STM32CubeMX教程31 USB_DEVICE - HID外设_模拟键盘或鼠标

目录

读者可访问 GitHub - lc-guo/STM32CubeMX-Series-Tutorial **获取原始工程代码**

1、准备材料

正点原子stm32f407探索者开发板V2.4

STM32CubeMX　软件　（Version 6.10.0）

keil　μVision5 IDE（MDK-Arm）

ST-LINK/V2驱动

野火DAP仿真器

XCOM V2.6串口助手

2、实验目标

使用STM32CubeMX软件配置STM32F407开发板**USB_OTG_FS为工作在Human Interface Device Class (HID)（人机接口设备类）模式下的USB_DEVICE（USB从机），利用上下左右四个用户按键模拟在Windwos上的鼠标或键盘操作**

3、模拟鼠标实验流程

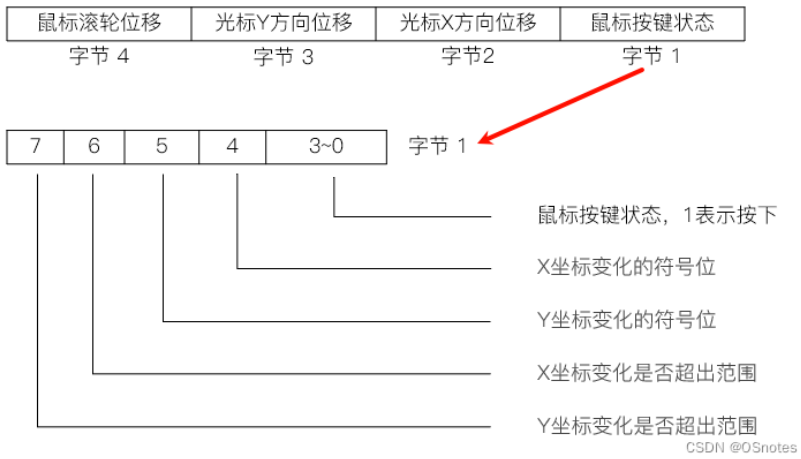
3.0、前提知识

关于USB的相关知识请读者阅读STM32CubeMX教程29 USB_HOST - 使用FatFs文件系统读写U盘实验“3、USB概述”小节内容，USB_SALVE从机接口 硬件　原理图请读者阅读其“4.0、前提知识”小节内容

关于USB从机参数配置中Device Descriptor 选项卡下的参数配置请阅读STM32CubeMX教程30 USB_DEVICE - MSC外设_读卡器实验”3.0、前提知识“小节

将USB设备接口配置工作在Human Interface Device Class (HID)模式下，然后通过USB线连接到Windows　电脑上就可以作为一个人体学输入设备出现在PC的设备管理器中，在此模式下可以将USB设备模拟为鼠标、键盘等其他的外设，**默认情况下CubeMX生成的HID外设为鼠标**

鼠标设备和计算机通过USB通信采用HID的鼠标协议，该协议由四个字节组成，用于向计算机报告当前鼠标的状态，四个字节代表的含义如下图所示



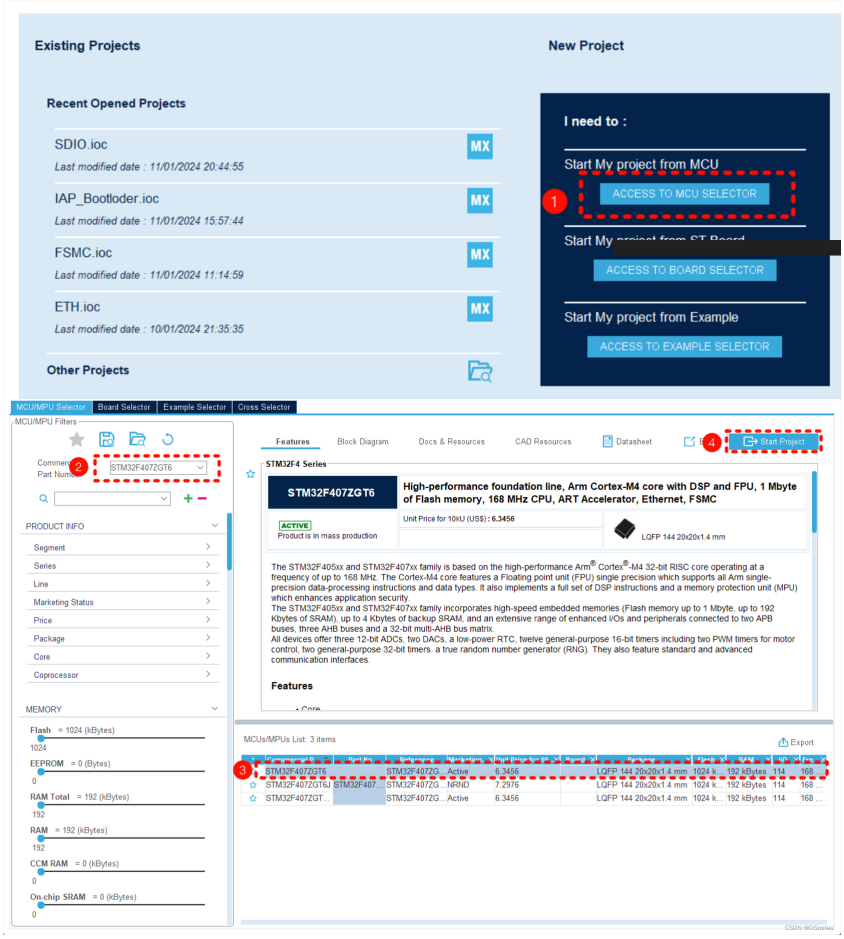
第一个字节共8位数据用于表示鼠标上的按键状态，每个位代表一个按钮，1表示按下，0表示未按下，最左边的Button位于字节的低位，通常下最低位表述鼠标左键，第一位表示鼠标右键，第二位表示鼠标中键，比如设置该字节数据为0x01，则表示鼠标左键被按下

第二个字节表示鼠标在水平（X）方向上的相对移动，比如设置该字节数据为10，则表示X正方向移动10刻度；第三个字节表示鼠标在竖直（Y）方向上的相对移动，比如设置该字节数据为-10，则表示Y负方向移动10刻度；第四个字节表示滚轮的状态，比如设置该字节数据为10表示向上滚动10刻度

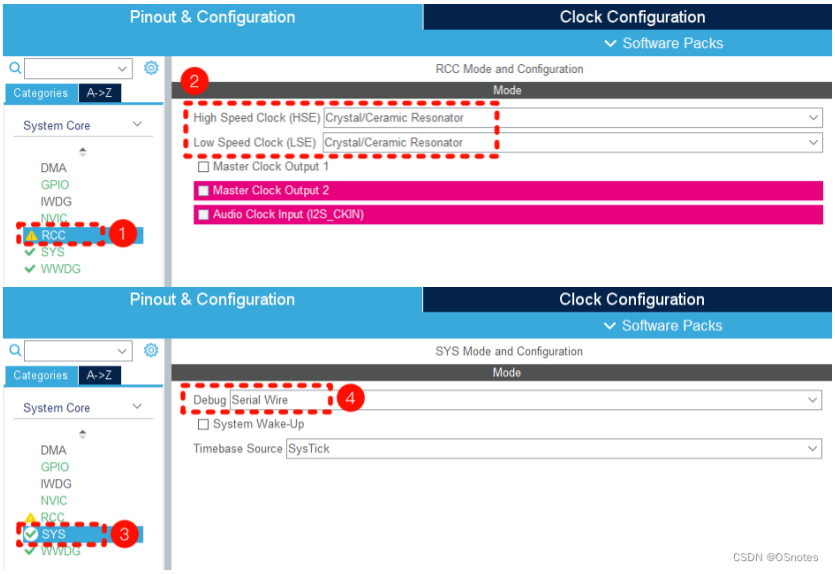
3.1、CubeMX相关配置

3.1.0、工程基本配置

打开STM32CubeMX软件，单击ACCESS TO MCU SELECTOR选择开发板MCU（选择你使用开发板的主控MCU型号），选中MCU型号后单击页面右上角Start Project开始工程，具体如下图所示



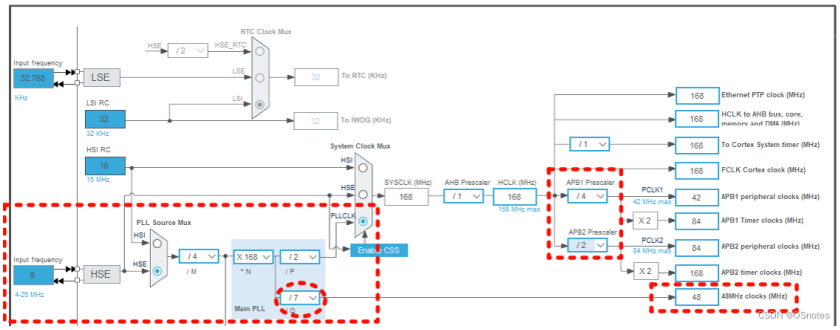
开始工程之后在配置主页面System Core/RCC中配置HSE/LSE晶振，在System Core/SYS中配置Debug模式，具体如下图所示



详细工程建立内容读者可以阅读“STM32CubeMX教程1 工程建立”

3.1.1、时钟树配置

将时钟树　中48MHz时钟配置为48MHz，也即将Main PLL（主锁相环）的Q参数调节为7，其他HCLK、PCLK1和PCLK2时钟仍然设置为STM32F407能达到的最高时钟频率，具体如下图所示



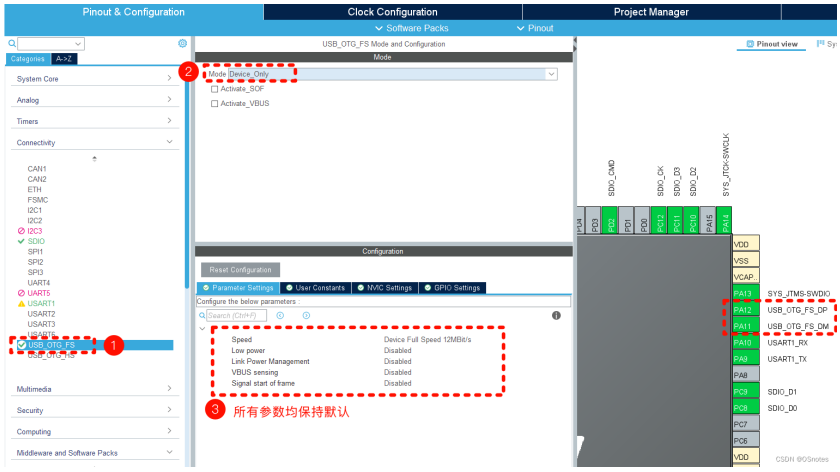
3.1.2、外设参数配置

本实验需要初始化　开发板上WK_UP、KEY2、KEY1和KEY0用户按键，具体配置步骤请阅读“STM32CubeMX教程3 GPIO输入 - 按键响应”，**注意两个实验使用的开发板不同，但配置步骤原理相同**

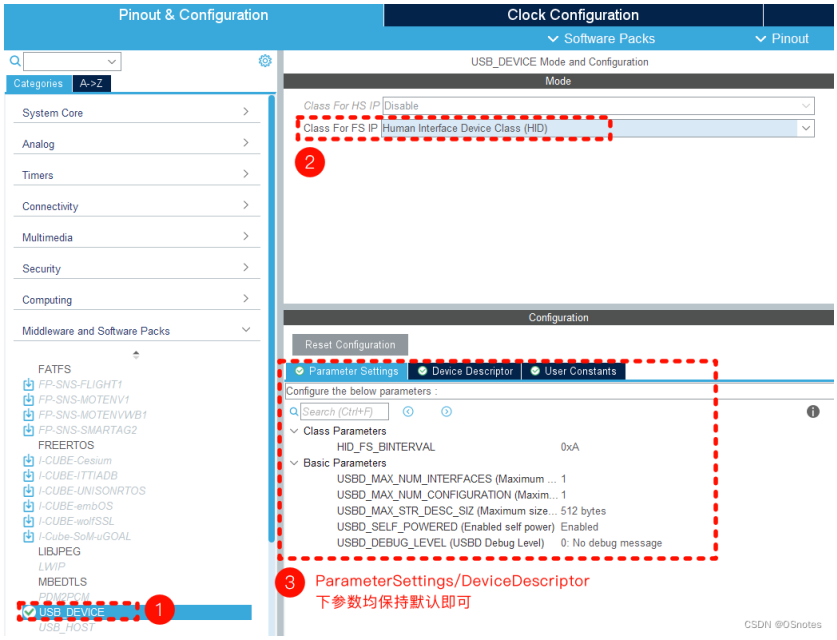
本实验需要初始化TIM6外设实现1ms定时，具体配置步骤请阅读“STM32CubeMX教程5 TIM 定时器概述及基本定时器”

本实验需要初始化USART1作为输出信息渠道，具体配置步骤请阅读“STM32CubeMX教程9 USART/UART 异步通信”

单击Pinout & Configuration页面左边功能分类栏目中**Connectivity/USB_OTG_FS，将其模式配置为仅从机（Device_Only），其他所有参数保持默认即可**，具体配置如下图所示



单击Pinout & Configuration页面左边功能分类栏目中**Middleware and Software Packs/USB DEVICE，将其模式配置为Human Interface Device Class (HID)（人机接口设备类），其他所有参数保持默认即可**，具体配置如下图所示



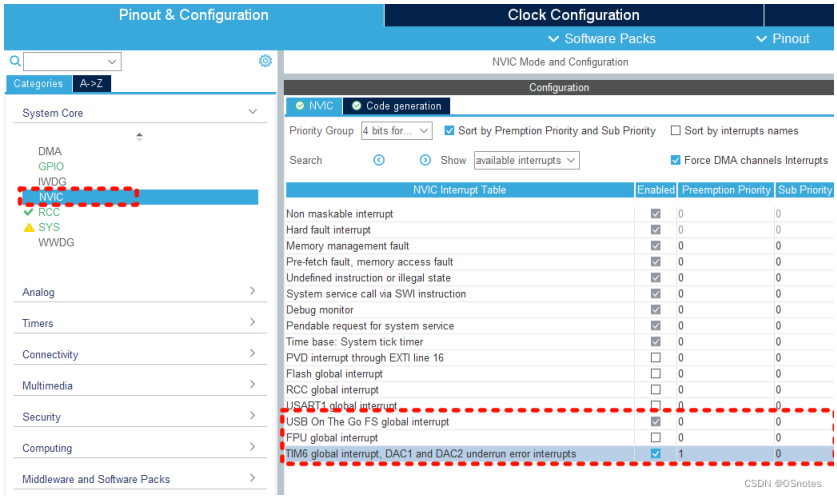
HID_FS_BINTERVAL （指定中断传输的轮询间隔）：可选0x01 ~ 0xFF，以毫秒为单位，此处设置为0XA表示USB主机每10ms轮询一次USB设备获取新的信息

Parameter Settings和Device Descriptor选项卡下其余参数请阅读STM32CubeMX教程30 USB_DEVICE - MSC外设_读卡器实验”3.0、前提知识“和”3.1.2、外设参数配置“两个小节

3.1.3、外设中断配置

当在Middleware and SoftwarePacks中配置了USB_DEVICE的模式不为Disable时，便会自动开启USB_OTG的全局中断，且不可关闭，用户配置合适的中断优先级即可

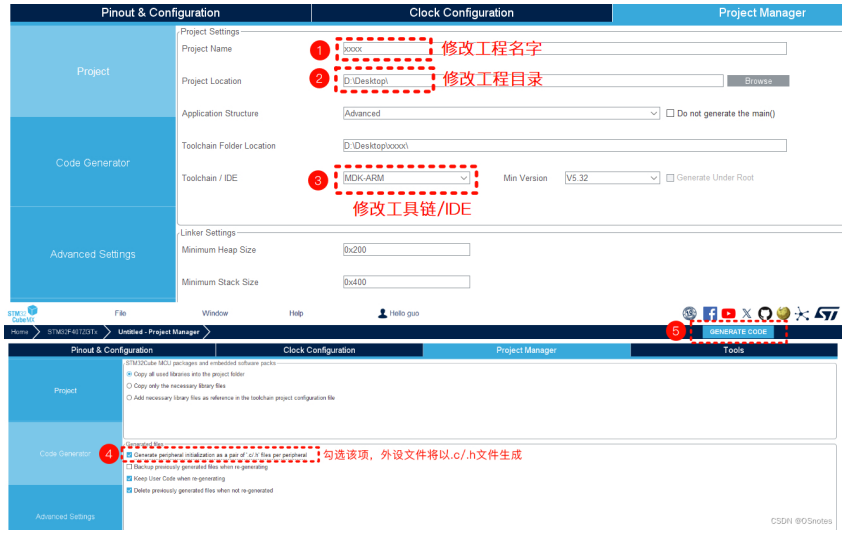
注意本实验需要开启基本定时器TIM6的全局中断，勾选NVIC下的全局中断，具体配置如下图所示



3.2、生成代码

3.2.0、配置Project Manager页面

单击进入Project Manager页面，在左边Project分栏中修改工程名称、工程目录和工具链，然后在Code Generator中勾选“Gnerate peripheral initialization as a pair of 'c/h' files per peripheral”，最后单击页面右上角GENERATE CODE生成工程，具体如下图所示



详细Project Manager配置内容读者可以阅读“STM32CubeMX教程1 工程建立”实验3.4.3小节

3.2.1、设初始化调用流程

暂无

3.2.2、外设中断调用流程

暂无

3.2.3、添加其他必要代码

在main.c文件最下方添加通过按键设置鼠标指针坐标值的函数 和 TIM6定时器1ms回调函数，具体 源代码 如下所示

```c
/*设置鼠标指针坐标值*/
static void GetPointerData(uint8_t *pbuf)
{
  int8_t  x = 0, y = 0, button = 0, Wheel=0;

    /*按键WK_UP被按下*/
    if(HAL_GPIO_ReadPin(WK_UP_GPIO_Port,WK_UP_Pin) == GPIO_PIN_SET)
    {
        if(HAL_GPIO_ReadPin(WK_UP_GPIO_Port,WK_UP_Pin) == GPIO_PIN_SET)
        {
            printf("Scroll the wheel up\r\n");
            //y -= CURSOR_STEP;
            Wheel = 10;
        }
    }
    /*按键KEY2被按下*/
    if(HAL_GPIO_ReadPin(KEY2_GPIO_Port,KEY2_Pin) == GPIO_PIN_RESET)
    {
        if(HAL_GPIO_ReadPin(KEY2_GPIO_Port,KEY2_Pin) == GPIO_PIN_RESET)
        {
            printf("←←←\r\n");
            x -= CURSOR_STEP;
        }
    }
    /*按键KEY1被按下*/
    if(HAL_GPIO_ReadPin(KEY1_GPIO_Port,KEY1_Pin) == GPIO_PIN_RESET)
    {
        if(HAL_GPIO_ReadPin(KEY1_GPIO_Port,KEY1_Pin) == GPIO_PIN_RESET)
        {
            printf("Left_Button_Pressed\r\n");
            //y += CURSOR_STEP;
            button = 0x01;
        }
    }
    /*按键KEY0被按下*/
    if(HAL_GPIO_ReadPin(KEY0_GPIO_Port,KEY0_Pin) == GPIO_PIN_RESET)
    {
        if(HAL_GPIO_ReadPin(KEY0_GPIO_Port,KEY0_Pin) == GPIO_PIN_RESET)
        {
            printf("→→→\r\n");
            x += CURSOR_STEP;
        }
    }
  pbuf[0] = button;
```

```
  pbuf[1] = x;  pbuf[2] = y;
  pbuf[3] = Wheel;
}

/*TIM6定时器1ms回调函数*/
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    static __IO uint32_t counter = 0;

    /* check Joystick state every polling interval (10ms) */
    if(counter++ == USBD_HID_GetPollingInterval(&hUsbDeviceFS))
    {
        GetPointerData(HID_Buffer);

        /* send data though IN endpoint*/
        if((HID_Buffer[0] != 0) || (HID_Buffer[1] != 0) || (HID_Buffer[2] != 0) || (HID_Buffer[3] != 0))
        {
            USBD_HID_SendReport(&hUsbDeviceFS, HID_Buffer, sizeof(HID_Buffer));
        }
        counter = 0;
    }
}
```

在main.c文件中包含使用到的头文件，以及定义/声明使用到的一些变量，最后在主函数main()初始化外设完毕后以中断方式打开TIM6定时器即可，具体源代码如下所示

```
/*main.c文件中*/
/*包含头文件*/
#include "usbd_hid.h"

/*定义/声明变量*/
extern USBD_HandleTypeDef hUsbDeviceFS;
#define CURSOR_STEP 7
uint8_t HID_Buffer[4];

/*主函数进入主循环前启动TIM6定时器*/
HAL_TIM_Base_Start_IT(&htim6);
```

4、烧录验证

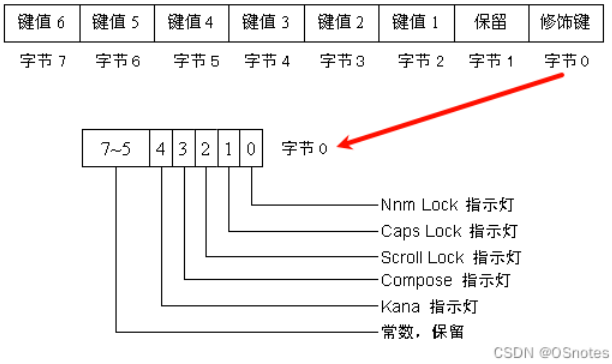
烧录程序，使用USB连接线将开发板上USB_SALVE接口与Windows电脑的USB接口连接，连接成功后可以通过串口助手监视系统的运行

首先按下开发板上的KEY2和KEY0左右两个用户按键，可以发现电脑上的鼠标光标会随着按键的按下向左或者向右移动，然后按下WK_UP上方用户按键可以发现串口助手显示的内容被拉到最上方，也即实现了滚轮向上滚动，然后将鼠标光标移动到串口助手的打开/关闭串口按钮上，按下KEY1按键之后发现可以控制串口的打开/关闭，具体现象如下图所示

5、模拟键盘实验流程简述

5.0、前提知识

键盘设备和计算机通过USB通信采用HID的键盘协议，该协议由八个字节组成，用于向计算机报告当前键盘的状态，八个字节代表的含义如下图所示 *(注释1)*



5.1、CubeMX相关配置

无需做任何修改，直接使用模拟鼠标时生成的工程代码

5.2、生成代码

打开生成的工程代码，由于CubeMX默认将设备描述为了鼠标设备，可以在usbd_hid.c文件中找到一个名为HID_MOUSE_ReportDesc的数组，该数组正式鼠标报告设备描述符，因此需要将该设备描述符修改为键盘的设备描述符，同时也应该修改该报告设备描述符数组的大小HID_MOUSE_REPORT_DESC_SIZE，具体修改内容如下所示 *（注释2）*

```
/*修改usbd_hid.c中的报告设备描述符*/
__ALIGN_BEGIN static uint8_t HID_MOUSE_ReportDesc[HID_MOUSE_REPORT_DESC_SIZE] __ALIGN_END =
{
    0x05, 0x01, // USAGE_PAGE (Generic Desktop) //63
    0x09, 0x06, // USAGE (Keyboard)
    0xa1, 0x01, // COLLECTION (Application)
    0x05, 0x07, // USAGE_PAGE (Keyboard)
    0x19, 0xe0, // USAGE_MINIMUM (Keyboard LeftControl)
    0x29, 0xe7, // USAGE_MAXIMUM (Keyboard Right GUI)
    0x15, 0x00, // LOGICAL_MINIMUM (0)
    0x25, 0x01, // LOGICAL_MAXIMUM (1)
    0x75, 0x01, // REPORT_SIZE (1)
    0x95, 0x08, // REPORT_COUNT (8)
    0x81, 0x02, // INPUT (Data,Var,Abs)
    0x95, 0x01, // REPORT_COUNT (1)
    0x75, 0x08, // REPORT_SIZE (8)
    0x81, 0x03, // INPUT (Cnst,Var,Abs)
    0x95, 0x05, // REPORT_COUNT (5)
    0x75, 0x01, // REPORT_SIZE (1)
    0x05, 0x08, // USAGE_PAGE (LEDs)
    0x19, 0x01, // USAGE_MINIMUM (Num Lock)
    0x29, 0x05, // USAGE_MAXIMUM (Kana)
    0x91, 0x02, // OUTPUT (Data,Var,Abs)
    0x95, 0x01, // REPORT_COUNT (1)
    0x75, 0x03, // REPORT_SIZE (3)
    0x91, 0x03, // OUTPUT (Cnst,Var,Abs)
    0x95, 0x06, // REPORT_COUNT (6)
    0x75, 0x08, // REPORT_SIZE (8)
    0x15, 0x00, // LOGICAL_MINIMUM (0)
    0x25, 0x65, // LOGICAL_MAXIMUM (101)
    0x05, 0x07, // USAGE_PAGE (Keyboard)
    0x19, 0x00, // USAGE_MINIMUM (Reserved (no event indicated))
    0x29, 0x65, // USAGE_MAXIMUM (Keyboard Application)
    0x81, 0x00, // INPUT (Data,Ary,Abs)
    0xc0,       // END_COLLECTION
};

/*修改usbd_hid.h中的报告设备描述符大小*/
#define HID_MOUSE_REPORT_DESC_SIZE 63U
```

修改报告设备描述符连接计算机之后，计算机就应该将其识别为一个键盘设备，计算机和该USB设备通信时就应该按照键盘设备的HID协议数据包进行数据解析，我们通过开发板上的四个按键来模拟键盘上的a/x/y/z四个按键，将程序直接实现在main.c文件中，具体源代码如下所示

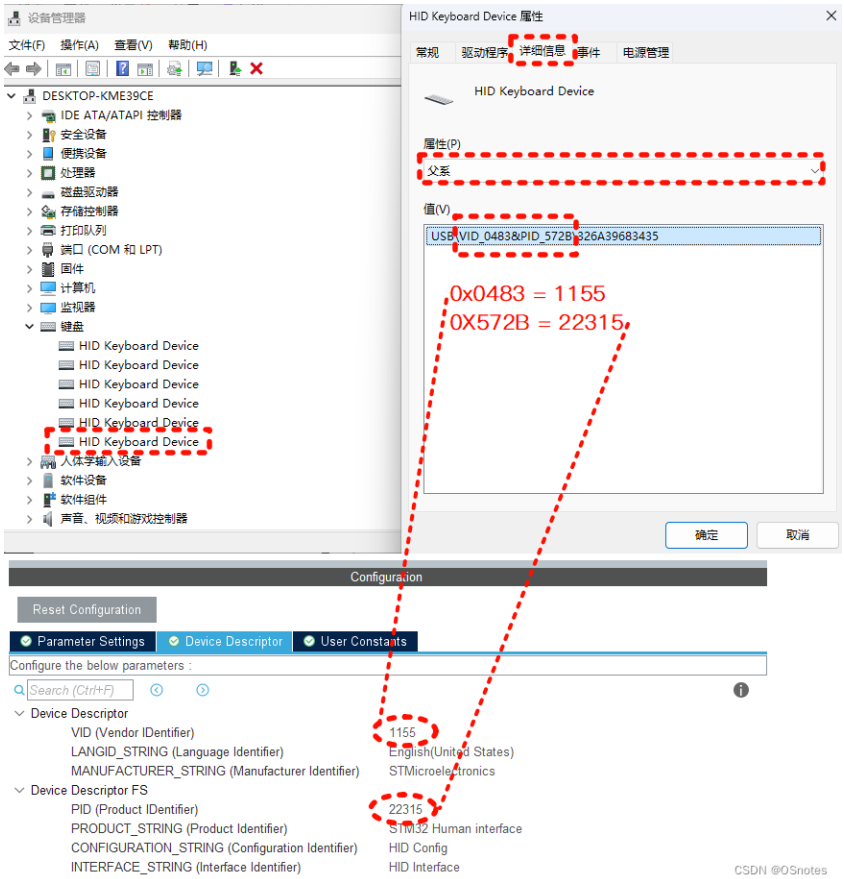
```
/*设置鼠标指针坐标值*/
static void GetPointerData(uint8_t *pbuf)
{
  int8_t keyboard = 0;

    /*按键WK_UP被按下*/
    if(HAL_GPIO_ReadPin(WK_UP_GPIO_Port,WK_UP_Pin) == GPIO_PIN_SET)
    {
```

```
        if(HAL_GPIO_ReadPin(WK_UP_GPIO_Port,WK_UP_Pin) == GPIO_PIN_SET)
        {
            printf("WK_UP Pressed : a/A\r\n");
            keyboard = 0x04;
            while(HAL_GPIO_ReadPin(WK_UP_GPIO_Port,WK_UP_Pin));
        }
    }
    /*按键KEY2被按下*/
    if(HAL_GPIO_ReadPin(KEY2_GPIO_Port,KEY2_Pin) == GPIO_PIN_RESET)
    {
        if(HAL_GPIO_ReadPin(KEY2_GPIO_Port,KEY2_Pin) == GPIO_PIN_RESET)
        {
            printf("KEY2 Pressed : x/X\r\n");
            keyboard = 0x1B;
            while(!HAL_GPIO_ReadPin(KEY2_GPIO_Port,KEY2_Pin));
        }
    }
    /*按键KEY1被按下*/
    if(HAL_GPIO_ReadPin(KEY1_GPIO_Port,KEY1_Pin) == GPIO_PIN_RESET)
    {
        if(HAL_GPIO_ReadPin(KEY1_GPIO_Port,KEY1_Pin) == GPIO_PIN_RESET)
        {
            printf("KEY1 Pressed : y/Y\r\n");
            keyboard = 0x1C;
            while(!HAL_GPIO_ReadPin(KEY1_GPIO_Port,KEY1_Pin));
        }
    }
    /*按键KEY0被按下*/
    if(HAL_GPIO_ReadPin(KEY0_GPIO_Port,KEY0_Pin) == GPIO_PIN_RESET)
    {
        if(HAL_GPIO_ReadPin(KEY0_GPIO_Port,KEY0_Pin) == GPIO_PIN_RESET)
        {
            printf("KEY0 Pressed : z/Z\r\n");
            keyboard = 0x1D;
            while(!HAL_GPIO_ReadPin(KEY0_GPIO_Port,KEY0_Pin));
        }
    }
    //合成键盘数据包
    for(uint8_t i=0;i<8;i++)
    {
        if(i == 2) pbuf[i] = keyboard;
        else pbuf[i] = 0;
    }
}

/*TIM6定时器1ms回调函数*/
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    static __IO uint32_t counter = 0;

    /* check Joystick state every polling interval (10ms) */
    if(counter++ == USBD_HID_GetPollingInterval(&hUsbDeviceFS))
    {
        GetPointerData(HID_Buffer);

        /* send data though IN endpoint*/
        USBD_HID_SendReport(&hUsbDeviceFS, HID_Buffer, sizeof(HID_Buffer));

        /* 重置counter */
        counter = 0;
    }
}
```

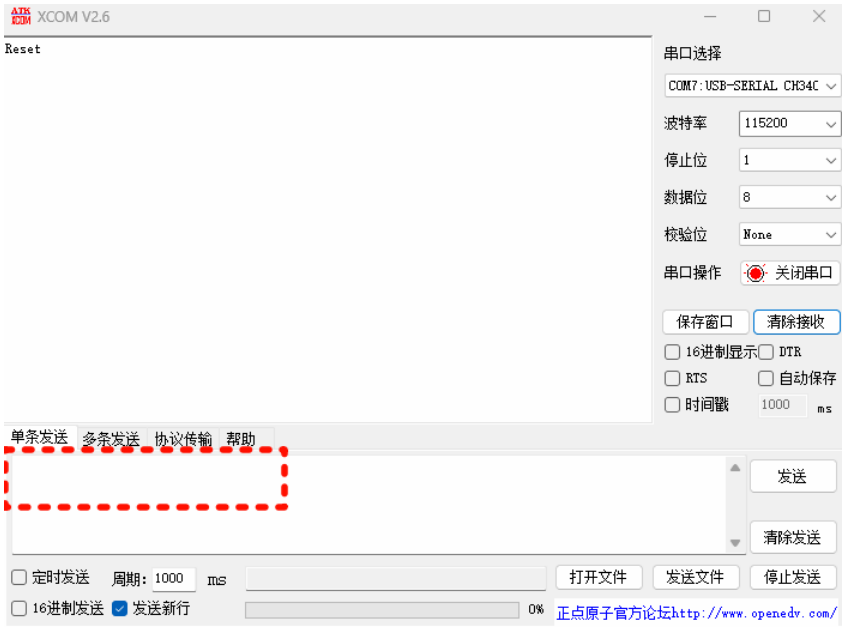
5.3、烧录验证

烧录程序，使用USB连接线将开发板上USB_SALVE接口与Windows电脑的USB接口连接，连接成功后可以通过串口助手监视系统的运行

首先我们可以通过设备管理器查找一下该设备，看看Windwos将其识别为了什么设备，打开设备管理器，在键盘中找到最后一个，右键查看其属性，在详细信息页面属性中找到父系，在下方可以查看到该设备的VID和PID，可以发现和我们配置的HID设备描述中的ID一致，具体如下图所示



然后打开串口助手，将鼠标光标点击串口助手的发送数据区域，然后随机按下开发板上的四个用户按键，可以在串口助手发送数据区域发现每按下一个按键都会对应输出a、x、y、z四个字符，并且同时串口会输出哪个按键被按下的提示，具体现象如下图所示



6、常用函数

```c
/*return polling interval from endpoint descriptor*/
uint32_t USBD_HID_GetPollingInterval(USBD_HandleTypeDef *pdev)
/*Send HID Report*/
uint8_t USBD_HID_SendReport(USBD_HandleTypeDef *pdev, uint8_t *report, uint16_t len)
```

7、注释详解

注释1：图片来源 3、USB接口的键盘描述符范例

注释2：键盘的报告设备描述符来源 STM32CubeMX学习笔记（44）——USB接口使用（HID按键）

参考资料

微雪课堂：STM32CubeMX系列教程25:USB Device