# M480 HSUSB 8K Mouse

Example Code Introduction for 32-bit NuMicro® Family

## Information

| | |
|---|---|
| Application | This example code demonstrates how to implement a High Speed USB Mouse. |
| BSP Version | M480 Series BSP CMSIS V3.05.001 |
| 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 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 Overview

This example code implements a function that simulates a USB 2.0 high-speed mouse device. General speaking, some USB applications such as mouse and keyboard only contain low speed interface. However, some special usages such as gaming mouse need high data transfer speed. For this purpose, the example utilizes USB 2.0 high-speed interface to design a mouse. Users can utilize this example to design their own USB 2.0 high-speed mouse devices.

## 1.1 Principle

The communication between a USB host and a target device happens in segments divided into frames. For a low/full speed device, a frame is 1 millisecond (ms) wide and a high speed device defines a microframe width of 125 microseconds. Each frame is distinguished by a frame number issued as a special packet called Start of Frame (SOF). The relationship between frame and microframe is shown in Figure 1-1 from USB 2.0 Specification.

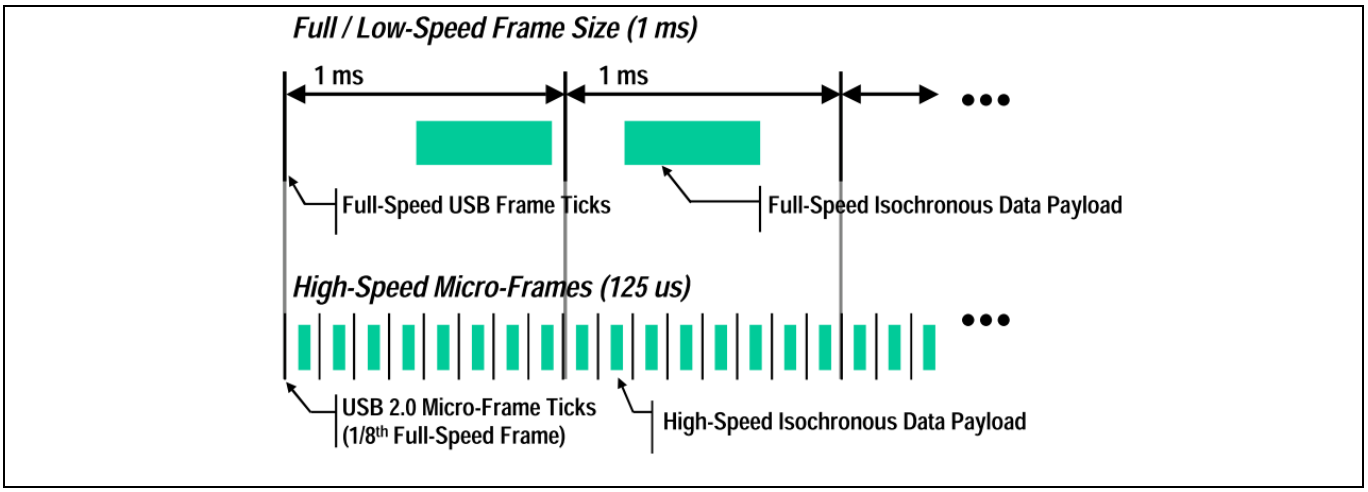


Figure 1-1 Relationship between Frame and Microframe

Within the frame there can be multiple packets of data indented to be delivered to different devices. Figure 1-2 shows the concepts.

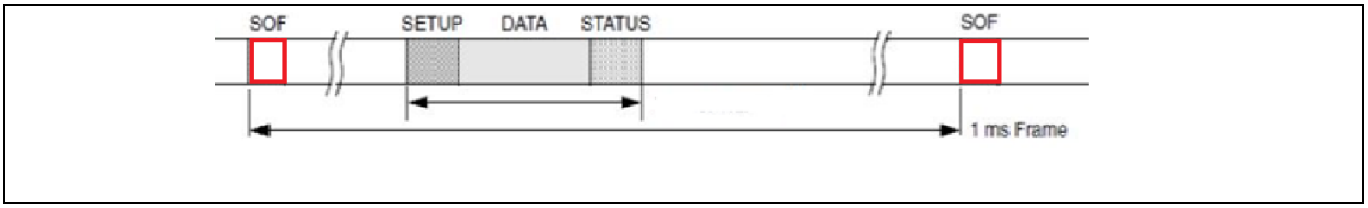


Figure 1-2 USB Packets within a Frame

For low/full speed interrupt endpoints, the descriptor's bInterval value indicates the requested maximum number of milliseconds between transaction attempts.

For high-speed interrupt endpoints, the interval is calculated differently. The value of bInterval must be between 1 and 16, the interval is in units of 125 microseconds, and the interval's value equals 2^(bInterval-1). If bInterval = 2, the interval is 2^(2-1) or two 125-microsecond periods, which means that the device is requesting a minimum of one poll per millisecond.

## 1.2 Demo Result

When connecting the USB 2.0 high-speed interface (CON1) of NuMaker-PFM-M487 to the host PC, the board is enumerated to be a USB high-speed mouse. Figure 1-3 shows the demo result.

```
        ---------------- Connection Information ---------------
Connection Index          : 0x02 (Port 2)
Connection Status         : 0x01 (DeviceConnected)
Current Config Value      : 0x01 (Configuration 1)
Device Address            : 0x10 (16)
Is Hub                    : 0x00 (no)
Device Bus Speed          : 0x02 (High-Speed)
Number Of Open Pipes      : 0x01 (1 pipe to data endpoints)
Pipe[0]                   : EndpointID=1  Direction=IN   ScheduleOffset=0
```

Figure 1-3 Demo Result

As long as SW3 button is pressed, the mouse pointer is moved to a different direction. Figure 1-4 shows the flow of button pressing and response of mouse pointer.

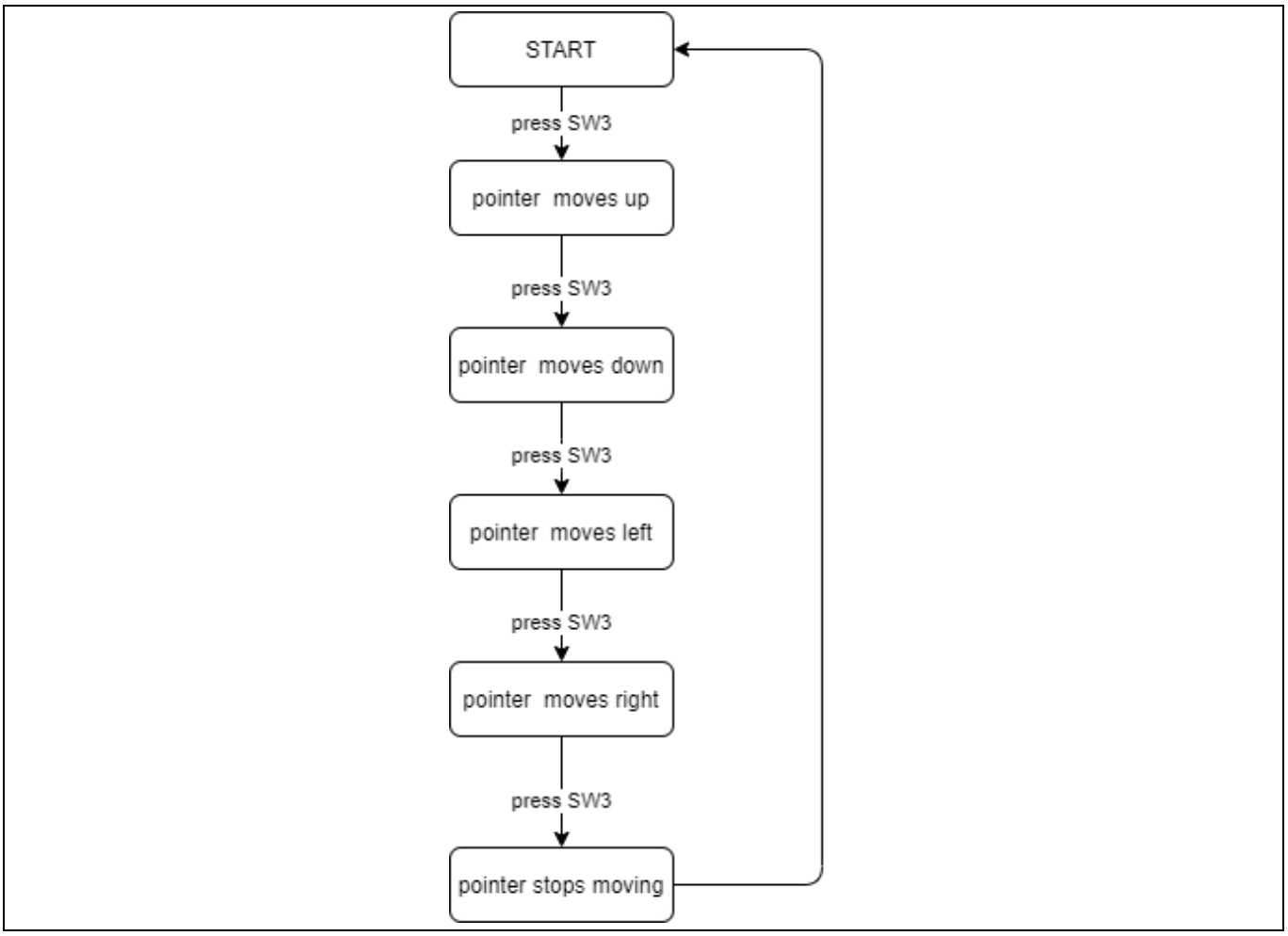


Figure 1-4 The Flow of Button Pressing and Response of Mouse Pointer

# 2 Code Description

In USBD20_IRQHandler, the mouse will be noticed that the moving information should be sent back to the host when receiving an endpoint A interrupt.

```
    void USBD20_IRQHandler(void)
    {
        __IO uint32_t IrqStL, IrqSt;

        IrqStL = HSUSBD->GINTSTS & HSUSBD->GINTEN;

        if (!IrqStL) {
            return;
        }
    ……
       if (IrqStL & HSUSBD_GINTSTS_EPAIF_Msk) {
            IrqSt = HSUSBD->EP[EPA].EPINTSTS & HSUSBD->EP[EPA].EPINTEN;

            HSUSBD_ENABLE_EP_INT(EPA, 0);
            EPA_Handler();
            HSUSBD_CLR_EP_INT_FLAG(EPA, IrqSt);
        }
    ……
    }
```

The moving information will be sent back to the host.

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

    if (g_u8EPAReady)
    {
        mouse_mode ^= 1;
        if (mouse_mode)
        {
            if (move_len > 14)
            {
                /* Update new report data */
                epa_buf[0] = 0x00;
```

```
                switch (g_MoveMouse)
                {
                    case 0:
                        epa_buf[1] = 0x00;
                        epa_buf[2] = 0x00;
                        break;

                    case 1:
                        epa_buf[1] = 0x00;
                        epa_buf[2] = mouse_table[0];
                        break;

                    case 2:
                        epa_buf[1] = 0x00;
                        epa_buf[2] = mouse_table[2];
                        break;

                    case 3:
                        epa_buf[1] = mouse_table[0];
                        epa_buf[2] = 0x00;
                        break;

                    case 4:
                        epa_buf[1] = mouse_table[2];
                        epa_buf[2] = 0x00;
                        break;

                    default:
                        epa_buf[1] = 0x00;
                        epa_buf[2] = 0x00;
                }

                epa_buf[3] = 0x00;
                move_len = 0;
            }
        }
        else
        {
            epa_buf[0] = epa_buf[1] = epa_buf[2] = epa_buf[3] = 0;
        }
```

```
        move_len++;
        g_u8EPAReady = 0;

        for (i = 0; i < 4; i++)
        {
            HSUSBD->EP[EPA].EPDAT_BYTE = epa_buf[i];
        }

        HSUSBD->EP[EPA].EPRSPCTL = HSUSBD_EP_RSPCTL_SHORTTXEN;
        HSUSBD_ENABLE_EP_INT(EPA, HSUSBD_EPINTEN_INTKIEN_Msk);
    }
}
```

The value of bInterval field in Endpoint descriptor is 0x01.

```
/* Define the interrupt In EP number */
#define MOUSE_INT_IN_EP_NUM        0x01
#define MOUSE_INT_IN_EP_INTERVAL   0x01
```

# 3 Software and Hardware Requirements

## 3.1 Software Requirements

- BSP version
  - M480 Series BSP CMSIS V3.05.001
- IDE version
  - Keil uVersion 5.26

## 3.2 Hardware Requirements

- Circuit components
  - NuMaker-PFM-M487
  - Micro USB cable
- Pin Connect

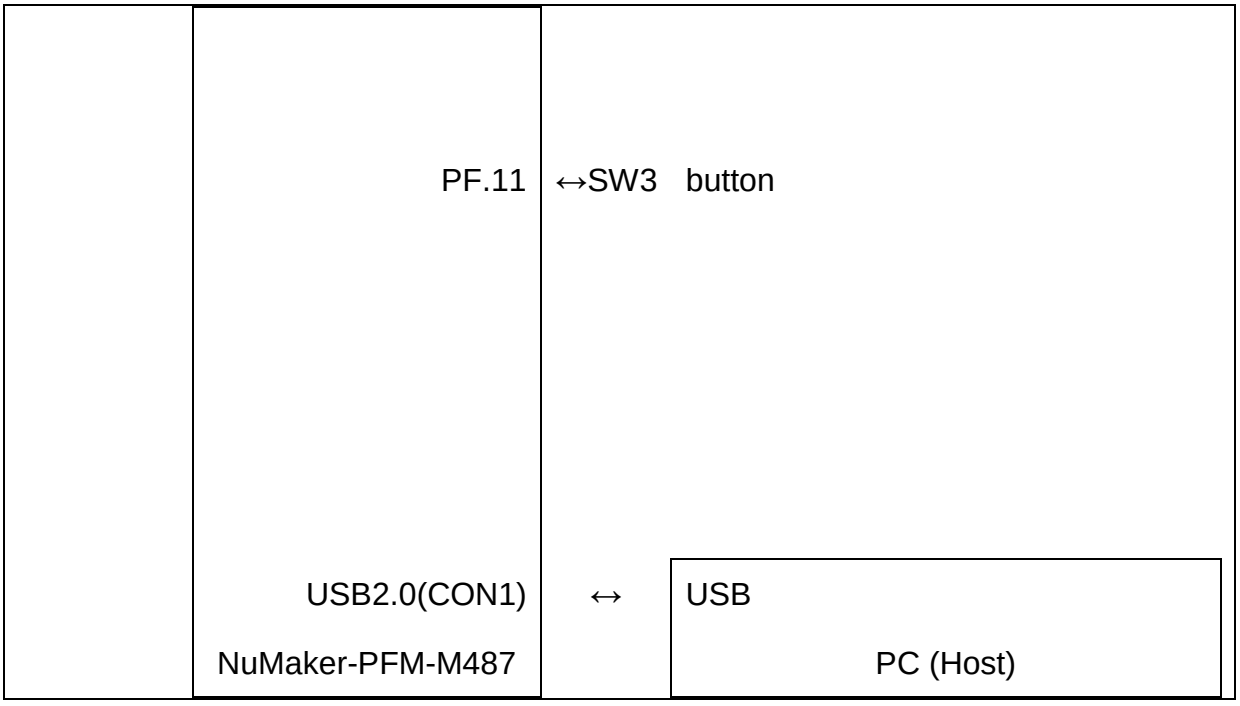


Figure 3-1 Pin Connect

# 4 Directory Information

The directory structure is shown below.

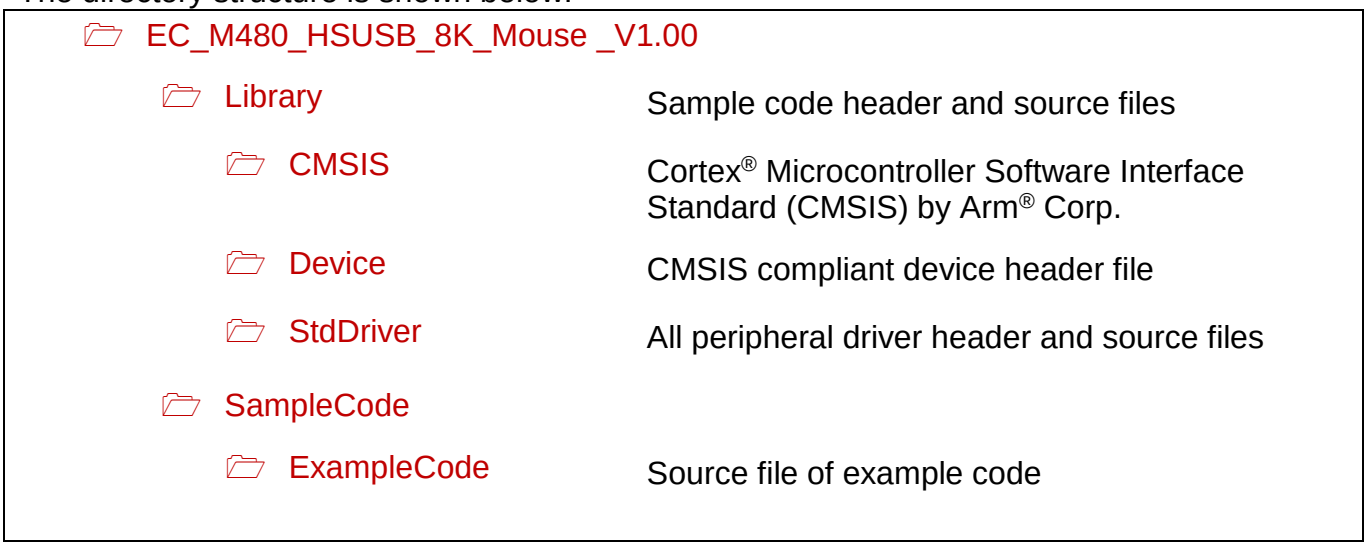


Figure 4-1 Directory Structure

# 5 Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click *HSUSB_8K_Mouse.uvprojx*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

# 6 Revision History

| Date | Revision | Description |
|---|---|---|
| 2021.06.02 | 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.*