文档编号：AN2021

上海东软载波微电子有限公司

应用笔记

ES32F36xx USB 例程

修订历史

版本	修订日期	修改概要
V1.0	2019-09-16	初版
V1.1	2020-09-29	添加 USB 模块推荐电路

地　址：中国上海市龙漕路 299 号天华信息科技园 2A 楼 5 层
邮　编：200235
E-mail：support@essemi.com
电　话：+86-21-60910333
传　真：+86-21-60914991
网　址：http://www.essemi.com/

目 录

内容目录

第1章 概述

1.1 简介

本文档介绍 ES32F36xx 系列芯片 USB 例程，包括从机模式例程、主机模式例程、OTG 例程以及高速 USB 眼图测试例程。

1.2 USB模块引脚说明

1. USB_DM/USB_DP：USB 差分数据线，对应高速 USB 板级走线注意防护；
2. VDD33_USB：USB 模块电源，需连接到 3.3V 电源上；
3. USB_REXT：串一个 1%精度 12.7KΩ电阻到地，高速 USB 专用，低速/全速无此引脚；
4. USB_ID：输入端口，高电平时 USB 模块工作在 Device 模式[默认]，低电平时 USB 模块工作在 Host 模式。采用 Micro-AB 接口时，Device 模式可以直接使用，Host 模式需连接 OTG 转接头，在 OTG 转接头中会将 USB_ID 引脚拉低，让 USB 控制器工作在 Host 模式。
5. USB_VBUS：接到 5V 电源，若此引脚电压低于 4.75V，USB 控制器可能会停止工作。

1.3 example例程

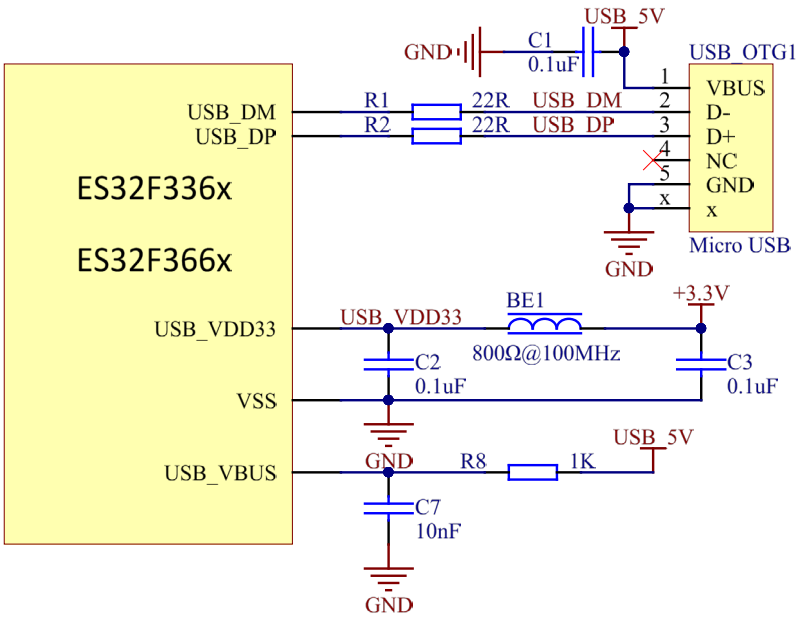
本例程为工程实例，USB 模块其他工程均与此工程保持一致。该工程还可以在 PA15 管脚上输出 USB 模块的时钟。时钟频率固定为 60MHz 的 256 分频，即 234.375KHz。可以在板级调试中使用，用来确认 USB 模块时钟是否正确。

工程路径：

~\ES32_SDK\Projects\ES32F36xx\Applications\USB\00_example\

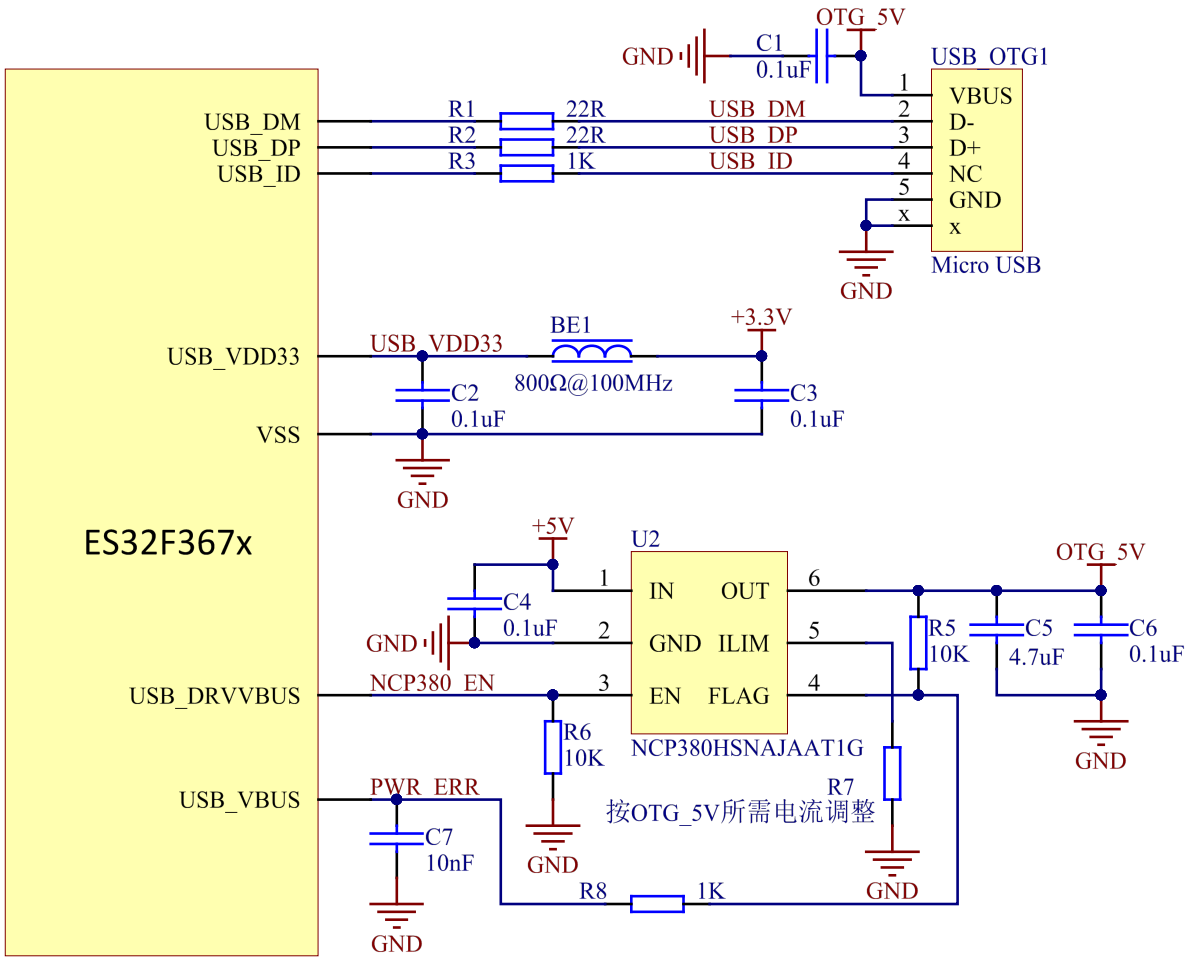
1.4 推荐电路

全速 USB 设备接口推荐电路如下：



注 1：C2、C7 尽量靠近芯片引脚。

全速 USB OTG 接口推荐电路如下：



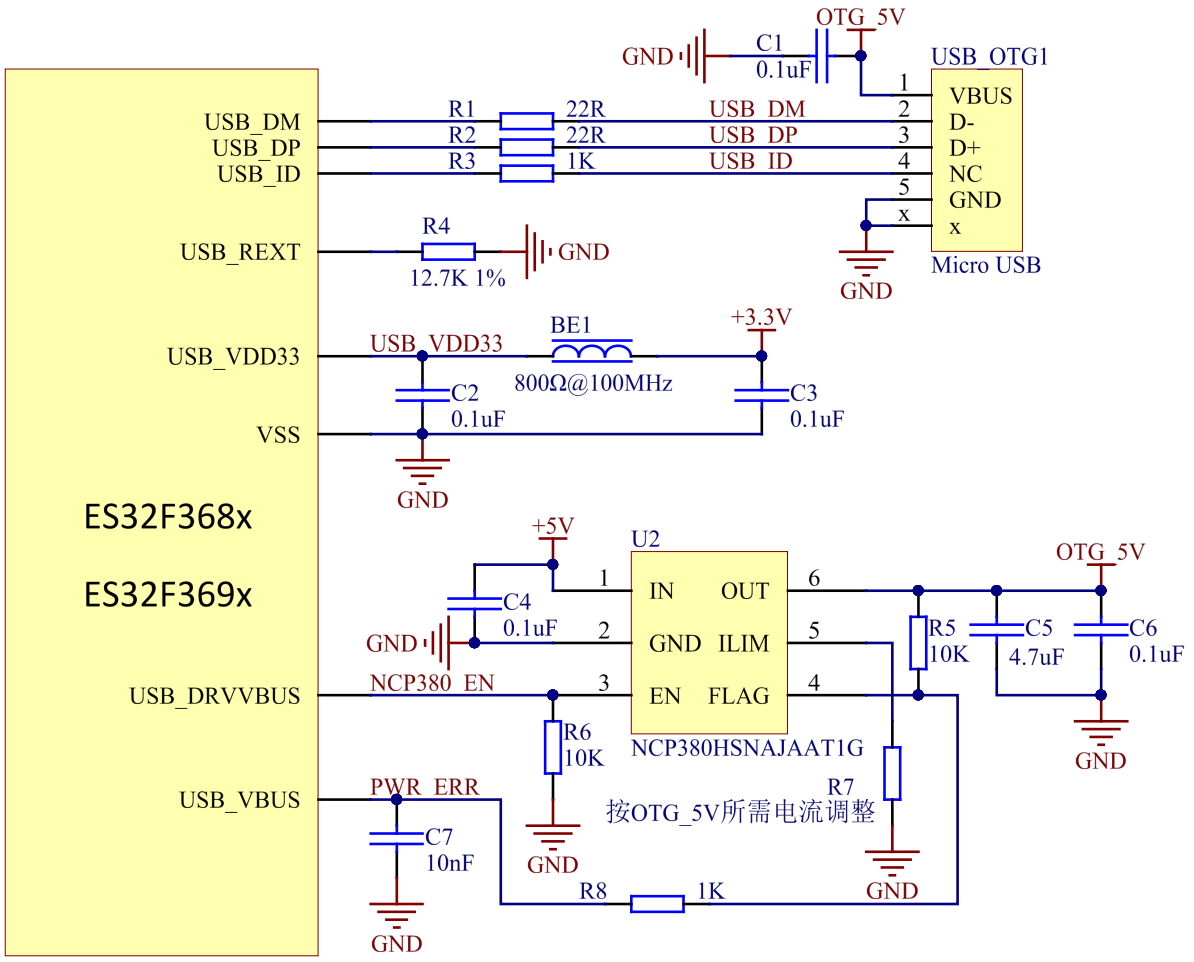
注 1：C2、C7 尽量靠近芯片引脚；

注 2：R7 电阻的阻值可根据 OTG_5V 所需的电流大小来确定，一般选 10K 左右；

注 3：NCP380 可以更换成功能类似的芯片；

注 4：USB_VBUS 是 OTG_5V 电源监测口，当存在异常时会停止数据传输。

高速 USB OTG 接口推荐电路如下：



注 1：C2、C7 尽量靠近芯片引脚；

注 2：R7 电阻的阻值可根据 OTG_5V 所需的电流大小来确定，一般选 10K 左右；

注 3：NCP380 可以更换成功能类似的芯片；

注 4：USB_VBUS 是 OTG_5V 电源监测口，当存在异常时会停止数据传输；

注 5：R4/12.7K 的精度要求 1%以内。

1.5 章节安排

第 1 章为概述部分；

第 2-10 章为单功能设备例程；

第 11-15 章为复合设备例程；

第 16-19 章为主机模式例程；

第 20 章为主机模块外扩 HUB 例程；

第 21 章为 OTG 模式例程；

第 22 章为高速模式眼图测试例程；

第2章 U盘设备(MSC)

2.1 功能概述

本例程为全速从机 U 盘设备(MSC)，将 MCU 虚拟成一个 80KBytes 的 U 盘。使用 USB-Micro 数据线将 MCU 连接到 PC 上后，在 PC 端可操作这个 80KBytes 的 U 盘，包括格式化、写入、读出等标准 U 盘操作。

2.2 主要API

2.2.1 MSC初始化函数

类型	描述
函数原型	void *usbd_msc_init(uint32_t idx, usbd_msc_dev_t *dev);
功能描述	将 MCU 初始化为一个 MSC 设备
输入参数	idx: USB 控制器的索引，总是为 0
	dev: MSC 设备结构体指针
返回值	成功：MSC 实例的指针
	失败：NULL
使用示例	usbd_msc_init(0, &msc_device);

表 2-1 MSC 初始化函数

2.2.2 USB事件回调函数

类型	描述
函数原型	uint32_t msc_event_handle(void *data, uint32_t event, uint32_t value, void *p_data);
功能描述	USB 协议栈向应用层通知相关事件
输入参数	data: 用户层参数，一般为 usbd_msc_init()函数返回值
	event: 事件类型
	value: 该事件携带的数值
	p_data:该事件携带的消息
返回值	成功：0
	失败：其他值
使用示例	应用层根据需要填充该函数

表 2-2 USB 事件回调函数

2.3 操作方法及演示效果

1. 工程路径：~\ES32_SDK\Projects\ES32F36xx\Applications\USB\01_dev_msc\
2. 编译工程，并将其下载到板子上；
3. 使用 Micro-USB 线将板子连接到 PC 上；
4. 若是第一次连接，PC 会自动安装相应驱动。驱动安装完成后 PC 会弹出格式化界面；

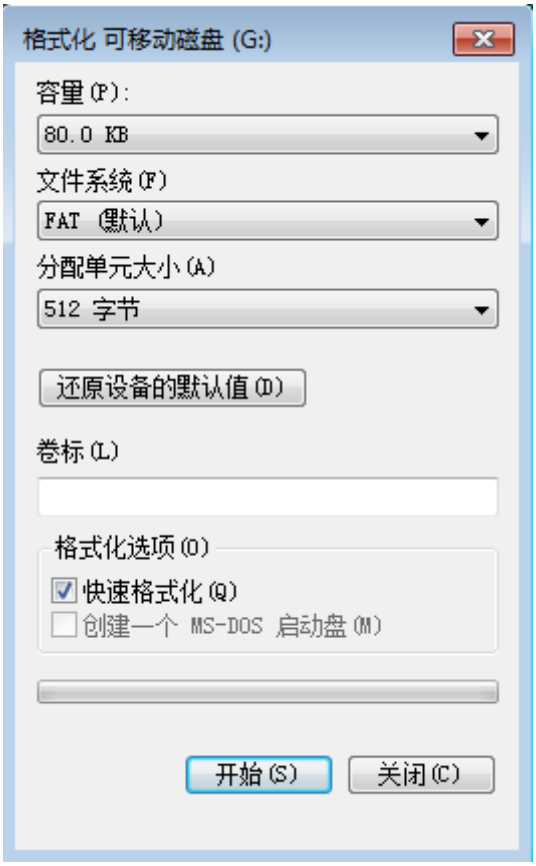


图 2-1 PC 端格式化界面

5. 格式化后会在“我的电脑”上显示对应盘符；

图 2-2 PC 端盘符界面

6. 此后可以像通用 U 盘一样使用；

2.4 注意事项

1. 由于例程使用的是将 SRAM 虚拟成 U 盘，所以断电后存在虚拟 U 盘的内容会丢失。

第3章　高速U盘设备(MSC)

3.1　功能概述

本例程为高速从机 U 盘设备(MSC)，将 MCU 虚拟成一个 80KBytes 的 U 盘。使用 USB-Micro 数据线将 MCU 连接到 PC 上后，在 PC 端可操作这个 80KBytes 的 U 盘，包括格式化、写入、读出等标准 U 盘操作。

本例程仅支持在具有高速 USB 接口的 MCU 上使用。

3.2　主要API

同 2.2 章节。

3.3　操作方法及演示效果

1. 工程路径：~\ES32_SDK\Projects\ES32F36xx\Applications\USB\02_dev_msc_hs\

下同 2.3 章节。

3.4　注意事项

同 2.4 章节。

第4章 鼠标设备（Mouse）

4. 1 功能概述

本例程为全速从机鼠标设备(Mouse)，将 MCU 虚拟成一个鼠标。使用 USB-Micro 数据线将 MCU 连接到 PC 上后，可以控制 PC 端鼠标指针的移动，鼠标左键、右键的点击事件。

本例程不依赖板级按键，采用延时控制。鼠标指针先向右下移动一段距离再向左上移动一段距离，以及鼠标左键、右键、中键的按下。

4. 2 主要API

4. 2. 1 Mouse初始化函数

类型	描述
函数原型	void *usbd_hid_mouse_init(uint32_t idx, usbd_hid_mouse_dev_t *dev);
功能描述	将 MCU 初始化为一个 Mouse 设备
输入参数	idx: USB 控制器的索引，总是为 0
	dev: Mouse 设备结构体指针
返回值	成功：Mouse 实例的指针
	失败：NULL
使用示例	usbd_hid_mouse_init(0, &mouse_device);

表 4-1 Mouse 初始化函数

4. 2. 2 USB事件回调函数

类型	描述
函数原型	uint32_t mouse_handle(void *data, uint32_t event, uint32_t value, void *p_data);
功能描述	USB 协议栈向应用层通知相关事件
输入参数	data: 用户层参数，一般为 usbd_hid_mouse_init()函数返回值
	event: 事件类型
	value: 该事件携带的数值
	p_data:该事件携带的消息
返回值	成功：0
	失败：其他值
使用示例	应用层根据需要填充该函数

表 4-2 USB 事件回调函数

4. 2. 3 Mouse状态改变函数

类型	描述
函数原型	uint32_t usbd_hid_mouse_state_change(void *device, int8_t x, int8_t y, uint8_t button);
功能描述	该函数将 Mouse 状态的改变发送给 USB 主机
输入参数	device:一般为 usbd_hid_mouse_init()函数返回值
	x: X 轴坐标变化量[-127, 127]
	y: Y 轴坐标变化量[-127, 127]
	button:按键状态，Bit-Mapping 格式
返回值	成功：0
	失败：其他值
使用示例	usbd_hid_mouse_state_change(&mouse_device, -5, -5, 0x1);

表 4-3 Mouse 状态改变函数

4. 3 操作方法及演示效果

1. 工程路径：
   ~\ES32_SDK\Projects\ES32F36xx\Applications\USB\03_dev_hid_mouse\
2. 编译工程，并将其下载到板子上；
3. 使用 Micro-USB 线将板子连接到 PC 上；
4. 若是第一次连接，PC 会自动安装相应驱动；
5. 之后可以在 PC 上观察鼠标指针的移动现象；

4. 4 注意事项

1. 本例程不依赖板级按键，采用延时控制。

第5章 键盘设备(Keyboard)

5.1 功能概述

本例程为全速从机键盘设备(Keyboard)，将 MCU 虚拟成一个键盘。使用 USB-Micro 数据线将 MCU 连接到 PC 上后，可以做为 PC 端的键盘使用。

本例程不依赖板级按键，采用延时控制。虚拟键盘会向 PC 端不断的输出“CAPSLOCK”按键、“T”按键。

5.2 主要API

5.2.1 Keyboard初始化函数

类型	描述
函数原型	void *usbd_hid_keyb_init(uint32_t idx, usbd_hid_keyb_dev_t *dev);
功能描述	将 MCU 初始化为一个 Keyboard 设备
输入参数	idx: USB 控制器的索引，总是为 0
	dev: Keyboard 设备结构体指针
返回值	成功：Keyboard 实例的指针
	失败：NULL
使用示例	usbd_hid_keyb_init(0, &keyb_device);

表 5-1 Keyboard 初始化函数

5.2.2 USB事件回调函数

类型	描述
函数原型	uint32_t keyb_handle(void *data, uint32_t event, uint32_t value, void *p_data);
功能描述	USB 协议栈向应用层通知相关事件
输入参数	data: 用户层参数，一般为 usbd_hid_keyb_init()函数返回值
	event: 事件类型
	value: 该事件携带的数值
	p_data:该事件携带的消息
返回值	成功：0
	失败：其他值
使用示例	应用层根据需要填充该函数

表 5-2 USB 事件回调函数

5. 2. 3 Keyboard状态改变函数

类型	描述
函数原型	uint32_t usbd_hid_keyb_state_change(void *device, uint8_t modify, uint8_t usage, uint8_t press);
功能描述	该函数将 Keyboard 状态的改变发送给 USB 主机
输入参数	device:一般为 usbd_hid_keyb_init()函数返回值
	modify: 功能按键代码
	usage: 普通按键代码
	press:按键状态，1：按下，0：释放
返回值	成功：0
	失败：其他值
使用示例	usbd_hid_keyb_state_change(&keyb_device, 0, HID_KEYB_USAGE_T, 1);

表 5-3 Keyboard 状态改变函数

5. 3 操作方法及演示效果

1. 工程路径：
   ~\ES32_SDK\Projects\ES32F36xx\Applications\USB\04_dev_hid_key\
2. 编译工程，并将其下载到板子上；
3. 使用 Micro-USB 线将板子连接到 PC 上；
4. 若是第一次连接，PC 会自动安装相应驱动；
5. 之后可以在 PC 端打开一个 txt 文件，并将光标定位到 txt 文件内，观察虚拟键盘的输入；

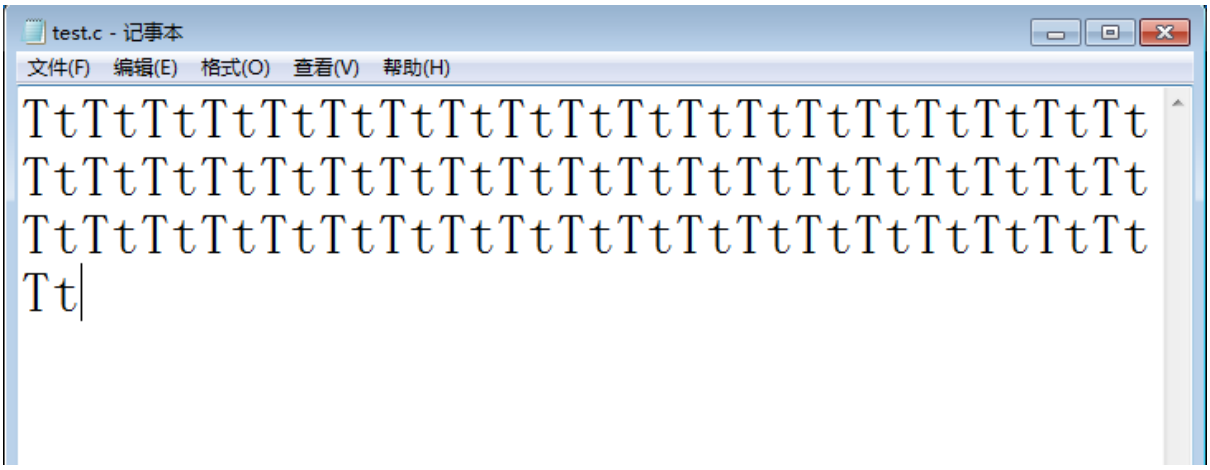


图 5-1 虚拟键盘输出到 PC 端的字符

5. 4 注意事项

1. 本例程不依赖板级按键，采用延时控制。

第6章 音频设备(Audio)

6. 1 功能概述

本例程为全速从机音频设备(Audio)，将 MCU 虚拟成一个音频设备。使用 USB-Micro 数据线将 MCU 连接到 PC 上后，可以做为 PC 端的音频设备使用。

6. 2 主要API

6. 2. 1 Audio初始化函数

类型	描述
函数原型	void *usbd_audio_init(uint32_t idx, usbd_audio_dev_t *dev);
功能描述	将 MCU 初始化为一个音频设备
输入参数	idx: USB 控制器的索引，总是为 0
	dev: Audio 设备结构体指针
返回值	成功：Audio 实例的指针
	失败：NULL
使用示例	usbd_audio_init(0, &audio_device);

表 6-1 Audio 初始化函数

6. 2. 2 USB事件回调函数

类型	描述
函数原型	uint32_t audio_handle(void *data, uint32_t event, uint32_t value, void *p_data);
功能描述	USB 协议栈向应用层通知相关事件
输入参数	data: 用户层参数，一般为 usbd_audio_init()函数返回值
	event: 事件类型
	value: 该事件携带的数值
	p_data:该事件携带的消息
返回值	成功：0
	失败：其他值
使用示例	应用层根据需要填充该函数

表 6-2 USB 事件回调函数

6.2.3 准备接收音频数据函数

类型	描述
函数原型	int32_t usbd_audio_data_out(usbd_audio_dev_t *dev, void *data, uint32_t size, usb_audio_cbk cbk);
功能描述	从 USB 主机获取音频数据，并放入指定 buffer 中
输入参数	dev:一般为 usbd_audio_init()函数返回值
	data: Buffer 地址
	size: Buffer 长度
	cbk:USB 协议栈接收到数据后的回调函数，应用层在此实现音频播放功能
返回值	成功：0
	失败：其他值
使用示例	usbd_audio_data_out(&audio_device, rx_buf, 192, data_handle);

表 6-3 准备接收音频数据函数

6.3 操作方法及演示效果

1. 工程路径：
   ~\ES32_SDK\Projects\ES32F36xx\Applications\USB\05_dev_audio\
2. 编译工程，并将其下载到板子上；
3. 使用 Micro-USB 线将板子连接到 PC 上；
4. 另外将板子的串口连接到 PC，在 PC 上打开串口终端，查看音频数据的接收情况；
5. 若是第一次连接，PC 会自动安装相应驱动，之后可在设备管理器中看到对应设备；

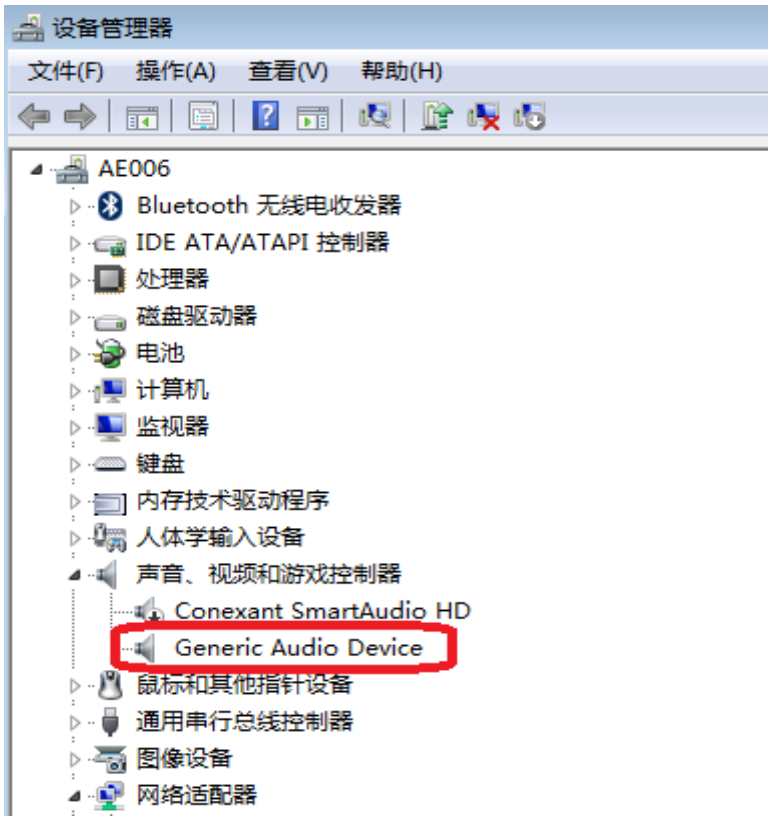


图 6-1 PC 端设备管理器中的虚拟音频设备

6. 在 PC 端播放音频文件，在 PC 的串口终端上查看音频数据的接收情况；

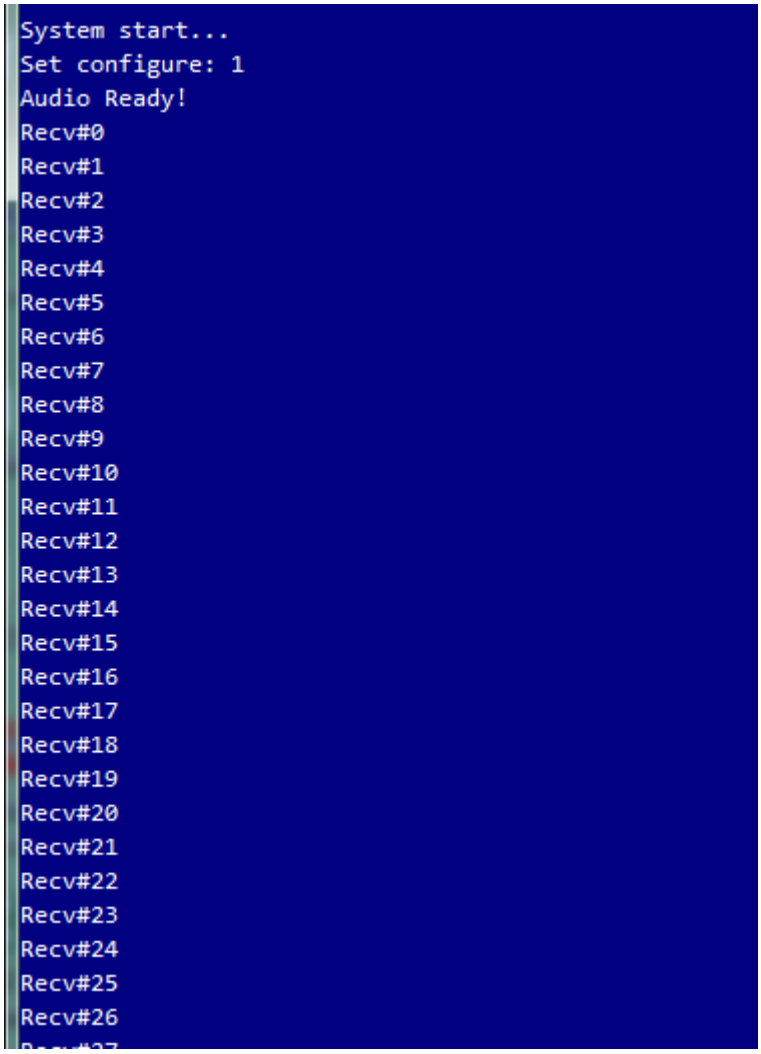


图 6-2 MCU 端接收到音频数据

6.4 注意事项

1. 若 PC 端已经有其他音频设备，在演示此例程时，需将其他音频设备暂时禁用。否则音频数据可能会发往其他音频设备上。

第7章 块设备(BULK)

7. 1 功能概述

本例程为全速从机块设备(BULK)。USB 批量传输适合传输大量数据，并且保证数据的正确性。本例程重点在于演示底层数据收发通道。用户在此基础上可以自行开发各类 USB 设备。

本例程的功能是数据回传，即将从主机接收到的数据回传给主机。

7. 2 主要API

7. 2. 1 BULK初始化函数

类型	描述
函数原型	void *usbd_bulk_init(uint32_t idx, usbd_bulk_dev_t *dev);
功能描述	将 MCU 初始化为一个 BULK 设备
输入参数	idx: USB 控制器的索引，总是为 0
	dev: BULK 设备结构体指针
返回值	成功：BULK 实例的指针
	失败：NULL
使用示例	usbd_bulk_init(0, &bulk_device);

表 7-1 BULK 初始化函数

7. 2. 2 USB事件回调RX函数

类型	描述
函数原型	uint32_t rx_handle(void *data, uint32_t event, uint32_t value, void *p_data);
功能描述	USB 协议栈向应用层通知相关事件
输入参数	data: 用户层参数，一般为 usbd_bulk_init()函数返回值
	event: 事件类型
	value: 该事件携带的数值
	p_data:该事件携带的消息
返回值	成功：0
	失败：其他值
使用示例	应用层根据需要填充该函数

表 7-2 USB 事件回调 RX 函数

7. 2. 3 USB事件回调TX函数

类型	描述
函数原型	uint32_t tx_handle(void *data, uint32_t event, uint32_t value, void *p_data);
功能描述	USB 协议栈通知应用层数据已发送完成
输入参数	data: 一般为 usbd_bulk_init()函数返回值
	event: 事件类型
	value: 该事件携带的数值
	p_data: 该事件携带的消息
返回值	成功：0
	失败：其他值
使用示例	应用层根据需要填充该函数

表 7-3 USB 事件回调 TX 函数

7. 2. 4 获取USB控制器接收到的数据个数函数

类型	描述
函数原型	uint32_t usbd_bulk_rx_packet_avail(usbd_bulk_dev_t *dev);
功能描述	获取 USB 控制器接收到的数据个数(Bytes)
输入参数	dev: 一般为 usbd_bulk_init()函数返回值
返回值	USB 控制器接收到的数据个数(Bytes)
使用示例	rx_nr = usbd_bulk_rx_packet_avail(&bulk_device);

表 7-4 获取 USB 控制器接收到的数据个数函数

7. 2. 5 从USB控制器读取数据函数

类型	描述
函数原型	uint32_t usbd_bulk_packet_read(usbd_bulk_dev_t *dev, uint8_t *data, uint32_t len, uint8_t last);
功能描述	从 USB 控制器中读取数据，该数据来自 USB 主机
输入参数	dev: 一般为 usbd_bulk_init()函数返回值
	data: 读取数据的存放地址
	len: Buffer 的大小
	last: 无关参数，忽略
返回值	读取到的数据个数(Bytes)
使用示例	rx_nr = usbd_bulk_packet_read(&bulk_device, rx_buf, rx_nr, 1);

表 7-5 从 USB 控制器中读取数据函数

7. 2. 6 向USB控制器写数据函数

类型	描述
函数原型	uint32_t usbd_bulk_packet_write(usbd_bulk_dev_t *dev, uint8_t *data, uint32_t len, uint8_t last);
功能描述	向 USB 控制器写数据函数，该数据将会被发送给 USB 主机
输入参数	dev: 一般为 usbd_bulk_init()函数返回值
	data: 待写入数据的存放地址
	len: Buffer 的大小
	last: 是否为最后一帧
返回值	写入数据的个数(Bytes)
使用示例	usbd_bulk_packet_write(&bulk_device, rx_buf, rx_nr, 1);

表 7-6 向 USB 控制器写数据函数

7. 3 操作方法及演示效果

1. 工程路径：
   ~\ES32_SDK\Projects\ES32F36xx\Applications\USB\06_dev_bulk\
2. 编译工程，并将其下载到板子上；
3. 使用 Micro-USB 线将板子连接到 PC 上；
4. 若是第一次连接，需要手动安装驱动，驱动安装方法参考注意事项部分；
5. 驱动安装成功后在 PC 端的设备管理器中能看到相应设备；
6. 在 PC 端打开 usb_bulk_example.exe 程序，若和 MCU 建立连接后，界面如下：

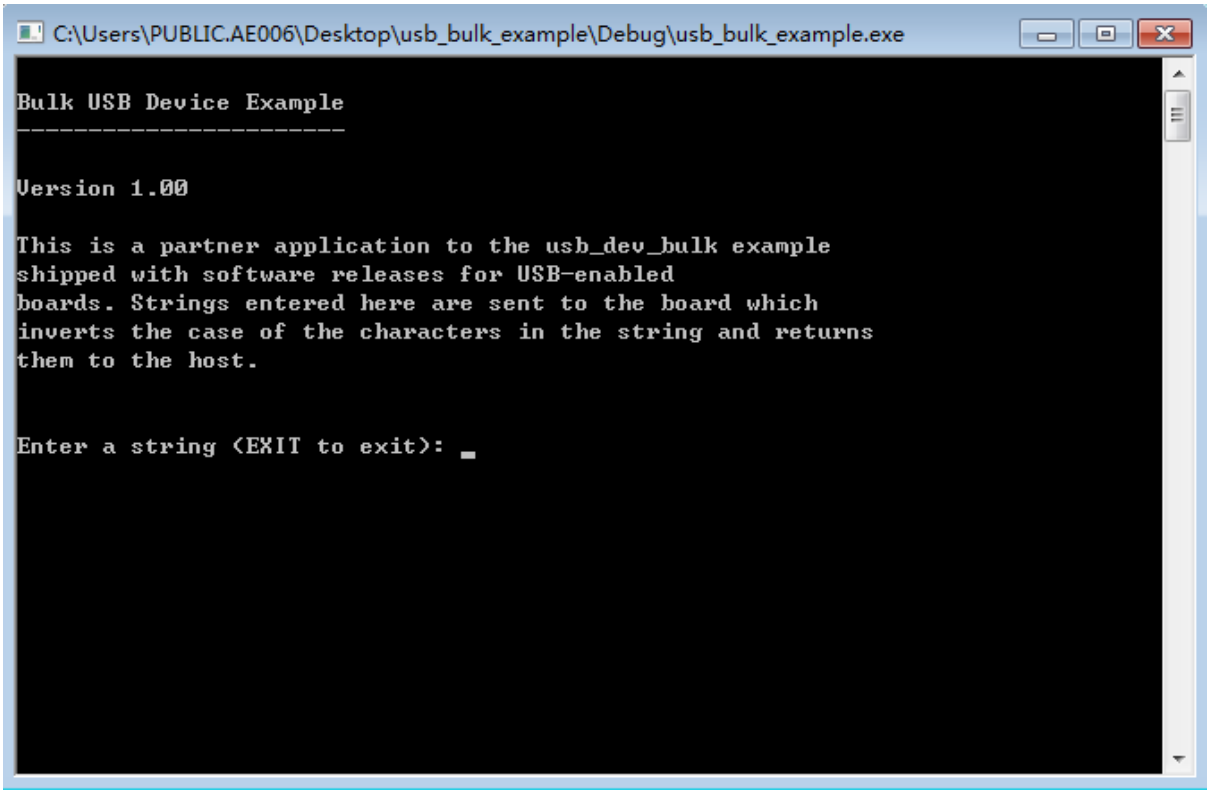


图 7-1 PC 端应用程序界面

7. 输入"abcdef123456789"，按"Enter"键，该字符串将发送给 MCU，MCU 在接收到该字符串后，返回同样的字符串给 PC：

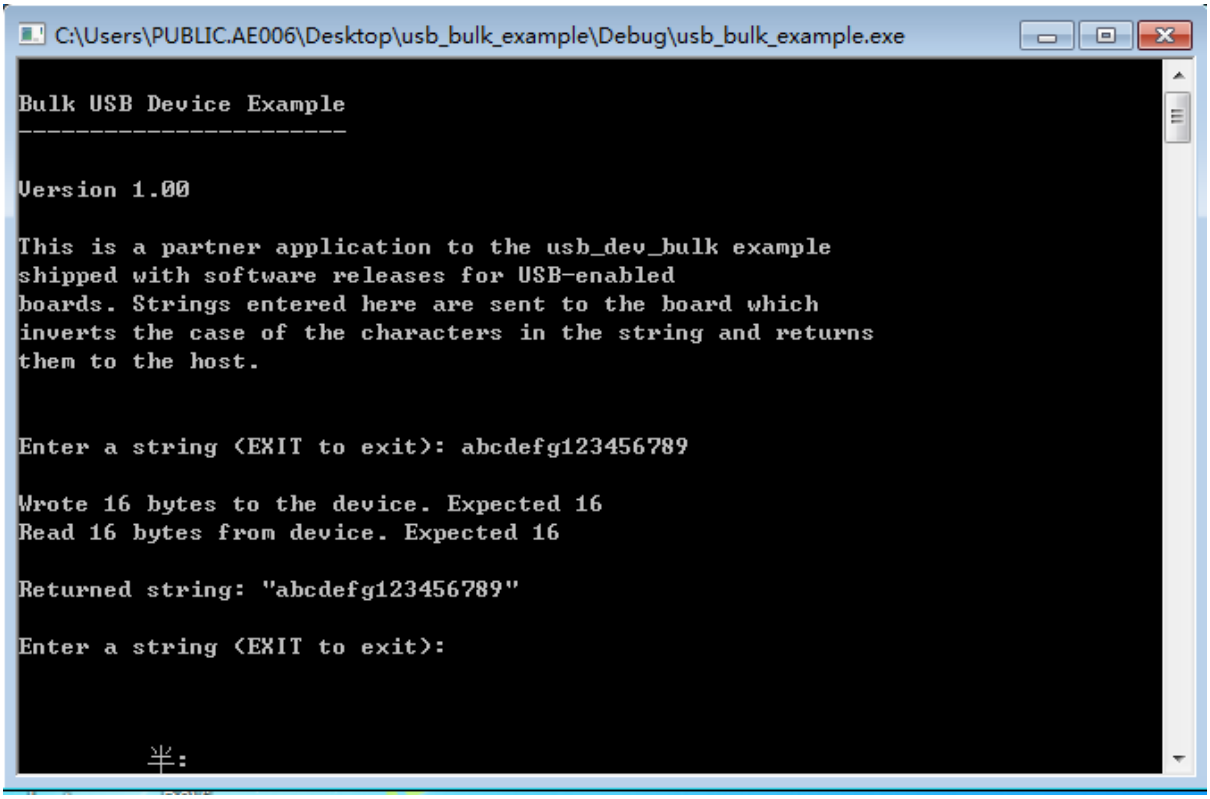


图 7-2 PC 端应用程序数据收发界面

7.4 注意事项

1. BULK 设备驱动安装(PC 端)：

1.1 将 MCU 通过 Micro-USB 线接入到 PC 端时，PC 端会出现如下界面：

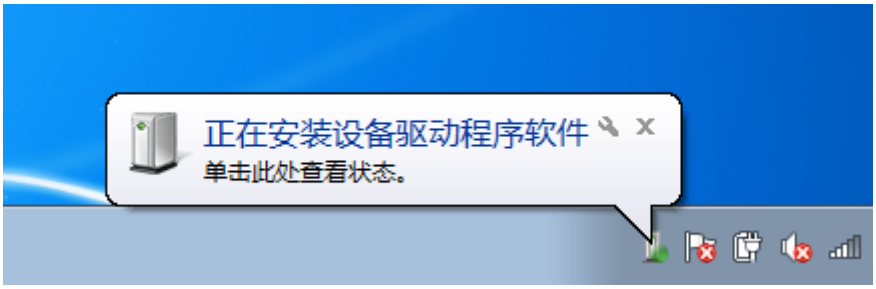


图 7-3 PC 端驱动搜索界面

1.2 自动搜索驱动会失败，失败后界面如下：

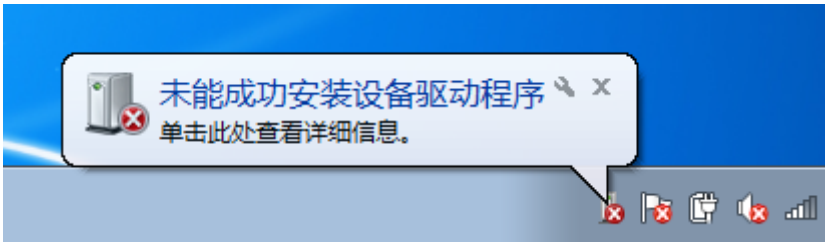


图 7-4 PC 端驱动搜索失败界面

1.3 打开设备管理器，找到对应设备， 并点击红框内按钮：

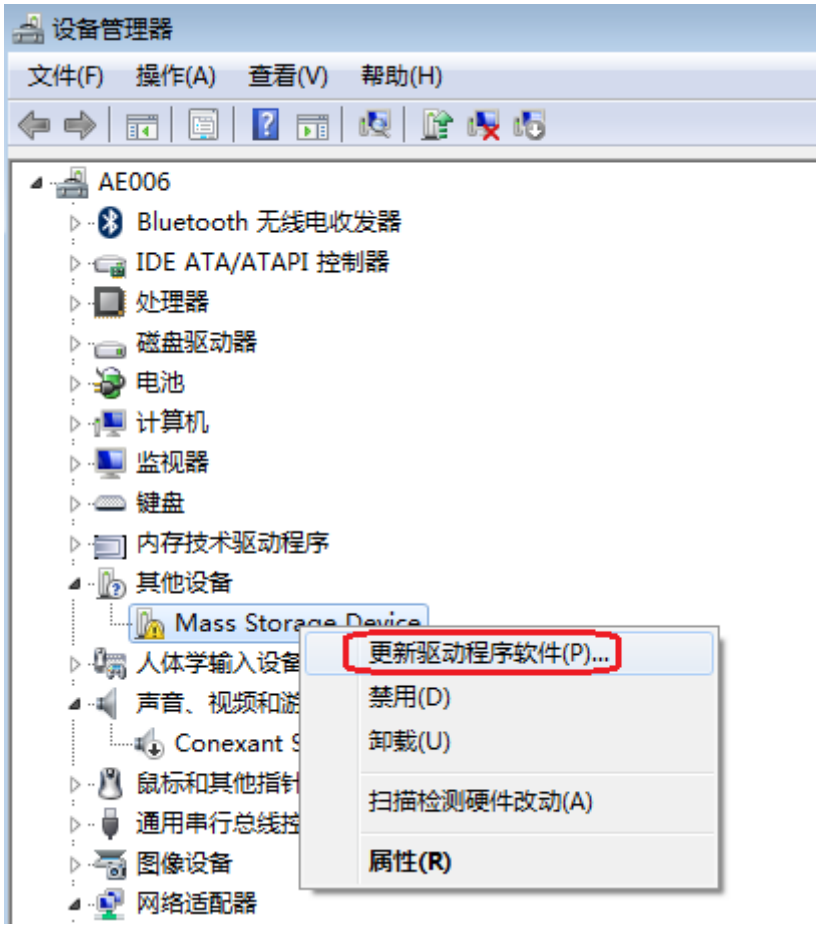


图 7-5 PC 端设备管理器界面

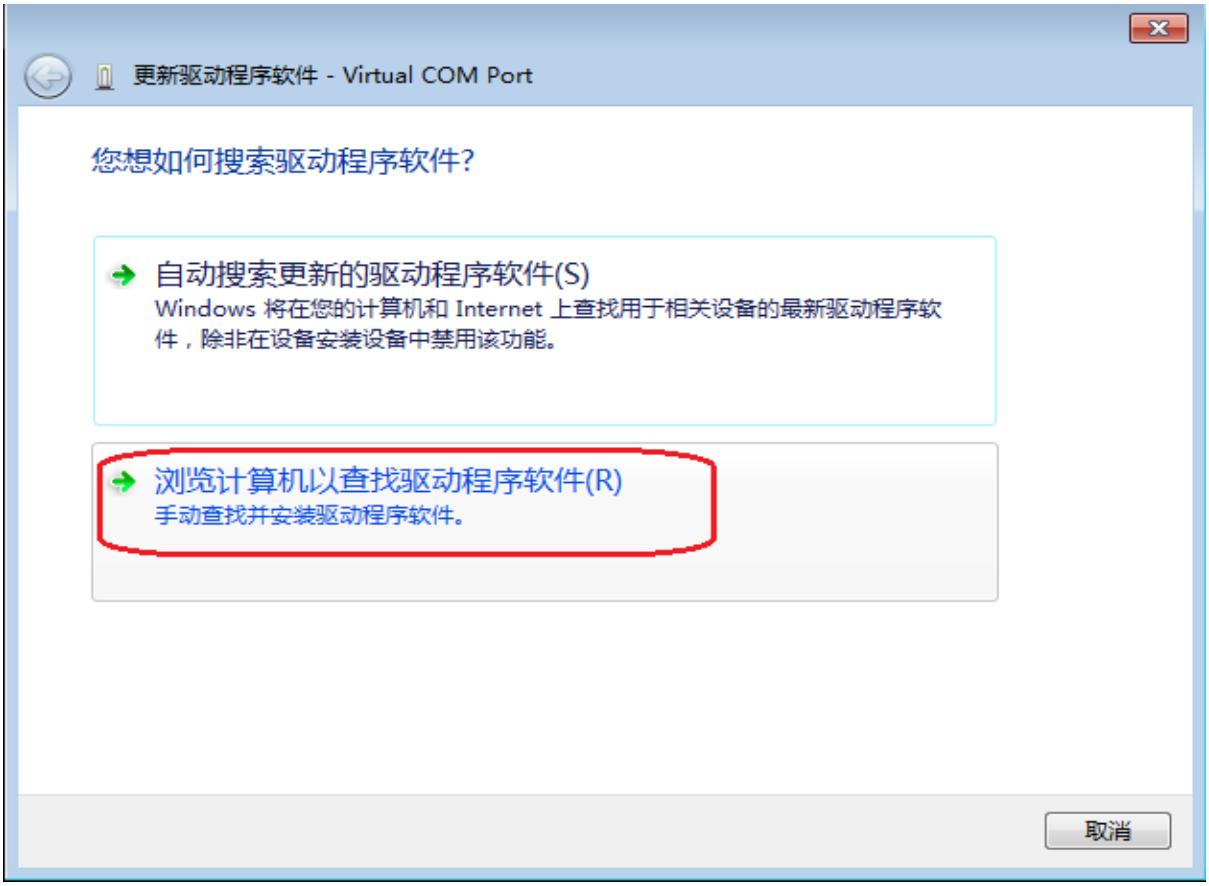


图 7-6 PC 端手动搜索驱动界面

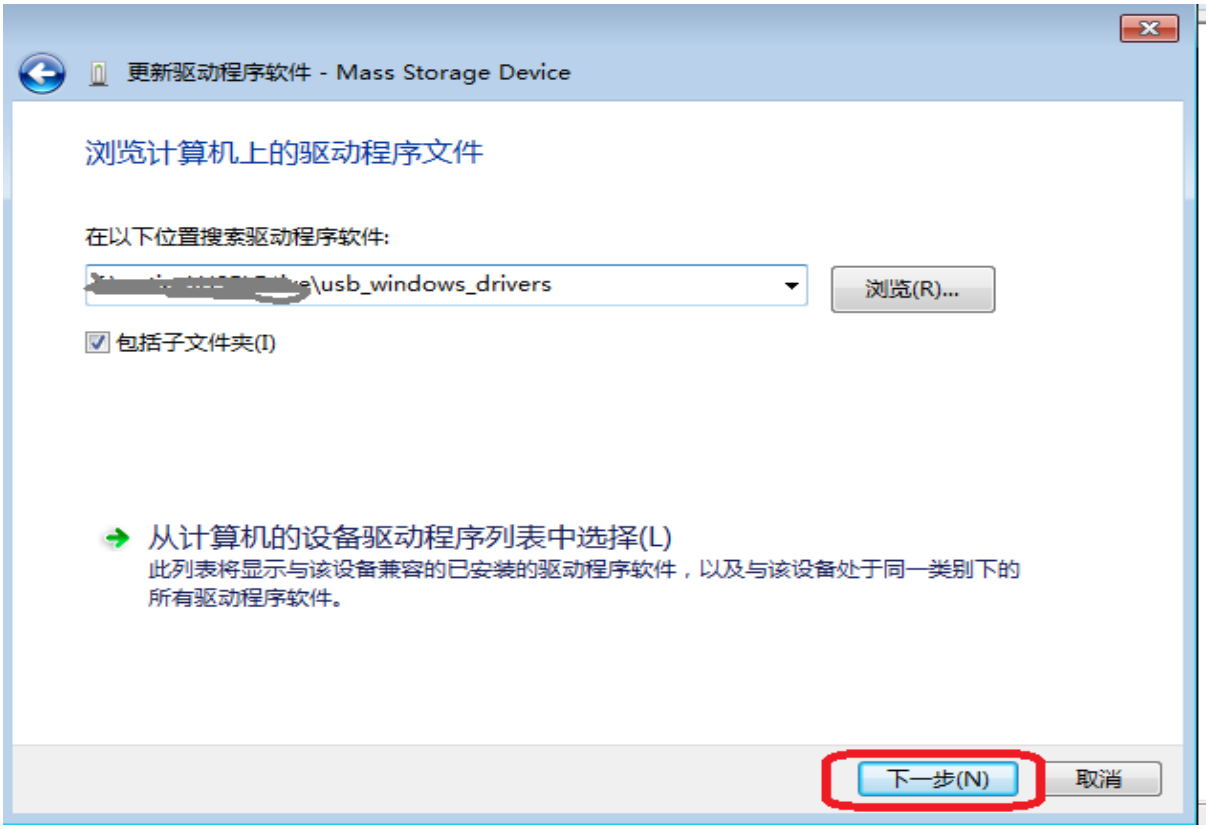


图 7-7 PC 端手动搜索驱动界面

其中 usb_windows_drivers 为本公司提供的 USB 驱动包，在此处选择正确的目录，点击“下一步”：

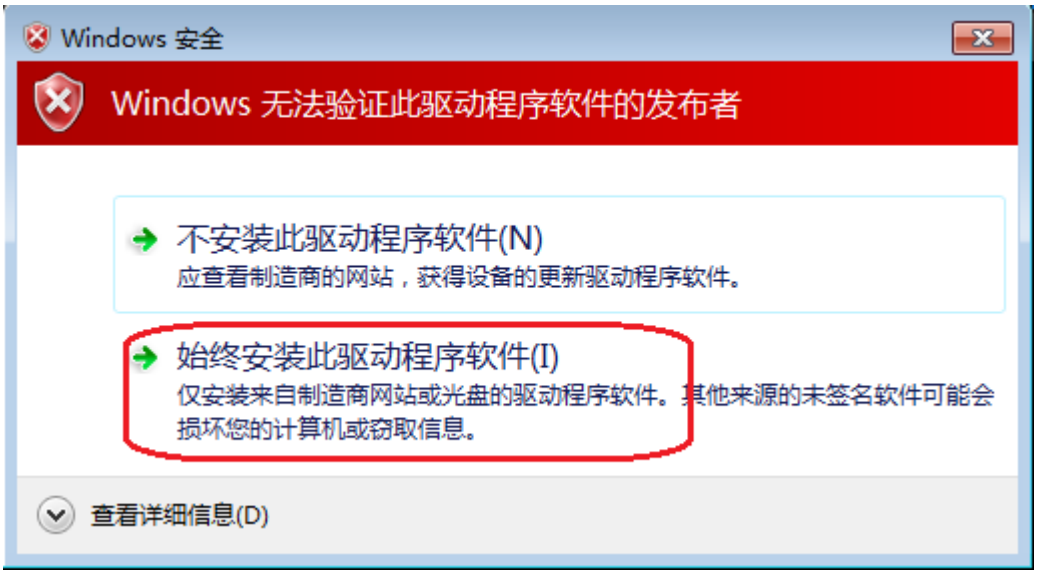


图 7-8 PC 端驱动安装界面

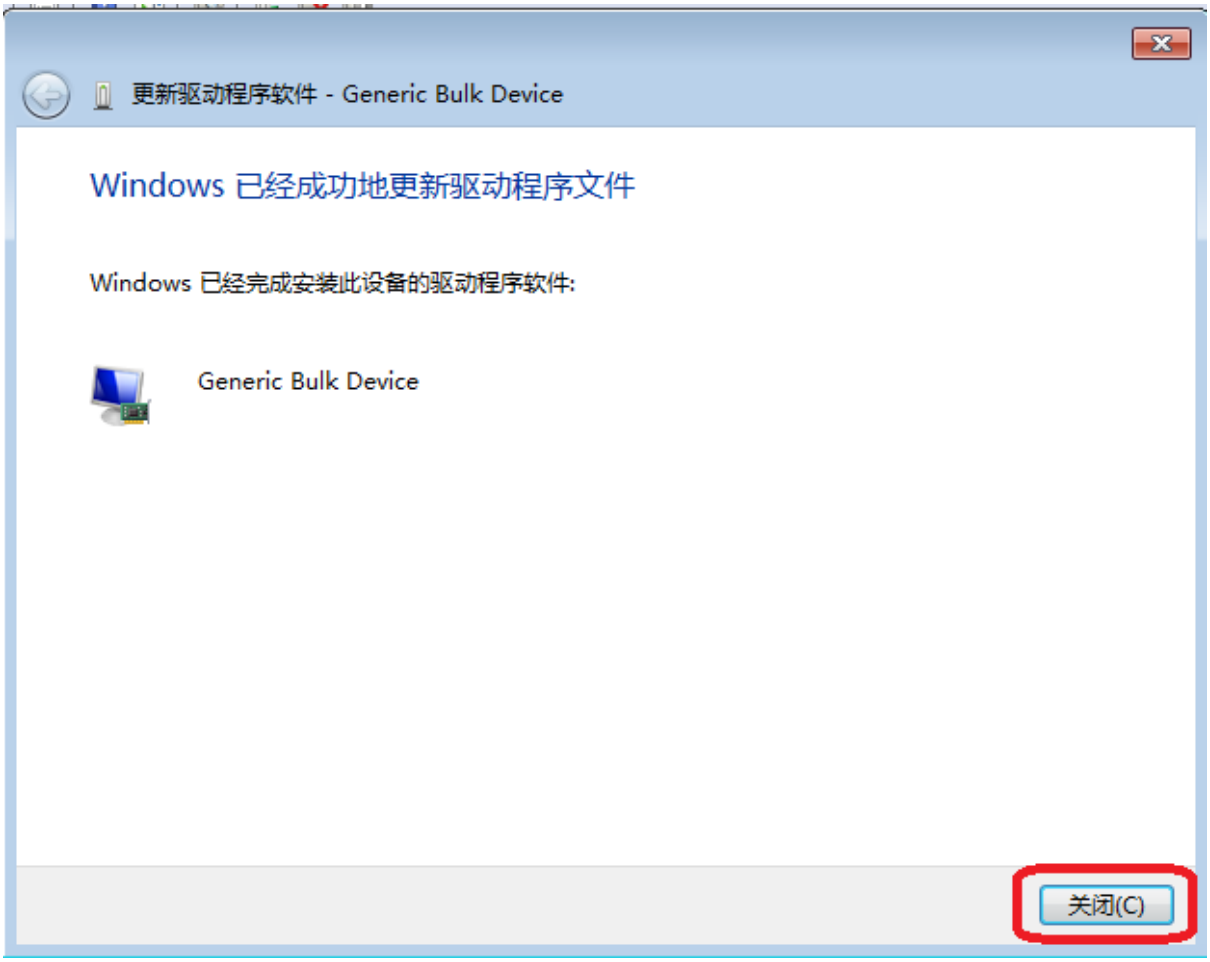


图 7-9 PC 端驱动安装完成界面

1.4 驱动安装成功后，在设备管理器界面会看到如下界面：

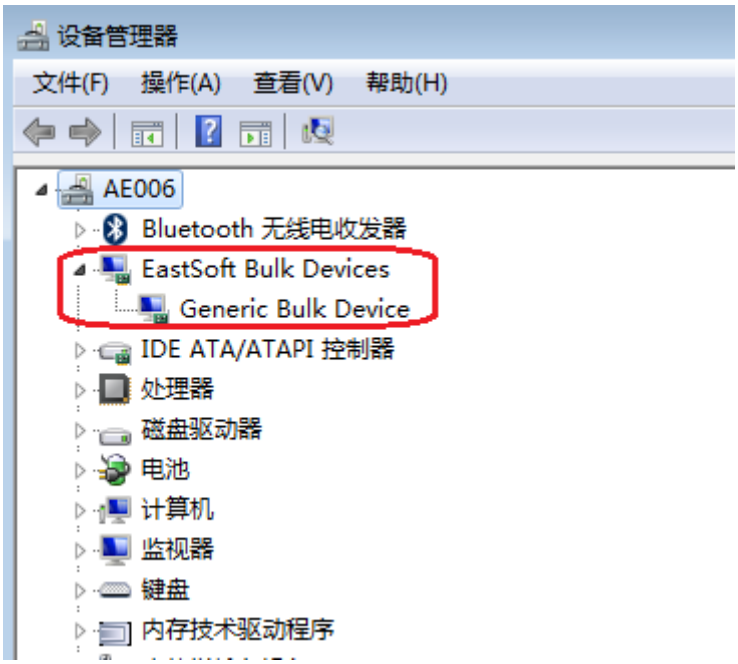


图 7-10 PC 端驱动安装完成后设备管理器界面

第8章　高速块设备(BULK)

8.1　功能概述

本例程为高速从机块设备(BULK)。USB 批量传输适合传输大量数据，并且保证数据的正确性。本例程重点在于演示底层数据收发通道。用户在此基础上可以自行开发各类 USB 设备。

本例程的功能是数据回传，即将从主机接收到的数据回传给主机。

本例程仅支持在具有高速 USB 接口的 MCU 上使用。

8.2　主要API

参考 7.2 章节

8.3　操作方法及演示效果

1. 工程路径：

   ~\ES32_SDK\Projects\ES32F36xx\Applications\USB\07_dev_bulk_hs\

参考 7.3 章节

8.4　注意事项

参考 7.4 章节

第9章 虚拟串口设备(CDC)

9. 1 功能概述

本例程为全速从机虚拟串口设备(CDC)。将 MCU 虚拟成一个串口。即通常说的"USB 转串口"设备

9. 2 主要API

9. 2. 1 CDC初始化函数

类型	描述
函数原型	void *usbd_cdc_init(uint32_t idx, usbd_cdc_dev_t *dev);
功能描述	将 MCU 初始化为一个 CDC 设备
输入参数	idx: USB 控制器的索引，总是为 0
	dev: CDC 设备结构体指针
返回值	成功：CDC 实例的指针
	失败：NULL
使用示例	usbd_cdc_init(0, &cdc_device);

表 9-1 CDC 初始化函数

9. 2. 2 CDC控制线回调函数

类型	描述
函数原型	uint32_t contrl_handle(void *data, uint32_t event, uint32_t value, void *p_data);
功能描述	USB 协议栈向应用层通知相关事件
输入参数	data: 用户层参数，一般为 usbd_cdc_init()函数返回值
	event: 事件类型
	value: 该事件携带的数值
	p_data:该事件携带的消息
返回值	成功：0
	失败：其他值
使用示例	应用层根据需要填充该函数

表 9-2 CDC 控制线回调函数

9. 2. 3 USB事件回调RX函数

类型	描述
函数原型	uint32_t rx_handle(void *data, uint32_t event, uint32_t value, void *p_data);
功能描述	USB 协议栈通知应用层已接收到数据
输入参数	data: 一般为 usbd_bulk_init()函数返回值
	event: 事件类型
	value: 该事件携带的数值
	p_data: 该事件携带的消息
返回值	成功：0
	失败：其他值
使用示例	应用层根据需要填充该函数

表 9-3 USB 事件回调 RX 函数

9. 2. 4 获取USB控制器接收到的数据个数函数

类型	描述
函数原型	uint32_t usbd_cdc_rx_packet_avail(usbd_cdc_dev_t *dev);
功能描述	获取 USB 控制器接收到的数据个数(Bytes)
输入参数	dev: 一般为 usbd_cdc_init()函数返回值
返回值	USB 控制器接收到的数据个数(Bytes)
使用示例	rx_nr = usbd_cdc_rx_packet_avail(&cdc_device);

表 9-4 获取 USB 控制器接收到的数据个数函数

9. 2. 5 从USB控制器读取数据函数

类型	描述
函数原型	uint32_t usbd_cdc_packet_read(usbd_cdc_dev_t *dev, uint8_t *data, uint32_t len, uint8_t last);
功能描述	从 USB 控制器中读取数据，该数据来自 USB 主机
输入参数	dev: 一般为 usbd_cdc_init()函数返回值
	data: 读取数据的存放地址
	len: Buffer 的大小
	last: 无关参数，忽略
返回值	读取到的数据个数(Bytes)
使用示例	rx_nr = usbd_cdc_packet_read(&cdc_device, rx_buf, rx_nr, 1);

表 9-5 从 USB 控制器中读取数据函数

9. 2. 6 向USB控制器写数据函数

类型	描述
函数原型	uint32_t usbd_cdc_packet_write(usbd_cdc_dev_t *dev, uint8_t *data, uint32_t len, uint8_t last);
功能描述	向 USB 控制器写数据函数，该数据将会被发送给 USB 主机
输入参数	dev: 一般为 usbd_cdc_init()函数返回值
	data: 待写入数据的存放地址
	len: Buffer 的大小
	last: 是否为最后一帧
返回值	写入数据的个数(Bytes)
使用示例	usbd_cdc_packet_write(&cdc_device, rx_buf, rx_nr, 1);

表 9-6 向 USB 控制器写数据函数

9. 3 操作方法及演示效果

1. 工程路径：
   ~\ES32_SDK\Projects\ES32F36xx\Applications\USB\08_dev_cdc\
2. 编译工程，并将其下载到板子上；
3. 使用 Micro-USB 线将板子连接到 PC 上；
4. 若是第一次连接，需要手动安装驱动，驱动安装方法参考注意事项部分；
5. 驱动安装成功后在 PC 端的设备管理器中能看到相应设备；
6. 在 PC 端打开串口工具，并进行数据收发；界面如下：

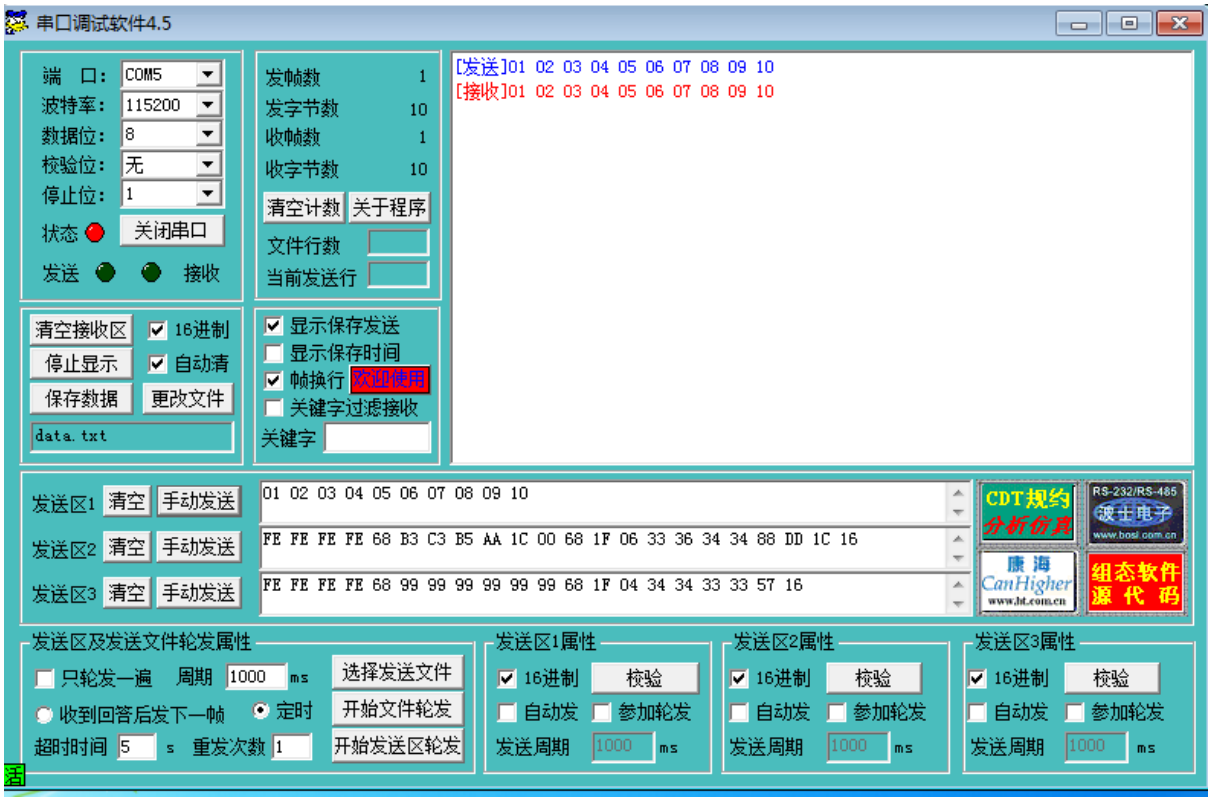


图 9-1 PC 端串口工具界面

9.4 注意事项

1. CDC 设备驱动安装(PC 端)：

1.1 将 MCU 通过 Micro-USB 线接入到 PC 端时，PC 端会出现如下界面：

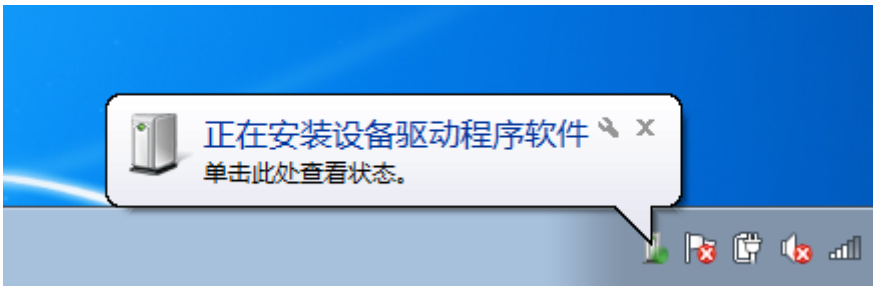


图 9-2 PC 端驱动搜索界面

1.2 自动搜索驱动会失败，失败后界面如下：

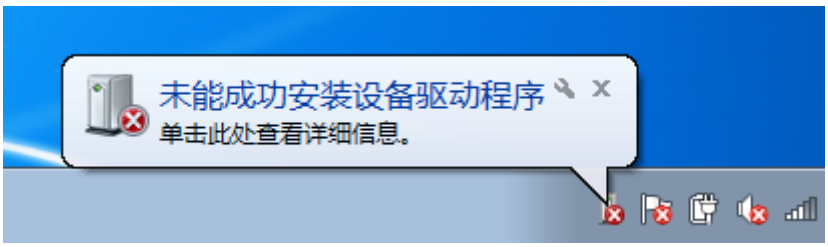


图 9-3 PC 端驱动搜索失败界面

1.3 打开设备管理器，找到对应设备， 并点击红框内按钮：

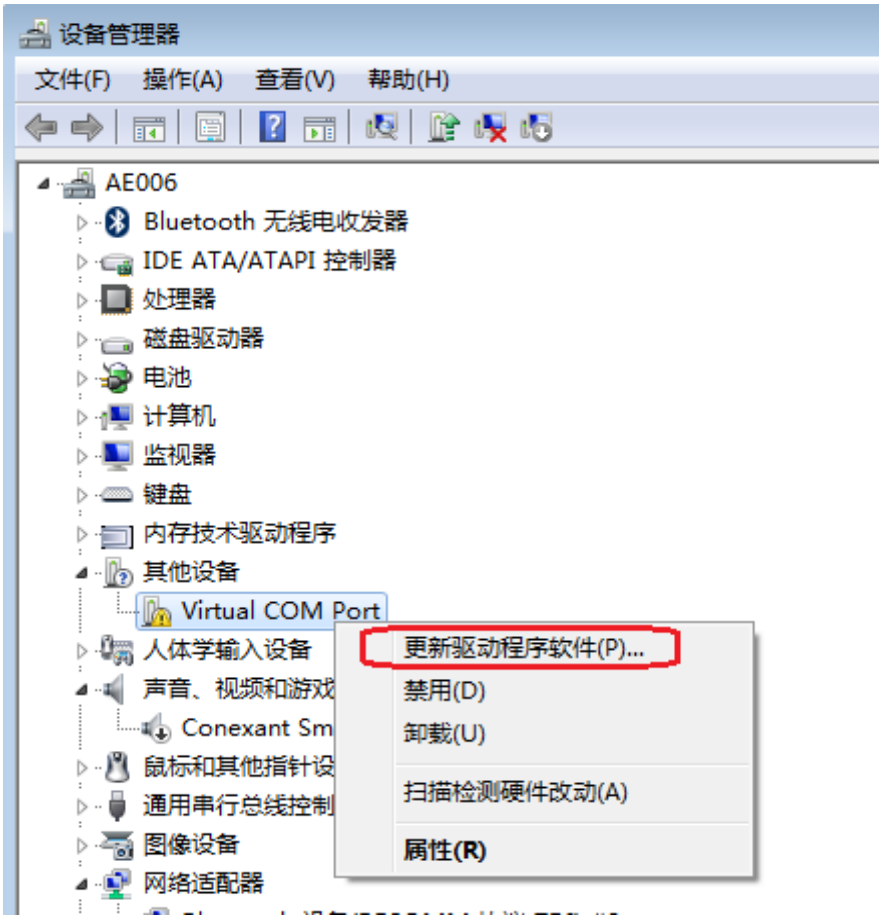


图 9-4 PC 端设备管理器界面

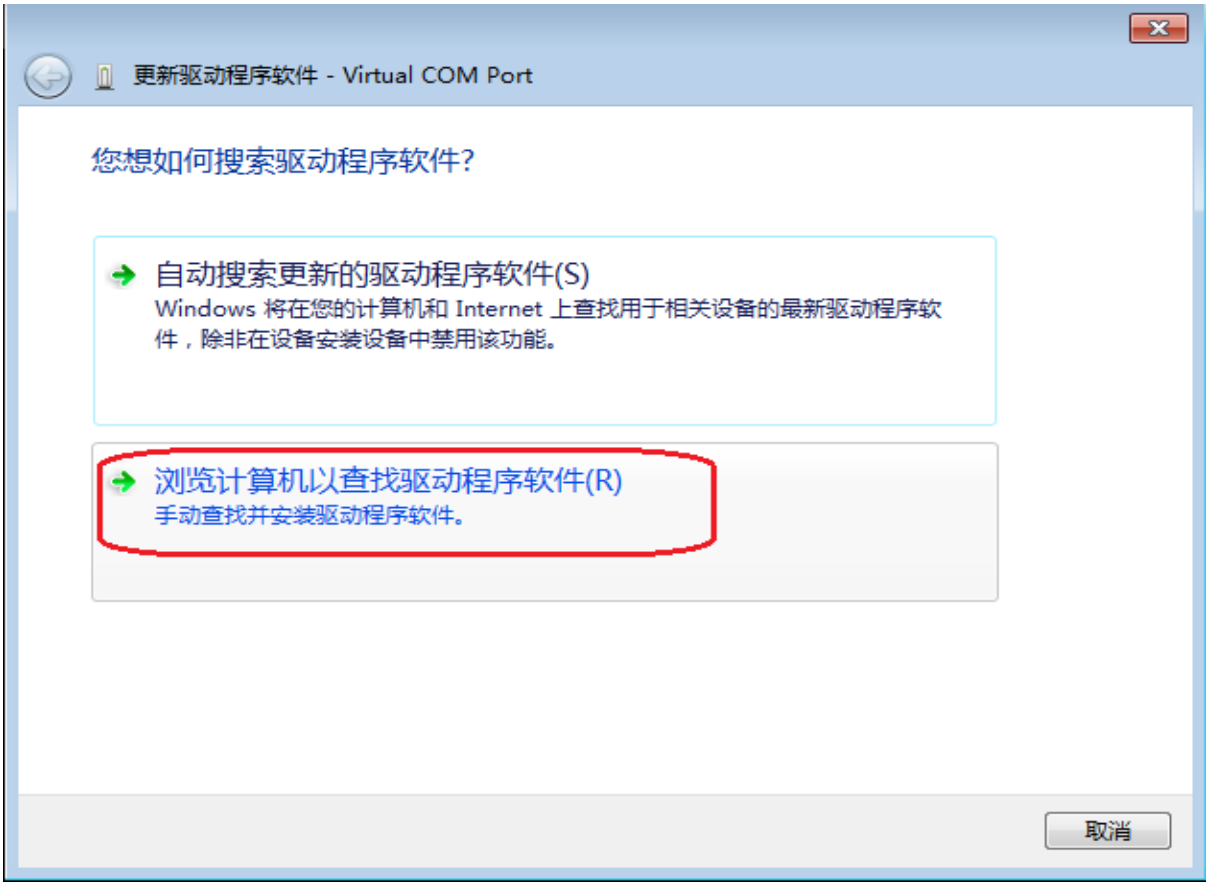


图 9-5 PC 端手动搜索驱动界面

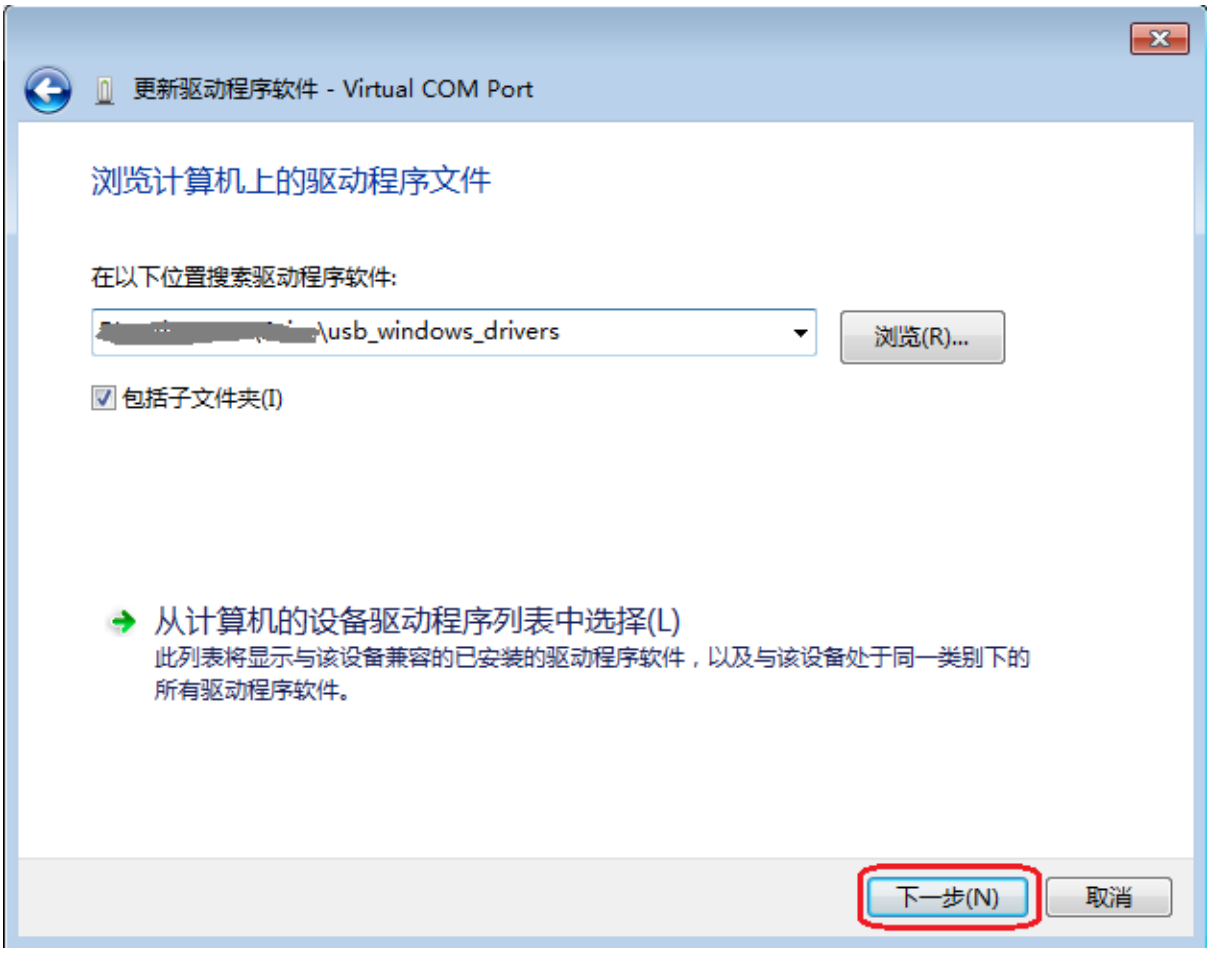


图 9-6 PC 端手动搜索驱动界面

其中 usb_windows_drivers 为本公司提供的 USB 驱动包，在此处选择正确的目录，点击“下一步”：

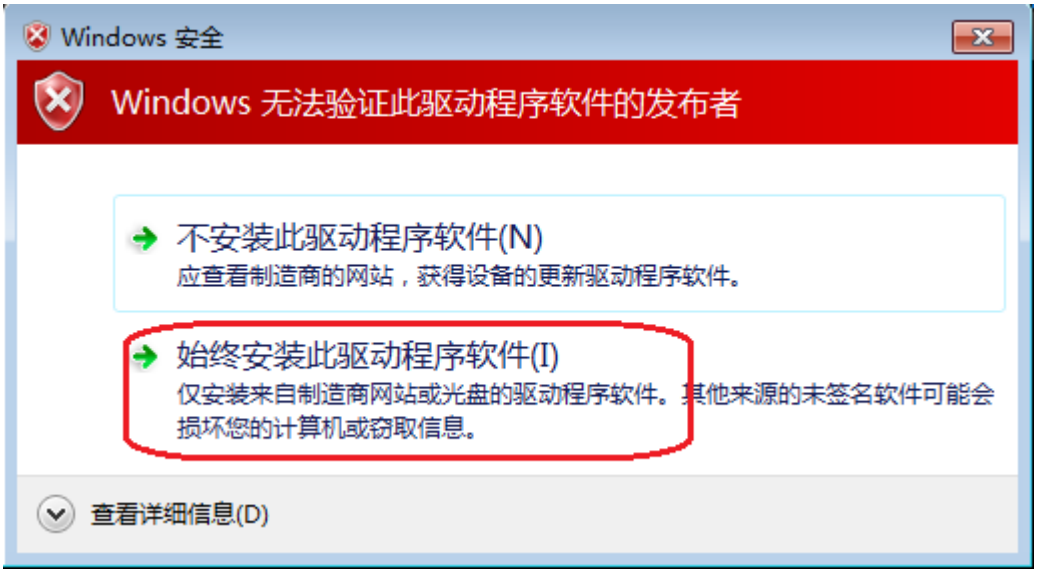


图 9-7 PC 端驱动安装界面

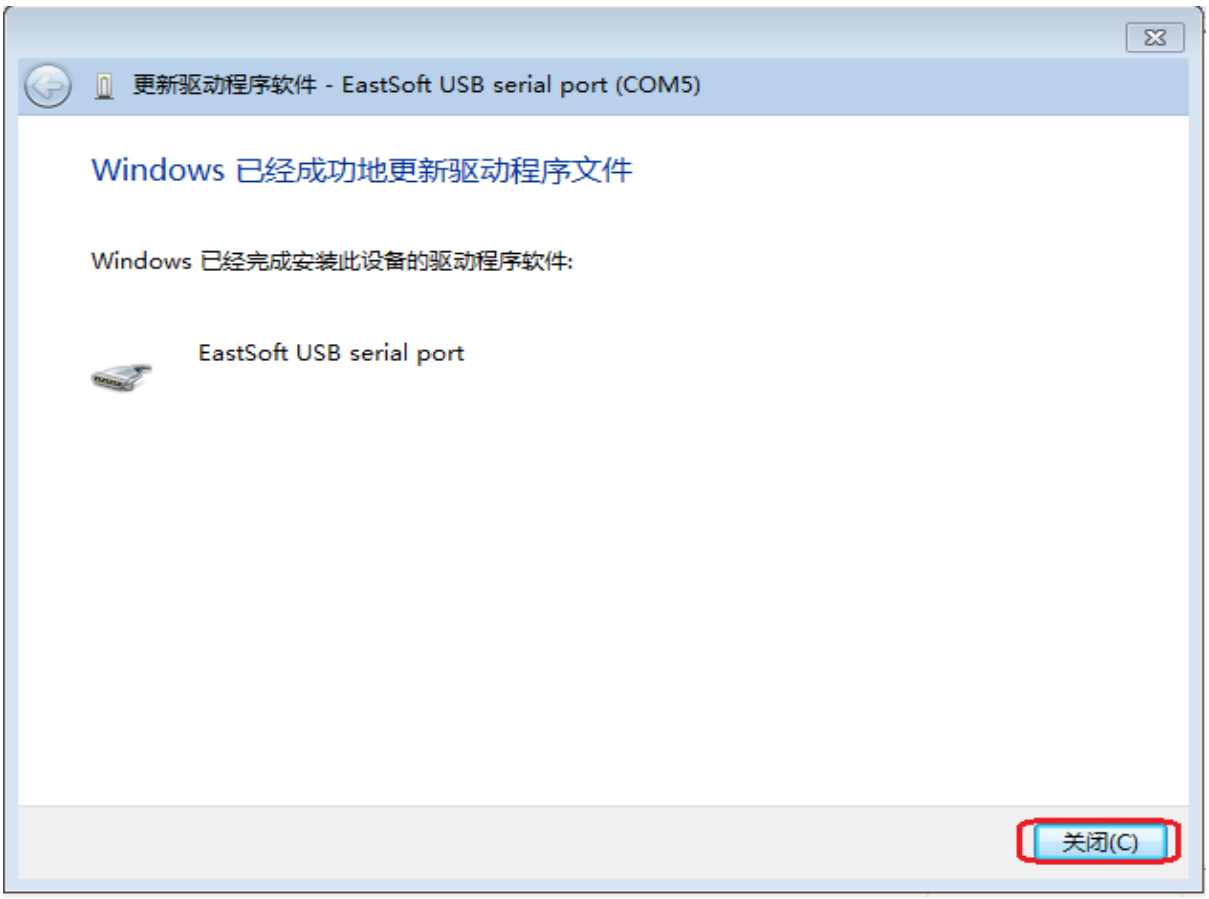


图 9-8 PC 端驱动安装完成界面

1.4 驱动安装成功后，在设备管理器界面会看到如下界面：

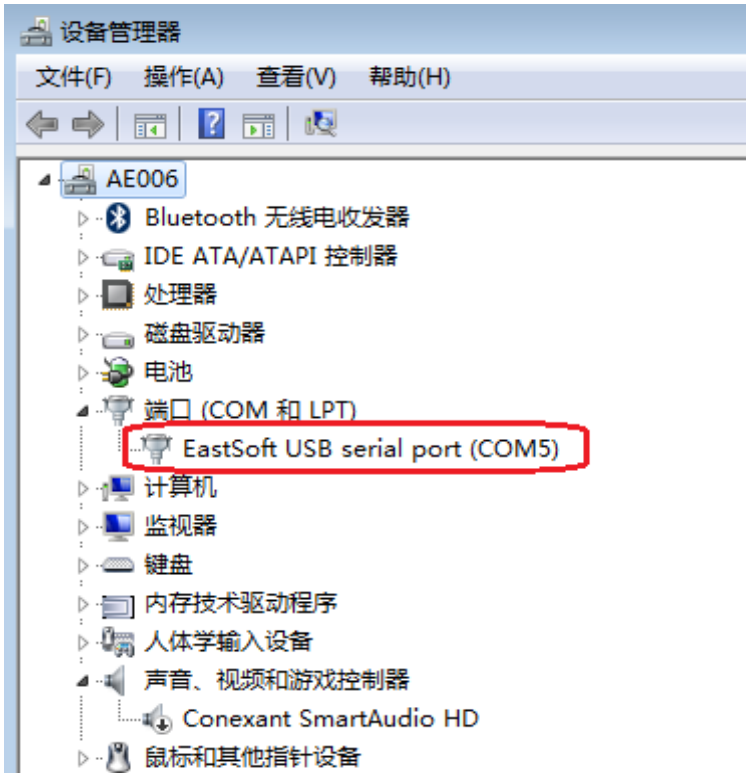


图 9-9 PC 端驱动安装完成后设备管理器界面

第10章 高速虚拟串口设备(CDC)

10. 1 功能概述

本例程为高速从机虚拟串口设备(CDC)。将 MCU 虚拟成一个串口。即通常说的"USB 转串口"设备

本例程仅支持在具有高速 USB 接口的 MCU 上使用。

10. 2 主要API

参考 9.2 章节。

10. 3 操作方法及演示效果

1. 工程路径：

   ~\ES32_SDK\Projects\ES32F36xx\Applications\USB\09_dev_cdc_hs\

参考 9.3 章节

10. 4 注意事项

参考 9.4 章节。

第11章 U盘键盘复合设备(Composite)

11. 1 功能概述

本例程为复合设备(Composite)，包含一个全速 U 盘设备(MSC)和一个全速键盘设备(Keyboard)。从 PC 端看 MCU 既有 U 盘的功能又有键盘的功能。

U 盘设备：将 MCU 虚拟成一个 80KBytes 的 U 盘；

键盘设备：将 MCU 虚拟成一个键盘设备。

11. 2 主要API

11. 2. 1 MSC复合设备初始化函数

类型	描述
函数原型	void *usbd_msc_init_comp(uint32_t idx, usbd_msc_dev_t *dev, comp_entry_t *entry);
功能描述	初始化 MSC 设备
输入参数	idx: USB 控制器的索引，总是为 0
	dev: MSC 设备结构体指针
	entry: 指向 comp_entry_t 结构的指针
返回值	成功：MSC 实例的指针
	失败：NULL
使用示例	comp_device.device[0].inst = usbd_msc_init_comp(0, &msc_device, &entry[0]);

表 11-1 MSC 复合设备初始化函数

11. 2. 2 Keyboard复合设备初始化函数

类型	描述
函数原型	void *usbd_hid_keyb_init_comp(uint32_t idx, usbd_hid_keyb_dev_t *dev, comp_entry_t *entry);
功能描述	初始化 Keyboard 设备
输入参数	idx: USB 控制器的索引，总是为 0
	dev: Keyboard 设备结构体指针
	entry: 指向 comp_entry_t 结构的指针
返回值	成功：Keyboard 实例的指针
	失败：NULL
使用示例	comp_device.device[1].inst = usbd_hid_keyb_init_comp(0, &keyb_device, &entry[1]);

表 11-2 Keyboard 复合设备初始化函数

11.2.3 复合设备初始化函数

类型	描述
函数原型	void *usbd_comp_init(uint32_t idx, usbd_comp_dev_t *dev, uint32_t size, uint8_t *data);
功能描述	初始化复合设备
输入参数	idx: USB 控制器的索引，总是为 0
	dev: 指向 usbd_comp_dev_t 结构体的指针
	size: Buffer 的大小
	data: 缓存复合设备配置描述符的 Buffer
返回值	成功：dev
	失败：NULL
使用示例	usbd_comp_init(0, &comp_device, DESC_BUF_SIZE, desc_buf);

表 11-3 复合设备初始化函数

11.3 操作方法及演示效果

1. 工程路径：
   ~\ES32_SDK\Projects\ES32F36xx\Applications\USB\10_dev_comp_msc_key\
2. 编译工程，并将其下载到板子上；
3. 使用 Micro-USB 线将板子连接到 PC 上；
4. 若是第一次连接，PC 端会自动安装相应驱动；
5. 驱动安装成功后在 PC 端的设备管理器中能看到相应设备；

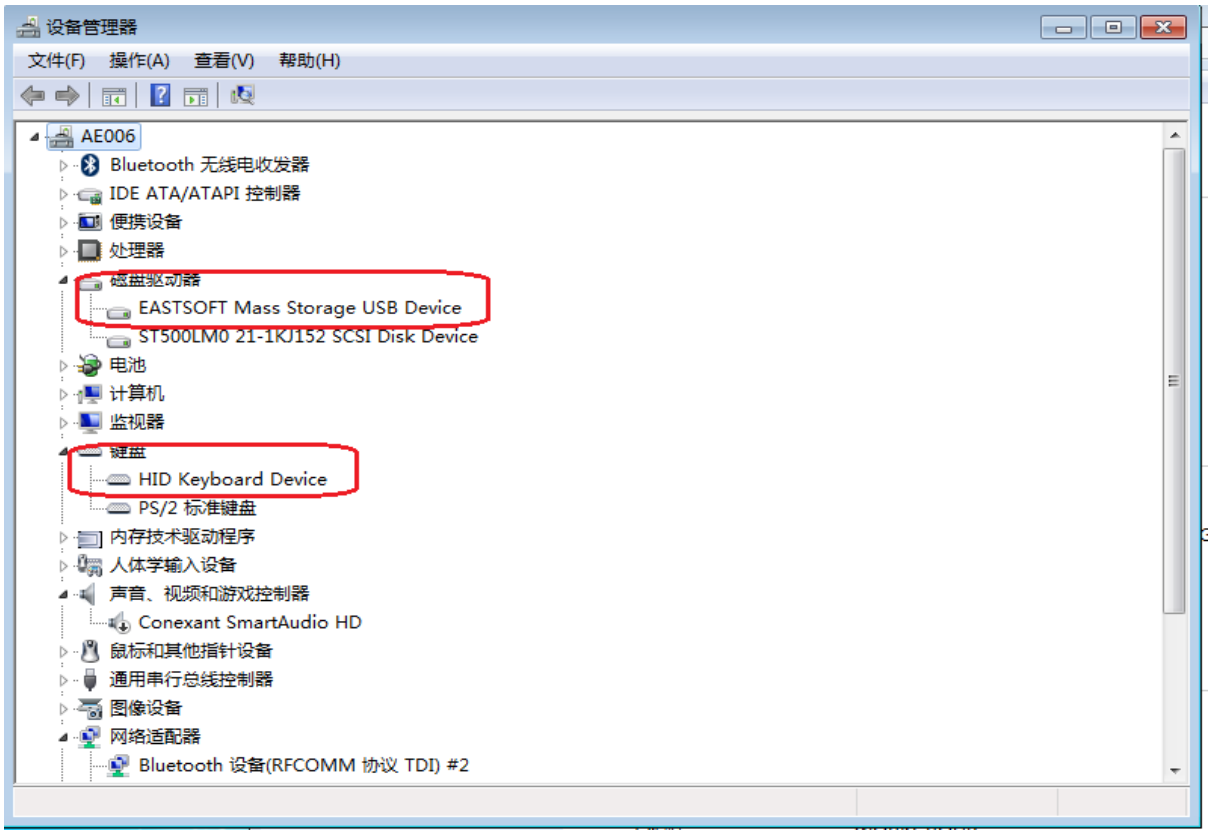


图 11-1 PC 端设备管理器界面

6. 演示效果参考第 2 章 MSC 设备、第 5 章 Keyboard 设备相应部分。

11. 4 注意事项

1. 复合设备相当于 1 个设备具备 2 种类型的功能；
2. 由于例程使用的是将 SRAM 虚拟成 U 盘，所以断电后存在虚拟 U 盘的内容会丢失。

第12章 键盘鼠标复合设备(Composite)

12. 1 功能概述

本例程为复合设备(Composite)，包含一个全速键盘设备(Keyboard)和一个全速鼠标设备(Mouse)。从 PC 端看，MCU 既有键盘的功能又有鼠标的功能，即是一个鼠键合一的设备。

键盘设备：将 MCU 虚拟成一个键盘设备；

鼠标设备：将 MCU 虚拟成一个鼠标设备。

12. 2 主要API

12. 2. 1 Mouse复合设备初始化函数

类型	描述
函数原型	void *usbd_hid_mouse_init_comp(uint32_t idx, usbd_hid_mouse_dev_t *dev, comp_entry_t *entry);
功能描述	初始化鼠标设备
输入参数	idx: USB 控制器的索引，总是为 0
	dev: Mouse 设备结构体指针
	entry: 指向 comp_entry_t 结构的指针
返回值	成功：Mouse 实例的指针
	失败：NULL
使用示例	comp_device.device[1].inst = usbd_hid_mouse_init_comp(0, &mouse_device, &entry[1]);

表 12-1 MSC 复合设备初始化函数

12. 2. 2 Keyboard复合设备初始化函数

类型	描述
函数原型	void *usbd_hid_keyb_init_comp(uint32_t idx, usbd_hid_keyb_dev_t *dev, comp_entry_t *entry);
功能描述	初始化 Keyboard 设备
输入参数	idx: USB 控制器的索引，总是为 0
	dev: Keyboard 设备结构体指针
	entry: 指向 comp_entry_t 结构的指针
返回值	成功：Keyboard 实例的指针
	失败：NULL
使用示例	comp_device.device[0].inst = usbd_hid_keyb_init_comp(0, &keyb_device, &entry[0]);

表 12-2 Keyboard 复合设备初始化函数

12. 2. 3 复合设备初始化函数

类型	描述
函数原型	void *usbd_comp_init(uint32_t idx, usbd_comp_dev_t *dev, uint32_t size, uint8_t *data);
功能描述	初始化复合设备
输入参数	idx: USB 控制器的索引，总是为 0
	dev: 指向 usbd_comp_dev_t 结构体的指针
	size: Buffer 的大小
	data: 缓存复合设备配置描述符的 Buffer
返回值	成功：dev
	失败：NULL
使用示例	usbd_comp_init(0, &comp_device, DESC_BUF_SIZE, desc_buf);

表 12-3 复合设备初始化函数

12. 3 操作方法及演示效果

1. 工程路径：
   ~\ES32_SDK\Projects\ES32F36xx\Applications\USB\11_dev_comp_keyb_mouse\
2. 编译工程，并将其下载到板子上；
3. 使用 Micro-USB 线将板子连接到 PC 上；
4. 若是第一次连接，PC 端会自动安装相应驱动；
5. 驱动安装成功后在 PC 端的设备管理器中能看到相应设备；

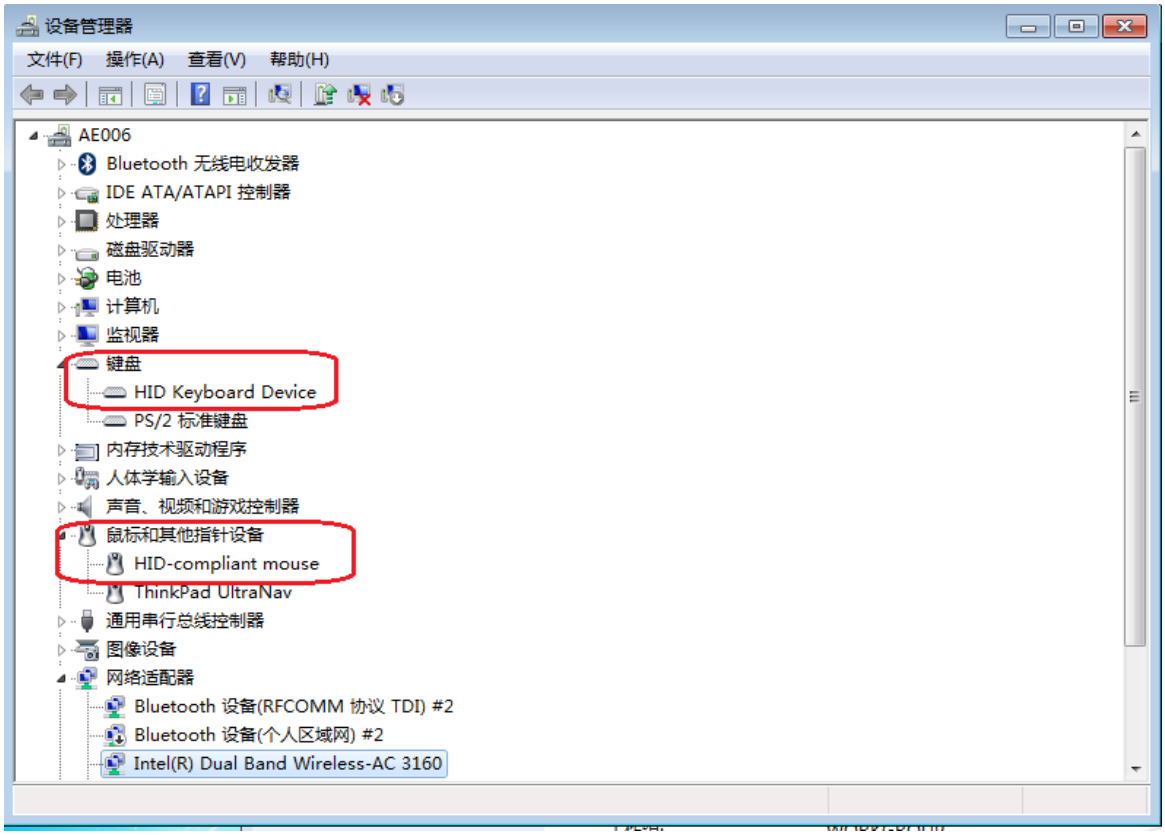


图 12-1 PC 端设备管理器界面

6. 演示效果参考第 4 章 Mouse 设备、第 5 章 Keyboard 设备相应部分。

12. 4 注意事项

1. 复合设备相当于 1 个设备具备 2 种类型的功能；

第13章 双虚拟串口复合设备(Composite)

13. 1 功能概述

本例程为复合设备(Composite)，包含两个全速虚拟串口设备(CDC)。从 PC 端看，MCU 具有两个虚拟串口功能。

13. 2 主要API

13. 2. 1 CDC复合设备初始化函数

类型	描述
函数原型	void *usbd_cdc_init_comp(uint32_t idx, usbd_cdc_dev_t *dev, comp_entry_t *entry, uint8_t second);
功能描述	初始化 CDC 设备
输入参数	idx: USB 控制器的索引，总是为 0
	dev: CDC 设备结构体指针
	entry: 指向 comp_entry_t 结构的指针
	second: 是否为复合 CDC 设备的第二个设备
返回值	成功：CDC 实例的指针
	失败：NULL
使用示例	comp_device.device[0].inst = usbd_cdc_init_comp(0, &cdc_device, &entry[0], 0); comp_device.device[1].inst = usbd_cdc_init_comp(0, &cdc_device1, &entry[1],1);

表 13-1 CDC 复合设备初始化函数

13. 2. 2 复合设备初始化函数

类型	描述
函数原型	void *usbd_comp_init(uint32_t idx, usbd_comp_dev_t *dev, uint32_t size, uint8_t *data);
功能描述	初始化复合设备
输入参数	idx: USB 控制器的索引，总是为 0
	dev: 指向 usbd_comp_dev_t 结构体的指针
	size: Buffer 的大小
	data: 缓存复合设备配置描述符的 Buffer
返回值	成功：dev
	失败：NULL
使用示例	usbd_comp_init(0, &comp_device, DESC_BUF_SIZE, desc_buf);

表 13-2 复合设备初始化函数

13. 3 操作方法及演示效果

1. 工程路径：
   ~\ES32_SDK\Projects\ES32F36xx\Applications\USB\12_dev_comp_dual_cdc\
2. 编译工程，并将其下载到板子上；
3. 使用 Micro-USB 线将板子连接到 PC 上；
4. 若是第一次连接，PC 端需手动安装相应驱动，驱动安装参考第 9 章相应部分；
5. 驱动安装成功后在 PC 端的设备管理器中能看到相应设备；

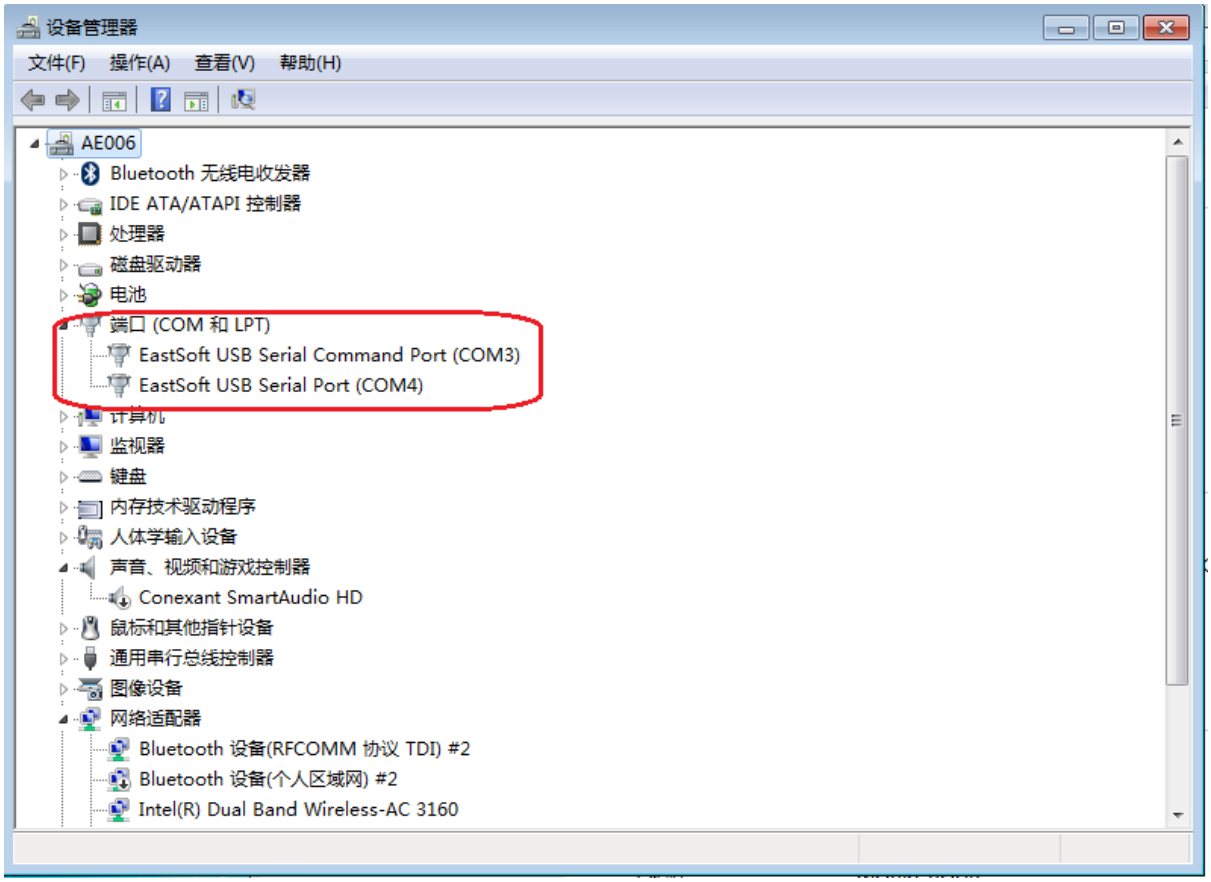


图 13-1 PC 端设备管理器界面

6. 在 PC 端打开 2 个串口终端，分别与不同的虚拟串口进行通信：

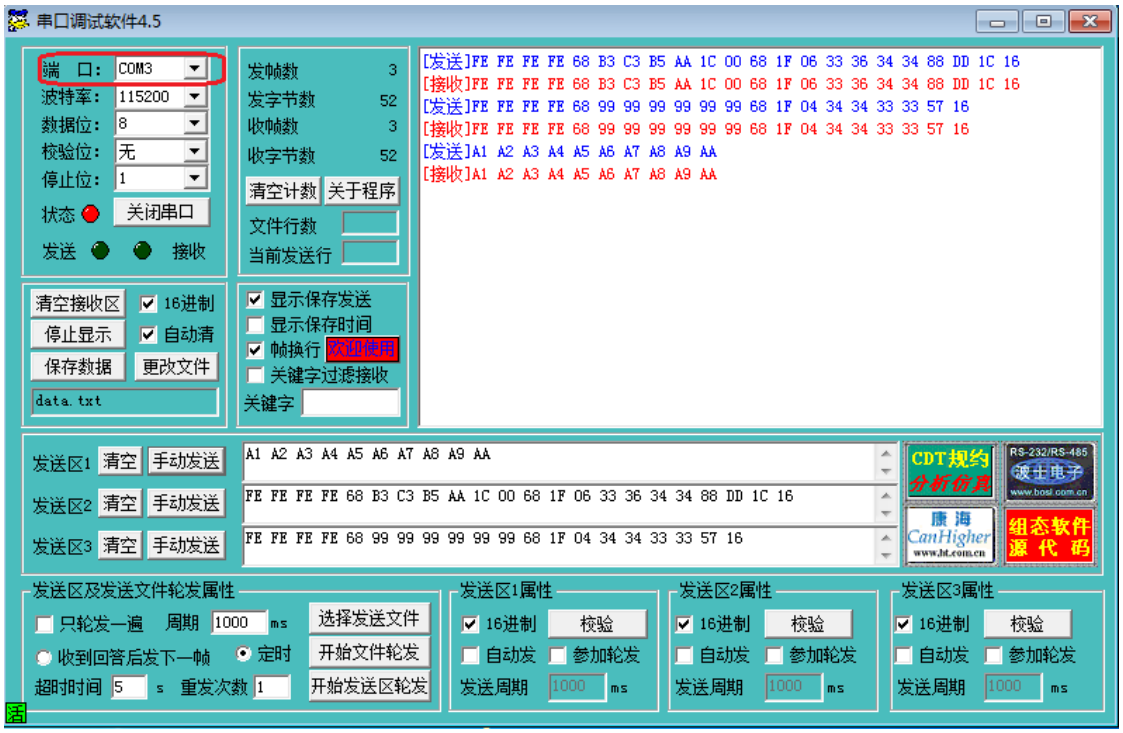


图 13-2 第一个虚拟串口界面

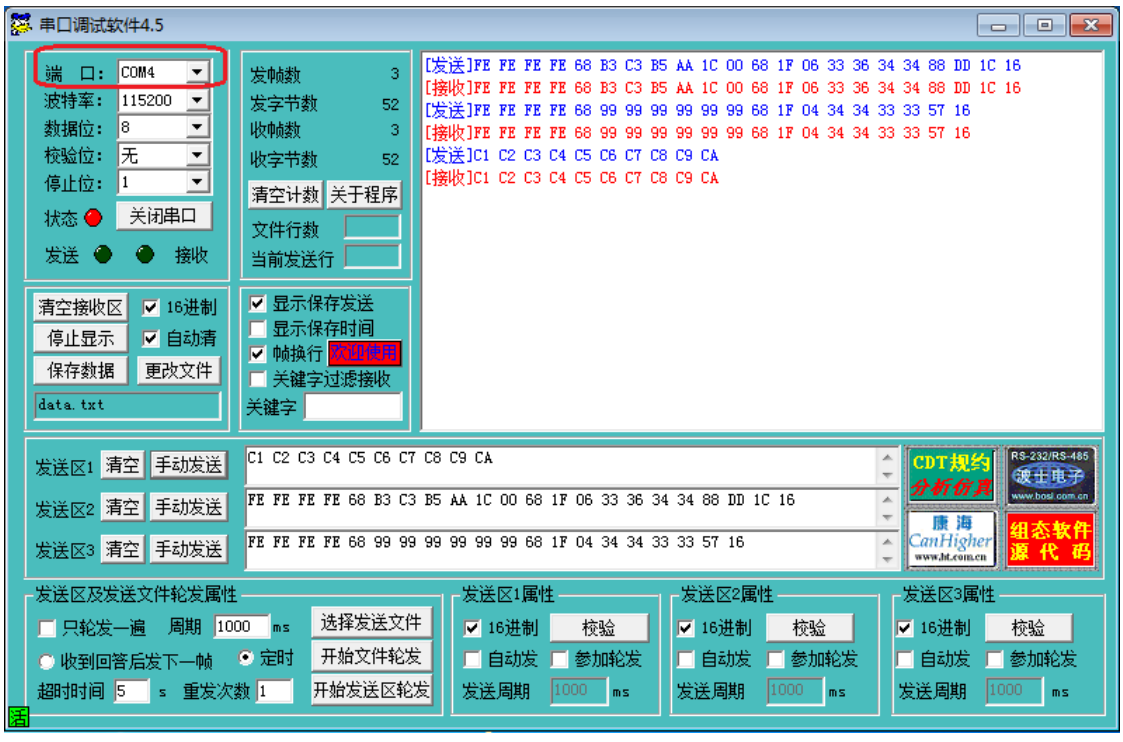


图 13-3 第二个虚拟串口界面

13. 4 注意事项

1. 复合设备相当于 1 个设备具备 2 种类型的功能；
2. MCU 内串口数据流被实现为环流：将从 USB 主机接收到的数据转发给 USB 主机；

第14章 双虚拟串口高速复合设备(Composite)

14. 1 功能概述

本例程为复合设备(Composite)，包含两个高速虚拟串口设备(CDC)。从 PC 端看，MCU 具有两个虚拟串口功能。

本例程仅支持在具有高速 USB 接口的 MCU 上使用。

14. 2 主要API

参考 13.2 章节。

14. 3 操作方法及演示效果

1. 工程路径：

   ~\ES32_SDK\Projects\ES32F36xx\Applications\USB\13_dev_comp_dual_cdc_hs\

参考 13.3 章节

14. 4 注意事项

参考 13.4 章节

第15章 虚拟串口鼠标复合设备(Composite)

15. 1 功能概述

本例程为复合设备(Composite)，包含 1 个全速虚拟串口设备(CDC)和 1 个鼠标设备(Mouse)。从 PC 端看 MCU 既有虚拟串口的功能又有鼠标的功能。

CDC 设备：将 MCU 虚拟成一个串口设备；

鼠标设备：将 MCU 虚拟成一个鼠标设备。

15. 2 主要API

15. 2. 1 CDC复合设备初始化函数

类型	描述
函数原型	void *usbd_cdc_init_comp(uint32_t idx, usbd_cdc_dev_t *dev, comp_entry_t *entry, uint8_t second);
功能描述	初始化 CDC 设备
输入参数	idx: USB 控制器的索引，总是为 0
	dev: CDC 设备结构体指针
	entry: 指向 comp_entry_t 结构的指针
	second: 是否为复合 CDC 设备的第二个设备
返回值	成功：CDC 实例的指针
	失败：NULL
使用示例	comp_device.device[0].inst = usbd_cdc_init_comp(0, &cdc_device, &entry[0], 0);

表 15-1 CDC 复合设备初始化函数

15. 2. 2 Mouse复合设备初始化函数

类型	描述
函数原型	void *usbd_hid_mouse_init_comp(uint32_t idx, usbd_hid_mouse_dev_t *dev, comp_entry_t *entry);
功能描述	初始化鼠标设备
输入参数	idx: USB 控制器的索引，总是为 0
	dev: Mouse 设备结构体指针
	entry: 指向 comp_entry_t 结构的指针
返回值	成功：Mouse 实例的指针
	失败：NULL
使用示例	comp_device.device[1].inst = usbd_hid_mouse_init_comp(0, &mouse_device, &entry[1]);

表 15-2 MSC 复合设备初始化函数

15. 2. 3 复合设备初始化函数

类型	描述
函数原型	void *usbd_comp_init(uint32_t idx, usbd_comp_dev_t *dev, uint32_t size, uint8_t *data);
功能描述	初始化复合设备
输入参数	idx: USB 控制器的索引，总是为 0
	dev: 指向 usbd_comp_dev_t 结构体的指针
	size: Buffer 的大小
	data: 缓存复合设备配置描述符的 Buffer
返回值	成功：dev
	失败：NULL
使用示例	usbd_comp_init(0, &comp_device, DESC_BUF_SIZE, desc_buf);

表 15-3 复合设备初始化函数

15. 3 操作方法及演示效果

1. 工程路径：
   ~\ES32_SDK\Projects\ES32F36xx\Applications\USB\14_dev_comp_cdc_mouse\
2. 编译工程，并将其下载到板子上；
3. 使用 Micro-USB 线将板子连接到 PC 上；
4. 若是第一次连接，PC 端需手动安装相应驱动，驱动安装参考第 9 章相应部分；
5. 驱动安装成功后在 PC 端的设备管理器中能看到相应设备；

图 15-1 PC 端设备管理器界面

6. 在 PC 端打开串口终端，与虚拟串口进行通信：

图 15-2 第一个虚拟串口界面

7. 鼠标设备演示效果：每隔一段时间会控制 PC 端的鼠标指针移动一段距离。

8. MCU 端串口输出如下：

图 15-3 MCU 端串口输出信息

15. 4 注意事项

1. 复合设备相当于 1 个设备具备 2 种类型的功能；
2. MCU 内串口数据流被实现为环流：将从 USB 主机接收到的数据转发给 USB 主机；
3. 鼠标功能不依赖板级资源，每隔一段时间会控制 PC 端的鼠标指针移动一段距离。

第16章 U盘主机(MSC)

16. 1 功能概述

本例程为 U 盘主机(MSC)，能识别连接到 MCU 的 U 盘设备，并对其进行扇区的读写。MCU 作为高速 USB 主机能适配各种速度的设备(LS/FS/HS)。

若 MCU 是 FS-USB，则只支持 USB1.1 设备；若是 HS-USB 则可支持 USB2.0 设备。

16. 2 主要API

16. 2. 1 主机驱动注册函数

类型	描述
函数原型	void usb_hcd_register_driver(uint32_t idx, const usbh_class_driver_t * const *drv, uint32_t nr);
功能描述	向 USB 主机注册驱动
输入参数	idx: USB 控制器的索引，总是为 0
	dev: 驱动结构体数组
	nr: 驱动的个数
返回值	无
使用示例	usb_hcd_register_driver(0, __host_driver, __nr_host_driver);

表 16-1 主机驱动注册函数

16. 2. 2 MSC驱动打开函数

类型	描述
函数原型	usbh_msc_inst_t *usbh_msc_driver_open(uint32_t drv, usbh_msc_cbk cbk);
功能描述	打开 MSC 驱动
输入参数	drv: 总是为 0
	cbk: MSC 驱动的回调函数
返回值	成功：MSC 驱动实例
	失败：NULL
使用示例	inst = usbh_msc_driver_open(0, usr_msc_cbk);

表 16-2 MSC 驱动打开函数

16. 2. 3 USB主机控制器初始化函数

类型	描述
函数原型	void usb_hcd_init(uint32_t idx, void *data, uint32_t size);
功能描述	USB 主机控制器初始化，所有主机程序都必须调用此函数
输入参数	idx: USB 控制器的索引，总是为 0
	data: 内存池，用来存放从机设备的配置描述符
	size: 内存池的大小(Bytes)
返回值	无
使用示例	usb_hcd_init(0, usbh_pool, USBH_POOL_LEN);

表 16-3 USB 主机控制器初始化函数

16. 2. 4 USB主机主处理函数

类型	描述
函数原型	void usb_hcd_main(void);
功能描述	USB 主机主处理函数，应用层需在主循环中调用
输入参数	无
返回值	无
使用示例	usb_hcd_main();

表 16-4 USB 主机主处理函数

16. 2. 5 查询MSC设备是否就绪函数

类型	描述
函数原型	int32_t usbh_msc_driver_ready(usbh_msc_inst_t *inst);
功能描述	查询 MSC 设备是否就绪
输入参数	inst: MSC 设备实例
返回值	0：MSC 设备已准备好
	其他值：MSC 设备未准备好
使用示例	if ((usbh_msc_driver_ready(inst)) != 0);

表 16-5 查询 MSC 设备是否就绪函数

16. 2. 6 读MSC设备扇区函数

类型	描述
函数原型	int32_t usbh_msc_block_read(usbh_msc_inst_t *inst, uint32_t lba, uint8_t *data, uint32_t nr_block);
功能描述	从 MSC 设备上读取扇区数据
输入参数	inst: MSC 设备实例
	lba: 起始扇区号
	data: 数据缓存区，
	nr_block: 扇区的个数
返回值	0：读取成功
	其他值：读取失败
使用示例	ret = usbh_msc_block_read(inst, 2, rx_buf, 1);

表 16-6 读 MSC 设备扇区函数

16. 2. 7 写MSC设备扇区函数

类型	描述
函数原型	int32_t usbh_msc_block_write(usbh_msc_inst_t *inst, uint32_t lba, uint8_t *data, uint32_t nr_block);
功能描述	向 MSC 设备上写入扇区数据
输入参数	inst: MSC 设备实例
	lba: 起始扇区号
	data: 数据缓存区，
	nr_block: 扇区的个数
返回值	0：写入成功
	其他值：写入失败
使用示例	ret = usbh_msc_block_write(inst, 2, tx_buf, 1);

表 16-7 写 MSC 设备扇区函数

16. 3 操作方法及演示效果

1. 工程路径：
   ~\ES32_SDK\Projects\ES32F36xx\Applications\USB\21_host_msc\
2. 编译工程，并将其下载到板子上；
3. 将 U 盘插到板子上(若板子是 OTG 接口则如要 OTG 转接头)；
4. MCU 开始对 U 盘设备进行枚举；
5. 枚举完成后，查询 MSC 设备是否准备好；
6. 如 MSC 设备已准备好，则进行一次读写操作。

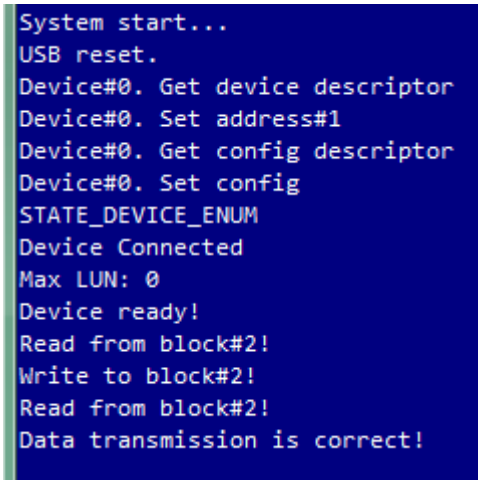


图 16-1 MCU 串口输出信息

第17章 鼠标主机(Mouse)

17. 1 功能概述

本例程为鼠标主机(Mouse)，能识别连接到 MCU 的鼠标设备，可以接收来自鼠标设备的输入：坐标移动量、左键、右键、中键的按下与释放。MCU 作为高速 USB 主机能适配各种速度的设备(LS/FS/HS)。

17. 2 主要API

17. 2. 1 主机驱动注册函数

类型	描述
函数原型	void usb_hcd_register_driver(uint32_t idx, const usbh_class_driver_t * const *drv, uint32_t nr);
功能描述	向 USB 主机注册驱动
输入参数	idx: USB 控制器的索引，总是为 0
	dev: 驱动结构体数组
	nr: 驱动的个数
返回值	无
使用示例	usb_hcd_register_driver(0, __host_driver, __nr_host_driver);

表 17-1 主机驱动注册函数

17. 2. 2 Mouse驱动打开函数

类型	描述
函数原型	usbh_mouse_t *usbh_mouse_open(usbh_mouse_cbk cbk, uint8_t *buf, uint32_t size);
功能描述	打开 Mouse 驱动
输入参数	cbk: Mouse 驱动的回调函数
	buf: 内存池，用来存放 HID 描述符；
	size: 内存池的大小
返回值	成功：Mouse 驱动实例
	失败：NULL
使用示例	mouse_inst = usbh_mouse_open(usr_mouse_cbk, rx_buf, 128);

表 17-2 Mouse 驱动打开函数

17. 2. 3 USB主机控制器初始化函数

类型	描述
函数原型	void usb_hcd_init(uint32_t idx, void *data, uint32_t size);
功能描述	USB 主机控制器初始化，所有主机程序都必须调用此函数
输入参数	idx: USB 控制器的索引，总是为 0
	data: 内存池，用来存放从机设备的配置描述符
	size: 内存池的大小(Bytes)
返回值	无
使用示例	usb_hcd_init(0, usbh_pool, USBH_POOL_LEN);

表 17-3 USB 主机控制器初始化函数

17. 2. 4 USB主机主处理函数

类型	描述
函数原型	void usb_hcd_main(void);
功能描述	USB 主机主处理函数，应用层需在主循环中调用
输入参数	无
返回值	无
使用示例	usb_hcd_main();

表 17-4 USB 主机主处理函数

17. 2. 5 鼠标驱动初始化函数

类型	描述
函数原型	uint32_t usbh_mouse_init(usbh_mouse_t *inst);
功能描述	初始化鼠标驱动
输入参数	inst: Mouse 设备实例
返回值	0：初始化成功
	其他值：初始化失败
使用示例	if ((usbh_mouse_init(mouse_inst)) != 0);

表 17-5 鼠标驱动初始化函数

17. 2. 6 鼠标驱动回调函数

类型	描述
函数原型	void usr_mouse_cbk(usbh_mouse_t *inst, uint32_t event, uint32_t param, void *data);
功能描述	鼠标驱动回调函数，通知应用层鼠标设备有数据更新
输入参数	inst: Mouse 设备实例
	event: Mouse 事件类型
	param: 事件携带的数值信息
	data: 事件携带的消息
返回值	无
使用示例	本函数为回调函数，应用层需要根据实际需求填充该函数

表 17-6 鼠标驱动回调函数

17.3 操作方法及演示效果

1. 工程路径：
   ~\ES32_SDK\Projects\ES32F36xx\Applications\USB\22_host_hid_mouse\
2. 编译工程，并将其下载到板子上；
3. 将鼠标插到板子上(若板子是 OTG 接口则如要 OTG 转接头)；
4. MCU 开始对鼠标设备进行枚举；
5. 枚举完成后，MCU 可以接收到鼠标设备发来的数据；

```
System start...
USB reset.
Device#0. Get device descriptor
Device#0. Set address#1
Device#0. Get config descriptor
Device#0. Set config
Device Connected
Mouse Ready!
Pos:[1, 0] Botton: 0x0
Pos:[1, -1] Botton: 0x0
Pos:[1, -2] Botton: 0x0
Pos:[1, -3] Botton: 0x0
Pos:[1, -4] Botton: 0x0
Pos:[1, -5] Botton: 0x0
Pos:[1, -5] Botton: 0x1
Pos:[1, -5] Botton: 0x0
Pos:[1, -5] Botton: 0x2
Pos:[1, -5] Botton: 0x0
Pos:[1, -6] Botton: 0x0
Pos:[2, -6] Botton: 0x0
Pos:[2, -7] Botton: 0x0
Pos:[3, -7] Botton: 0x0
Pos:[3, -8] Botton: 0x0
Pos:[3, -8] Botton: 0x4
Pos:[1, -8] Botton: 0x4
Pos:[1, -8] Botton: 0x0
Pos:[0, -8] Botton: 0x0
Pos:[-1, -8] Botton: 0x0
Pos:[-2, -8] Botton: 0x0
Pos:[-2, -9] Botton: 0x0
Pos:[-3, -9] Botton: 0x0
Pos:[-3, -11] Botton: 0x0
Pos:[-3, -12] Botton: 0x0
Pos:[-3, -14] Botton: 0x0
Pos:[-3, -15] Botton: 0x0
Pos:[-3, -16] Botton: 0x0
Pos:[-3, -16] Botton: 0x1
Pos:[-3, -16] Botton: 0x3
Pos:[-3, -16] Botton: 0x1
Pos:[-3, -16] Botton: 0x0
Pos:[-4, -16] Botton: 0x0
Pos:[-4, -17] Botton: 0x0
```

图 17-1 MCU 串口输出信息

第18章 键盘主机(Keyboard)

18. 1 功能概述

本例程为键盘主机(Keyboard)，能识别连接到 MCU 的键盘设备，可以接收来自键盘设备的输入，并能通知键盘修改灯的状态。MCU 作为高速 USB 主机能适配各种速度的设备(LS/FS/HS)。

18. 2 主要API

18. 2. 1 主机驱动注册函数

类型	描述
函数原型	void usb_hcd_register_driver(uint32_t idx, const usbh_class_driver_t * const *drv, uint32_t nr);
功能描述	向 USB 主机注册驱动
输入参数	idx: USB 控制器的索引，总是为 0
	dev: 驱动结构体数组
	nr: 驱动的个数
返回值	无
使用示例	usb_hcd_register_driver(0, __host_driver, __nr_host_driver);

表 18-1 主机驱动注册函数

18. 2. 2 Keyboard驱动打开函数

类型	描述
函数原型	usbh_keyb_t *usbh_keyb_open(usbh_keyb_cbk cbk, uint8_t *buf, uint32_t size);
功能描述	打开 Keyboard 驱动
输入参数	cbk: Keyboard 驱动的回调函数
	buf: 内存池，用来存放 HID 描述符；
	size: 内存池的大小
返回值	成功：Keyboard 驱动实例
	失败：NULL
使用示例	keyb_inst = usbh_keyb_open(usr_keyb_cbk, rx_buf, 128);

表 18-2 Keyboard 驱动打开函数

18. 2. 3 USB主机控制器初始化函数

类型	描述
函数原型	void usb_hcd_init(uint32_t idx, void *data, uint32_t size);
功能描述	USB 主机控制器初始化，所有主机程序都必须调用此函数
输入参数	idx: USB 控制器的索引，总是为 0
	data: 内存池，用来存放从机设备的配置描述符
	size: 内存池的大小(Bytes)
返回值	无
使用示例	usb_hcd_init(0, usbh_pool, USBH_POOL_LEN);

表 18-3 USB 主机控制器初始化函数

18. 2. 4 USB主机主处理函数

类型	描述
函数原型	void usb_hcd_main(void);
功能描述	USB 主机主处理函数，应用层需在主循环中调用
输入参数	无
返回值	无
使用示例	usb_hcd_main();

表 18-4 USB 主机主处理函数

18. 2. 5 Keyboard驱动初始化函数

类型	描述
函数原型	uint32_t usbh_keyb_init(usbh_keyb_t *inst);
功能描述	初始化键盘驱动
输入参数	inst: Keyboard 设备实例
返回值	0：初始化成功
	其他值：初始化失败
使用示例	if ((usbh_keyb_init(keyb_inst)) != 0);

表 18-5 Keyoard 驱动初始化函数

18. 2. 6 Keyboard驱动回调函数

类型	描述
函数原型	void usr_keyb_cbk(usbh_keyb_t *inst, uint32_t event, uint32_t param, void *data);
功能描述	键盘驱动回调函数，通知应用层键盘设备有数据更新
输入参数	inst: Keyboard 设备实例
	event: Keyboard 事件类型
	param: 事件携带的数值信息
	data: 事件携带的消息
返回值	无
使用示例	本函数为回调函数，应用层需要根据实际需求填充该函数

表 18-6 Keyboard 驱动回调函数

18. 2. 7 设置按键状态函数

类型	描述
函数原型	uint32_t usbh_keyb_modifier_set(usbh_keyb_t *inst, uint32_t modifier);
功能描述	设置键盘按键状态，主要用来修改键盘大小写灯的状态
输入参数	inst: Keyboard 设备实例
	modifier: 需要修改的屏蔽位，主要参数为： HID_KEYB_NUM_LOCK HID_KEYB_CAPS_LOCK HID_KEYB_SCROLL_LOCK
返回值	成功：0
	失败：非 0 值
使用示例	usbh_keyb_modifier_set(keyb_inst, env.modify);

表 18-7 设置按键状态函数

18. 3 操作方法及演示效果

1. 工程路径：
   ~\ES32_SDK\Projects\ES32F36xx\Applications\USB\23_host_hid_keyb\
2. 编译工程，并将其下载到板子上；
3. 将键盘插到板子上(若板子是 OTG 接口则如要 OTG 转接头)；
4. MCU 开始对键盘设备进行枚举；
5. 枚举完成后，MCU 可以接收到键盘设备发来的数据；

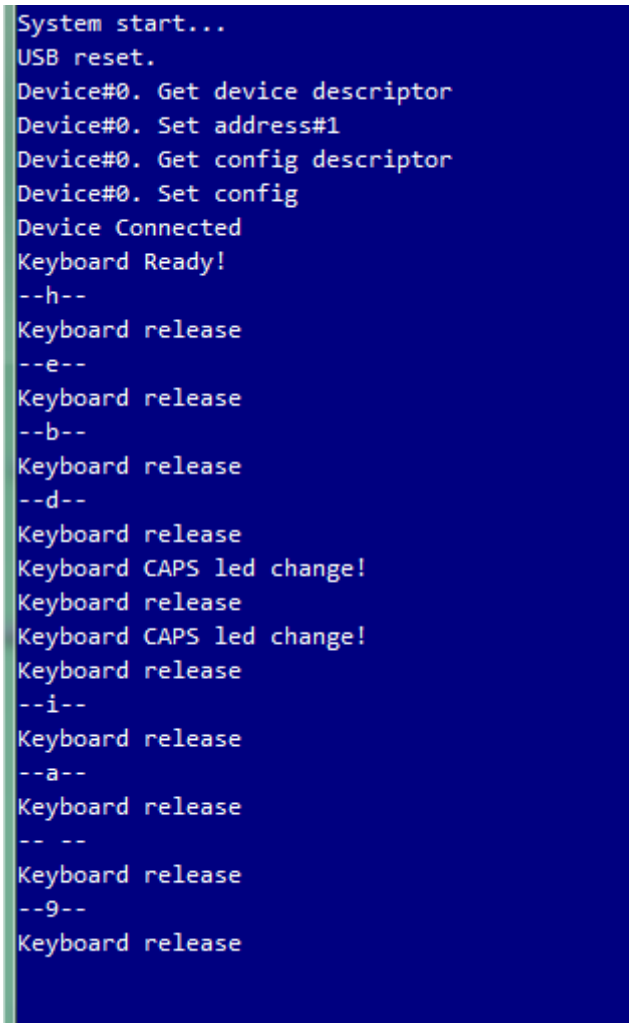


图 18-1 MCU 串口输出信息

第19章 块设备主机(BULK)

19. 1 功能概述

本例程为块主机(BULK)，能识别连接到 MCU 的块设备，可以接收来自块设备的数据，并能向块设备发送数据。MCU 作为高速 USB 主机能适配各种速度的设备(LS/FS/HS)。

若 MCU 是 FS-USB，则只支持 USB1.1 设备；若是 HS-USB 则可支持 USB2.0 设备。

19. 2 主要API

19. 2. 1 主机驱动注册函数

类型	描述
函数原型	void usb_hcd_register_driver(uint32_t idx, const usbh_class_driver_t * const *drv, uint32_t nr);
功能描述	向 USB 主机注册驱动
输入参数	idx: USB 控制器的索引，总是为 0
	dev: 驱动结构体数组
	nr: 驱动的个数
返回值	无
使用示例	usb_hcd_register_driver(0, __host_driver, __nr_host_driver);

表 19-1 主机驱动注册函数

19. 2. 2 BULK驱动打开函数

类型	描述
函数原型	usbh_bulk_inst_t *usbh_bulk_driver_open(usbh_bulk_cbk cbk);
功能描述	打开 BULK 驱动
输入参数	cbk: BULK 驱动的回调函数
返回值	成功：BULK 驱动实例
	失败：NULL
使用示例	inst = usbh_bulk_driver_open(usr_bulk_cbk);

表 19-2 BULK 驱动打开函数

19. 2. 3 USB主机控制器初始化函数

类型	描述
函数原型	void usb_hcd_init(uint32_t idx, void *data, uint32_t size);
功能描述	USB 主机控制器初始化，所有主机程序都必须调用此函数
输入参数	idx: USB 控制器的索引，总是为 0
	data: 内存池，用来存放从机设备的配置描述符
	size: 内存池的大小(Bytes)
返回值	无
使用示例	usb_hcd_init(0, usbh_pool, USBH_POOL_LEN);

表 19-3 USB 主机控制器初始化函数

19. 2. 4 USB主机主处理函数

类型	描述
函数原型	void usb_hcd_main(void);
功能描述	USB 主机主处理函数，应用层需在主循环中调用
输入参数	无
返回值	无
使用示例	usb_hcd_main();

表 19-4 USB 主机主处理函数

19. 2. 5 查询目标设备速度函数

类型	描述
函数原型	uint32_t usbh_bulk_target_speed_get(usbh_bulk_inst_t *inst);
功能描述	查询目标设备速度
输入参数	inst: BULK 设备实例
返回值	目标设备速度
	0xFFFFFFFF：查询失败
使用示例	ret = usbh_bulk_target_speed_get(inst);

表 19-5 查询目标设备速度函数

19. 2. 6 读BULK设备数据函数

类型	描述
函数原型	int32_t usbh_bulk_read(usbh_bulk_inst_t *inst, uint8_t *data, uint32_t size);
功能描述	从 BULK 设备上读取数据
输入参数	inst: BULK 设备实例
	data: 数据缓存区
	size: 数据缓存区大小
返回值	读到的数据个数(Bytes)
	-1: 读取失败
使用示例	ret = usbh_bulk_read(inst, rx_buf, len);

表 19-6 读 BULK 设备数据函数

19. 2. 7 写BULK设备数据函数

类型	描述
函数原型	int32_t usbh_bulk_write(usbh_bulk_inst_t *inst, uint8_t *data, uint32_t size);
功能描述	向 BULK 设备上写入数据
输入参数	inst: BULK 设备实例
	data: 数据缓存区
	size: 数据缓存区大小
返回值	写入的数据个数(Bytes)
	-1: 写入失败
使用示例	ret = usbh_bulk_write(inst, tx_buf, len);

表 19-7 写 BULK 设备数据函数

19. 3 操作方法及演示效果

1. 工程路径：
   ~\ES32_SDK\Projects\ES32F36xx\Applications\USB\24_host_bulk\
2. 编译工程，并将其下载到板子上；
3. 将 BULK 设备插到板子上(若板子是 OTG 接口则如要 OTG 转接头)；
4. MCU 开始对 BULK 设备进行枚举；
5. 枚举完成后，MCU 可以读写 BULK 设备；

```
System start...
USB reset.
Device#0. Get device descriptor
Device#0. Set address#1
Device#0. Get config descriptor
Device#0. Set config
STATE_DEVICE_ENUM
Device Connected
Device ready!
--------------1---------------
Write to device, size#512
Read from device, size#512
Data transmission is correct!
--------------2---------------
Write to device, size#512
Read from device, size#512
Data transmission is correct!
--------------3---------------
Write to device, size#512
Read from device, size#512
Data transmission is correct!
--------------4---------------
Write to device, size#512
Read from device, size#512
Data transmission is correct!
```

图 19-1 MCU 串口输出信息

第20章 通用主机(HUB)

20. 1 功能概述

本例程为通用主机，通过 HUB 外扩接口，能同时连接鼠标设备、键盘设备、U 盘设备。各设备之间相互独立运行，不受彼此干扰。MCU 作为 USB 主机能适配各种速度的设备(LS/FS/HS)。不同速度的设备在 HUB 端进行转换，HUB 与 MCU 之间为高速通道。

本例程仅支持在具有高速 USB 接口的 MCU 上使用。

20. 2 主要API

本例程没有新的 API 函数，各模块 API 请参考：

16.2 章节 MSC 模块的 API；

17.2 章节 Mouse 模块的 API；

18.2 章节 Keyboard 模块的 API；

相互组合关系，请参考本例程源代码。

20. 3 操作方法及演示效果

1. 工程路径：
   ~\ES32_SDK\Projects\ES32F36xx\Applications\USB\25_host_hub_keyb_mouse_msc\
2. 编译工程，并将其下载到板子上；
3. 将 USB2.0 HUB 设备插到板子上(若板子是 OTG 接口则如要 OTG 转接头)；
4. 在 HUB 上依次插入 U 盘设备、键盘设备、鼠标设备
5. 各设备可以独立工作；
6. MCU 端串口输出信息如下：

```
System start...
USB reset.
Device#0. Get device descriptor
Device#0. Set address#1
Device#0. Get config descriptor
Device#0. Set config
Device Connected
Connect change on port 4
Connected
Device connect at port 4
Start enumeration for port 4
Reset deassert for port 4
Connection from hub 1, port 4.
Allocating device 1
Device#1. Get device descriptor
Device#1. Set address#2
Device#1. Get config descriptor
Device#1. Set config
STATE_DEVICE_ENUM
Device Connected
Enumeration complete for hub 1, port 4
Max LUN: 0
Device ready!
Read block#2!
Write block#2!
Read block#2!
Connect change on port 3
Connected
Device connect at port 3
Start enumeration for port 3
Reset deassert for port 3
Connection from hub 1, port 3.
Allocating device 2
Device#2. Get device descriptor
Device#2. Set address#3
Device#2. Get config descriptor
Device#2. Set config
Device Connected
Enumeration complete for hub 1, port 3
Keyboard Ready!
--h--
Keyboard release
```

图 20-1 MCU 串口输出信息 1

```
Enumeration complete for hub 1, port 3
Keyboard Ready!
--h--
Keyboard release
--j--
Keyboard release
Connect change on port 1
Connected
Device connect at port 1
Start enumeration for port 1
Reset deassert for port 1
Connection from hub 1, port 1.
Allocating device 3
Device#3. Get device descriptor
Device#3. Set address#4
Device#3. Get config descriptor
Device#3. Set config
Device Connected
Enumeration complete for hub 1, port 1
Mouse Ready!
Pos:[0, 0] Botton: 0x1
Pos:[0, 0] Botton: 0x0
Pos:[-1, 0] Botton: 0x0
Pos:[-2, 0] Botton: 0x0
Pos:[-4, 0] Botton: 0x0
Pos:[-4, 0] Botton: 0x4
Pos:[-5, 0] Botton: 0x4
Pos:[-5, 1] Botton: 0x4
Pos:[-6, 1] Botton: 0x4
Pos:[-6, 2] Botton: 0x4
Pos:[-7, 2] Botton: 0x4
Pos:[-7, 5] Botton: 0x4
Pos:[-10, 5] Botton: 0x4
Pos:[-10, 10] Botton: 0x4
Pos:[-11, 10] Botton: 0x4
Pos:[-11, 16] Botton: 0x4
Pos:[-12, 16] Botton: 0x4
Pos:[-12, 21] Botton: 0x4
Pos:[-15, 21] Botton: 0x4
Pos:[-15, 28] Botton: 0x4
Pos:[-16, 28] Botton: 0x4
Pos:[-16, 31] Botton: 0x4
Pos:[-16, 35] Botton: 0x4
Pos:[-16, 36] Botton: 0x4
Pos:[-16, 36] Botton: 0x0
```

图 20-2 MCU 串口输出信息 2

第21章 OTG模式(OTG)

21. 1 功能概述

本例程为 OTG 模式，将该例程分别下载到 2 块板子上，将其中一块板子的 ID 接高电平(USB 从机)，另一块板子的 ID 接低电平(USB 主机)。两者使用 Micro-USB 线互联，可构成 USB 主从系统。

MCU 作为高速 USB 主机能适配各种速度的设备(LS/FS/HS)。若 MCU 为全速 USB 主机则只支持(LS/FS)设备。

21. 2 主要API

21. 2. 1 USB模式设置函数

类型	描述
函数原型	void usb_mode_set(uint32_t idx, usb_mode_t mode, usb_mode_cbk cbk);
功能描述	设置 USB 控制器模式
输入参数	idx: USB 控制器的索引，总是为 0
	mode: 待设置的模式
	cbk: 模式变化的回调函数
返回值	无
使用示例	usb_mode_set(0, USB_LIB_MODE_NONE, usb_otg_mode_cbk);

表 21-1 USB 模式设置函数

21. 2. 2 OTG模式初始化函数

类型	描述
函数原型	void usb_mode_otg_init(uint32_t idx, uint32_t rate, void *pool, uint32_t size);
功能描述	OTG 模式初始化
输入参数	idx: USB 控制器的索引，总是为 0
	rate: 轮询的速率
	pool: 内存池，存放枚举过程中的配置描述符
	size: 内存池的大小
返回值	无
使用示例	usb_mode_otg_init(0, 100, usbh_pool, USBH_POOL_LEN);

表 21-2 OTG 模式初始化函数

21. 2. 3 OTG模式主函数

类型	描述
函数原型	void usb_mode_otg_main(void);
功能描述	OTG 模式主函数，需放在应用层大循环中一直调用
输入参数	无
返回值	无
使用示例	usb_mode_otg_main();

表 21-3 OTG 模式主函数

21. 3 操作方法及演示效果

1. 工程路径：
   ~\ES32_SDK\Projects\ES32F36xx\Applications\USB\31_otg_bulk\
2. 编译工程，将该例程分别下载到 2 块板子上，将其中一块板子的 ID 接高电平(USB 从机)，另一块板子的 ID 接低电平(USB 主机)；
3. 两者使用 Micro-USB 线互联；
4. 若板子是 OTG 接口则忽略第二步，OTG 接口会自动将其中一端的 ID 拉低，另一端的 ID 拉高，从而构成主从系统；
5. 主机端开始对设备端进行枚举；
6. 主机端串口输出如下：

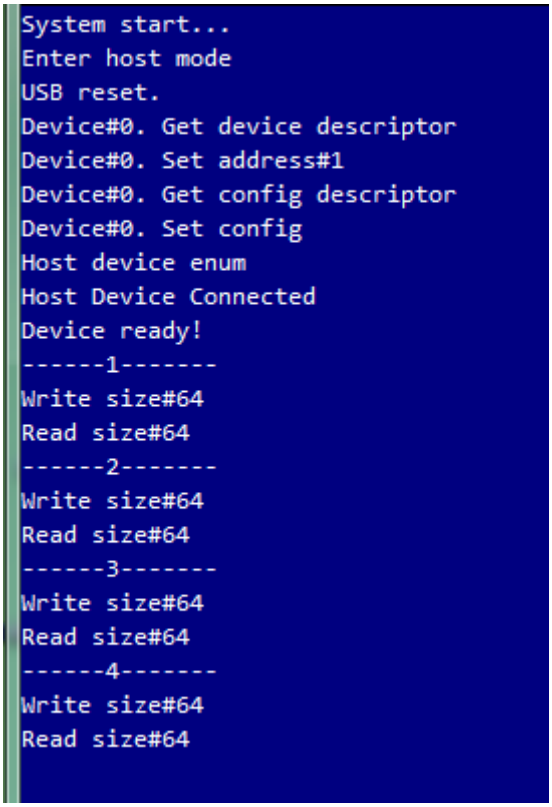


图 21-1 主机端串口输出信息

7. 设备端串口输出如下：

```
System start...
Enter device mode
Set configure: 1
BULK Ready!
Recv size#64:
0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51
51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x
 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51 0x51
Send size#64
Recv size#64:
0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52
52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x
 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52 0x52
Send size#64
Recv size#64:
0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53
53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x
 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53 0x53
Send size#64
Recv size#64:
0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54
54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x
 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54 0x54
Send size#64
```

图 21-2 设备端串口输出信息

第22章 眼图测试(Eye Diagram)

22. 1 功能概述

本例程为高速 USB 主机眼图测试例程。将该例程下载到支持高速 USB 的 MCU 上，在接入专业眼图测试仪器后，可测试眼图。

22. 2 主要API

本例程无 API。

22. 3 操作方法及演示效果

1. 工程路径：
   ~\ES32_SDK\Projects\ES32F36xx\Applications\USB\41_test_eye_diagram\
2. 编译工程，将该例程下载到支持高速 USB 的 MCU 上；
3. 接入专业眼图测试仪器；
4. 可观察到眼图：

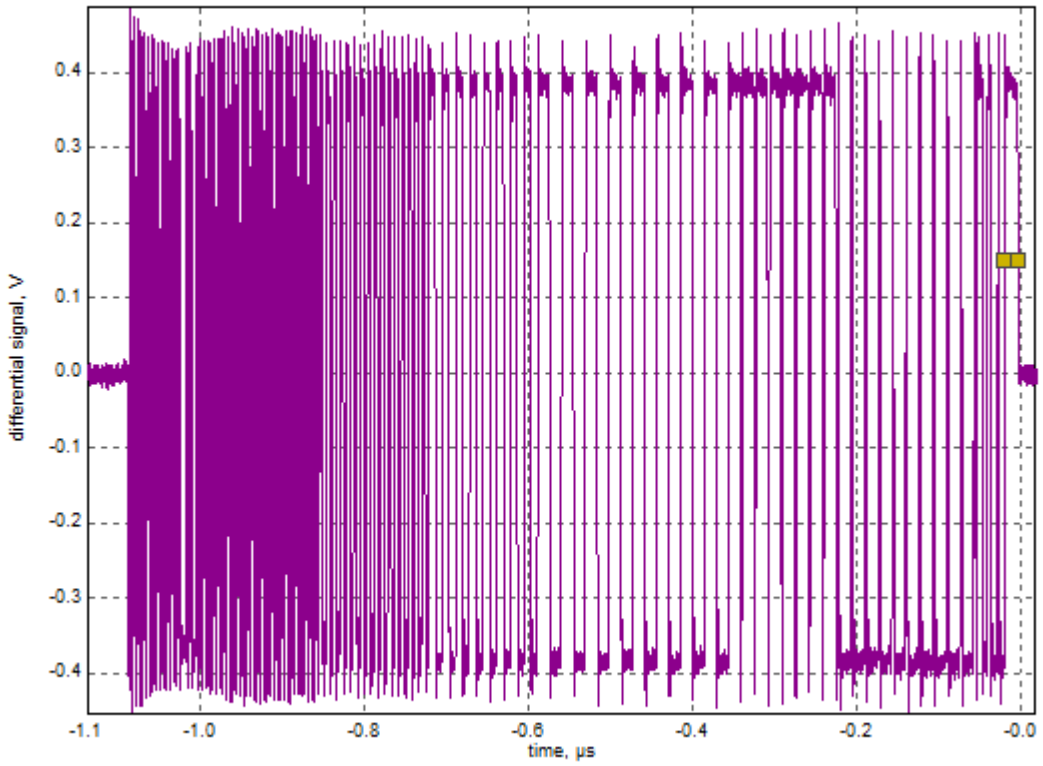


图 22-1 MCU 输出波形

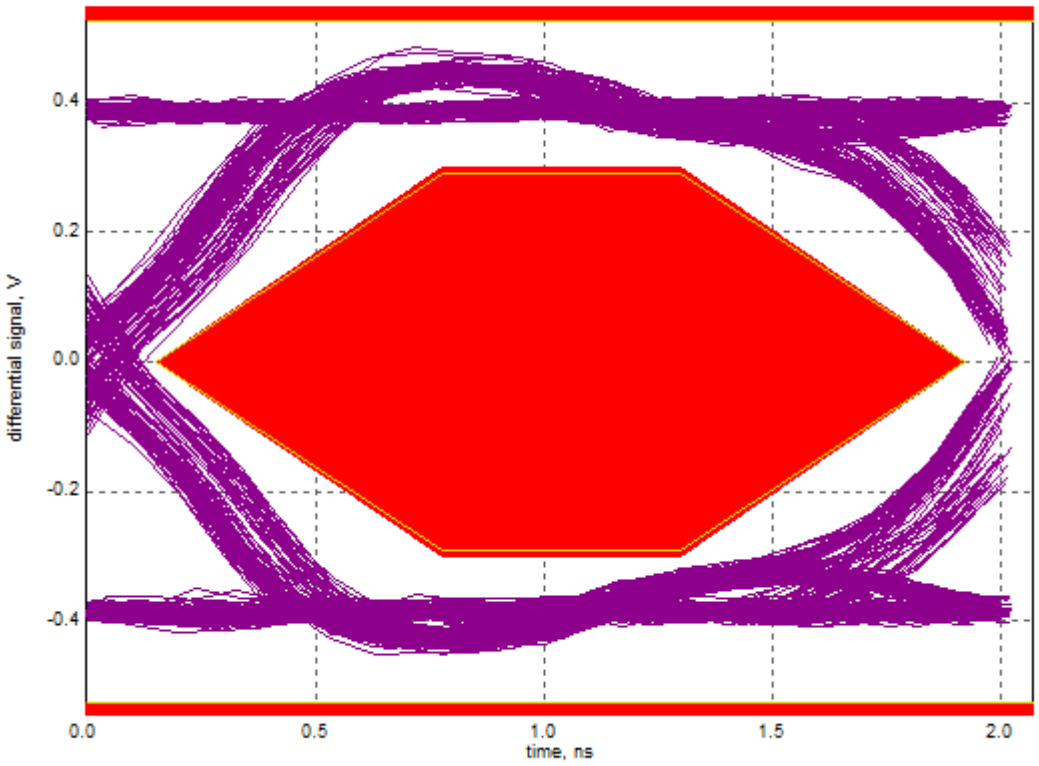


图 22-2 眼图