USB Composite Device of Virtual COM Port and Multimedia Key

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This sample code demonstrates how to implement a USB HID composite device (Virtual COM Port and Multimedia Key).
BSP Version	M031 BSP CMSIS V3.03.000
Hardware	NuMaker-M032KI Ver1.0

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

A USB Composite Device is a peripheral device that supports more than one device class at the same time. This example code consists of Virtual Com port and HID media key in one USB device. The HID class is for reporting the key buttons inputs to control the mute, brightness, keyboard brightness and volume up/down of multimedia functions through the USB protocol.

1.1 Principle

This example code includes three interfaces that follow CDC (Communication Device Class) and HID (Human Interface Devices) specification. Based on USB spec, endpoint 0 for control transfer is used for USB enumeration. There are two endpoints to upload and download Bulk report and one endpoint to upload HID report. In this example, control pipe (endpoint 0) is made of two hardware endpoints 0 and 1 in device. The interfaces and endpoints configuration are shown in Table 1-1.

Function	USB Composite Device	M032 Hardware Setting
Control Transfer	Control Pipe endpoint 0: Control IN/OUT	endpoint 0: Control IN endpoint 1: Control OUT
Virtual COM	Interface 0 endpoint 4: Interrupt IN Interface 1 endpoint 2: Bulk IN endpoint 3: Bulk OUT	endpoint 2: Bulk IN endpoint 3: Bulk OUT endpoint 4: Interrupt IN
HID Multimedia Key	Interface 2 endpoint 5: Interrupt IN	endpoint 5: Interrupt IN

Table 1-1 Configuration of Composite Interfaces and Endpoints

1.2 Windows Driver Installation

The Windows driver installation steps of Virtual Com are depicted below.

1. Connect the M032 USB to the computer, open the [Device Manager] console, and you can see that the USB Device needs to install Windows driver in the Other devices section, as shown in Figure 1-1.

2. Right-click the mouse button and then select **Update Driver Software** and **Browse my computer for driver software** to install Windows driver**.**
3. Set the path to Windows Driver folder of the example code and then click **Next** button, as shown in Figure 1-4.

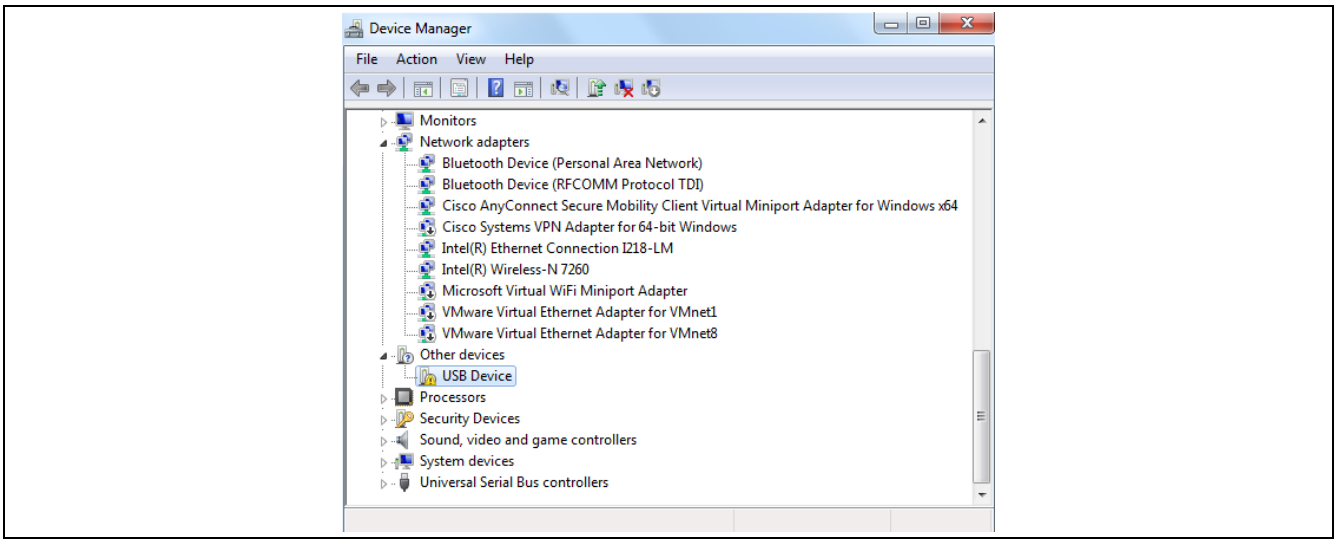

Figure 1-1 Device Manager

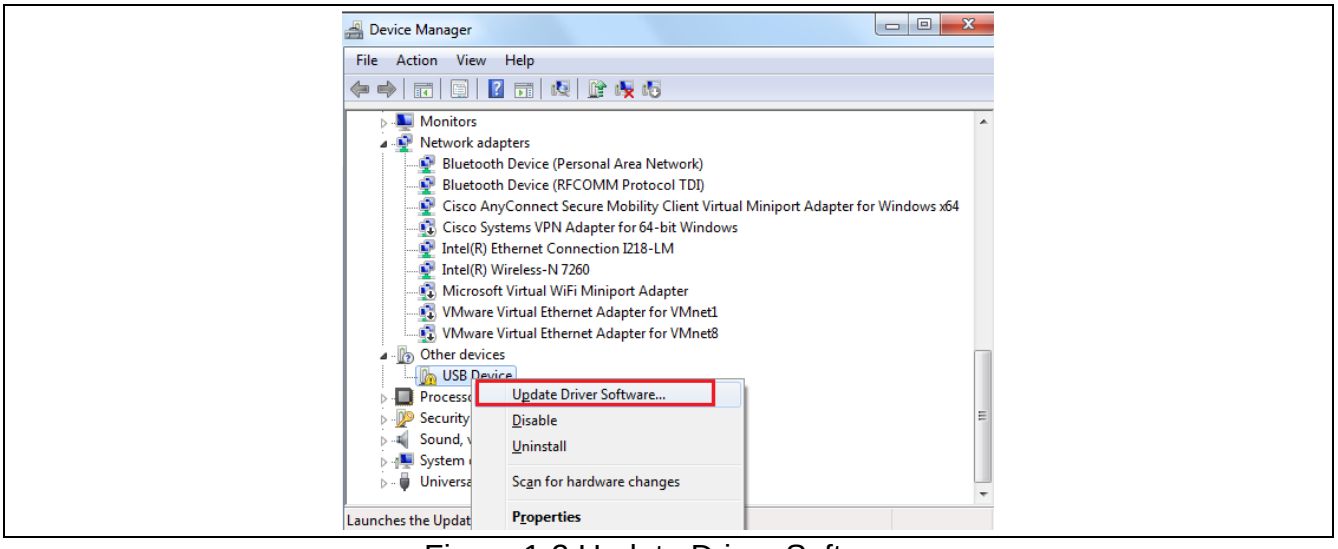

Figure 1-2 Update Driver Software

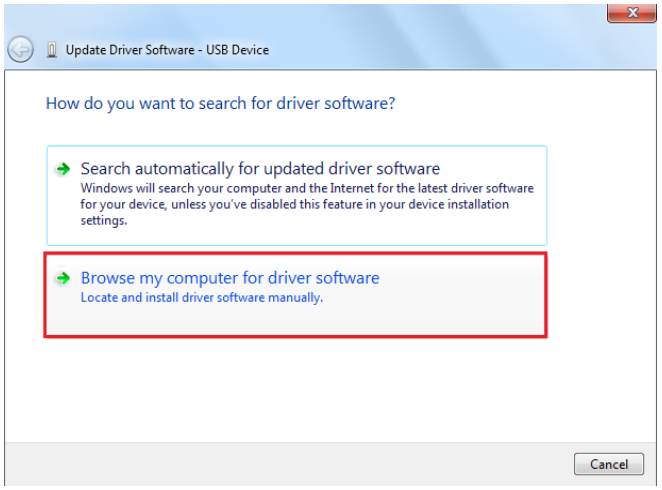

Figure 1-3 Install Driver Software Manually

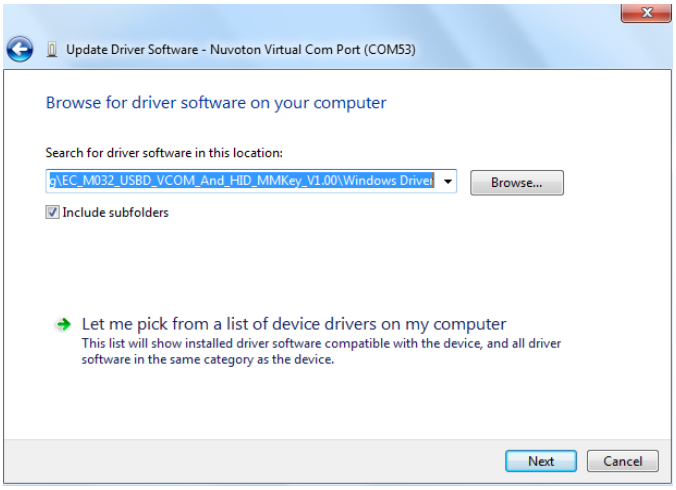

Figure 1-4 Driver Software Location

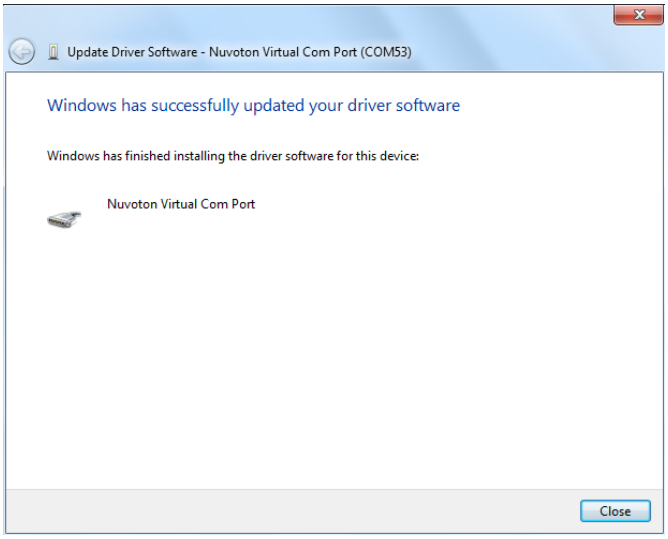

Figure 1-5 Install Driver Software Successfully

1.3 Demo Result

After executing this example code, user can plug-in the USB cable that connects the USB connector with PC Host. Open the [Device Manager] console, and you can see the Nuvoton Virtual Com Port in the Ports section, as shown in Figure 1-6. User can enter characters on UART0 terminal and the characters will be printed on Virtual Com port terminal. The volume setting gets up or down during the PD.2 or PD.3 is connected to ground.

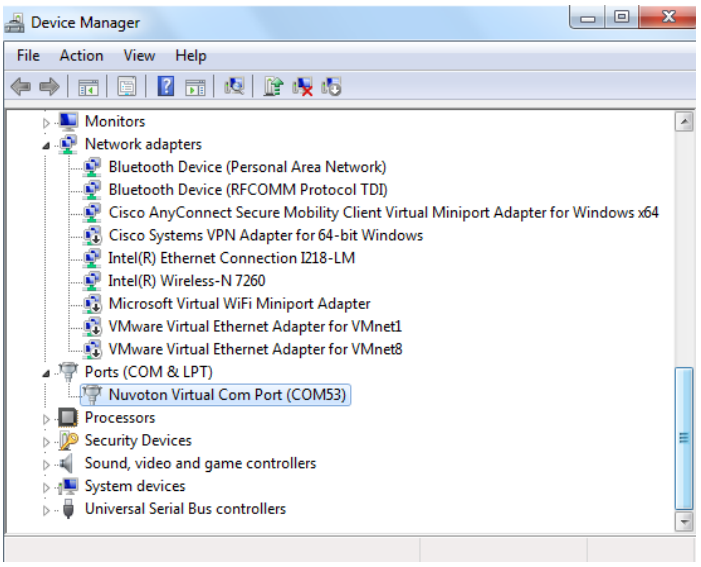

Figure 1-6 Virtual Com Port

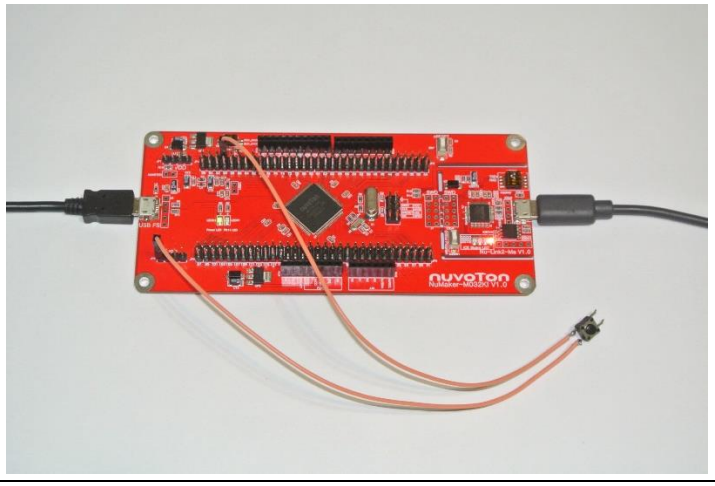

Figure 1-7 Volume Up Button

2 Code Description

This sample code is always polling the function of VCOM_TransferData() and HID_UpdateKbData() to check the status of Virtual Com transfer data and simulate a multimedia key input.

```
    while(1)
    {
#if CRYSTAL_LESS
        /* Start USB trim if it is not enabled. */
        if((SYS->HIRCTRIMCTL & SYS_HIRCTRIMCTL_FREQSEL_Msk) != 1)
        {
            /* Start USB trim only when SOF */
            if(USBD->INTSTS & USBD_INTSTS_SOFIF_Msk)
            {
                /* Clear SOF */
                USBD->INTSTS = USBD_INTSTS_SOFIF_Msk;

                /* Re-enable crystal-less */
                SYS->HIRCTRIMCTL = 0x01;
                SYS->HIRCTRIMCTL |= SYS_HIRCTRIMCTL_REFCKSEL_Msk;
            }
        }

        /* Disable USB Trim when error */
        if(SYS->HIRCTRIMSTS & (SYS_HIRCTRIMSTS_CLKERIF_Msk | SYS_HIRCTRIMSTS_TFAILIF_Msk))
        {
            /* Init TRIM */
            M32(TRIM_INIT) = u32TrimInit;

            /* Disable crystal-less */
            SYS->HIRCTRIMCTL = 0;

            /* Clear error flags */
            SYS->HIRCTRIMSTS = SYS_HIRCTRIMSTS_CLKERIF_Msk | SYS_HIRCTRIMSTS_TFAILIF_Msk;

            /* Clear SOF */
            USBD->INTSTS = USBD_INTSTS_SOFIF_Msk;
        }
#endif
        /* Enter power down when USB suspend */
        if(g_u8Suspend)
            PowerDown();

        VCOM_TransferData();

        HID_UpdateKbData();
    }
```

The EP2_Handler() and VCOM_TransferData() functions transfer the data received from UART0 to PC and then print the data on Virtual Com port terminal.

```
void EP2_Handler(void)
{
    gu32TxSize = 0;
}
```

```
void VCOM_TransferData(void)
{
    int32_t i, i32Len;

    /* Check whether USB is ready for next packet or not*/
    if(gu32TxSize == 0)
    {
        /* Check whether we have new COM Rx data to send to USB or not */
        if(comRbytes)
        {
            i32Len = comRbytes;
            if(i32Len > EP2_MAX_PKT_SIZE)
                i32Len = EP2_MAX_PKT_SIZE;

            for(i = 0; i < i32Len; i++)
            {
                gRxBuf[i] = comRbuf[comRhead++];
                if(comRhead >= RX_BUFSIZE)
                    comRhead = 0;
            }

            __set_PRIMASK(1);
            comRbytes -= i32Len;
            __set_PRIMASK(0);

            gu32TxSize = i32Len;
            USBD_MemCopy((uint8_t *)(USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP2)), (uint8_t
             *)gRxBuf, i32Len);
            USBD_SET_PAYLOAD_LEN(EP2, i32Len);
        }
        else
        {
            /* Prepare a zero packet if previous packet size is EP2_MAX_PKT_SIZE and
               no more data to send at this moment to note Host the transfer has been done
             */
            i32Len = USBD_GET_PAYLOAD_LEN(EP2);
            if(i32Len == EP2_MAX_PKT_SIZE)
                USBD_SET_PAYLOAD_LEN(EP2, 0);
        }
    }

    ...
}
```

The EP3_Handler() and VCOM_TransferData() functions are used to get the data received from Virtual Com port from PC and then print the data on UART0 terminal.

```
void EP3_Handler(void)
{
    /* Bulk OUT */
    if (g_u32OutToggle == (USBD->EPSTS0 & USBD_EPSTS0_EPSTS3_Msk))
    {
        USBD_SET_PAYLOAD_LEN(EP3, EP3_MAX_PKT_SIZE);
    }
    else
    {
        gu32RxSize = USBD_GET_PAYLOAD_LEN(EP3);
        gpu8RxBuf = (uint8_t *)(USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP3));
```

```
        g_u32OutToggle = USBD->EPSTS & USBD_EPSTS0_EPSTS3_Msk;
        /* Set a flag to indicate bulk out ready */
        gi8BulkOutReady = 1;
    }
}

void VCOM_TransferData(void)
{
    int32_t i, i32Len;

    ...

    /* Process the Bulk out data when bulk out data is ready. */
    if(gi8BulkOutReady && (gu32RxSize <= TX_BUFSIZE - comTbytes))
    {
        for(i = 0; i < gu32RxSize; i++)
        {
            comTbuf[comTtail++] = gpu8RxBuf[i];
            if(comTtail >= TX_BUFSIZE)
                comTtail = 0;
        }

        __set_PRIMASK(1);
        comTbytes += gu32RxSize;
        __set_PRIMASK(0);

        gu32RxSize = 0;
        gi8BulkOutReady = 0; /* Clear bulk out ready flag */

        /* Ready to get next BULK out */
        USBD_SET_PAYLOAD_LEN(EP3, EP3_MAX_PKT_SIZE);
    }

    /* Process the software Tx FIFO */
    if(comTbytes)
    {
        /* Check if Tx is working */
        if((UART0->INTEN & UART_INTEN_THREIEN_Msk) == 0)
        {
            /* Send one bytes out */
            UART0->DAT = comTbuf[comThead++];
            if(comThead >= TX_BUFSIZE)
                comThead = 0;

            comTbytes--;

            /* Enable Tx Empty Interrupt. (Trigger first one) */
            UART_ENABLE_INT(UART0, UART_INTEN_THREIEN_Msk);
        }
    }
}
```

The HID_KeyboardReportDescriptor[] table is the HID keyboard report descriptor for the multimedia keys. The EP5_Handler() and HID_UpdateKbData() functions are used to report the HID multimedia key status to Host when the push buttons are pressed or released.

```
/*!<USB HID Report Descriptor */
uint8_t HID_KeyboardReportDescriptor[] =
{
    0x05, 0x01,     /* Usage Page(Generic Desktop Controls) */
```

```
    0x09, 0x06,     /* Usage(Keyboard) */
    0xA1, 0x01,     /* Collection(Application) */
    0x85, 0x01,     /* Report ID (1) */
    0x05, 0x07,     /* Usage Page(Keyboard/Keypad) */
    0x19, 0xE0,     /* Usage Minimum(0xE0) */
    0x29, 0xE7,     /* Usage Maximum(0xE7) */
    0x15, 0x00,     /* Logical Minimum(0x0) */
    0x25, 0x01,     /* Logical Maximum(0x1) */
    0x75, 0x01,     /* Report Size(0x1) */
    0x95, 0x08,     /* Report Count(0x8) */
    0x81, 0x02,     /* Input (Data) => Modifier byte */
    0x95, 0x01,     /* Report Count(0x1) */
    0x75, 0x08,     /* Report Size(0x8) */
    0x81, 0x01,     /* Input (Constant) => Reserved byte */
    0x95, 0x05,     /* Report Count(0x5) */
    0x75, 0x01,     /* Report Size(0x1) */
    0x05, 0x08,     /* Usage Page(LEDs) */
    0x19, 0x01,     /* Usage Minimum(0x1) */
    0x29, 0x05,     /* Usage Maximum(0x5) */
    0x91, 0x02,     /* Output (Data) => LED report */
    0x95, 0x01,     /* Report Count(0x1) */
    0x75, 0x03,     /* Report Size(0x3) */
    0x91, 0x01,     /* Output (Constant) => LED report padding */
    0x95, 0x06,     /* Report Count(0x6) */
    0x75, 0x08,     /* Report Size(0x8) */
    0x15, 0x00,     /* Logical Minimum(0x0) */
    0x25, 0x65,     /* Logical Maximum(0x65) */
    0x05, 0x07,     /* Usage Page(Keyboard/Keypad) */
    0x19, 0x00,     /* Usage Minimum(0x0) */
    0x29, 0x65,     /* Usage Maximum(0x65) */
    0x81, 0x00,     /* Input (Data) */
    0xC0,           /* End Collection */

    0x05, 0x0C,     /* Usage Page (Consumer) */
    0x09, 0x01,     /* Usage (Consumer Control) */
    0xA1, 0x01,     /* Collection (Application) */
    0x85, 0x02,     /* Report ID (2) */
    0x09, 0x6f,     /* Usage (Brightness  Increment) */
    0x09, 0x70,     /* Usage (Brightness Decrement) */
    0x09, 0xE9,     /* Usage (Volume Increment) */
    0x09, 0xEA,     /* Usage (Volume Decrement) */
    0x09, 0x79,     /* Usage (keyboard Brightness  Increment) */
    0x09, 0x7a,     /* Usage (keyboard Brightness Decrement) */
    0x09, 0xe2,     /* Usage (mute) */
    0x15, 0x00,     /* Logical Minimum (0) */
    0x25, 0x01,     /* Logical Maximum (1) */
    0x95, 0x07,     /* Report Count (7) */
    0x75, 0x01,     /* Report Size (1) */
    0x81, 0x02,     /* Input (Data,Var,Abs,No Wrap,Linear,Preferred State,No Null
                    Position) */
    0x75, 0x01,     /* Report Size (1) */
    0x95, 0x01,     /* Report Count (1) */
    0x81, 0x03,     /* Input: (Constant, Variable, Absolute) */
    0xC0,           /* End Collection */

    0x05, 0x0C,     /* Usage Page (Consumer Devices) */
    0x09, 0x01,     /* Usage (Consumer Control) */
    0xA1, 0x01,     /* Collection: (Application) */
    0x85, 0x03,     /* Report ID (3) */
    0x15, 0x00,     /* Logical Minimum */
    0x25, 0x01,     /* Logical Maximum */
    0x05, 0x01,     /* Usage Page (Generic Desktop Controls) */
    0x09, 0x82,     /* Usage (System Sleep) */
    0x75, 0x01,     /* Report Size (1) */
    0x95, 0x01,     /* Report Count (1) */
    0x81, 0x02,     /* Input: (Data, Variable, Absolute) */
```

```
    0x75, 0x01,     /* Report Size (1) */
    0x95, 0x07,     /* Report Count (7) */
    0x81, 0x03,     /* Input: (Constant, Variable, Absolute) */
    0xC0,           /* End Collection */
};

void EP5_Handler(void)  /* Interrupt IN handler */
{
    g_u8EP5Ready = 1;
}

void HID_UpdateKbData(void)
{
    int32_t i;
    uint8_t *buf;
    uint32_t u32Key;
    static uint32_t u32Prekey;

    if (g_u8EP5Ready)
    {
        buf = (uint8_t *)(USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP5));

        u32Key = (!PD0) | (!PD1 << 1) | (!PD2 << 2) | (!PD3 << 3) | (!PD4 << 4) | (!PD5 <<
5) | (!PD6 << 6);

        g_u8EP5Ready = 0;

        buf[0] = 0x02;    /* Report ID - MediaKey */

        for (i = 1; i < 9; i++)
        {
            buf[i] = 0;
        }

        if (u32Key != u32Prekey)
        {

            if(!PD6)
                buf[1] = HID_CTRL_MUTE;    /* Mute */
            else if(!PD5)
                buf[1] = HID_CTRL_KEYBOARD_BRIGHTNESS_DEC;    /* Keyboard Brightess - */
            else if(!PD4)
                buf[1] = HID_CTRL_KEYBOARD_BRIGHTNESS_INC;    /* Keyboard Brightess + */
            else if(!PD3)
                buf[1] = HID_CTRL_VOLUME_DEC;    /* Volume - */
            else if(!PD2)
                buf[1] = HID_CTRL_VOLUME_INC;    /* Volume + */
            else if(!PD1)
                buf[1] = HID_CTRL_BRIGHTNESS_DEC;    /* Brightess - */
            else if(!PD0)
                buf[1] = HID_CTRL_BRIGHTNESS_INC;    /* Brightess + */

            u32Prekey = u32Key;
        }
        USBD_SET_PAYLOAD_LEN(EP5, 9);
    }
}
```

3 Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - M031 Series BSP CMSIS v3.03.000
- IDE version
  - Keil uVersion 5.26
- Windows Driver
  - NuvotonCDC.inf

3.2 Hardware Requirements

- Circuit components
  - NuMaker-M032KI Ver1.0
  - Micro USB cable
  - Button

- Pin Connect

PD.0	←	Button (Brightness up key)
PD.1	←	Button (Brightness down key)
PD.2	←	Button (Volume up key)
PD.3	←	Button (Volume down key)
PD.4	←	Button (Keyboard Brightness up key)
PD.5	←	Button (Keyboard Brightness down key)
PD.6	←	Button (Mute key)
UART0_RX (PB.12)	←	UART_TX
UART0_TX (PB.13)	→	UART_RX
USB	↔	PC(USB Host)
NuMaker-M032KI		

Figure 3-1 Pin Connect

4 Directory Information

The directory structure is shown below.

EC_M032_USBD_VCOM_And_HID_MMKey_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
SampleCode	
ExampleCode	Source file of example code
Windows Driver	PC Software Driver

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 *USBD_VCOM_And_HID_MMKey.uvproj*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6 Revision History

Date	Revision	Description
2021.11.05	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.