M463 使用 TinyUSB 实现鼠标与键盘

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

应用简述	本范例代码使用 M463 MCU 基于 TinyUSB 架构实现鼠标与键盘的功能
BSP 版本	M460_Series_BSP_CMSIS_V3.00.004
开发平台	NuMaker-M463KG V1.0

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. 概述

本范例代码使用 Nuvoton M463 系列微控制器基于 TinyUSB Library 来实现 USB HID（Human Interface Device）鼠标与键盘的功能。透过本范例，用户可以了解如何初始化 MCU、处理 HID 类别的鼠标与键盘事件。本范例适用于对 USB HID 协议有初步认识，并希望快速建立应用雏型的开发者。本应用采用 M463 的 USB 高速（USB HS）接口与 GPIO 来仿真鼠标与键盘输入，并透过 TinyUSB 提供的 API 进行 HID 数据传输。

1.1 原理

Nuvoton M463 系列微控制器具备 USB High-Speed Device，支持 USB 2.0 装置功能。此范例透过 TinyUSB 对 USB 装置初始化、设定描述符，让开发板在接上计算机后被辨识为 HID 类型的鼠标与键盘。

TinyUSB 是一个开源、跨平台的 USB Host/Device Stack，设计目标是提供简单易用、可移植性高且资源占用小的 USB 协议支持，适合嵌入式系统开发。TinyUSB 架构模块化，依功能可分为以下几个主要部分：

- **Device Stack**：支持 USB 装置端功能，包含多种 USB 类别如 CDC（虚拟串口）、MSC（大容量储存装置）、HID（人机接口装置）、MIDI、DFU 等。
- **Host Stack**：支持 USB 主机端功能，可控制外接装置（如读取 USB 随身碟、鼠标等）。
- **Class Drivers**：每种 USB 类别对应的驱动模块，用于解析数据与处理控制流程。
- **Board Support Layer**：用于将 TinyUSB 连接至 MCU 的 USB 控制器，实作底层传输逻辑。

在本范例中，我们主要使用其 Device Stack 中的 HID 类别驱动，使 M463 能够在与计算机连接后被辨识为 USB 鼠标与键盘装置。TinyUSB 提供统一且直观的 API，例如 tud_hid_keyboard_report() 与 tud_hid_mouse_report()，让开发者无需关心底层传输协议，即可轻松实现 HID 数据传输。

本范例使用 tud_hid_keyboard_report()仿真键盘数据的传输：

```
bool tud_hid_keyboard_report(uint8_t instance, uint8_t report_id, uint8_t modifier, uint8_t const keycode[6]);
```

详细的参数说明如下：

- **instance**：HID 实例编号，通常为 0。
- **report_id**：若 HID 描述符中启用了 Report ID，则填入对应值，否则设为 0。
- **modifier**：修饰键，如 LEFT SHIFT、RIGHT CTRL 等，亦可设为 0 表示无修饰键。
- **keycode[6]**：一次最多传送 6 个按键代码，若未使用的位置需填入 0。

另外，本范例也有使用 tud_hid_mouse_report()仿真鼠标数据的传输：

bool tud_hid_mouse_report(uint8_t instance, uint8_t report_id, uint8_t buttons, int8_t x, int8_t y, int8_t vertical, int8_t horizontal);

详细的参数说明如下：

- **instance**：HID 实例编号，通常为 0。
- **report_id**：若 HID 描述符中启用了 Report ID，则需填入对应值，否则设为 0。
- **buttons**：鼠标按钮状态，最多 8 位（bit 0~2 分别为左、中、右键）。
- **x**：光标在 X 轴的相对移动（-127 ~ 127）。
- **y**：光标在 Y 轴的相对移动（-127 ~ 127）。
- **vertical**：垂直滚轮的移动量（可为正负）。
- **horizontal**：水平滚轮的移动量（可为正负）。

1.2 执行结果

本范例代码执行编译和烧录后，通过 UART 串口可以看到范例代码的相关信息：

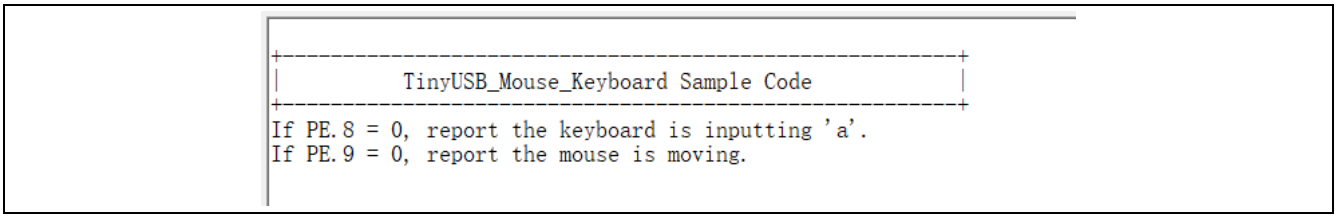


图 1-1 UART 信息

将 USB HS Port 连接到 PC 后，可以根据信息按下 BTN0(PE.8)仿真键盘按下'a'按键：

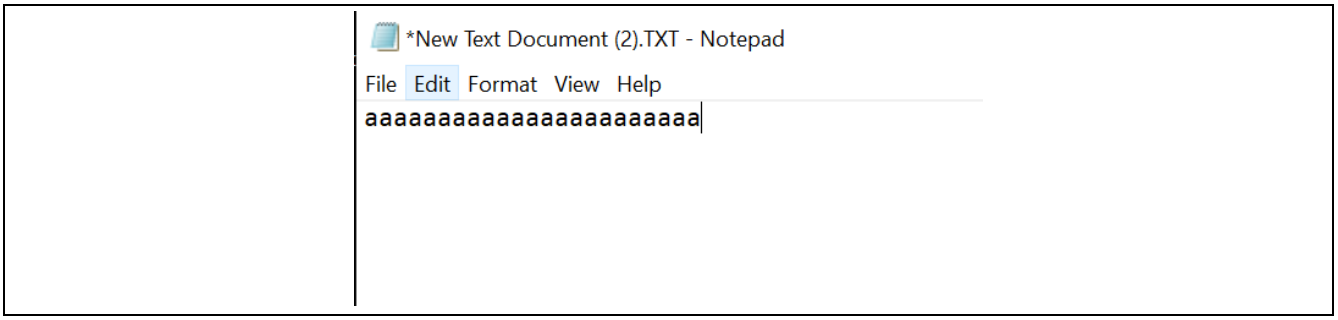


图 1-2 仿真键盘按键

按下 BTN1(PE.9)仿真鼠标向右下角移动：

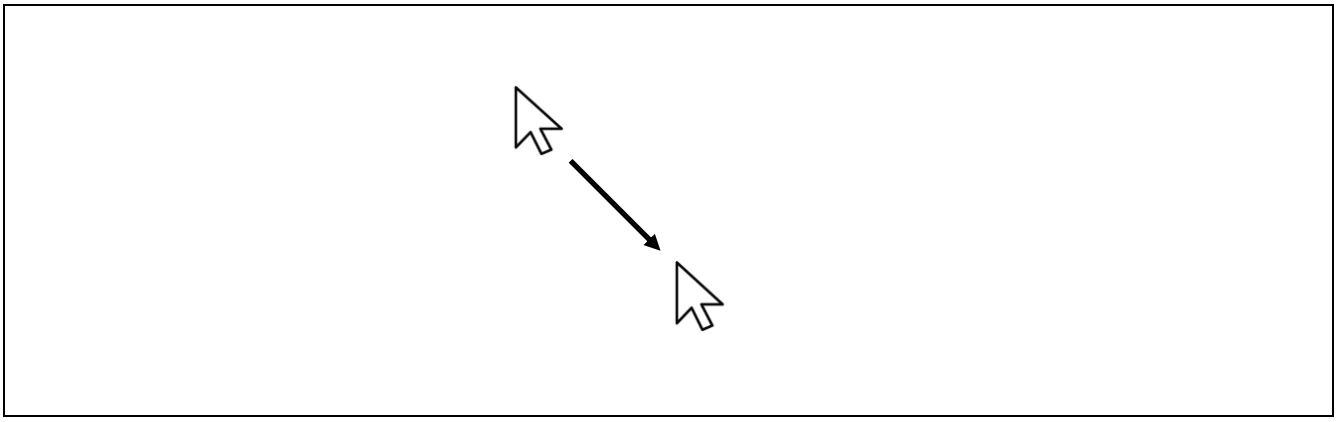


图 1-3 仿真鼠标移动

2. 代码介绍

初始化 MCU 并调用 TinyUSB Library 的函数，代码位于 *main.c*。

```
void SYS_Init(void)
{
    uint32_t volatile i;

    /* Unlock protected registers */
    SYS_UnlockReg();

    /*---------------------------------------------------*/
    /* Init System Clock                                 */
    /*---------------------------------------------------*/

    /* Enable HIRC and HXT clock */
    CLK_EnableXtalRC(CLK_PWRCTL_HIRCEN_Msk | CLK_PWRCTL_HXTEN_Msk);

    /* Wait for HIRC and HXT clock ready */
    CLK_WaitClockReady(CLK_STATUS_HIRCSTB_Msk | CLK_STATUS_HXTSTB_Msk);

    /* Set PCLK0 and PCLK1 to HCLK/2 */
    CLK->PCLKDIV = (CLK_PCLKDIV_APB0DIV_DIV2 | CLK_PCLKDIV_APB1DIV_DIV2);

    /* Set core clock to 200MHz */
    CLK_SetCoreClock(FREQ_200MHZ);

    /* Enable all GPIO clock */
    CLK->AHBCLK0 |= CLK_AHBCLK0_GPACKEN_Msk | CLK_AHBCLK0_GPBCKEN_Msk |
CLK_AHBCLK0_GPCCKEN_Msk | CLK_AHBCLK0_GPDCKEN_Msk |
                    CLK_AHBCLK0_GPECKEN_Msk | CLK_AHBCLK0_GPFCKEN_Msk |
CLK_AHBCLK0_GPGCKEN_Msk | CLK_AHBCLK0_GPHCKEN_Msk;
    CLK->AHBCLK1 |= CLK_AHBCLK1_GPICKEN_Msk | CLK_AHBCLK1_GPJCKEN_Msk;

    /* Enable UART0 module clock */
    CLK_EnableModuleClock(UART0_MODULE);

    /* Select UART0 module clock source as HIRC and UART0 module clock divider as 1 */
    CLK_SetModuleClock(UART0_MODULE, CLK_CLKSEL1_UART0SEL_HIRC, CLK_CLKDIV0_UART0(1));

    /* Select HSUSBD */
    SYS->USBPHY &= ~SYS_USBPHY_HSUSBROLE_Msk;

    /* Enable USB PHY */
    SYS->USBPHY = (SYS->USBPHY & ~(SYS_USBPHY_HSUSBROLE_Msk | SYS_USBPHY_HSUSBACT_Msk)) |
SYS_USBPHY_HSUSBEN_Msk;
    for(i = 0; i < 0x1000; i++);    // delay > 10 us
    SYS->USBPHY |= SYS_USBPHY_HSUSBACT_Msk;

    /* Enable HSUSBD module clock */
    CLK_EnableModuleClock(HSUSBD_MODULE);

    /*---------------------------------------------------*/
    /* Init I/O Multi-function                           */
    /*---------------------------------------------------*/

    /* Set multi-function pins for UART0 RXD and TXD */
    SET_UART0_RXD_PB12();
    SET_UART0_TXD_PB13();

    /* Lock protected registers */
    SYS_LockReg();
}
```

```
/*---------------------------------------------------*/
/*  Main Function                                    */
/*---------------------------------------------------*/
int32_t main(void)
{
    /* Unlock protected registers */
    SYS_UnlockReg();

    /* Init System, peripheral clock and multi-function I/O */
    SYS_Init();

    /* Init UART0 to 115200-8n1 for print message */
    UART_Open(UART0, 115200);

    printf("\n");
    printf("+------------------------------------------------------------+\n");
    printf("|          TinyUSB_Mouse_Keyboard Sample Code                |\n");
    printf("+------------------------------------------------------------+\n");
    printf("If PE.8 = 0, report the keyboard is inputting 'a'.\n");
    printf("If PE.9 = 0, report the mouse is moving.\n");

    /* Set PE.8 & PE.9 as Quasi-bidirectional mode */
    GPIO_SetMode(PE, BIT8| BIT9, GPIO_MODE_QUASI);
    PE8 = 1;
    PE9 = 1;

    // 1ms tick timer
    SysTick_Config(200000000 / 1000);

    // init device stack on configured roothub port
    tud_init(BOARD_TUD_RHPORT);

    while(1) {
        tud_task(); // tinyusb device task

        hid_task();
    }
}
```

使用 BTN0(PE.8)及 BTN1(PE.9) 按键来触发鼠标事件和键盘事件，当按下 BTN0(PE.8)调用 tud_hid_n_keyboard_report() 传输键盘事件的数据，当按下 BTN1(PE.9) 调用 tud_hid_n_mouse_report()传输鼠标事件的数据，代码位于 *main.c*。

```
    ……
void hid_task(void)
{
    // Poll every 10ms
    const uint32_t interval_ms = 10;
    static uint32_t start_ms = 0;

    if(board_millis() - start_ms < interval_ms) return;   // not enough time
    start_ms += interval_ms;

    /*------------- Keyboard -------------*/
    if(tud_hid_n_ready(ITF_KEYBOARD)) {
        // use to avoid send multiple consecutive zero report for keyboard
        static bool has_key = false;

        if(PE8==0) {
            uint8_t keycode[6] = { 0 };
```

```
            keycode[0] = HID_KEY_A;

            tud_hid_n_keyboard_report(ITF_KEYBOARD, 0, 0, keycode);

            has_key = true;
        } else {
            // send empty key report if previously has key pressed
            if(has_key) tud_hid_n_keyboard_report(ITF_KEYBOARD, 0, 0, NULL);
            has_key = false;
        }
    }

    /*------------- Mouse -------------*/
    if(tud_hid_n_ready(ITF_MOUSE)) {
        if(PE9==0) {
            int8_t const delta = 5;

            // no button, right + down, no scroll pan
            tud_hid_n_mouse_report(ITF_MOUSE, 0, 0x00, delta, delta, 0, 0);
        }
    }
}
```

Board Support Layer 代码，用来处理 USB 控制器收发 USB 封包的中断函数，将封包数据传递到 TinyUSB Library 中，位于 *dcd_M463.c*。

```
void dcd_int_handler(uint8_t rhport)
{
    ……
    if(IrqStL & (HSUSBD_GINTSTS_EPAIF_Msk | HSUSBD_GINTSTS_EPBIF_Msk |
HSUSBD_GINTSTS_EPCIF_Msk | HSUSBD_GINTSTS_EPDIF_Msk | HSUSBD_GINTSTS_EPEIF_Msk |
HSUSBD_GINTSTS_EPFIF_Msk | HSUSBD_GINTSTS_EPGIF_Msk | HSUSBD_GINTSTS_EPHIF_Msk |
HSUSBD_GINTSTS_EPIIF_Msk | HSUSBD_GINTSTS_EPJIF_Msk | HSUSBD_GINTSTS_EPKIF_Msk |
HSUSBD_GINTSTS_EPLIF_Msk)) {
        /* service PERIPH_EPA through PERIPH_EPL */
        for(ep_index = PERIPH_EPA, mask = HSUSBD_GINTSTS_EPAIF_Msk, xfer =
&xfer_table[PERIPH_EPA], ep = &HSUSBD->EP[PERIPH_EPA]; ep_index < PERIPH_MAX_EP;
ep_index++, mask <<= 1, xfer++, ep++) {
            if(IrqStL & mask) {
                uint8_t const ep_addr = xfer->ep_addr;
                bool const out_ep = !(ep_addr & TUSB_DIR_IN_MASK);
                uint32_t ep_state = ep->EPINTSTS & ep->EPINTEN;

                if(out_ep) {
                    uint16_t const available_bytes = ep->EPDATCNT &
HSUSBD_EPDATCNT_DATCNT_Msk;
                    /* copy the data from the PC to the previously provided buffer */
                    for(int count = 0; (count < available_bytes) && (xfer-
>out_bytes_so_far < xfer->total_bytes); count++, xfer->out_bytes_so_far++) {
                        *xfer->data_ptr++ = ep->EPDAT_BYTE;
                    }

                    /* when the transfer is finished, alert TinyUSB; otherwise, continue
accepting more data */
                    if((xfer->total_bytes == xfer->out_bytes_so_far) || (available_bytes <
xfer->max_packet_size)) {
```

```
                        dcd_event_xfer_complete(0, ep_addr, xfer->out_bytes_so_far,
XFER_RESULT_SUCCESS, true);
                    }

                } else if(ep_state & HSUSBD_EPINTSTS_BUFEMPTYIF_Msk) {
                    /* send any remaining data */
                    dcd_userEP_in_xfer(xfer, ep);
                } else if(ep_state & HSUSBD_EPINTSTS_TXPKIF_Msk) {
                    /* alert TinyUSB that we've finished */
                    dcd_event_xfer_complete(0, ep_addr, xfer->total_bytes,
XFER_RESULT_SUCCESS, true);
                    ep->EPINTEN = 0;
                }

                ep->EPINTSTS = ep_state;
            }
        }
    }
}
```

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - M460_Series_BSP_CMSIS_V3.00.004
- IDE 版本
  - Keil uVision 5.38

3.2 硬件需求

- 电路组件
  - NuMaker-M463KG V1.0
- 线路示意图
  - 将 UART0 TX (PB.13) pin 脚连接到 PC UART RX，以显示范例代码的信息。
  - 将 USB 线从 PC 连接到开发板的 USB HS Port，作为 TinyUSB 鼠标及键盘的连接。

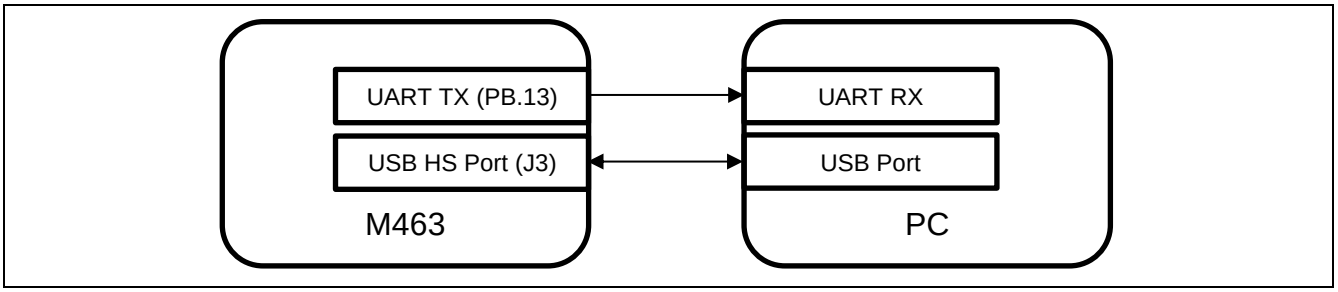


图 3-1 线路示意图

4. 目录信息

目录结构如下图所示。

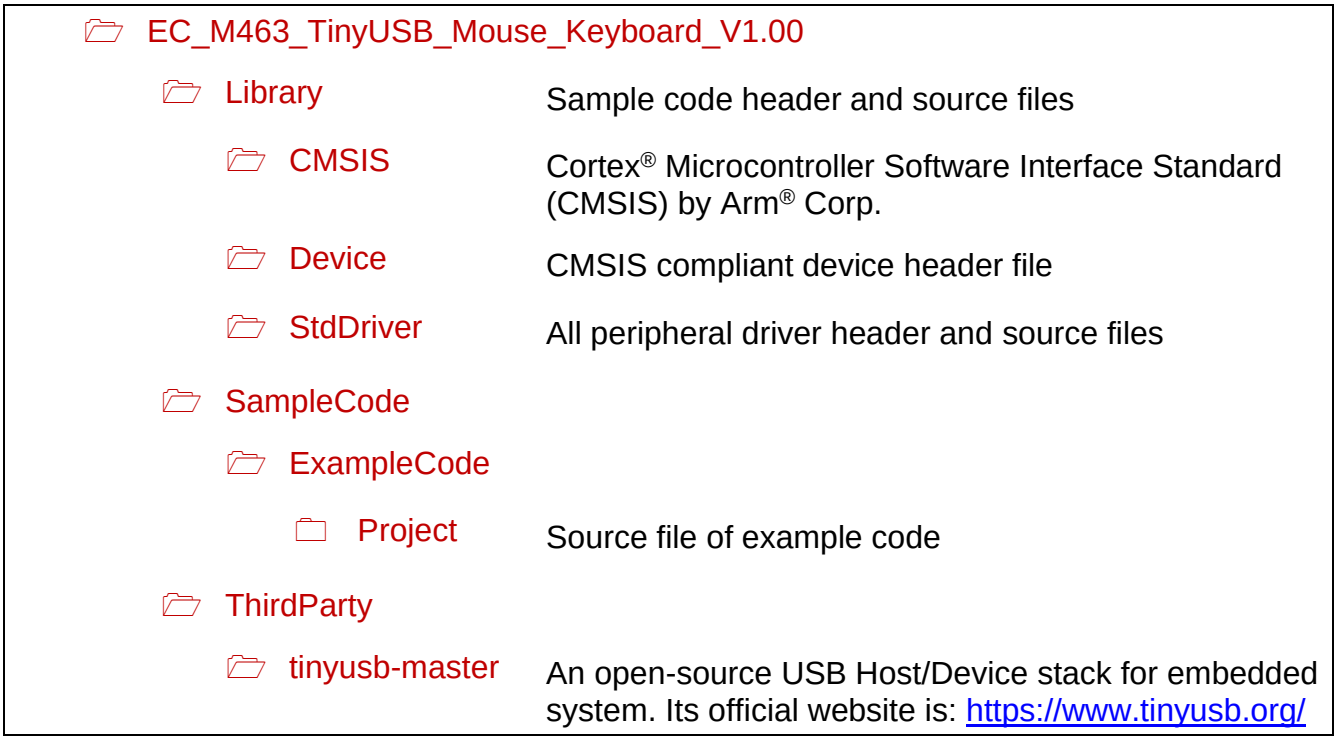


图 4-1 目录信息

5. 范例程序执行

1. 根据目录信息章节进入 ExampleCode 路径中的 KEIL 文件夹，双击 *TinyUSB_Mouse_Keyboard.uvprojx*。
2. 进入编译模式界面
   - 编译
   - 下载代码至内存
   - 进入 / 离开仿真模式
3. 进入仿真模式界面
   - 执行代码

6. 修订纪录

Date	Revision	Description
2025.04.04	1.00	初始发布。

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.