应用笔记

Application Note

文档编号：**AN1095**

APM32 USB Device Library 应用笔记

版本：**V1.0**

1 引言

本应用笔记提供如何配置和使用 APM32 USB device library，包括目录文件结构、函数介绍及应用方法等。

USB 是通用串行总线（Universal Serial Bus）的缩写，是一种串口总线标准，也是一种输入输出接口的技术规范，广泛应用于鼠标、键盘、打印机、扫描仪、数码相机等设备中。

适用产品系列
APM32F072xx, APM32L072xx，APM32F10x，APM32E10x， APM32S10x，APM32F4xx

目录

2 USB Device Library 简介

USB 设备库适用于 APM32 所有带 USB 外设的 MCU，在所提供的 APM32 USB SDK 中包含以下内容：

1. 各系列芯片的底层 USB 驱动；
2. 各系列芯片的 USB 设备标准库驱动；
3. 公用的 USB 类驱动；
4. USB 设备常用的应用程序，如支持 USB 全速、高速的控制、中断、批量和同步传输操作；
5. 包括以下 USB class 的例程：

   - Communication Device（CDC）

   虚拟 COM 口，用于 USB 转 RS232 桥接。

   - Human Interface Device（HID）

   适用开发板的按键输入来模拟鼠标滚轮操作。

   - Custom Human Interface Device（CHID）

   包括 HID 方式的输出和输入。

   - Mass storage

   基于 SRAM 或 SD Card 或 Nor Flash 的 MSC 设备，也可用于测试 MSC 传输速度。

   - Winusb

   基于微软 winusb（Microsoft OS 1.0）免驱方式实现的批量传输。

3 USB Device Library 架构

USB Device Library 的架构主要分为五层，其中应用层位于架构的顶端。第二层包括 core 和 class 两部分驱动：

1. Core 驱动
   - Core 模块提供内部设备库管理状态机及 USB 中断的回调。
   - Translation 模块提供 USB 各种传输处理函数。
   - Request 模块提供 chapter 9 相关的请求。
2. Class 驱动

   Class 驱动包括一系列由预定义类驱动程序组成的驱动，由 USBD_Init()函数进行链接到 USB 内核。

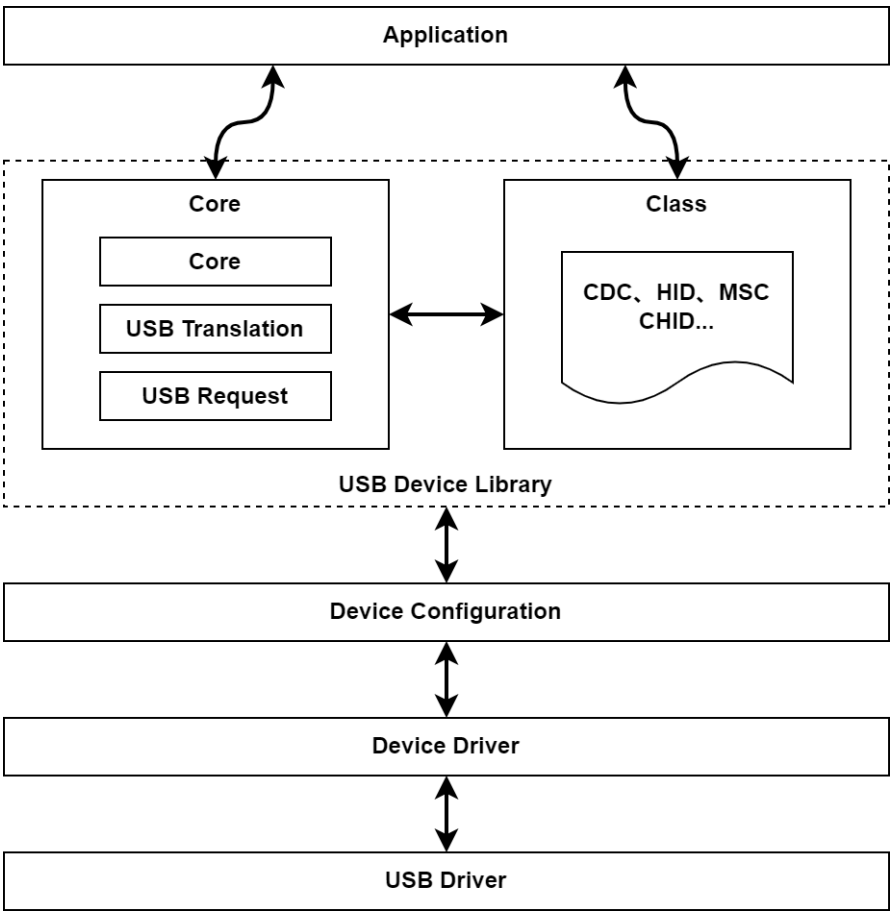

图 1 USB 设备库架构

4 USB Device Library 文件目录结构

4.1 目录结构

USB 设备库基于通用的 USB 协议栈底层驱动，适用于全速和高速模式。其文件包含在 APM32_USB_Library 目录下的 Device 文件夹中。

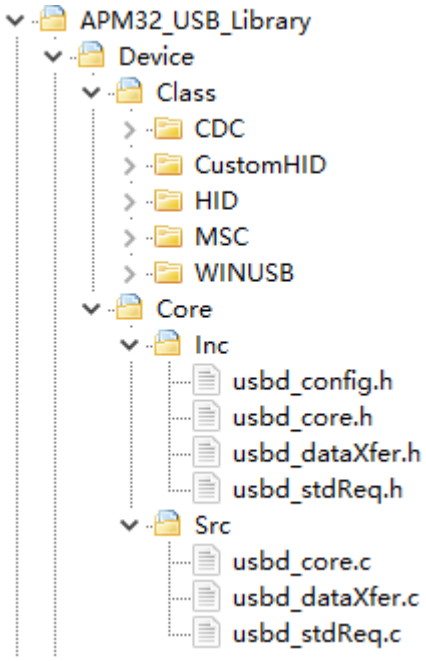

图 2 USB 设备库文件目录

4.2 文件结构

USB 设备库的文件包括底层共用的驱动、时基单元、中间件 core 和 class 文件及用户配置文件。

1. 共用的驱动
    - apm32xxx.h
    - apm32xxx_usb.c/h
    - apm32xxx_usb_device.c/h
2. 时基单元
    - bsp_delay.c/h
3. 中间件
    - Core 文件
    - Class 文件
4. 用户配置文件
    - usbd_board.c/h：USB 设备配置文件
    - usbd_device_user.c/h：USB 用户初始化文件
    - usb_descriptor.c/h：描述符配置文件
    - usb_interface.c/h：数据接口配置文件

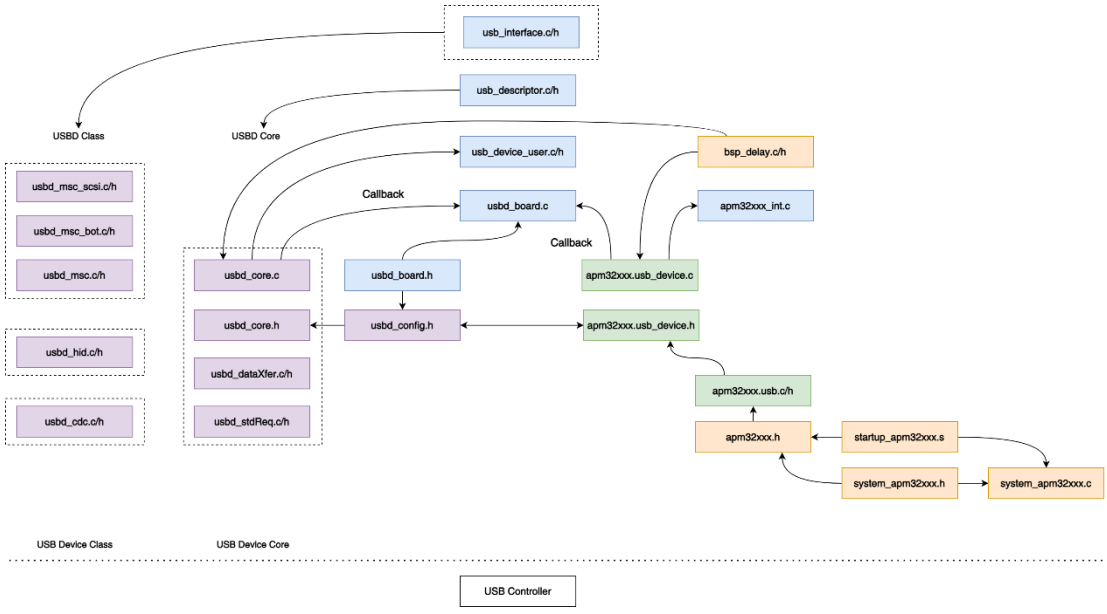

图 3 文件结构

4.3 USB 设备 Configuration

4.3.1 usbd_board(.c/h)

该文件包含 USB 设备的配置和回调函数。

4.3.2 usbd_descriptor(.c/h)

该文件包含 USB 设备的基本描述符。

4.3.3 usb_device_user(.c/h)

该文件用于初始化 USB 设备。

4.4 USB 设备 core

4.4.1 usbd_core(.c/h)

该文件包含 USB 核心状态机的控制和不同事件的处理及回调函数。下表是该文件主要函数的说明。

函数	描述
USBD_Init	初始化 USB 设备及状态机并链接 USB 设备结构体到类处理单元
USBD_DeInit	取消 USB 设备初始化
USBD_SetupStage	处理 setup 阶段的请求
USBD_DataOutStage	在合适端点上处理来自 Host 的数据
USBD_DataInStage	在合适端上上发送数据到 Host
USBD_SetSpeed	设置 USB 设备速度
USBD_Resume	在挂起前恢复设备状态到之前的状态
USBD_Suspend	设置设备状态为挂起
USBD_Reset	重新初始化设备
USBD_HandleSOF	处理 SOF 事件
USBD_IsoInInComplete	处理 ISO 输入不完整事件
USBD_IsoOutInComplete	处理 ISO 输出不完整事件
USBD_Connect	处理设备连接事件
USBD_Disconnect	处理设备断开连接事件

表 1 usbd core 函数

4.4.2 usbd_dataXfer(.c/h)

该文件包含在端点 0 上实现数据发送和接收的函数。下表是该文件主要函数的说明。

函数	描述
USBD_CtrlSendData	在端点 0 上发送数据
USBD_CtrlSendNextData	在端点 0 上发送剩余数据
USBD_CtrlReceiveData	在端点 0 上接收数据
USBD_CtrlSendStatus	在端点 0 上发送零长度数据包
USBD_CtrlReceiveStatus	在端点 0 上接收零长度数据包
USBH_SetupReqParse	解析 setup 请求

表 2 usbd data xfer 函数

4.4.3 usbd_stdReq(.c/h)

该文件包含一个标准请求回调函数数组，用于确保 USB 能够完成 Host 发来的标准请求。下表是该文件主要函数的说明。

函数	描述
USBD_REQ_GetStatus	处理“Get Status”请求
USBD_REQ_ClearFeature	处理“Clear Feature”请求
USBD_REQ_SetFeature	处理“Set Feature”请求
USBD_REQ_SetAddress	处理“Set Address”请求
USBD_REQ_GetDesc	处理“Get Descriptor”请求
USBD_REQ_GetCfg	处理“Get Configuration”请求
USBD_REQ_SetCfg	处理“Set Configuration”请求
USBD_REQ_CtrlError	处理控制传输出现的错误

表 3 usbd stdReq 函数

4.4.4 usbd_config(.h)

该文件包含常用的 USB 设备变量和结构体等的定义。

4.5 USB 设备 Class

4.5.1 Human Interface Devices (HID) Class

该类用于实现人机接口的设备，用于人与机器（如游戏控制器、鼠标、键盘）之间进行交互。该类的代码是基于《Device Class Definition for Human Interface Devices (HID) version 1.11》来实现的。

4.5.1.1 usbd_hid(.c/h)

该文件包含所有 HID 类的变量和特定的 API，默认配置描述符为鼠标。下表是该文件主要函数的说明。

函数	描述
USBD_HID_ClassInitHandler	初始化 HID 类的结构体及端点
USBD_HID_ClassDeInitHandler	解除 HID 类结构体及端点的初始化
USBD_HID_SetupHandler	处理 HID 类的 setup 请求
USBD_HID_DataInHandler	处理 HID 类从 IN 端点发送到主机的数据
USBD_HID_SOFHandler	处理 SOF 事件
USBD_HID_ReadInterval	读取轮询间隔
USBD_HID_TxReport	发送报告描述符给主机

表 4 usbd hid 函数

4.5.1.2 usbd_hid_keyboard.c

该文件包含所有 HID 类的变量和特定的 API，默认 HID 报告描述符为键盘。函数与 usbd_hid.c 一致。

4.5.2 Custom HID Class

该类的代码根据《Device Class Definition for Human Interface Devices (HID) version 1.11》来实现。

4.5.2.1 usbd_customhid(.c/h)

该文件包含所有 HID 类的变量和特定的 API， HID 报告描述符可自定义。下表是该文件主要函数的说明。

函数	描述
USBD_CUSTOM_HID_ClassInitHandler	初始化 CHID 类的结构体及端点
USBD_CUSTOM_HID_ClassDeInitHandler	解除 CHID 类结构体及端点的初始化
USBD_CUSTOM_HID_SOFHandler	处理 SOF 事件
USBD_CUSTOM_HID_SetupHandler	处理 CHID 类的 setup 请求
USBD_CUSTOM_HID_RxEP0Handler	处理端点 0 上的接收就绪事件
USBD_CUSTOM_HID_DataInHandler	处理 CHID 类从 IN 端点发送到主机的数据
USBD_CUSTOM_HID_DataOutHandler	处理 CHID 类从 OUT 端点接收到的主机数据
USBD_CUSTOM_HID_ReportDescHandler	发送报告描述符给主机
USBD_CUSTOM_HID_ReadInterval	读取轮询间隔
USBD_CUSTOM_HID_RegisterItf	注册 CHID 类接口处理函数

表 5 usbd custom hid 函数

4.5.3 Mass Storage Class (MSC)

该类用于实现大容量存储设备，通过 USB 进行数据存储和交换。该类的代码是基于《Universal Serial Bus Mass Storage Class (MSC) Bulk-Only Transport (BOT) Version 1.0》来实现的。

4.5.3.1 usbd_msc(.c/h)

该文件包含所有 MSC 类的变量和特定的 API。下表是该文件主要函数的说明。

函数	描述
USBD_MSC_ClassInitHandler	初始化 MSC 类的结构体及端点
USBD_MSC_ClassDeInitHandler	解除 MSC 类结构体及端点的初始化
USBD_MSC_SOFHandler	处理 SOF 事件
USBD_MSC_SetupHandler	处理 MSC 类的 setup 请求
USBD_MSC_DataInHandler	处理 MSC 类从 IN 端点发送到主机的数据
USBD_MSC_DataOutHandler	处理 MSC 类从 OUT 端点接收到的主机数据
USBD_MSC_RegisterMemory	注册 MSC 类接口处理函数

表 6 usbd msc 函数

4.5.3.2 usbd_msc_bot(.c/h)

该文件包含的仅批量传输（BOT）处理函数。

4.5.3.3 usbd_msc_scsi(.c/h)

该文件包含必需的 SCSI 命令处理函数及 MSC 的 Inquiry 和 Sense 信息。

4.5.4 Communications Devices Class (CDC)

该类用于实现虚拟 COM 端口和调制解调器设备。该类的代码是根据《Universal Serial Bus Class Definitions for Communications Devices Revision 1.2》来实现的。

4.5.4.1 usbd_cdc(.c/h)

该文件包含所有 CDC 类的变量和特定的 API。下表是该文件主要函数的说明。

函数	描述
USBD_CDC_ClassInitHandler	初始化 CDC 类的结构体及端点
USBD_CDC_ClassDeInitHandler	解除 CDC 类结构体及端点的初始化
USBD_CDC_SOFHandler	处理 SOF 事件
USBD_CDC_SetupHandler	处理 CDC 类的 setup 请求
USBD_CDC_RxEP0Handler	处理端点 0 上的接收就绪事件
USBD_CDC_DataInHandler	处理 CDC 类从 IN 端点发送到主机的数据
USBD_CDC_DataOutHandler	处理 CDC 类从 OUT 端点接收到的主机数据
USBD_CDC_ReadInterval	读取轮询间隔
USBD_CDC_RegisterItf	注册 CDC 类接口处理函数

表 7 usbd cdc 函数

4.5.5 WinUSB

该类用于实现 windows 环境下的免驱设备。该类的代码是根据”WinUSB 1.0”来实现的。

4.5.5.1 usbd_winusb(.c/h)

该文件包含所有 winusb 1.0 供应商特定类的变量和特定的 API。下表是该文件主要函数的说明。

函数	描述
USBD_WINUSB_ClassInitHandler	初始化 WINUSB 类的结构体及端点
USBD_WINUSB_ClassDeInitHandler	解除 WINUSB 类结构体及端点的初始化
USBD_WINUSB_SOFHandler	处理 SOF 事件
USBD_WINUSB_SetupHandler	处理 WINUSB 类的 setup 请求
USBD_WINUSB_DataInHandler	处理 WINUSB 类从 IN 端点发送到主机的数据
USBD_WINUSB_DataOutHandler	处理 WINUSB 类从 OUT 端点接收到的主机数据
USBD_WINUSB_ReadInterval	读取轮询间隔
USBD_WINUSB_RegisterItf	注册 WINUSB 类接口处理函数

表 8 usbd winusb 函数

5 USB Device Library 数据结构

5.1 Setup 请求处理结构体

当从主机收到请求时，会对其进行解码并放入以下结构中以简化其处理。

```
typedef struct
{
    union
    {
        uint8_t REQ_DATA[8];

        struct
        {
            USBD_REQ_TYPE_T     bmRequest;
            uint8_t             bRequest;
            uint8_t             wValue[2];
            uint8_t             wIndex[2];
            uint8_t             wLength[2];
        } DATA_FIELD;
    };
} USBD_REQ_SETUP_T;
```

5.2 描述符处理结构体

该结构体包含了所有常见的描述符的配置函数。

```
typedef struct
{
    const char*             descName;
    USBD_DescCallback_T     deviceDescHandler;
    USBD_DescCallback_T     configDescHandler;
    USBD_DescCallback_T     configStrDescHandler;
    USBD_DescCallback_T     interfaceStrDescHandler;
    USBD_DescCallback_T     langIdStrDescHandler;
    USBD_DescCallback_T     manufacturerStrDescHandler;
    USBD_DescCallback_T     productStrDescHandler;
    USBD_DescCallback_T     serialStrDescHandler;
#if USBD_SUP_LPM
    USBD_DescCallback_T     bosDescHandler;
#endif
    USBD_DescCallback_T     winUsbOsStrDescHandler;
    USBD_DescCallback_T     otherSpeedConfigDescHandler;
    USBD_DescCallback_T     devQualifierDescHandler;
} USBD_DESC_T;
```

5.3 USB 设备类结构体

该结构体包含了常见类所需的功能。

```
typedef struct
{
    /* Class handler */
    const char*          className;
    void*                classData;
    USBD_STA_T(*ClassInitHandler)(struct _USBD_INFO_T* usbInfo, uint8_t
cfgIndex);
    USBD_STA_T(*ClassDeInitHandler)(struct _USBD_INFO_T* usbInfo, uint8_t
cfgIndex);
    USBD_STA_T(*ClassSofHandler)(struct _USBD_INFO_T* usbInfo);

    /* Control endpoint */
    USBD_STA_T(*ClassSetup)(struct _USBD_INFO_T* usbInfo, USBD_REQ_SETUP_T*
req);
    USBD_STA_T(*ClassTxEP0)(struct _USBD_INFO_T* usbInfo);
    USBD_STA_T(*ClassRxEP0)(struct _USBD_INFO_T* usbInfo);
    /* Specific endpoint */
    USBD_STA_T(*ClassDataIn)(struct _USBD_INFO_T* usbInfo, uint8_t epNum);
    USBD_STA_T(*ClassDataOut)(struct _USBD_INFO_T* usbInfo, uint8_t epNum);
    USBD_STA_T(*ClassIsoOutIncomplete)(struct _USBD_INFO_T* usbInfo, uint8_t
epNum);
    USBD_STA_T(*ClassIsoInIncomplete)(struct _USBD_INFO_T* usbInfo, uint8_t
epNum);
} USBD_CLASS_T;
```

5.4 USB 设备处理结构体

该结构体包含 USB 设备数据和信息。

```
typedef struct _USBD_INFO_T
{
    __IO uint8_t            devState;
    __IO uint8_t            preDevState;
    __IO uint8_t            devEp0State;
    uint32_t                devEp0DataLen;

    uint8_t                 devSpeed;
    uint8_t                 devAddr;

    USBD_DESC_T*            devDesc;
    USBD_CLASS_T*           devClass[USBD_SUP_CLASS_MAX_NUM];

    void*                   devClassUserData[USBD_SUP_CLASS_MAX_NUM];
    uint32_t                classID;
    uint32_t                classNum;

    void*                   cfgDesc;
    USBD_REQ_SETUP_T        reqSetup;

    uint32_t                devCfg;
    uint32_t                devCfgStatus;
    uint32_t                devCfgDefault;
    uint8_t                 devTestModeStatus;
    uint32_t                devRemoteWakeUpStatus;

    USBD_EP_INFO_T          devEpIn[16];
    USBD_EP_INFO_T          devEpOut[16];
    void (*userCallback)(struct _USBD_INFO_T* usbInfo, uint8_t userStatus);
    void*                   dataPoint;
} USBD_INFO_T;
```

6 USB Device Library 应用方法

6.1 配置 USB Device Library

6.1.1 库配置项

库配置在 usbd_board.h 文件中，USB 设备库支持以下配置项，在使用相关功能时，需配置相关功能项。

```
#define USBD_SUP_CLASS_MAX_NUM              1       //最大 class 支持数量
#define USBD_SUP_INTERFACE_MAX_NUM          1       //最大 interface 支持数量
#define USBD_SUP_CONFIGURATION_MAX_NUM      1       //最大 configuration 支持数量
#define USBD_SUP_STR_DESC_MAX_NUM           512     //字符串描述符最大字符长度
#define USBD_SUP_LPM                        0       //LPM 开关宏
#define USBD_SUP_SELF_PWR                   1       //自供电开关宏
```

6.1.2 Class 配置项

除了 class 头文件中的配置项外，有些 class 还需在源文件中配置相关描述符，如 HID class 的配置项如下：

```
#define USBD_HID_MOUSE_REPORT_DESC_SIZE         74      //鼠标 HID 报告描述符长度
#define USBD_HID_KEYBOARD_REPORT_DESC_SIZE      63      //键盘 HID 报告描述符长度
#define USBD_HID_DESC_SIZE                      9       //HID 描述符长度
#define USBD_HID_FS_INTERVAL                    10      //HID FS 轮询间隔
#define USBD_HID_HS_INTERVAL                    7       //HID HS 轮询间隔
#define USBD_HID_IN_EP_ADDR                     0x81    //HID IN 端点地址
#define USBD_HID_IN_EP_SIZE                     0x04    //HID IN 端点大小
#define USBD_HID_FS_MP_SIZE                     0x40    //HID FS 最大包长
```

6.2 配置 USB 底层

USB 设备的延时、IO、时钟、中断、端点大小的配置项在 usbd_board.c 文件中的 USBD_HardwareInit()函数。

```
void USBD_HardwareInit(USBD_INFO_T* usbInfo)
{
    ...
    /* Configure FIFO */
    /* Configure RX FIFO */
    USB_OTG_ConfigRxFifoSize(usbDeviceHandler.usbGlobal, USBD_FS_RX_FIFO_SIZE);
    /* Configure TX FIFO */
    USBD_OTG_ConfigDeviceTxFifo(&usbDeviceHandler, 0, USBD_FS_TX_FIFO_0_SIZE);
    USBD_OTG_ConfigDeviceTxFifo(&usbDeviceHandler, 1, USBD_FS_TX_FIFO_1_SIZE);

    ...
}
```

需要注意的是，端点号及 size 也需要在对应引用的 class 文件中修改，如 CDC Class 中下列的配置项和初始化函数。

```
#define USBD_CDC_FS_MP_SIZE                     0x40
#define USBD_CDC_HS_MP_SIZE                     0x200
#define USBD_CDC_CMD_MP_SIZE                    0x08
#define USBD_CDC_DATA_MP_SIZE                   0x07
#define USBD_CDC_CMD_EP_ADDR                    0x82
#define USBD_CDC_DATA_IN_EP_ADDR                0x81
#define USBD_CDC_DATA_OUT_EP_ADDR               0x01
#define USBD_CDC_FS_INTERVAL                    16
#define USBD_CDC_HS_INTERVAL                    16


USBD_STA_T USBD_CDC_ClassInitHandler(USBD_INFO_T* usbInfo, uint8_t cfgIndex)
```

6.3 配置中断服务函数

USB 中断请求函数需放在相关中断服务函数中。

6.4 配置 USB 描述符

USB 设备常用的描述符都在 usbd_descriptor.c 文件中，包括：

1. 设备描述符
2. 配置描述符
3. 配置字符串描述符
4. 接口字符串描述符
5. 语言 ID 字符串描述符
6. 制造商字符串描述符
7. 产品字符串描述符
8. 序列号字符串描述符
9. 其他速度描述符
10. 设备修饰描述符

6.5 配置 USB 数据接口

有些 class 需要有数据处理接口，则需在 USB 设备初始化时注册到 USB 内核中。

6.6 初始化 USB 设备

USB 设备的初始化函数 USB_DeviceInit()在 usb_device_user.c 文件中，包括以下功能：

1. Class 数据处理接口的注册
2. 内核速度选择
3. 描述符注册
4. Class 处理函数注册
5. USB 设备用户回调函数注册

```
USBD_STA_T USBD_Init(USBD_INFO_T* usbInfo, USBD_SPEED_T usbDevSpeed, \
                     USBD_DESC_T* usbDevDesc, \
                     USBD_CLASS_T* usbDevClass, \
                     void (*userCallbackFunc)(struct _USBD_INFO_T*,
uint8_t))
{
    USBD_STA_T usbStatus = USBD_OK;

    /* Register descriptor function */
    if (usbDevDesc != NULL)
    {
        usbInfo->devDesc = usbDevDesc;
    }

    /* Register class function */
    if (usbDevClass == NULL)
    {
        usbStatus = USBD_FAIL;
        return usbStatus;
    }
    else
    {
        usbInfo->devClass[usbInfo->classNum++] = usbDevClass;
    }

    /* Register user application */
    usbInfo->userCallback = userCallbackFunc;

    usbInfo->devState = USBD_DEV_DEFAULT;
    /* Set USB device speed */
    usbInfo->devSpeed = usbDevSpeed;

    /* Init USB hardware */
    USBD_HardwareInit(usbInfo);

    return usbStatus;
}
```

7 修订历史

表 9 文件修订历史

日期	修订	变化
2023.07.20	1.0	初版

声明

本手册由珠海极海半导体有限公司（以下简称“极海”）制订并发布，所列内容均受商标、著作权、软件著作权相关法律法规保护，极海保留随时更正、修改本手册的权利。使用极海产品前请仔细阅读本手册，一旦使用产品则表明您（以下称“用户”）已知悉并接受本手册的所有内容。用户必须按照相关法律法规和本手册的要求使用极海产品。

1、权利所有

本手册仅应当被用于与极海所提供的对应型号的芯片产品、软件产品搭配使用，未经极海许可，任何单位或个人均不得以任何理由或方式对本手册的全部或部分内容进行复制、抄录、修改、编辑或传播。

本手册中所列带有“®”或“TM”的“极海”或“Geehy”字样或图形均为极海的商标，其他在极海产品上显示的产品或服务名称均为其各自所有者的财产。

2、无知识产权许可

极海拥有本手册所涉及的全部权利、所有权及知识产权。

极海不应因销售、分发极海产品及本手册而被视为将任何知识产权的许可或权利明示或默示地授予用户。

如果本手册中涉及任何第三方的产品、服务或知识产权，不应被视为极海授权用户使用前述第三方产品、服务或知识产权，除非在极海销售订单或销售合同中另有约定。

3、版本更新

用户在下单购买极海产品时可获取相应产品的最新版的手册。

如果本手册中所述的内容与极海产品不一致的，应以极海销售订单或销售合同中的约定为准。

4、信息可靠性

本手册相关数据经极海实验室或合作的第三方测试机构批量测试获得，但本手册相关数据难免会出现校正笔误或因测试环境差异所导致的误差，因此用户应当理解，极海对本手册中可能出现的该等错误无需承担任何责任。本手册相关数据仅用于指导用户作为性能参数参照，不构成极海对任何产品性能方面的保证。

用户应根据自身需求选择合适的极海产品，并对极海产品的应用适用性进行有效验证和测试，以确认极海产品满足用户自身的需求、相应标准、安全或其它可靠性要求；若因用户未充分对极海产品进行有效验证和测试而致使用户损失的，极海不承担任何责任。

5、合规要求

用户在使用本手册及所搭配的极海产品时，应遵守当地所适用的所有法律法规。用户应了解产品可能受到产品供应商、极海、极海经销商及用户所在地等各国有关出口、再出口或其它法律的限制，用户（代表其本身、子公司及关联企业）应同意并保证遵守所有关于取得极海产品及／或技术与直接产品的出口和再出口适用法律与法规。

6、免责声明

本手册由极海“按原样”（as is）提供，在适用法律所允许的范围内，极海不提供任何形式的明示或暗示担保，包括但不限于对产品适销性和特定用途适用性的担保。

对于用户后续在针对极海产品进行设计、使用的过程中所引起的任何纠纷，极海概不承担责任。

7、责任限制

在任何情况下，除非适用法律要求或书面同意，否则极海和/或以“按原样”形式提供本手册的任何第三方均不承担损害赔偿责任，包括任何一般、特殊因使用或无法使用本手册相关信息而产生的直接、间接或附带损害（包括但不限于数据丢失或数据不准确，或用户或第三方遭受的损失）。

8、适用范围

本手册的信息用以取代本手册所有早期版本所提供的信息。