关于 STM32F102/103 的 USB 模块和 USB 库函数

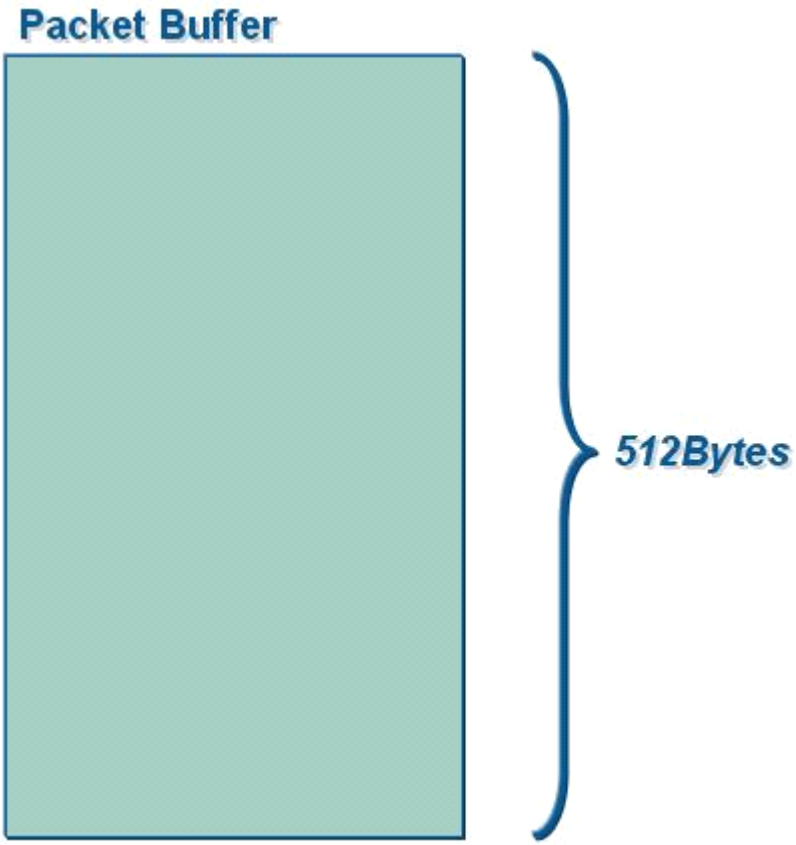
(By　vigia)

今天有空，开贴讲讲，怎样配合 ST 提供的库函数理解 STM32F102/103 的 **USB** 模块，以及怎么调用这些库函数来实现基本的 **USB** 通信。

题目很大，先只讲讲最简单的应用。

1 关于 512 字节的 Packet Buffer

在 STM32F103 的 **USB** 模块中有一个 RAM 区，称为 Packet Buffer，共有 512 字节。



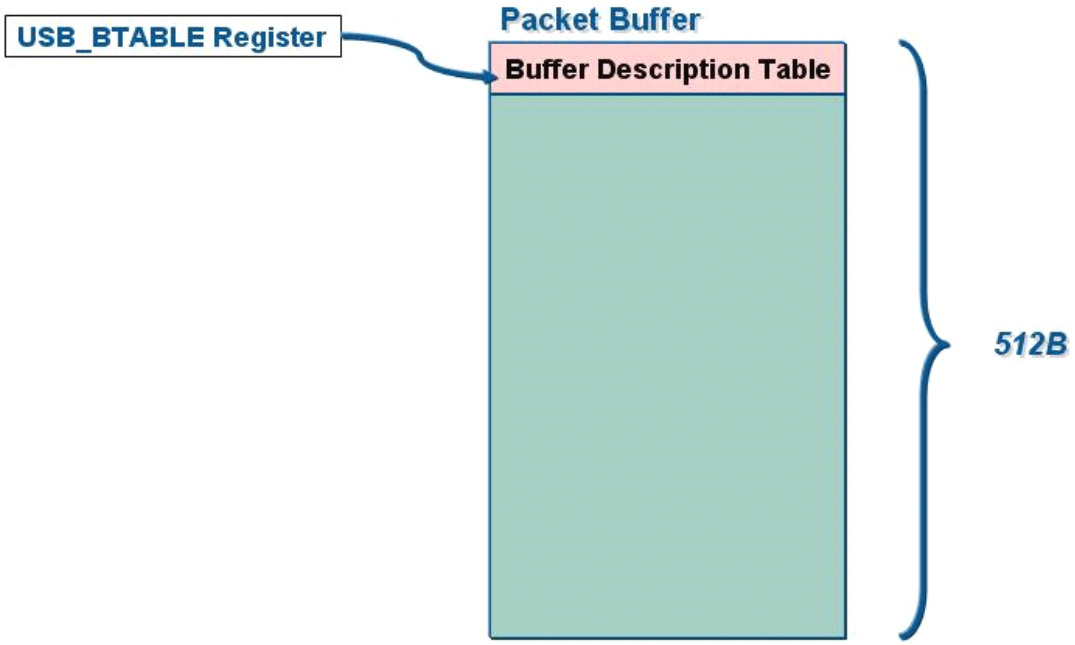
USB 模块中有个 Buffer Description Table，这个 Table 位于 512 字节的 Packet Buffer 中，可以在 Packet Buffer 的任意位置。

USB 模块提供一个寄存器 **USB**_BTABLE 来设置 Buffer Description Table 在 Packet

Buffer 的偏移地址。

在库函数中，Define 了这个偏移地址：
usb_conf.h：
#define BTABLE_ADDRESS　(0x00)

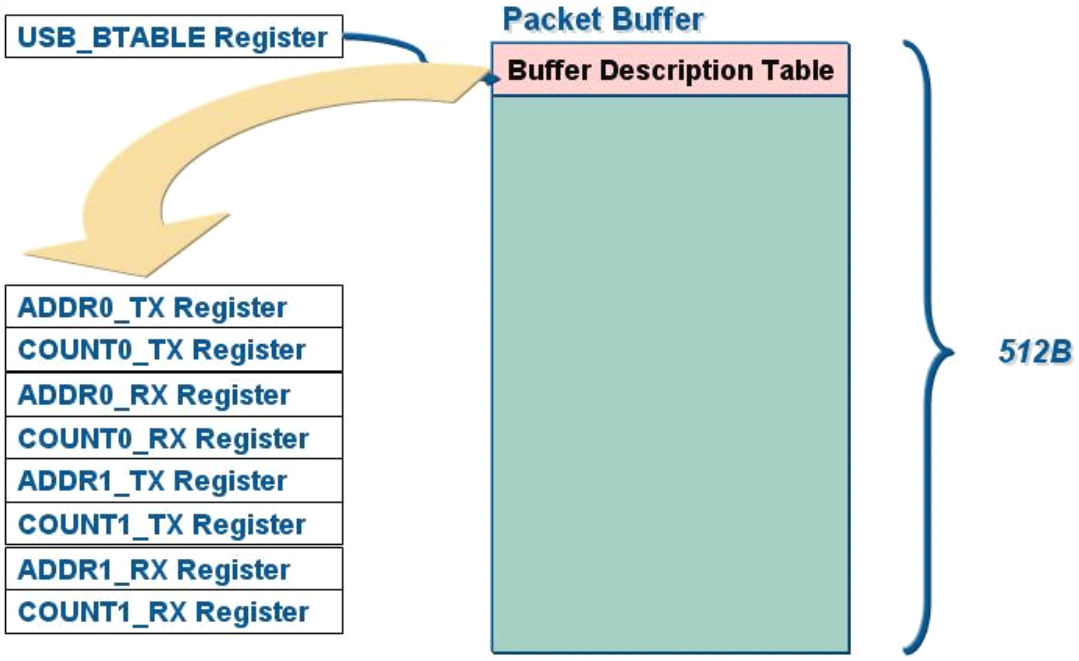
这意味着 Buffer Description Table 位于 Packet Buffer 的首地址。



在 Buffer Description Table 中的，是所用到的端点的缓存区地址寄存器和缓存区长度寄存器。所有用到的端点的这两个寄存器都位于这个 Table 中。

如上所说，由于这个 Table 位于 Packet Buffer 的首地址。所以端点 0 的发送缓冲区地址寄存器就位于 Packet Buffer 的首地址，紧接的是端点 0 发送缓冲区长度寄存器，接着的是端点 0 接收缓存区的地址寄存器，跟着是端点 0 的接收缓存区的长度寄存器，等等等等，一直到最后一个端点 8 的接收缓存区的长度寄存器。

每个端点的一个方向有 2 个寄存器，共 8 个端点 16 个方向，一共 32 个寄存器，每个寄存器为 4 个字节，所以这个 Table 一共占有 128 字节。



在端点 0 发送缓存区的地址寄存器中的值，是端点 0 发送缓存区在 Packet Buffer 中的偏移地址。而端点 0 接收缓存区的地址寄存器中的值，是端点 0 接收缓存区在 Packet Buffer 中的偏移地址。

如前所说，Buffer Description Table 从理论上占有 128 个字节。但对于具体的应用，不是每个应用都会用到 8 个端点的 16 个方向的。所以，对于那些没有用到的端点寄存器，我们可以不考虑为他们预留位置。

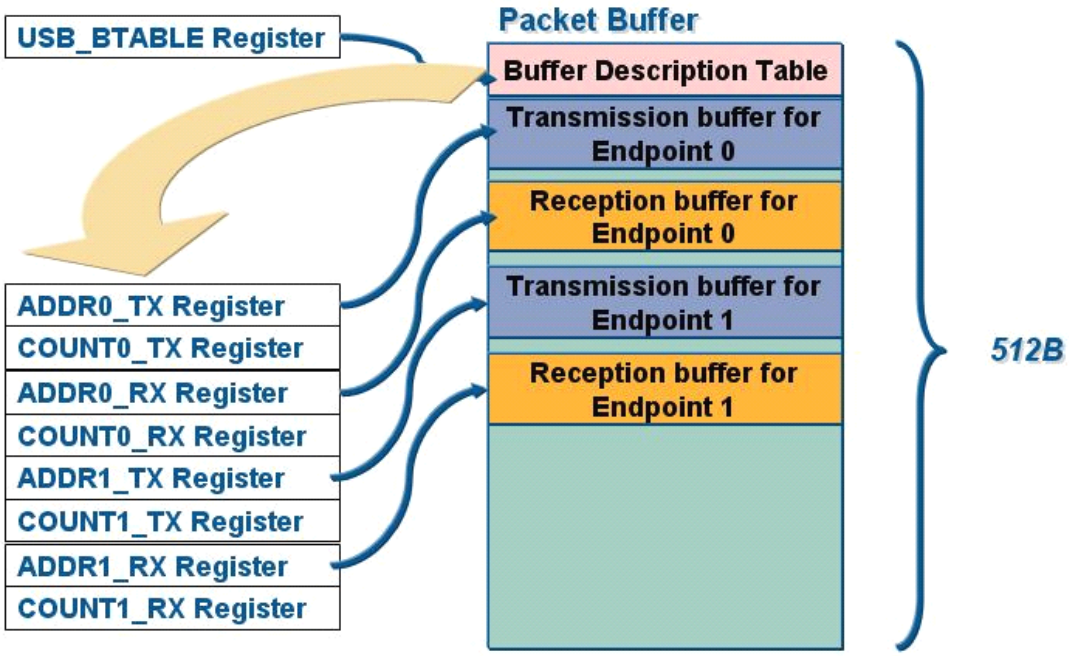
在 ST 提供的例程中，通常这么定义：

```
#define BTABLE_ADDRESS      (0x00)

/* EP0 */
/* rx/tx buffer base address */
#define ENDP0_RXADDR    (0x18)
#define ENDP0_TXADDR    (0x58)
```

这 3 句定义，意味着：

1， 端点 0 的接收缓存区位于 Packet Buffer 的 0x18 地址。
2， 端点 0 的发送缓存区位于 Packet Buffer 的 0x58 地址。
3， Buffer Description Table 位于 Packet Buffer 的前 24 字节。24 个字节意味着应用需要使用 6 个寄存器，即 3 个端点。
4， 端点 0 的接收缓存区长度为 64 字节。



好了，关于这个 Packet Buffer 讲解完毕。

要做一个 **USB** 应用，第一步就是要根据应用合理的分配好这个 Packet Buffer。

出个题目给大家做做

假设，需要使用端点 0 的 IN,OUT 传输，端点长度为 8 字节，端点 1 的 IN 传输，长度为 16 字节。端点 2 的 OUT 传输，长度为 64 字节。端点 2 的 IN 传输，长度为 64 字节。

该怎么分配这个 Packet Buffer？

2 使用 STM32F102/103 **USB** 函数库 进行 **USB** 通信

第一步：
根据应用的需求，定义使用到的端点数量

usb_conf.h

```
#define EP_NUM (3)
```

以上意味着应用需要使用到 EP0，EP1 和 EP2

第二步：
初始化每个使用到的端点

usb_prop.c

SetEPType(ENDP2, EP_INTERRUPT);
定义端点 2 为中断端点

SetEPTxAddr(ENDP2, ENDP2_TXADDR);
如果需要进行 EP2 IN 通信，需要定义端点 2 的发送缓存区的地址，也就是在 Packet Buffer 中的偏移地址

SetEPRxAddr(ENDP2, ENDP2_RXADDR);
如果需要进行 EP2 OUT 通信，需要定义端点 2 的接收缓存区在 Packet Buffer 中的偏移地址

SetEPRxStatus(ENDP2, EP_RX_NAK);
设置端点 2 的接收状态为 NAK，设备将以 NAK 来响应主机发起的所有 OUT 通信。

SetEPTxStatus(ENDP2, EP_TX_NAK);
设置端点 2 的发送状态为 NAK，设备将以 NAK 来响应主机发起的所有 IN 通信。

第三步：
使能端点的通信

对于 IN 端点的使能：
UserToPMABufferCopy(Send_Buffer, ENDP2_TXADDR, 8);
拷贝用户数据到端点 2 的发送缓存区

SetEPTxCount(ENDP2, 8);
设置端点 2 发送数据长度

SetEPTxValid(ENDP2);
设置端点 2 的发送状态为 VALID

以上三句可以在应用代码的任意位置调用，一旦调用，即使能了一次 **USB** IN 通信。

USB 设备将在收到主机的 IN TOKEN 后，自动发送缓存区中的数据到主机，并在发送完毕后产生 EP2_IN_Callback 中断，同时将端点 2 的发送状态自动改为 NAK。

如果需要再次进行数据传送，需要再次调用以上的三句函数。
对于 OUT 端点的使能：

SetEPRxValid(ENDP2);
设置端点 2 的接收状态为 VALID。

以上的这句函数即使能了端点 2 的 OUT 通信，可以在任意位置调用。

一旦调用，即使能了一次 OUT 通信。**USB** 设备将以 ACK 来响应主机随后的 OUT 通信，并在接收数据完毕后，产生 EP2_OUT_Callback 中断，同时自动将端点的接收状态改为 NAK。

在 EP2_OUT_Callback 中断函数中调用
USB_SIL_Read(EP2_OUT, Receive_Buffer);
可以将端点 2 接收缓存区中收到的数据拷贝到用户数据区。

作者: vigia

整理:　CANSTAR