设备软件必须将下一个 DD 的 DMA_buffer_length 字段设置为 0，这样，设备就发送一个空包来标志主机的 USB 传输结束。

3. 设置DMA传输

对于 OUT 端点，主机硬件需对 DD 中的 Message_length_position 字段进行设置。它表示消息长度在输入数据包中的起始位置。而且，设备软件必须将 DMA_buffer_length 字段设置为 0，因为该字段是在提取了缓冲区长度之后由设备硬件进行更新的。

对于 IN 端点，描述符的设置与正常模式相同。

由于一个包可以分在两个 DD 中，因此，软件应始终准备好两个 DD，以短包或空包结束的最后一次 DMA 传输除外。

4. 查找DMA描述符

DMA 描述符的查找操作与正常模式相同。

5. 传输数据

（1）OUT 端点

如果状态字段中的 LS_byte_extracted 和 MS_byte_extracted 位没有置位，则硬件将从数据流中提取传输长度并对 DMA_buffer_length 进行设置。一旦完成提取，LS_byte_extracted 和 MS_byte_extracted 将置位。

（2）IN 端点

IN 端点的 DMA 传输处理与在正常模式下的处理相同，该操作持续进行直到已传输的字节数等于 DMA_buffer_length 字段的值。

6. 结束包传输

DMA 引擎持续与 USB RAM 进行传输，直到传输的字节数已达到 DMA_buffer_length 中指定的值。然后，EOT 中断产生。如果在信息包中间出现这种情况，则加载相连的 DD 并且该包中的剩余字节将针对新 DD 指向的地址进行传输。

（1）OUT 端点

如果链接的 DD 无效，并且包中已有一部分信息传输到了存储器中，则 DD 将以设置 DataOverrun 状态代码结束，并且该端点的 DMA 将被禁止。否则 DD_status 将更新为 NormalCompletion 状态代码。

（2）IN 端点

如果链接的 DD 无效，并且包中已有一部分信息传输到了 USB，则 DD 将以 DD_status 字段的 NormalCompletion 的状态代码结束。这种情况与 USB 传输结束一致，信息包将作为一个短包发送。当链接的 DD 有效并且缓冲区长度为 0 时，将发送一个空包表示 USB 传输结束。

11.16 双缓冲的端点操作

USB 设备控制器的批量端点和同步端点采用双缓冲功能来增加数据吞吐量。

当某个具有双缓冲属性的端点实现时，设备将在 EP_RAM 中自动分配两个端点缓冲区的空间。见“EP RAM 要求”。

在下面的讨论中，当前可由 CPU 或 DMA 引擎进行读和写访问的端点缓冲区被看作是有效缓冲区。

11.16.1 批量端点

对于批量端点，有效的端点缓冲区通过 SIE 清空缓冲区命令或确认缓冲区命令来切换。下面的例子阐述了在从模式中，批量 OUT 端点的双缓冲是如何工作的。

假定缓冲区 1(B_1)和缓冲区 2(B_2)为空，并且有效缓冲区为 B_1。

a）主机发送一个数据包到端点。设备硬件将该数据包放入 B_1，并产生一个端点中断。

b）软件清除端点中断并开始从 B_1 中读取包中的数据。当正在读 B_1 时，主机发送第 2 个数据包，设备硬件将它放入 B_2，并产生一个端点中断。

c）当主机试图发送第 3 个包时软件仍在读 B_1。由于 B_1 和 B_2 都是满的，所以设备硬件以 NAK 响应。

d）软件完成第 1 个包的读取操作，并发送 SIE 清空缓冲区命令将 B_1 释放以接收其它的包。B_2 变为有效缓冲区。

e）软件发送 SIE 选择端点命令来读取选择端点寄存器并测试 FE 位。软件发现有效缓冲区（B_2）中已经有数据（FE=1）。软件清除端点中断并开始读取 B_2 的内容。

f）主机重新发送第 3 个包，设备硬件将它放入 B_1 中。端点中断产生。

g）软件完成第 2 个包的读取操作，并发送 SIE 清空缓冲区命令来释放 B_2 以接收其它的包。B_1 变为有效缓冲区。软件等待下一个端点中断出现（它已经在步骤 6 中产生）。

h）软件将端点中断清零作为对它的响应并开始从 B_1 中读取第 3 个包。

i）软件完成第 3 个包的读取操作，并发送 SIE 清空缓冲区命令来释放 B_1 以接收其它包。B_2 变为有效缓冲区。

j）软件测试 FE 位，并发现有效缓冲区（B_2）为空（FE=0）。

k）B_1 和 B_2 都为空。软件等待下一个端点中断的发生。现在有效缓冲区为 B_2。主机发送的下一个数据包将放在 B_2 中。

下面的例子阐述了在从模式中，批量 IN 端点的双缓冲是如何工作的。

假定缓冲区 1（B_1）和缓冲区 2（B_2）都为空，并且有效缓冲区为 B_1。NAK 特性的中断被使能。

a）主机通过发送 IN 令牌包来请求一个数据包。设备以 NAK 作为应答并产生一个端点中断。

b）软件清除端点中断。设备有 3 个包要发送。软件将第 1 个包填入 B_1 并发送 SIE 确认缓冲区命令。有效缓冲区切换为 B_2。

c）软件发送 SIE 选择端点命令来读取选择端点寄存器并测试 FE 位。软件发现 B_2 为空（FE=0），它用第 2 个数据包来填充 B_2。软件发送 SIE 确认缓冲区命令，有效缓冲区切换为 B_1。

d）软件等待端点中断的产生。

e）设备成功地发送 B_1 中的包并清空缓冲区。端点中断产生。

f）软件清除端点中断。软件将第 3 个包填充到 B_1 中并使用 SIE 确认缓冲区命令来激活它。有效缓冲区切换为 B_2。

g）设备成功地发送了 B_2 中的第 2 包并产生一个端点中断。

h）软件没有其它包要发送，因此只简单地将中断清除。

i）设备成功地发送了 B_1 中的第 3 个包并产生一个端点中断。

j）软件没有其它包要发送，因此只简单地将中断清除。

k）B_1 和 B_2 均为空，有效缓冲区为 B_2。下一个由软件写入的包将进入 B_2。

在 DMA 模式中，有效缓冲区的切换自动在硬件中处理。对于批量 IN 端点，通过使用 USBDMARSet 寄存器手动地开始一个包的传输，可以提前对一个端点缓冲区进行填充，从而充分利用双缓冲的功能。

11.16.2 同步端点

对于同步端点，有效数据缓冲区在出现 FRAME 中断时由硬件切换。SIE 清空缓冲区命令和确认缓冲区命令不会引起有效缓冲区的切换。

当正在总线上发送或接收另一个缓冲区中的数据包时，双缓冲允许软件充分利用帧间隔将一个包写入有效缓冲区或从有效缓冲区中读取一个包。

对于 OUT 同步端点，任何没有在帧结束之前从有效缓冲区中读出的数据都将在有效缓冲区切换时丢失。

对于 IN 同步端点，如果没有在帧结束之前对有效缓冲区进行确认，则当有效缓冲区切换时将在总线上发送一个空包，并且当该缓冲区再次变为有效时，它的内容将被覆盖。