M480 HSUSBH HID to SPI

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	The example code is M480 High speed USB interface receive mouse data then SPI interface send data with PDMA mode.
BSP Version	M480 Series BSP CMSIS V3.03.001
Hardware	NuMaker-PFM-M487 V3.0

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 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

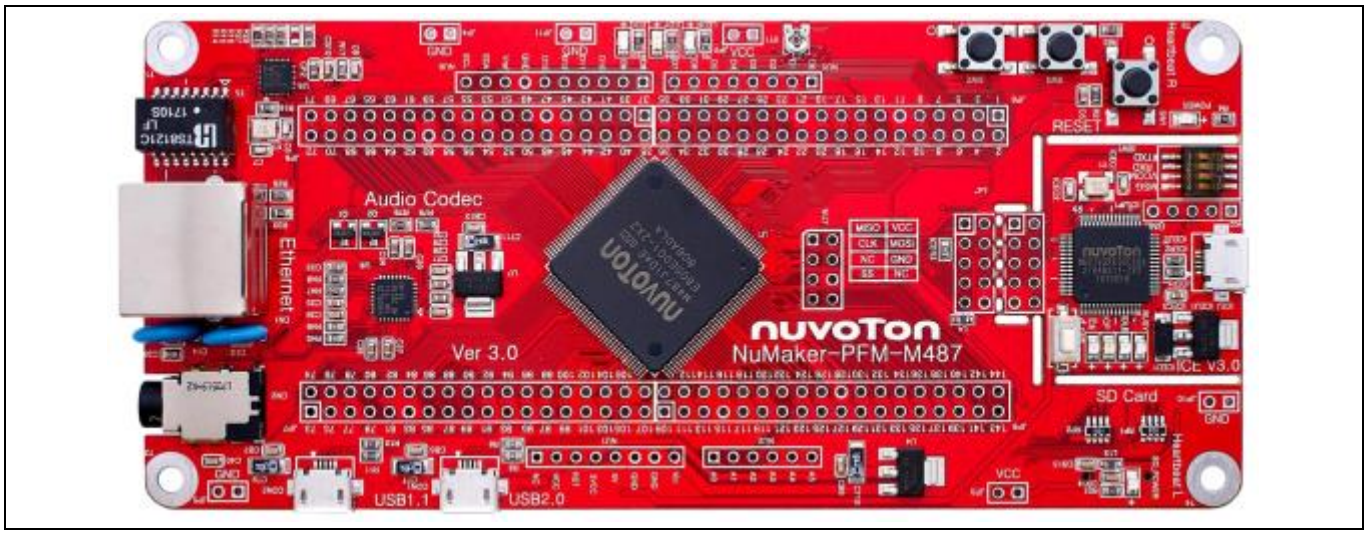
1 Function Description

1.1 Introduction

This chip is equipped with a high speed usb and a full speed usb. Both of high speed usb and full speed usb can be as host and device. The example code uses M480 high speed usb as host to receive mouse data and then transmits data via the SPI interface with PDMA on the NuMaker-PFM-M487 V3.0 board.

For more information about the NuMaker-PFM-M487 board, please visit Nuvoton website to download the UM_NuMaker-PFM_M487_User_Manual:

https://www.nuvoton.com/hq/products/microcontrollers/arm-cortex-m4-mcus/User-Manual/?__locale=en&resourcePage=Y&category=&pageIndex=3



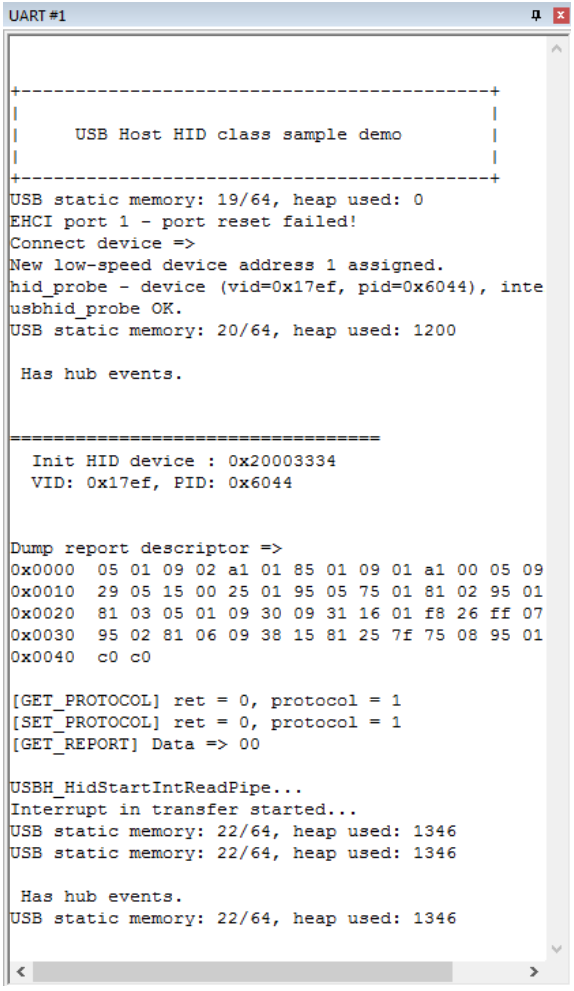
NuMaker-PFM-M487 Board

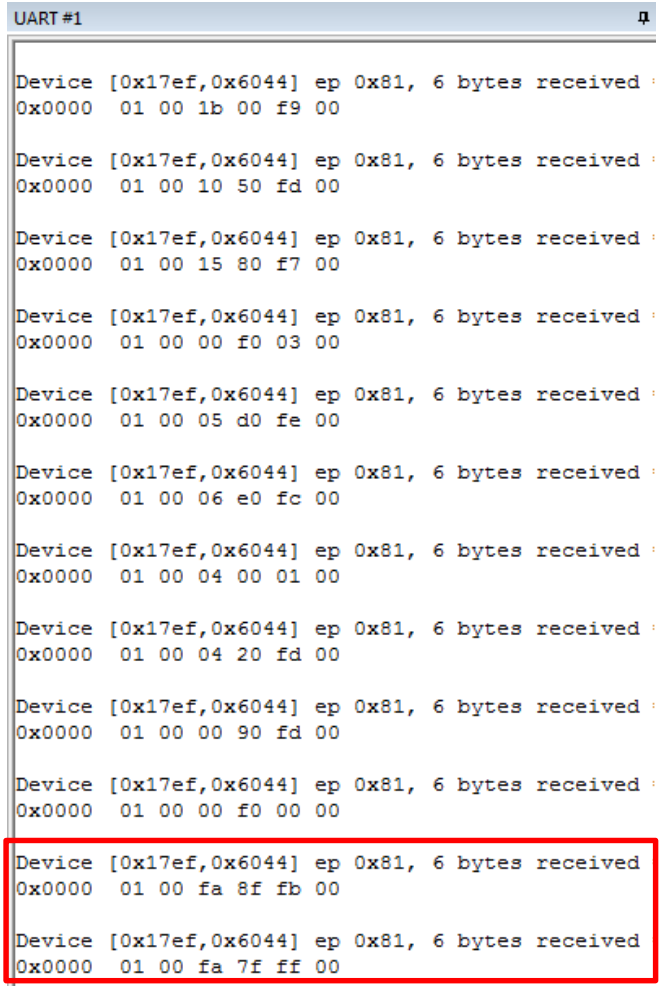
1.2 Principle

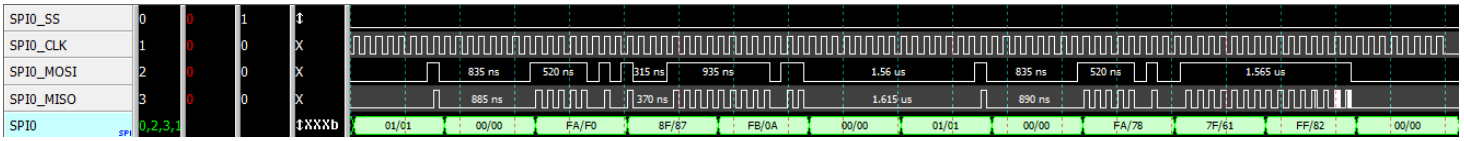
The MCU will use the USB host driver to obtain mouse data, and then print the data via SPI with PDMA. For example, when the mouse is connected to M480 that can recognize the mouse and receive the information of the mouse.

1.3 Demo Result

After executed this example code, user will be seeing below picture. M480 high speed usb will receive mouse data and send data by SPI and print data to terminal.







2 Code Description

This main() function is used to initialize platform for high speed usb, SPI and PDMA. After that, it will receive mouse data and send data with SPI PDMA.

```
void main(void)
{
    HID_DEV_T    *hdev, *hdev_list;
    SYS_Init();                         /* Init System, IP clock and multi-function I/O */
    UART0_Init();                       /* Initialize UART0 */
    SPI_Init();                         /* Init SPI */
    …
    usbh_core_init();
    usbh_hid_init();
    usbh_memory_used();
    PDMA_Init();
    while (1) {
        if (usbh_pooling_hubs()) {      /* USB Host port detect polling and management */
            usbh_memory_used();         /* print out USB memory allocating information */
            printf("\n Has hub events.\n");
            hdev_list = usbh_hid_get_device_list();
            hdev = hdev_list;
            while (hdev != NULL) {
                if (is_a_new_hid_device(hdev)) {
                    init_hid_device(hdev);
                }
                hdev = hdev->next;
            }
            update_hid_device_list(hdev_list);
            usbh_memory_used();
        }
        if((Idx-T_Idx)>=(TEST_COUNT-1))
        {
                SPI_TRIGGER_TX_PDMA(SPI0);
                while((PDMA_GET_INT_STATUS(PDMA) & PDMA_INTSTS_TDIF_Msk)==0);
                /* Clear the DMA transfer done flags */
                PDMA_CLR_TD_FLAG(PDMA, 1 << SPI_MASTER_TX_DMA_CH);
                while(SPI_IS_BUSY(SPI0));
                printf("SS1 data %d bytes => ", TEST_COUNT);
                printf("0x%02x 0x%02x 0x%02x 0x%02x\n",
HIDbuff[T_Idx],HIDbuff[T_Idx+1],HIDbuff[T_Idx+2],HIDbuff[T_Idx+3]);
```

```
                /* Master PDMA TX channel configuration */
                /* Set transfer width (32 bits) and transfer count */
                PDMA_SetTransferCnt(PDMA, SPI_MASTER_TX_DMA_CH, PDMA_WIDTH_8, TEST_COUNT);
                /* Set source/destination address and attributes */
                PDMA_SetTransferAddr(PDMA, SPI_MASTER_TX_DMA_CH,
(uint32_t)&HIDbuff[T_Idx], PDMA_SAR_INC, (uint32_t)&SPI0->TX, PDMA_DAR_FIX);
                /* Set request source; set basic mode. */
                PDMA_SetTransferMode(PDMA, SPI_MASTER_TX_DMA_CH, PDMA_SPI0_TX, FALSE, 0);
                T_Idx+=TEST_COUNT;
                if(T_Idx==1024) T_Idx=0;
        }
    }
```

This int_read_callback () function is used to read mouse usb data to SPI buffer.

```
void int_read_callback(HID_DEV_T *hdev, uint16_t ep_addr, int status, uint8_t *rdata,
uint32_t data_len)
{
    int  i;
    /*
     *  USB host HID driver notify user the transfer status via <status> parameter.
     *  If <status> is 0, the USB transfer is fine. If <status> is not zero, this
interrupt in
     *  transfer failed and HID driver will stop this pipe. It can be caused by USB
transfer error
     *  or device disconnected.
     */
    if (status < 0)
    {
        printf("Interrupt in transfer failed! status: %d\n", status);
        return;
    }
    printf("Device [0x%x,0x%x] ep 0x%x, %d bytes received =>\n", hdev->idVendor, hdev-
>idProduct, ep_addr, data_len);
    dump_buff_hex(rdata, data_len);

    for(i = 0; i < data_len; i++)
    {
        HIDbuff[Idx++]=rdata[i];
        if(Idx==1024) Idx=0;
    }
}
```

3 Software and Hardware Environment

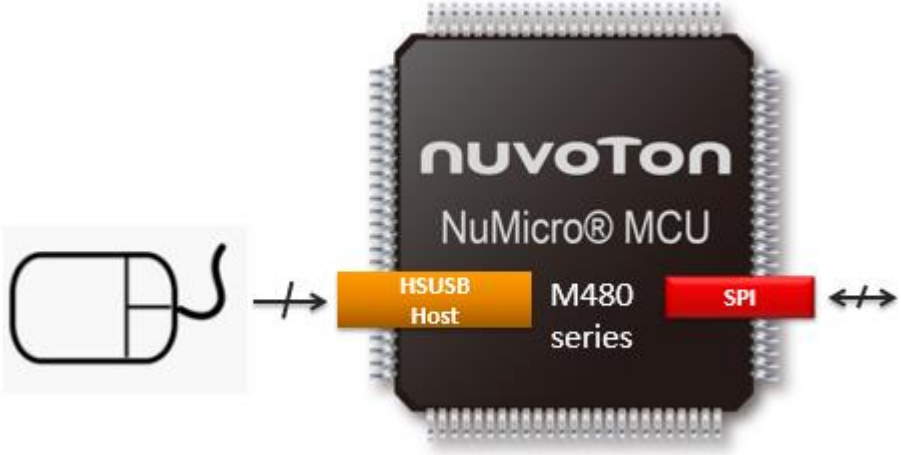
Software Environment

- BSP version
  - M480 Series BSP CMSIS V3.03.001
- IDE version
  - Keil uVersion 5.26

Hardware Environment

- Circuit components
  - NuMaker-PFM-M487 V3.0 board
- Diagram

The below picture is to show that is the diagram of example code.



4 Directory Information

- EC_M480_HSUSBH_HID_to_SPI_V1.00
  - Library — Sample code header and source files
    - CMSIS — Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
    - Device — CMSIS compliant device header file
    - StdDriver — All peripheral driver header and source files
    - UsbHostLib — USB host library source code
  - SampleCode
    - ExampleCode — Source file of example code

5 How to Execute Example Code?

1. Browsing into sample code folder by Directory Information (section 4) and double click HSUSBH_HID_to_SPI.uvproj.
2. Enter Keil compile mode
   a. Build
   b. Download
   c. Start/Stop debug session
3. Enter debug mode
   a. Run

6 Revision History

Date	Revision	Description
Jul. 18, 2019	1.00	1. Initially issued.

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.