GigaDevice Semiconductor Inc.

GD32 USBD 固件库使用指南

应用笔记

AN049

目录

图索引

表索引

1. 前言

本文基于 GD32 MCU 产品的 USBD 模块，分析 GD32 USBD 固件库架构，简要描述了固件库函数的功能，通过具体的应用实例，阐明部分 USBD 设备类的实现过程，为用户的后续开发提供借鉴。

本文主要分为两部分：固件库说明和应用类协议及例程说明。在固件库说明章节中，包含着 main.c 文件及函数说明和 usbd_driver 底层文件及库函数说明。在应用类协议及例程说明章节中，按照 USB 协议，GD32 MCU USBD 模块支持四种数据传输类型：中断传输、批量传输、控制传输和同步传输，如*表 1-1. USBD 示例*所示，在后文中，通过依次介绍 HID 设备、CDC 设备、DFU 设备和 UAC 设备，展示 USBD 设备的应用类协议、描述符、应用类请求、数据传输和输出结果。

表 1-1. USBD 示例

DEMO 名称	USB 传输类型	基本功能
standard_hid_keyboard	中断传输	枚举为键盘，打印字符
cdc_acm	批量传输	枚举为虚拟串口，收发数据
dev_firmware_update	控制传输	枚举为 DFU 设备，升级固件
audio_headphone	同步传输	枚举为音频设备，播放音乐

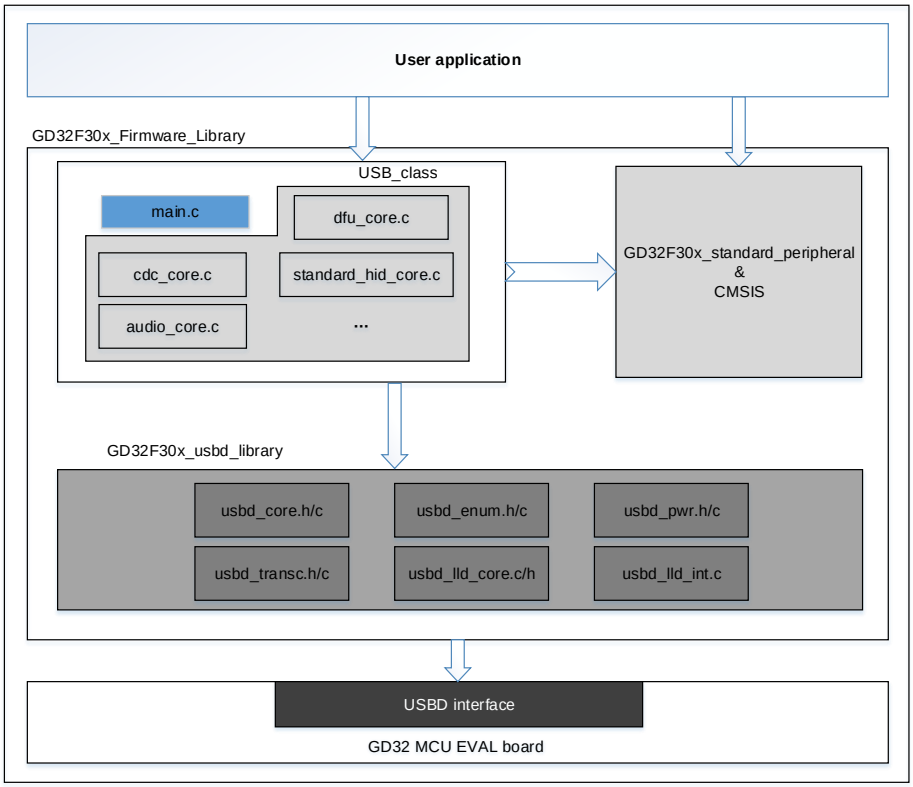
本文所适用的产品系列如*表 1-2. 适用产品*所示，在本文中，以 GD32F303xx 为例，其他各系列产品的固件库和应用实例与该系列相似。

表 1-2. 适用产品

类型	型号
MCU	GD32F103xx 系列
	GD32F150xx 系列
	GD32F303xx 系列
	GD32E503xx 系列
	GD32EPRTxx 系列
	GD32L23x 系列

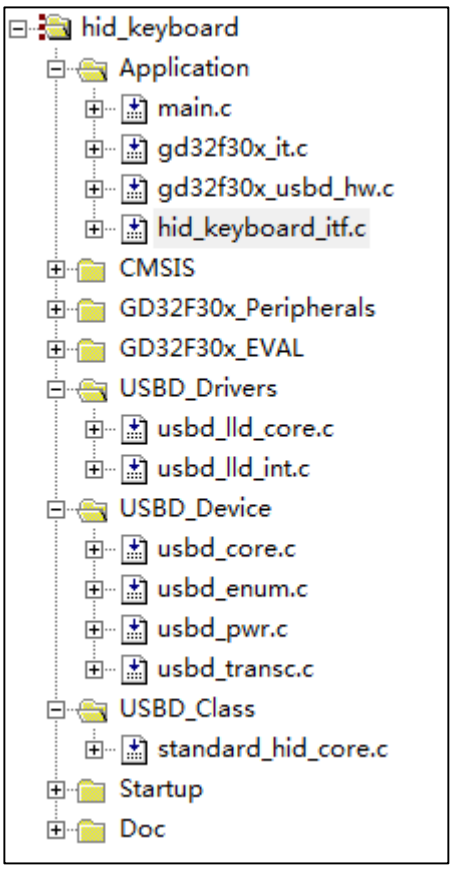
2. 固件库说明

图 2-1. 固件库框架



GD32F30x 系列固件库如*图 2-1. 固件库框架*所示。用户应用程序(User application)调用 GD32 全速 USB 设备固件库中的接口实现 USB 设备与主机之间的通信，架构的最底层为 GD32 MCU 开发板的硬件层。其中，GD32 全速 USB 设备固件库(GD32F30x_usbd_Library)分为两层，顶层为应用接口层，用户可以修改，包含 main.c 和 USB 相关设备类驱动；底层为 USBD 设备驱动层，不建议用户修改，该驱动层包含实现 USB 通信相关协议以及 USBD 底层模块操作。

图 2-2. USBD 工程框架图



以 HID 键盘例程为例，其工程结构如*图 2-2. USBD 工程框架图*所示，除了通用的外设库、启动文件和开发板硬件库文件外，需要调用 USBD 固件库的底层文件，如 usbd_lld_core.c、usbd_enum.c 等，这部分库函数是相对固定的，不建议用户修改。应用接口层文件，如 standard_hid_core.c、main.c 等，用户可根据应用的实际需求对其进行修改。

2.1. main.c 文件及函数说明

表 2-1. Main 函数

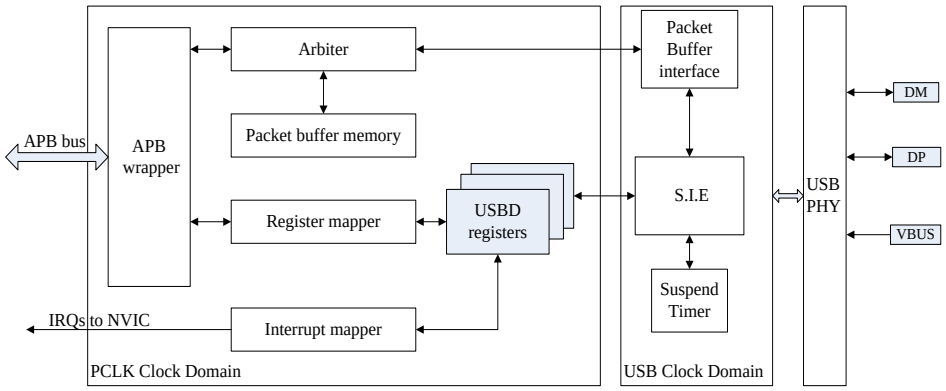
```
int main(void)
{
    /* system clocks configuration */
    rcu_config();

    /* GPIO configuration */
    gpio_config();

    hid_itfop_register (&usb_hid, &fop_handler);

    /* USB device configuration */
    usbd_init(&usb_hid, &hid_desc, &hid_class);
```

```
    /* NVIC configuration */
    nvic_config();

    usbd_connect(&usb_hid);

    while(USBD_CONFIGURED != usb_hid.cur_status){
    }

    while (1) {
        fop_handler.hid_itf_data_process(&usb_hid);
    }
}
```

如*表 2-1. Main 函数*所示，在用户 main 函数中，需要配置时钟、USBD 相关引脚、中断优先级、USBD 模块初始化等函数，在执行完 usbd_connect 函数后，MCU 需要等待与主机之间枚举交互完成，然后再执行相关应用的操作。

2.1.1. 时钟配置

图 2-3. USBD 时钟域



如*图 2-3. USBD 时钟域*所示，配置 USBD 寄存器等操作是在 PCLK 时钟下完成的，而 USBD 模块与主机进行数据交互需基于 USBD 时钟域（48MHz 时钟）完成。

表 2-2. USBD 系统时钟和分频

产品系列名称	适用 USBD 系统时钟	分频系数
GD32F103xx	48/72/96 MHz	1 / 1.5 / 2 分频
GD32F150xx	48/72 MHz	1 / 1.5 分频
GD32F303xx	48/72/96/120 MHz	1 / 1.5 / 2 / 2.5 分频
GD32E503xx / GD32EPRTxx	48/72/96/120/144/168 MHz	1 / 1.5 / 2 / 2.5 / 3 / 3.5 分频
GD32L23xx	48MHz	无

在应用实践中，如*表 2-2. USBD 系统时钟和分频*所示，用户通常会把系统时钟配置为 24MHz 的整数倍，通过分频系数，为 USBD 的控制器提供 48MHz 时钟。GD32L23xx、GD32F303xx、GD32EPRTxx 和 GD32E503xx 产品系列支持 IRC48M 时钟，可以使用 CTC 模块基于外部高精度的参考信号源来校准 IRC48M 的时钟频率，从而为 USBD 模块提供 48MHz 时钟。

表 2-3. RCU 配置代码

```
void rcu_config(void)
{
    uint32_t system_clock = rcu_clock_freq_get(CK_SYS);

    /* enable USB pull-up pin clock */
    rcu_periph_clock_enable(RCC_AHBPeriph_GPIO_PULLUP);

    if (48000000U == system_clock) {
        rcu_usb_clock_config(RCU_CKUSB_CKPLL_DIV1);
    } else if (72000000U == system_clock) {
        rcu_usb_clock_config(RCU_CKUSB_CKPLL_DIV1_5);
    } else if (96000000U == system_clock) {
        rcu_usb_clock_config(RCU_CKUSB_CKPLL_DIV2);
```

```
    } else if (120000000U == system_clock) {
        rcu_usb_clock_config(RCU_CKUSB_CKPLL_DIV2_5);
    } else {
        /* reserved */
    }

    /* enable USB APB1 clock */
    rcu_periph_clock_enable(RCU_USBD);
}
```

如*表 2-3. RCU 配置代码*所示，在多数产品系列中，48M 时钟来源于系统时钟依据相应分频系数分频所得，如在后文的例程中，系统时钟被配置为 120MHz，对其进行 2.5 倍分频，即可获取 USBD 时钟域所需要的 48MHz 时钟。

2.1.2. GPIO 配置

表 2-4. GPIO 配置代码

```
void gpio_config(void)
{
    /* configure usb pull-up pin */
    gpio_init(USB_PULLUP,          GPIO_MODE_OUT_PP,          GPIO_OSPEED_50MHZ,
USB_PULLUP_PIN);

    /* USB wakeup EXTI line configuration */
    exti_interrupt_flag_clear(EXTI_18);
    exti_init(EXTI_18, EXTI_INTERRUPT, EXTI_TRIG_RISING);
}
```

如*表 2-4. GPIO 配置代码*所示，依据 USB 协议，DP 线连接上拉电阻，需要通过控制 USB_PULLUP_PIN 引脚完成上拉动作，从而保证被识别为 USB 全速设备。对于 L23xx 系列，DP 的上拉动作由配置 USBD_DPC 寄存器实现。此外，如果用户需要实现 USBD 唤醒功能，需要配置 EXTI_18 作为触发源。

2.1.3. 中断配置

表 2-5. 中断配置代码

```
void nvic_config(void)
{
    /* 2 bits for preemption priority, 2 bits for subpriority */
    nvic_priority_group_set(NVIC_PRIGROUP_PRE1_SUB3);

    /* enable the USB low priority interrupt */
    nvic_irq_enable((uint8_t)USBD_LP_CAN0_RX0_IRQn, 1U, 0U);

    /* enable the USB Wake-up interrupt */
    nvic_irq_enable((uint8_t)USBD_WKUP_IRQn, 0U, 0U);
}
```

如*表 2-5. 中断配置代码*所示，在 USBD 相关应用中，鉴于 USBD 中断被频繁调用，需要尽可能保证 USBD 的中断不被长时间阻塞，否则会导致 USBD 数据传输异常，因此，USBD 的中断优先级尽可能高，从而保证总线不被别的中断长时间抢占。

2.1.4. 应用配置

表 2-6. 应用配置代码

```
static void hid_key_data_send(usb_dev *udev)
{
    standard_hid_handler *hid =
(standard_hid_handler*)udev->class_data[USBD_HID_INTERFACE];
    if (hid->prev_transfer_complete) {
        switch (key_state()) {
        case CHAR_A:
            hid->data[2] = 0x04U;
            break;
        case CHAR_B:
            hid->data[2] = 0x05U;
            break;
        case CHAR_C:
            hid->data[2] = 0x06U;
            break;
        default:
            break;
        }
        if (0U != hid->data[2]) {
            hid_report_send(udev, hid->data, HID_IN_PACKET);
        }
    }
}
```

在 USBD 应用实例中，如*表 2-6. 应用配置代码*所示，USBD 设备枚举完成后，需要调用 MCU 其他模块，实现应用数据更新，进而通过 USBD 端点进行输入或输出。如上文代码所示，通过按下不同按键，hid->data[2]被赋予不同数值，以报文的形式发送数据。程序调用 usbd_ep_send 函数，将待发送数据写入 USBD buffer RAM 中，并修改端点状态为有效，当 USBD 设备收到 Host 下发的 IN 令牌包后，将数据传输给 Host。当数据发送完成后，MCU 进入 USBD_LP_CAN0_TX_IRQHandler，代码执行到 IN 事务处理分支 udev->ep_transc[ep_num][TRANSC_IN](udev, ep_num)，即 data in 事务处理函数，表明此次 IN 传输完成。在 data in 事务处理函数中，用户可以添加相应处理，如置位发送完成标志等。

在其他应用中，可能会存在 OUT 方向的数据传输流程，调用 usbd_ep_recev 函数，MCU 进入 USBD_LP_CAN0_RX0_IRQHandler，代码执行到 OUT 事务处理分支，udev->ep_transc[ep_num][TRANSC_OUT](udev, ep_num)，即 data out 事务处理函数。在 data out 事务处理函数中，用户可以处理接收到的数据或者置位接收完成标志等。

2.2. usbd_driver 底层文件及库函数说明

usbd_driver 设备驱动层包含两个文件夹，分别为 Include 和 Source，其中，Include 为底层头文件，Source 为底层源文件。该设备驱动层文件说明如*表 2-7. 设备驱动层文件说明列表*所示。

表 2-7. 设备驱动层文件说明列表

文件名称	说明
usbd_core.h/.c	USBD 设备驱动核心接口层驱动
usbd_enum.h/c	USBD 设备枚举函数驱动
usbd_pwr.h/c	USBD 设备电源管理驱动
usbd_transc.h/c	USB 事务传输驱动
usb_ch9_std.h	USB2.0 协议第 9 章定义

其中，usbd_core.h/c 文件中的库函数说明如*表 2-8. usbd_core.h/c 库函数说明列表*所示。

表 2-8. usbd_core.h/c 库函数说明列表

文件名称	说明
usbd_init	设备核心寄存器初始化
usbd_ep_send	端点准备发送数据
usbd_ep_recev	端点准备接收数据
usbd_connect	设备连接
usbd_disconnect	设备断开连接
usbd_core_deinit	设备核心寄存器去初始化
usbd_ep_init	端点初始化
usbd_ep_deinit	端点去初始化
usbd_ep_stall	设置端点为 STALL 状态
usbd_ep_clear_stall	清除端点 STALL 状态
usbd_ep_status_get	获取端点状态

usbd_transc.h/.c 文件中的库函数说明如*表 2-9. usbd_transc.c 文件中的库函数说明列表*所示。

表 2-9. usbd_transc.c 文件中的库函数说明列表

文件名称	说明
_usb_setup_transc	USB SETUP 事务处理
_usb_out0_transc	端点 0 数据 OUT 阶段处理
_usb_in0_transc	端点 0 数据 IN 阶段处理
usb_stall_transc	USB STALL 事务
usb_ctl_status_in	控制传输状态 IN 阶段
usb_ctl_data_in	控制传输数据 IN 阶段
usb_ctl_out	控制传输数据或者状态 OUT 阶段
usb_0len_packet_send	USB 发送 0 长度的数据包

usbd_pwr.h/.c 文件中的库函数说明如*表 2-10. usbd_pwr.h/.c 文件中的库函数说明列表*所

示。

表 2-10. usbd_pwr.h/.c 文件中的库函数说明列表

文件名称	说明
resume_mcu	MCU 唤醒函数
usbd_remote_wakeup_active	启动远程唤醒
usbd_suspend	设置 USB 设备到挂起模式

usbd_enum.h/.c 文件中的库函数说明如*表 2-11. usbd_enum.h/.c 文件中的库函数说明列表*所示。

表 2-11. usbd_enum.h/.c 文件中的库函数说明列表

文件名称	说明
usbd_standard_request	处理 USB 标准设备请求
usbd_class_request	处理 USB 设备类请求
usbd_vendor_request	处理 USB 供应商的请求
_usb_std_reserved	保留，无操作
_usb_dev_desc_get	获取设备描述符
_usb_config_desc_get	获取配置描述符
_usb_str_desc_get	获取字符串描述符
_usb_bos_desc_get	获取 BOS 描述符
_usb_std_getstatus	处理 Get_Status 请求
_usb_std_clearfeature	处理 USB Clear_Feature 请求
_usb_std_setfeature	处理 USB Set_Feature 请求
_usb_std_setaddress	处理 USB Set_Address 请求
_usb_std_getdescriptor	处理 USB Get_Descriptor 请求
_usb_std_setdescriptor	处理 USB Set_Descriptor 请求
_usb_std_getconfiguration	处理 USB Get_Configuration 请求
_usb_std_setconfiguration	处理 USB Set_Configuration 请求
_usb_std_getinterface	处理 USB Get_Interface 请求
_usb_std_setinterface	处理 USB Set_Interface 请求
_usb_std_synchframe	处理 USB SynchFrame 请求
int_to_unicode	将十六进制的 32 位值转换为 unicode 字符
serial_string_get	获取序列号字符串

3. 应用类协议及例程说明

3.1. HID

3.1.1. 协议简介

HID(Human Interface Device，人机接口设备)是 USB 设备中常用的设备类型，此类设备非常繁多，如 USB 鼠标、USB 键盘、USB 游戏操纵杆等。HID 设备枚举阶段使用控制传输，而应用数据传输阶段，采用中断传输，中断时间间隔由后文端点描述符的 bInterval 域配置。

HID 设备的描述符除了标准描述符外，还支持三个 HID 设备类特定的描述符：HID 描述符、报文描述符、实体描述符，前两种描述符后文有所描述。HID 描述符和接口描述符关联，包含 HID 规范的版本号，报表描述符的长度等。报文描述符较为复杂，没有固定长度，定义设备输入输出数据格式等。实体描述符是可选的，用来描述设备的行为特性。

3.1.2. 描述符解析

本章节展示 HID 键盘的配置描述符、接口描述符、HID 描述符、端点描述符和报文描述符。

图 3-1. HID 配置描述符

Configuration descriptor

Name	Value	Dec	Hex	Bin
bLength	Valid	9	0x09	00001001
bDescriptorType	CONFIGURATION	2	0x02	00000010
wTotalLength	34 bytes	34	0x0022	00000000 00100010
bNumInterface	1	1	0x01	00000001
bConfigurationValue	1	1	0x01	00000001
iConfiguration	0	0	0x00	00000000
bmAttributes. Reserved	Zero	0	0x00	00000
bmAttributes. RemoteWakeup	Supported	1	0x1	1
bmAttributes. SelfPowered	No, Bus Powered	0	0x0	0
bmAttributes. Reserved7	One	1	0x1	1
bMaxPower	100 mA	50	0x32	00110010

配置描述符定义配置描述符集长度、接口数量和供电特性等。

图 3-2. HID 接口描述符

Interface descriptor				
Name	Value	Dec	Hex	Bin
bLength	Valid	9	0x09	00001001
bDescriptorType	INTERFACE	4	0x04	00000100
bInterfaceNumber	0	0	0x00	00000000
bAlternateSetting	0	0	0x00	00000000
bNumEndpoints	1	1	0x01	00000001
bInterfaceClass	Human Interface Device (Find out more online)	3	0x03	00000011
bInterfaceSubClass	Boot Interface	1	0x01	00000001
bInterfaceProtocol	Keyboard	1	0x01	00000001
iInterface	0	0	0x00	00000000

接口描述符定义接口类、接口协议等，如*图 3-2. HID 接口描述符*所示，定义 bInterfaceClass 为 0x03，即 HID 设备，定义 bInterfaceProtocal 为 0x01，即键盘设备。

图 3-3. HID 描述符

HID descriptor				
Name	Value	Dec	Hex	Bin
bLength	Valid	9	0x09	00001001
bDescriptorType	HID	33	0x21	00100001
bcdHID	1.1.1	273	0x0111	00000001 00010001
bCountryCode	Not Supported	0	0x00	00000000
bNumDescriptors	1	1	0x01	00000001
bDescriptorType[0]	REPORT	34	0x22	00100010
wDescriptorLength[0]	46 bytes	46	0x002E	00000000 00101110

HID 描述符定义 HID 规范的版本号，报文描述符的长度，如*图 3-3. HID 描述符*所示，定义 wDescriptorLength 为 0x2E，即后文的报文描述符长度为 46 字节。

图 3-4. HID 端点描述符

Endpoint descriptor				
Name	Value	Dec	Hex	Bin
bLength	Valid	7	0x07	00000111
bDescriptorType	ENDPOINT	5	0x05	00000101
bEndpointAddress	1 IN	129	0x81	10000001
bmAttributes. TransferType	Interrupt	3	0x3	11
bmAttributes. Reserved	Zero	0	0x00	000000
wMaxPacketSize	8 bytes	8	0x0008	00000000 00001000
bInterval	64 frames (64 ms)	64	0x40	01000000

端点描述符定义端点传输类型、时间间隔等，如*图 3-4. HID 端点描述符*所示，定义 bmAttributes.TransferType 为 0x3，即中断传输，定义 bInternal 为 0x40，即中断时间间隔为 64 毫秒。

图 3-5. HID 报文描述符

HID Report Descriptor	
Item	**Data**
Usage Page *(Generic Desktop)*	05 01
Usage *(Keyboard)*	09 06
Collection *(Application)*	A1 01
Usage Page *(Keyboard)*	05 07
Usage Minimum *(Keyboard Left Control)*	19 E0
Usage Maximum *(Keyboard Right GUI)*	29 E7
Logical minimum *(0)*	15 00
Logical maximum *(1)*	25 01
Report Count *(8)*	95 08
Report Size *(1)*	75 01
Input *(Data,Value,Absolute,Bit Field)*	81 02
Report Count *(1)*	95 01
Report Size *(8)*	75 08
Input *(Constant,Value,Absolute,Bit Field)*	81 03
Report Count *(6)*	95 06
Report Size *(8)*	75 08
Logical minimum *(0)*	15 00
Logical maximum *(255)*	26 FF 00
Usage Page *(Keyboard)*	05 07
Usage Minimum *(No event indicated)*	19 00
Usage Maximum *(Keyboard Application)*	29 65
Input *(Data,Array,Absolute,Bit Field)*	81 00
End Collection	C0

报文描述符定义输入输出数据的格式，在应用的数据传输阶段，设备收发数据需要符合报文描述符。

3.1.3. 应用类请求简介

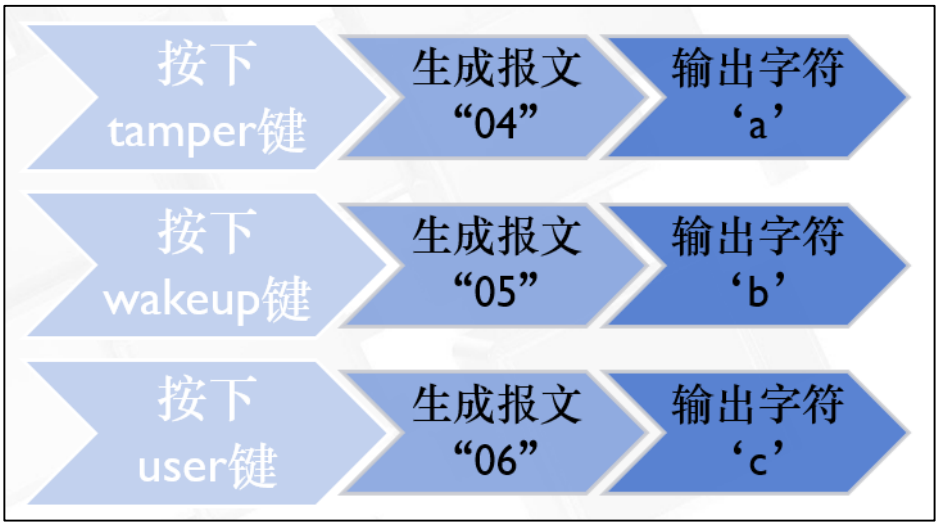
表 3-1. HID 部分设备部分应用类请求简介

类请求	功能描述
USB_GET_DESCRIPTOR	获取报文描述符
GET_REPORT	获取报文控制信息
SET_REPORT	设置报文控制信息
GET_IDLE	获取闲置状态
SET_IDLE	设置闲置状态

鉴于 USBD 设备应用类请求较多，本文仅列举 Demo 中处理的部分请求，读者在实际开发的过程中，需要结合自身的需求，处理应用类请求。以此类推，后文的 CDC 设备、UAC 设备的应用类请求简介也是如此。

3.1.4. 数据传输

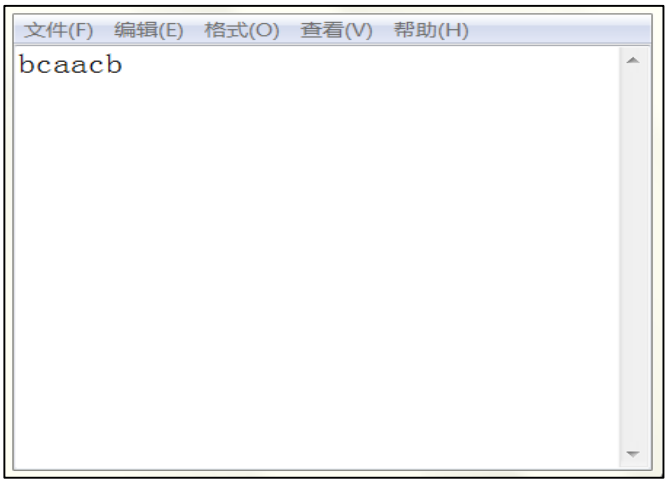
图 3-6. HID 键盘应用示例



在 HID 键盘设备枚举完成后，在主机的设备管理器上会增加一个键盘设备，如*图 3-6. HID 键盘应用示例*所示，按下开发板 Tamper 按键，赋予报文 hid->data[2]数值为“04”，通过设备 IN 端点发送报文至主机，主机显示“a”，按下 Wakeup 键，主机显示“b”，按下 User 键，主机显示“c”。

3.1.5. 输出结果

图 3-7. HID 键盘打印



如*图 3-7. HID 键盘打印*所示，打开文本编辑器，按下开发板的相应按键，文本编辑器上就会打印出相应的字符。

3.2. CDC

3.2.1. 协议简介

CDC（Communication Device Class）协议，是 USB 组织为各种通信设备定义的 USB 子类，后文所提及的虚拟串口即电话业务模型下的抽象控制模型。

在 CDC 设备描述符中，bDeviceClass 值为 2，即 CDC 类，定义两个接口，在接口层，所有 CDC 功能都被包括在 CDC 类的子类中。当 bInterfaceClass 设置为 02 时，即为 Communication control，通信控制类，当该接口的 bInterfaceSubClass 设置为 02 时，即为 Abstract line control model，抽象电话控制模型；当 bInterfaceClass 设置为 10 时，即为 Communication data，通信数据类。

对于 Win10 之前的操作系统，CDC 设备需要安装设备相应的设备驱动，兆易创新公司开发了适配 GD32 MCU 的 CDC 设备驱动（USB Virtual COM Port Driver），下载地址为 ***http://www.gd32mcu.com/cn/download/7***。上位机软件采用串口调试助手即可。

3.2.2. 描述符解析

本章节展示 CDC 设备的配置描述符、接口描述符和功能描述符。配置描述符定义配置描述符集长度、接口数量和供电特性等。接口描述符定义接口类、接口子类等。

图 3-8. CDC 配置描述符

Configuration descriptor				
Name	Value	Dec	Hex	Bin
bLength	Valid	9	0x09	00001001
bDescriptorType	CONFIGURATION	2	0x02	00000010
wTotalLength	67 bytes	67	0x0043	00000000 01000011
bNumInterface	2	2	0x02	00000010
bConfigurationValue	1	1	0x01	00000001
iConfiguration	0	0	0x00	00000000
bmAttributes. Reserved	Zero	0	0x00	00000
bmAttributes. RemoteWakeup	Not supported	0	0x0	0
bmAttributes. SelfPowered	No, Bus Powered	0	0x0	0
bmAttributes. Reserved7	One	1	0x1	1
bMaxPower	100 mA	50	0x32	00110010

如*图 3-8. CDC 配置描述符*所示，bNumInterface 为 2，CDC 设备有两个接口描述符，一个是 CDC 控制接口描述符，另一个是 CDC 数据接口描述符。

图 3-9. CDC 控制接口描述符

Interface descriptor				
Name	**Value**	**Dec**	**Hex**	**Bin**
bInterfaceNumber	0	0	0x00	00000000
bAlternateSetting	0	0	0x00	00000000
bNumEndpoints	1	1	0x01	00000001
bInterfaceClass	Communication Control (Find out more online)	2	0x02	00000010
bInterfaceSubClass	Abstract Line Control Model	2	0x02	00000010
bInterfaceProtocol	Common AT commands	1	0x01	00000001

如*图 3-9. CDC 控制接口描述符*所示，bInterfaceNumber 为 0，表示接口编号为 0，bInterfaceClass 为 2，表示为通信控制类接口描述符。

图 3-10. CDC 标题描述符

Functional Descriptor				
Name	**Value**	**Dec**	**Hex**	**Bin**
bFunctionLength	Valid	5	0x05	00000101
bDescriptorType	CS_INTERFACE	36	0x24	00100100
bDescriptorSubtype	Header	0	0x00	00000000
bcdCDC	1.1	272	0x0110	00000001 00010000

图 3-11. CDC 通话管理描述符

Functional Descriptor				
Name	**Value**	**Dec**	**Hex**	**Bin**
bFunctionLength	Valid	5	0x05	00000101
bDescriptorType	CS_INTERFACE	36	0x24	00100100
bDescriptorSubtype	Call Management	1	0x01	00000001
bmCapabilities. HandleManagement	No	0	0x0	0
bmCapabilities. DataClass	No	0	0x0	0
bmCapabilities. Reserved	0x00	0	0x00	000000
bDataInterface	1	1	0x01	00000001

如*图 3-11. CDC 通话管理描述符*所示，bDataInterface 为 1，表示后续的数据接口描述符接口号为 1，在多 CDC 设备中，通话管理描述符的该域数值与对应的数据接口描述符接口号相等。

图 3-12. CDC 抽象控制管理描述符

Functional Descriptor				
Name	Value	Dec	Hex	Bin
bFunctionLength	Valid	4	0x04	00000100
bDescriptorType	CS_INTERFACE	36	0x24	00100100
bDescriptorSubtype	Abstract Control Management	2	0x02	00000010
bmCapabilities. CommFeature	No	0	0x0	0
bmCapabilities. LineCoding	Yes	1	0x1	1
bmCapabilities. SendBreak	No	0	0x0	0
bmCapabilities. NetworkConnection	No	0	0x0	0
bmCapabilities. Reserved	0x0	0	0x0	0000

图 3-13. CDC 联合功能描述符

Functional Descriptor				
Name	Value	Dec	Hex	Bin
bFunctionLength	Valid	5	0x05	00000101
bDescriptorType	CS_INTERFACE	36	0x24	00100100
bDescriptorSubtype	Union Functional descriptor	6	0x06	00000110
bMasterInterface	0	0	0x00	00000000
bSlaveInterface0	1	1	0x01	00000001

图 3-14. CDC 命令端点描述符

Endpoint descriptor				
Name	Value	Dec	Hex	Bin
bLength	Valid	7	0x07	00000111
bDescriptorType	ENDPOINT	5	0x05	00000101
bEndpointAddress	2 IN	130	0x82	10000010
bmAttributes. TransferType	Interrupt	3	0x3	11
bmAttributes. Reserved	Zero	0	0x00	000000
wMaxPacketSize	8 bytes	8	0x0008	00000000 00001000
bInterval	10 frames (10 ms)	10	0x0A	00001010

如*图 3-14. CDC 命令端点描述符*所示，bmAttributes.TransferType 为 3，为中断传输，命令端点描述符在虚拟串口设备中，没有数据输入，但不可删除，否则会出现异常。

图 3-15. CDC 数据接口描述符

Interface descriptor				
Name	Value	Dec	Hex	Bin
bLength	Valid	9	0x09	00001001
bDescriptorType	INTERFACE	4	0x04	00000100
bInterfaceNumber	1	1	0x01	00000001
bAlternateSetting	0	0	0x00	00000000
bNumEndpoints	2	2	0x02	00000010
bInterfaceClass	Communication Data (Find out more online)	10	0x0A	00001010
bInterfaceSubClass	Unknown (0x00)	0	0x00	00000000
bInterfaceProtocol	None	0	0x00	00000000
iInterface	0	0	0x00	00000000

如*图 3-15. CDC 数据接口描述符*所示，bInterfaceNumber 为 1，表示接口编号为 1，bInterfaceClass 为 0x0A，表示为通信数据类接口描述符，bNumEndpoints 为 2，表示该接口下，有两个端点。

图 3-16. CDC OUT 描述符

Endpoint descriptor				
Name	Value	Dec	Hex	Bin
bLength	Valid	7	0x07	00000111
bDescriptorType	ENDPOINT	5	0x05	00000101
bEndpointAddress	3 OUT	3	0x03	00000011
bmAttributes. TransferType	Bulk	2	0x2	10
bmAttributes. Reserved	Zero	0	0x00	000000
wMaxPacketSize	64 bytes	64	0x0040	00000000 01000000
bInterval	Ignored for full speed Bulk endpoints	0	0x00	00000000

如*图 3-16. CDC OUT 描述符*所示，bmAttributes.TransferType 为 2，为批量传输，OUT 端点为 CDC 设备的接收数据端点。

图 3-17. CDC IN 描述符

Endpoint descriptor				
Name	Value	Dec	Hex	Bin
bLength	Valid	7	0x07	00000111
bDescriptorType	ENDPOINT	5	0x05	00000101
bEndpointAddress	1 IN	129	0x81	10000001
bmAttributes. TransferType	Bulk	2	0x2	10
bmAttributes. Reserved	Zero	0	0x00	000000
wMaxPacketSize	64 bytes	64	0x0040	00000000 01000000
bInterval	Ignored for full speed Bulk endpoints	0	0x00	00000000

如*图 3-17. CDC IN 描述符*所示，bmAttributes.TransferType 为 2，为批量传输，IN 端点为 CDC

设备的发送数据端点。

3.2.3. 应用类请求简介

表 3-2. CDC 设备部分应用类请求简介

类请求	功能描述
SET_LINE_CODING	设置波特率、停止位等串口属性
GET_LINE_CODING	获取波特率、停止位等串口属性
SET_CONTROL_LINE_STATE	设置串口状态，打开或关闭串口

3.2.4. 数据传输

在设备与主机连接之前，安装 CDC 设备驱动。CDC 设备枚举完成后，如*图 3-18. CDC 设备枚举*所示，设备管理器的子项“端口（COM 和 LPT）”会出现设备“GD32 Virtual COM Port”，COM 编号取决于所安装的本地串口实际情况。

图 3-18. CDC 设备枚举

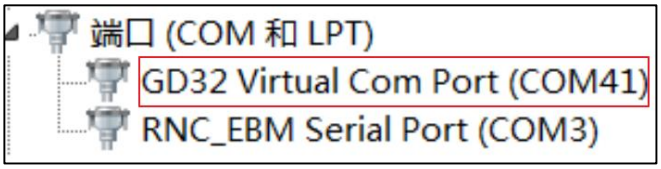
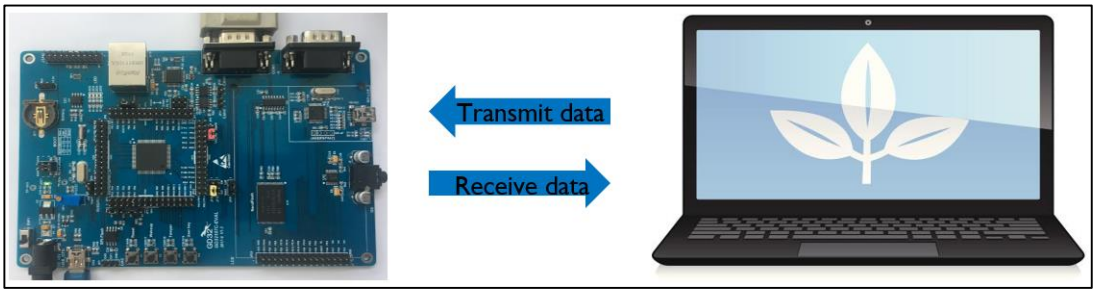


图 3-19. 虚拟串口数据交互



如*图 3-19. 虚拟串口数据交互*所示，CDC 例程实现了回传功能。当主机向设备下发数据时，主机通过串口助手向 CDC 设备的 OUT 端点传输数据，OUT 端点将所接收的数据存入应用缓存空间。当设备向主机上传数据时，设备将应用缓存空间的数据写到发送 RAM，通过 IN 端点将 CDC 设备接收的数据发送到主机，通过串口助手可以显示接收的数据内容。

3.2.5. 输出结果

将 CDC 例程下载到开发板中，并运行。通过键盘输入某些数据，虚拟串口可以接收并显示这些数据。如*图 3-20. 虚拟串口打印*所示，比如通过虚拟串口的输入框输入“GigaDevice MCU”，PC 回传这些信息给虚拟串口，并得以显示。

图 3-20. 虚拟串口打印

3.3. DFU

3.3.1. 协议简介

DFU（Device Firmware Upgrade）协议，主要使用于 USB 接口，实现固件的上传与下载。DFU 设备是 MCU 与编程工具（上位机）之间的数据通道。鉴于 DFU 设备需要安装相应的设备驱动以及保证其正常执行功能的上位机，兆易创新公司开发了多接口编程环境（GD32 All-In-One Programmer）和适配 GD32 MCU 的 DFU 设备驱动（GD32 DFU Drivers），下载地址为 ***http://www.gd32mcu.com/cn/download/7***。

3.3.2. 描述符解析

本章节展示 DFU 设备的配置描述符、接口描述符和功能描述符。配置描述符定义配置描述符集长度、接口数量和供电特性等。接口描述符定义接口类、接口子类等。

图 3-21. DFU 配置描述符

Configuration descriptor

Name	Value	Dec	Hex	Bin
bLength	Valid	9	0x09	00001001
bDescriptorType	CONFIGURATION	2	0x02	00000010
wTotalLength	27 bytes	27	0x001B	00000000 00011011
bNumInterface	1	1	0x01	00000001
bConfigurationValue	1	1	0x01	00000001
iConfiguration	0	0	0x00	00000000
bmAttributes. Reserved	Zero	0	0x00	00000
bmAttributes. RemoteWakeup	Not supported	0	0x0	0
bmAttributes. SelfPowered	No, Bus Powered	0	0x0	0
bmAttributes. Reserved7	One	1	0x1	1
bMaxPower	100 mA	50	0x32	00110010

图 3-22. DFU 接口描述符

Interface descriptor				
Name	Value	Dec	Hex	Bin
bLength	Valid	9	0x09	00001001
bDescriptorType	INTERFACE	4	0x04	00000100
bInterfaceNumber	0	0	0x00	00000000
bAlternateSetting	0	0	0x00	00000000
bNumEndpoints	0	0	0x00	00000000
bInterfaceClass	Application-specific (Find out more online)	254	0xFE	11111110
bInterfaceSubClass	Device Firmware Upgrade	1	0x01	00000001
bInterfaceProtocol	DFU Mode v1.1	2	0x02	00000010
iInterface	0	0	0x00	00000000

如*图 3-22. DFU 接口描述符*所示，定义 bInterfaceClass 为 0xFE，即特定应用设备，定义 bInterfaceSubClass 为 0x01，即 DFU 设备。

图 3-23. DFU 功能描述符

DFU functional descriptor				
Name	Value	Dec	Hex	Bin
bLength	Valid	9	0x09	00001001
bDescriptorType	DFU_FUNCTIONAL	33	0x21	00100001
bmAttributes.bitCanDnload	Yes	1	0x1	1
bmAttributes.bitCanUpload	Yes	1	0x1	1
bmAttributes. bitManifestationTolerant	No, must see bus reset	0	0x0	0
bmAttributes.bitWillDetach	Yes	1	0x1	1
bmAttributes.reserved	Zero	0	0x0	0000
wDetachTimeOut	255 ms	255	0x00FF	00000000 11111111
wTransferSize	2,048 bytes	2,048	0x0800	00001000 00000000
bcdDfuVersion	1.1	272	0x0110	00000001 00010000

3.3.3. 应用类请求简介

表 3-3. DFU 设备应用类请求

请求编码	类请求	功能描述
0	DFU_DETACH	请求设备离开 DFU 模式，进入应用程序。
1	DFU_DNLOAD	请求主机端的数据发送到设备端，将数据加载到设备存储介质，包含擦除操作。
2	DFU_UPLOAD	请求设备端的数据传输到主机端，将设备存储介质的相应数据加载到主机端的目标文件。

请求编码	类请求	功能描述
3	DFU_GETSTATUS	请求设备发送状态到主机端。
4	DFU_CLRSTATUS	请求设备清除错误状态并移动到下一步。
5	DFU_GETSTATE	请求设备仅仅发送当前即将进入的状态。
6	DFU_ABORT	请求设备离开当前状态/操作，进入空闲状态。

表 3-4. DFU 特定类请求的参数总结

bmRequest	bRequest	wValue	wIndex	wLength	Data
00100001b	DETACH	wTimeout	接口	0	0
00100001b	DNLOAD	wBloackNum	接口	长度	固件
10100001b	UPLOAD	0	接口	长度	固件
00100001b	GETSTATUS	0	接口	6	状态
00100001b	CLRSTATUS	0	接口	0	无
00100001b	GETSTATE	0	接口	1	状态
00100001b	ABORT	0	接口	0	无

3.3.4. 数据传输

在设备与主机连接之前，安装 DFU 设备驱动。DFU 设备枚举完成后，如*图 3-24. DFU 设备枚举*所示，设备管理器的子项“通用串行总线控制器”会出现设备“GD32 Device in DFU Mode”。

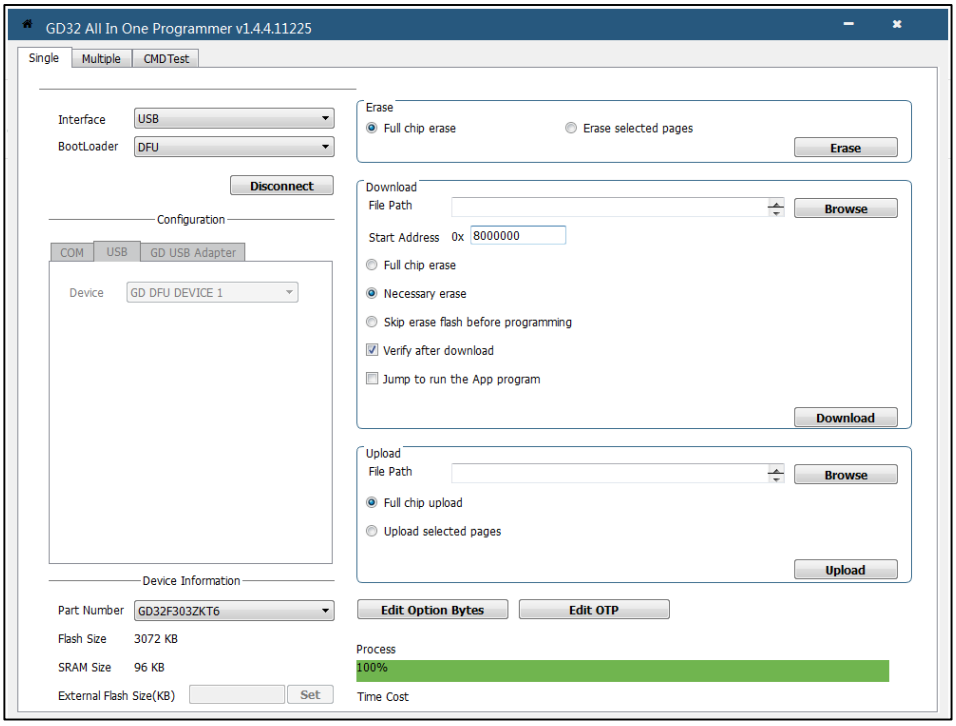
图 3-24. DFU 设备枚举

通过上位机，执行下载功能时，主机通过 USB 总线发送数据到 DFU 设备，进而加载到存储介质。在执行上传功能时，主机通过 USB 总线接收到来自 DFU 设备的数据，并生成 bin 文件。

3.3.5. 输出结果

如*图 3-25. DFU 上位机*所示，打开“GD32 All In One Programmer”上位机，下拉框选择接口为 USB，可见“GD DFU DEVICE 1”，点击 Connect，即可执行 DFU 设备的各项功能，如全片擦除、页擦除、文件下载，文件上传和选项字节操作等。

图 3-25. DFU 上位机


GD32 All In One Programmer v1.4.4.11225
Single
Multiple
CMDTest
Interface
USB
BootLoader
DFU
Disconnect
Configuration
COM
USB
GD USB Adapter
Device
GD DFU DEVICE 1
Device Information
Part Number
GD32F303ZKT6
Flash Size
3072 KB
SRAM Size
96 KB
External Flash Size(KB)
Set
Erase
Full chip erase
Erase selected pages
Erase
Download
File Path
Browse
Start Address
0x
8000000
Full chip erase
Necessary erase
Skip erase flash before programming
Verify after download
Jump to run the App program
Download
Upload
File Path
Browse
Full chip upload
Upload selected pages
Upload
Edit Option Bytes
Edit OTP
Process
100%
Time Cost

3.4. UAC

3.4.1. 协议简介

UAC(USB Audio Class，USB 音频类)，可以实现数字音频数据的传输。USB 音频类定义在接口层，而 USB 音频类分为不同的子类（SubClass）以便于进一步的细节枚举和设置。所有的 USB 音频功能都被包括在 USB 音频类的子类中。当 bInterfaceSubClass 设置为 01 时，即为 AudioControl Interface Subclass，音频控制接口子类；当 bInterfaceSubClass 设置为 02 时，即为 AudioStreaming Interface Subclass 音频流接口子类。

在 AudioControl 接口子类中，Output Terminal 描述符的 wTerminalType 定义为 0301（Speaker），通过 OUT 端点播放主机向设备发送的音频数据，如果 wTerminalType 为定义为 0101（Micphone）时，通过 IN 端点采集音频数据向主机发送。

UAC 设备数据传输采用同步传输，其传输时间间隔由后文的端点描述符的 bInterval 位确定。

3.4.2. 描述符解析

本章节展示 UAC 设备的配置描述符、接口描述符和端点描述符。配置描述符定义配置描述符集长度、接口数量和供电特性等。接口描述符定义接口类、接口子类等。

图 3-26. UAC 配置描述符

Configuration descriptor

Name	Value	Dec	Hex	Bin
bNumInterface	2	2	0x02	00000010
bConfigurationValue	1	1	0x01	00000001
bmAttributes. RemoteWakeup	Not supported	0	0x0	0
bmAttributes. SelfPowered	Yes	1	0x1	1
bMaxPower	100 mA	50	0x32	00110010

图 3-27. UAC 控制接口描述符

Interface descriptor

Name	Value	Dec	Hex	Bin
bInterfaceNumber	0	0	0x00	00000000
bAlternateSetting	0	0	0x00	00000000
bNumEndpoints	0	0	0x00	00000000
bInterfaceClass	Audio (Find out more online)	1	0x01	00000001
bInterfaceSubClass	Audio Control	1	0x01	00000001

如*图 3-27. UAC 控制接口描述符*所示，定义 bInterfaceClass 为 0x01，即 Audio 设备，定义 bInterfaceSubClass 为 0x01，即 Audio Control 子类。

图 3-28. UAC 标题描述符

Audio Descriptor				
Name	**Value**	**Dec**	**Hex**	**Bin**
bDescriptorSubtype	HEADER	1	0x01	00000001
bcdADC	1.0	256	0x0100	00000001 00000000
wTotalLength	39 bytes	39	0x0027	00000000 00100111
baInterfaceNr(1)	1	1	0x01	00000001

如*图 3-28. UAC 标题描述符*所示，HEADER 描述符的域 wTotalLength 定义的长度是自身长度+input terminal 描述符长度+feature unit 描述符长度+output terminal 描述符长度。

图 3-29. UAC 输入终端描述符

Audio Descriptor				
Name	**Value**	**Dec**	**Hex**	**Bin**
bDescriptorSubtype	INPUT_TERMINAL	2	0x02	00000010
bTerminalID	1	1	0x01	00000001
wTerminalType	USB streaming	257	0x0101	00000001 00000001
bNrChannels	1	1	0x01	00000001
wChannelConfig. Left Front (L)	Not present	0	0x0	0
wChannelConfig. Right Front (R)	Not present	0	0x0	0
wChannelConfig. Center Front (C)	Not present	0	0x0	0
wChannelConfig. Low Frequency Enhancement (LFE)	Not present	0	0x0	0
wChannelConfig. Left Surround (LS)	Not present	0	0x0	0
wChannelConfig. Right Surround (RS)	Not present	0	0x0	0
wChannelConfig. Left of Center (LC)	Not present	0	0x0	0
wChannelConfig. Right of Center (RC)	Not present	0	0x0	0
wChannelConfig. Surround (S)	Not present	0	0x0	0
wChannelConfig. Side Left (SL)	Not present	0	0x0	0
wChannelConfig. Side Right (SR)	Not present	0	0x0	0
wChannelConfig. Top (T)	Not present	0	0x0	0

图 3-30. UAC 特性单元描述符

Audio Descriptor				
Name	**Value**	**Dec**	**Hex**	**Bin**
bDescriptorSubtype	FEATURE_UNIT	6	0x06	00000110
bUnitID	2	2	0x02	00000010
bSourceID	1	1	0x01	00000001
bmaControls(0). Mute	Supported	1	0x1	1
bmaControls(0). Volume	Not supported	0	0x0	0
bmaControls(0). Bass	Not supported	0	0x0	0
bmaControls(0). Mid	Not supported	0	0x0	0
bmaControls(0). Treble	Not supported	0	0x0	0
bmaControls(0). Graphic Equalizer	Not supported	0	0x0	0
bmaControls(0). Automatic Gain	Not supported	0	0x0	0
bmaControls(0). Delay	Not supported	0	0x0	0
bmaControls(1). Mute	Not supported	0	0x0	0
bmaControls(1). Volume	Not supported	0	0x0	0
bmaControls(1). Bass	Not supported	0	0x0	0
bmaControls(1). Mid	Not supported	0	0x0	0
bmaControls(1). Treble	Not supported	0	0x0	0
bmaControls(1). Graphic Equalizer	Not supported	0	0x0	0
bmaControls(1). Automatic Gain	Not supported	0	0x0	0
bmaControls(1). Delay	Not supported	0	0x0	0

图 3-31. UAC 输出终端描述符

Audio Descriptor				
Name	**Value**	**Dec**	**Hex**	**Bin**
bDescriptorSubtype	OUTPUT_TERMINAL	3	0x03	00000011
bTerminalID	3	3	0x03	00000011
wTerminalType	Speaker	769	0x0301	00000011 00000001
bSourceID	2	2	0x02	00000010

图 3-32. UAC 标准数据流零带宽接口描述符

Interface descriptor				
Name	Value	Dec	Hex	Bin
bInterfaceNumber	1	1	0x01	00000001
bAlternateSetting	0	0	0x00	00000000
bNumEndpoints	0	0	0x00	00000000
bInterfaceClass	Audio (Find out more online)	1	0x01	00000001
bInterfaceSubClass	Audio Streaming	2	0x02	00000010

如*图 3-32. UAC 标准数据流零带宽接口描述符*所示，定义 bInterfaceClass 为 0x01，即 Audio 设备，定义 bInterfaceSubClass 为 0x02，即 Audio Streaming 子类。

图 3-33. UAC 标准数据流接口描述符

Interface descriptor				
Name	Value	Dec	Hex	Bin
bInterfaceNumber	1	1	0x01	00000001
bAlternateSetting	1	1	0x01	00000001
bNumEndpoints	1	1	0x01	00000001
bInterfaceClass	Audio (Find out more online)	1	0x01	00000001
bInterfaceSubClass	Audio Streaming	2	0x02	00000010

图 3-34. UAC 通用数据流描述符

Audio Descriptor				
Name	Value	Dec	Hex	Bin
bDescriptorSubtype	AS_GENERAL	1	0x01	00000001
bTerminalLink	1	1	0x01	00000001
bDelay	1 frame	1	0x01	00000001
wFormatTag	PCM	1	0x0001	00000000 00000001

图 3-35. UAC 音频格式描述符

Audio Descriptor				
Name	Value	Dec	Hex	Bin
bDescriptorSubtype	FORMAT_TYPE	2	0x02	00000010
bFormatType	FORMAT_TYPE_III	3	0x03	00000011
bNrChannels	2	2	0x02	00000010
bSubframeSize	2 bytes	2	0x02	00000010
bBitResolution	16 bits	16	0x10	00010000
bSamFreqType	1 discrete sampling frequencies	1	0x01	00000001
tSamFreq(1)	16.0 kHz	16,000	0x003E80	00000000 00111110 10000000

如*图 3-35. UAC 音频格式描述符*所示，bFormatType 定义音频格式为 FORMAT_TYPE_III，bBitResolution 定位为 0x10，表示按照 16 位格式播放音频，tSamFreq 定义音频采集频率。

图 3-36. UAC 标准端点描述符

Endpoint descriptor

Name	Value	Dec	Hex	Bin
bEndpointAddress	1 OUT	1	0x01	00000001
bmAttributes. TransferType	Isochronous	1	0x1	01
wMaxPacketSize	64 bytes	64	0x0040	00000000 01000000
bInterval	1 frame (1000 us)	1	0x01	00000001
bRefresh	Not used	0	0x00	00000000
bSynchAddress	Not used	0	0x00	00000000

图 3-37. UAC 数据流端点描述符

Audio Descriptor

Name	Value	Dec	Hex	Bin
bDescriptorSubtype	EP_GENERAL	1	0x01	00000001
bmAttributes. Sampling Frequency	Not supported	0	0x0	0
bmAttributes. Pitch	Not supported	0	0x0	0
bmAttributes. MaxPacketsOnly	Not supported	0	0x0	0
bLockDelayUnits	Undefined	0	0x00	00000000
wLockDelay	0	0	0x0000	00000000 00000000

3.4.3. 应用类请求简介

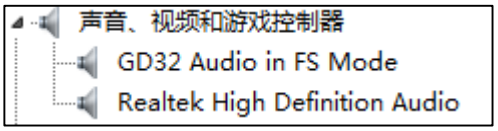
表 3-5. UAC 设备部分应用类请求简介

类请求	功能描述
AUDIO_REQ_GET_CUR	获取当前音频参数
AUDIO_REQ_SET_CUR	设置当前音频参数

3.4.4. 数据传输

在 UAC 设备枚举完成后，如*图 3-38. UAC 设备枚举*所示，设备管理器的子项“声音、视频和游戏控制器”会出现设备“GD32 Audio in FS Mode”。在主机中选择音频文件，播放音频，音频数据通过 USB 总线传输数据到 UAC 设备，UAC 设备将所获取的数据通过 I2S 总线传输到耳机接口，通过耳机即可听到主机所播放的音频文件。

图 3-38. UAC 设备枚举



3.4.5. 输出结果

在主机声音配置的子项“播放”中，如*图 3-39. UAC 声道配置*所示，选择“GD32 Audio in FS Mode”作为默认的扬声器，将耳机插入开发板耳机接口中，如*图 3-40. 音频播放*所示，双击音频文件，即可通过开发板的耳机接口听到主机所播放的音频。

图 3-39. UAC 声道配置

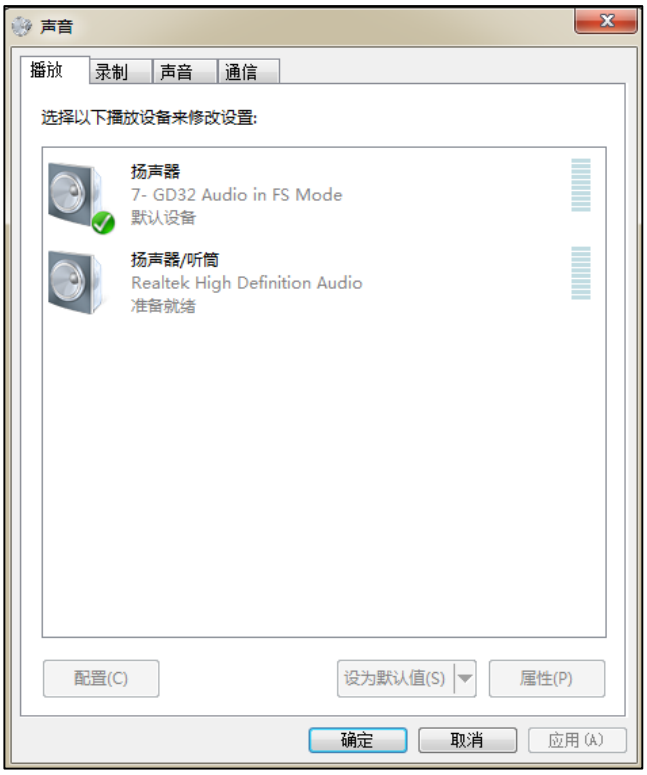


图 3-40. 音频播放

4. 版本历史

表 4-1. 版本历史

版本号	说明	日期
1.0	首次发布	2022 年 03 月 28 日

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the “Product”), is owned by the Company under the intellectual property laws and treaties of the People’s Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as it’s suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as it’s suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.