USB Composite Device with Caps Lock LED

Example Code Introduction for 32-bit NuMicro®Family

Information

Application	Demonstrate how to implement a USB HID composite device (Mouse, Keyboard with Caps Lock LED and Media key).
BSP Version	NUC122 Series BSP CMSIS v3.00.003
Hardware	NuTiny-EVB-122-LQFP64 V2.0

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

www.nuvoton.com

1 Function Description

1.1 Introduction

An USB Composite Device is a peripheral device that supports more than one device class at the same time. This example code consists of HID mouse, HID keyboard and HID media key in one USB device. The PD.1 ~ PD.4 are used to select content of HID report for report test and PD5 is used to be the Cap Lock LED.

1.2 Principle

This example code includes 2 interfaces that follow HID (Human Interface Devices) specification. Based on USB spec, endpoint 0 for control transfer is used for USB enumeration. There are 2 endpoints to upload HID report. In this example, control pipe (endpoint 0) is made of two hardware endpoints 0 and 1 in device. Every interface works as a specific function and contains a HID interrupt IN endpoint for IN data transfer. The interfaces and endpoints configuration are shown in Table 1.

Function	USB Composite Device	NUC122 Hardware setting
Control Transfer	Control Pipe endpoint 0: Control IN/OUT	endpoint 0: Control IN endpoint 1: Control OUT
HID Mouse	Interface 0 endpoint 1: Interrupt IN	endpoint 2: Interrupt IN
HID Keyboard	Interface 1 endpoint 2: Interrupt IN	endpoint 3: Interrupt IN

Table 1 Configuration of HID composite interfaces and endpoints

HID report descriptor is used to define HID report format and can be customized based on application demand. This example includes 2 HID report descriptors to support HID report formats for mouse and keyboard/media key that are listed in Table 2 and Table 3.

Byte	Bits	Description
0	0~2	Button 1~3
0	3~7	Padding
1	0~7	X-axis
2	0~7	Y-axis

Table 2 HID Mouse report format

Byte	Bits	Description	
0	0~7	Report ID :0x01	Report ID :0x02
1	0	Modifier Keys	Mute
1	1	Modifier Keys	Volume+
1	2	Modifier Keys	Volume-
1	3	Modifier Keys	Brightness+
1	4	Modifier Keys	Brightness-
1	5~7	Modifier Keys	Reserved
2	0~7	Reserved	Reserved
3	0~7	Key code 1	Reserved
4	0~7	Key code 2	Reserved
5	0~7	Key code 3	Reserved
6	0~7	Key code 4	Reserved
7	0~7	Key code 5	Reserved
8	0~7	Key code 6	Reserved

Table 3 HID Keyboard / Media key report format

The Set_Report request is one of the class-specific requests; it allows the host to send a report to the device, possibly setting the state of input, output or feature controls. These transactions are done through the control pipe; therefore, they must follow the control pipe request format defined in the USB specification. The Set_Report Request format is listed in Table 4.

Field	Length (bits)	Description
bmRequestType	8	00100001
bRequest	8	SET_REPORT (0x09)
wValue	16	Report Type and Report ID
wIndex	16	Interface
wLength	16	Report Length

Table 4 Set_Report Request Format

Among of them, the *wValue* field specifies Report Type in the high byte, and the Report Type is specified as Table 5. Therefore, user can read the output report transferred from control OUT endpoint in this example code if the Report Type is 0x2.

Value	Report Type
0x1	Input
0x2	Output
0x3	Feature
0x4-0xFF	Reserved

Table 5 Report Type

With this HID report descriptor, PC sends back one byte of output report for the LED on-off status. The bitmap of the output report is as follows,

Byte	Bits	Description
0	0~7	Report ID :0x01
1	0	Num Lock
1	1	Caps Lock
1	2	Scroll Lock
1	3	Compose
1	4	Kana
1	5~7	Reserved

Table 6 Output Report Format

1.3 Demo Result

After plug-in USB device, the USB device tree information could be got by USBLyzer in Figure 1. In this demo, the USB composite device includes 2 interfaces to support HID mouse, keyboard and media key function.

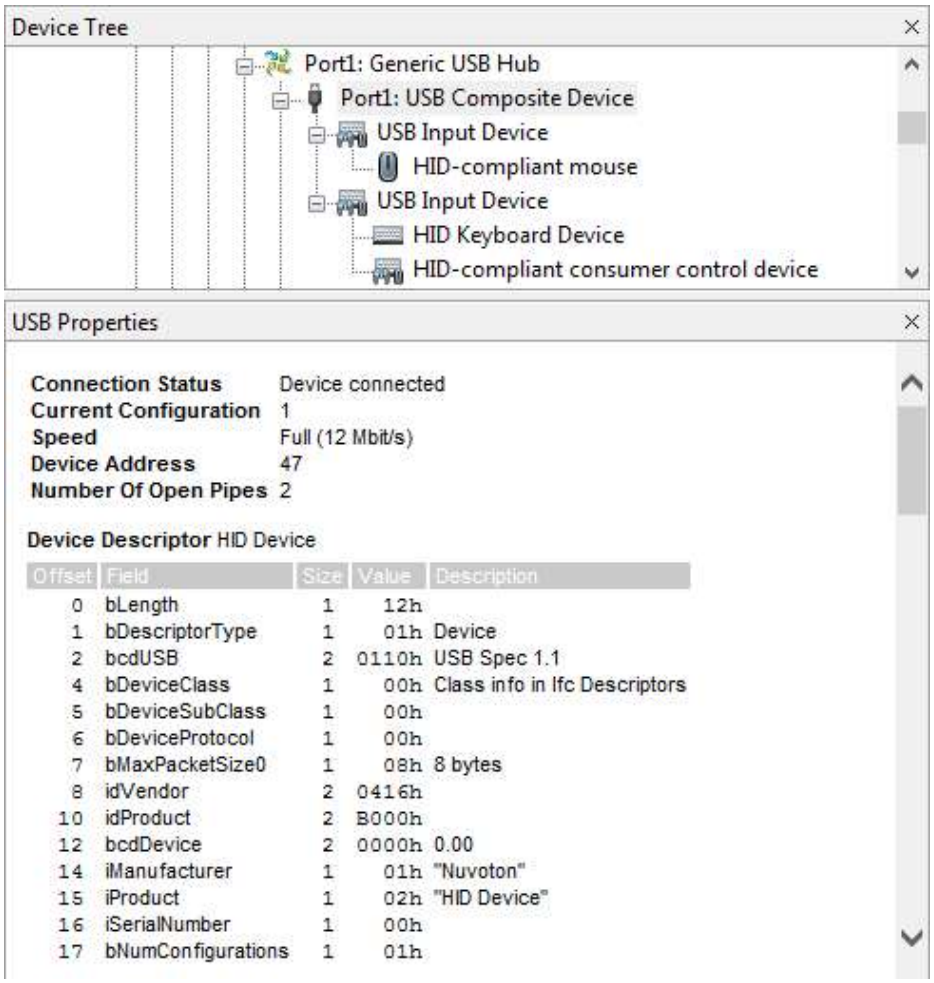


Figure 1 Device Tree

1.3.1 HID Mouse Report

In Figure 2 and Table 2, Data field of X-axis (Byte 1) is changed to 0x1 during the PD.1 is connected to ground (Button Pressed). At this time, the mouse pointer is moving right.

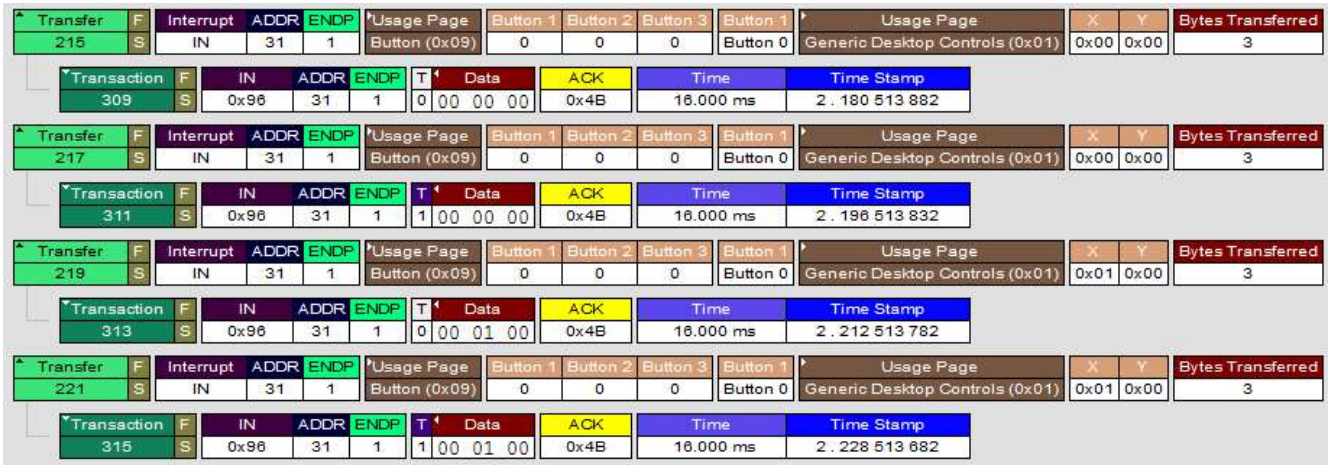


Figure 2 Mouse Report Packets

1.3.2 HID Keyboard Report

In Figure 3 and Table 3, Data field of key code 1 (Byte 3) is changed to 0x04 during the PD.2 is connected to ground (button pressed). At this time, the character of keyboard input is ‘A’ and showed at input area of notepad displayed in Figure 4. The key code, 0x04, defined in “HID USAGE TABLE V1.12” is character ‘A’.

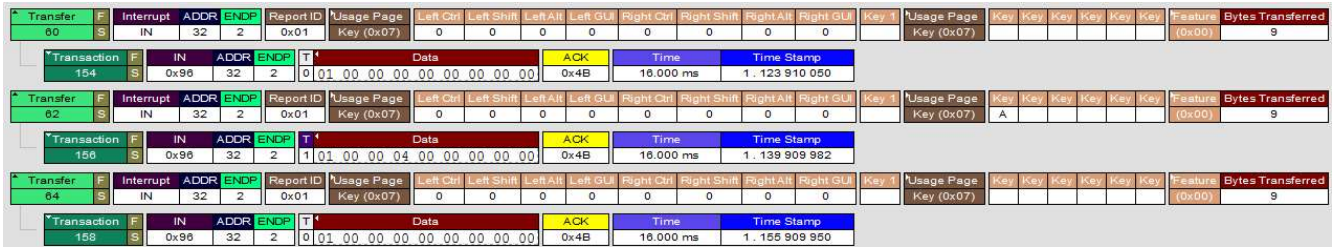


Figure 3 Keyboard packets

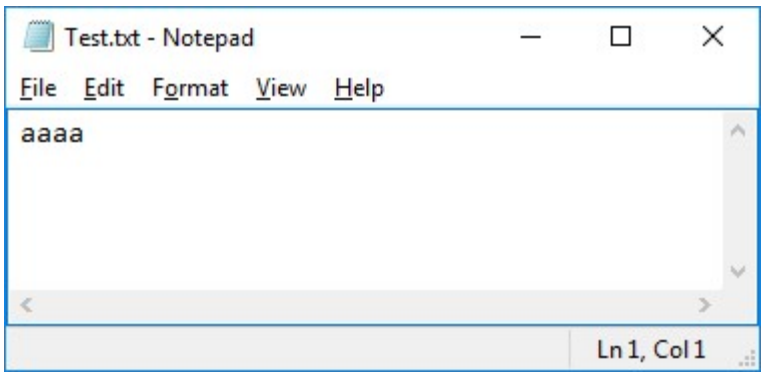


Figure 4 Keyboard input to notepad

1.3.3 HID Media key Report

In Figure 5, data fields (Bit 1 or Bit 2 of Byte1) is changed during the PD.3 or PD.4 is connected to ground. At this time, the volume setting is getting up or down.

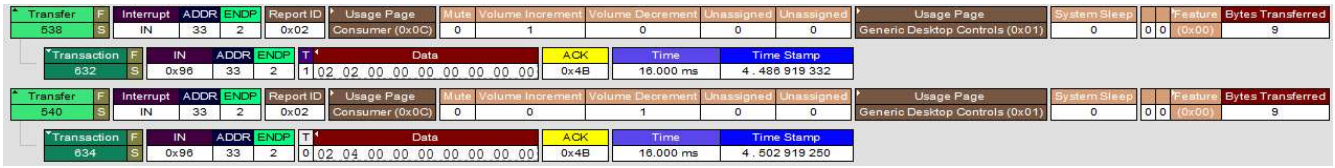


Figure 5 Media key packets

1.3.4 LED On-Off Status Output Report

When Caps Lock status is changed (the bit 1 of byte 1), you can check if the LED light (GPD5) changes or not. Moreover, you can see the data of set report request you get as shown in Figure 6.

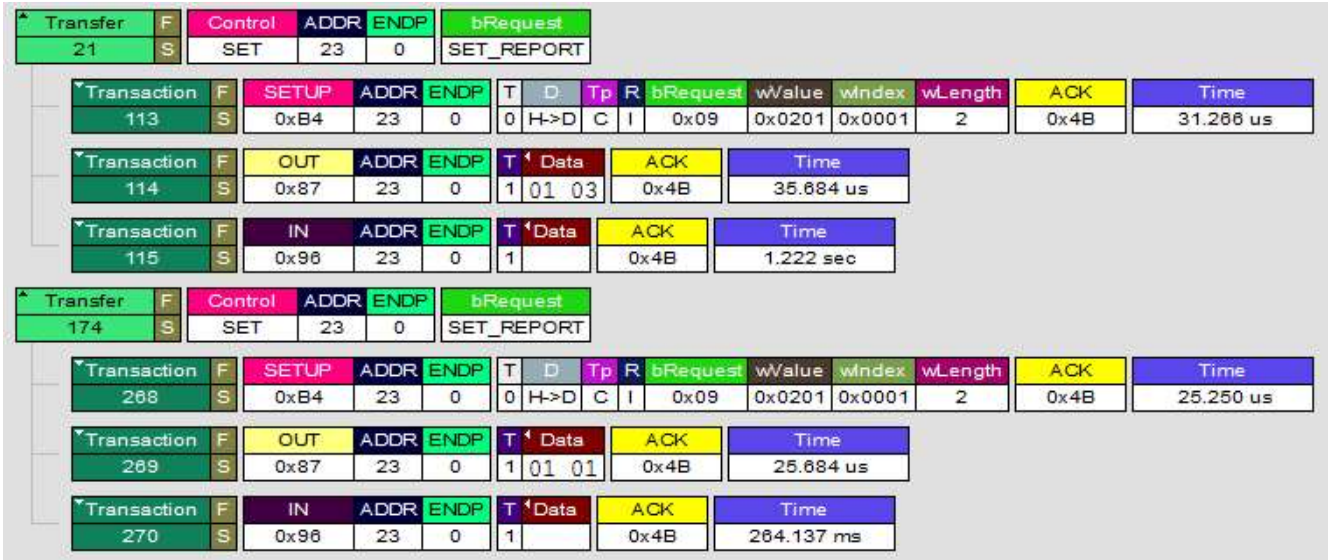
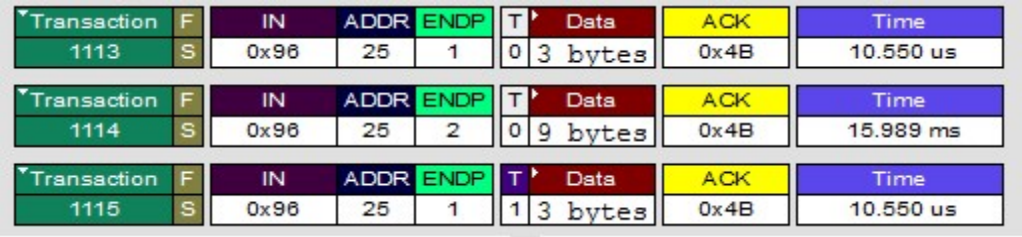


Figure 6 Set Report packets

2 Code Description

This example code supports HID mouse, HID keyboard and HID media key function that follows the HID specification. Therefore, the implementation and transfer flow of the two interfaces are similar. Each function uses an interrupt IN endpoint. In Figure 7, the endpoint will respond to USB host with the data whose format is defined in HID report descriptor when USB host issues IN token to request data.



Transaction	F	IN	ADDR	ENDP	T	Data	ACK	Time
1113	S	0x96	25	1	0	3 bytes	0x4B	10.550 us
Transaction	F	IN	ADDR	ENDP	T	Data	ACK	Time
1114	S	0x96	25	2	0	9 bytes	0x4B	15.989 ms
Transaction	F	IN	ADDR	ENDP	T	Data	ACK	Time
1115	S	0x96	25	1	1	3 bytes	0x4B	10.550 us

Figure 7 Interrupt IN Transaction for two endpoints

2.1 Mouse, Keyboard and Media key

2.1.1 main.c

In the main program, it is always polling the function of HID_UpdateKbData() and HID_UpdateMsData() to simulate a key input, and checking what data we got when Set_Report request occurred.

```
    while (1) {
        HID_UpdateKbData();
        HID_UpdateMsData();

        /* Enter power down when USB suspend */
        if(g_u8Suspend)
            PowerDown();

        if(memcmp(SetReport, PreSetReport, 2))
        {
            if(SetReport[1] &  0x2)
                PD5 = 1;    /* Turn On CapsLock LED */
            else
                PD5 = 0;    /* Turn Off CapsLock LED */
            memcpy(PreSetReport, SetReport, 2);
        }
    }
```

2.1.2 hid.c

Only take mouse as example. USB Host issues IN token to request the data from the endpoint 1. While IN token received, g_u8EP2Ready will be set to 1 in endpoint 2 interrupt handler. HID_UpdateMsData() will be executed to setup the payload and then send data if g_u8EP2Ready is 1.

Implementation of HID_UpdateKbData() is almost the same with HID_UpdateMsData(). The few differences are the settings of buffer and endpoint number. For keyboard/media key, interrupt IN buffer is 9 bytes (include 1 byte for Report ID) and NUC122 endpoint number is 3.

```
    void HID_UpdateMouseData(void)
    {
        uint8_t *buf;
        if (g_u8EP2Ready)
        {
            buf = (uint8_t *)(USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP2));
            g_u8EP2Ready = 0; /* Clear flag */
            for(i = 0; i < 3; i++)
                buf[0] = 0x00;
            if ((PD->PIN & BIT1) == 0)
                buf[1] = 0x01;    /* Left */
            /* Set transfer length and trigger IN transfer */
            USBD_SET_PAYLOAD_LEN(EP2, 3);
        }
    }
```

In the function of HID_ClassRequest(), PC will issue a SET_REPORT request within HID class-specific requests to device when user keys in Caps Lock. In the meanwhile, device can receive the new state data of key input stored in Set_Report.

```
void HID_ClassRequest(void)
{
    uint8_t buf[8];

    USBD_GetSetupPacket(buf);

    if (buf[0] & 0x80)   /* request data transfer direction */
    {
// Device to host
        switch (buf[1])
```

```
        {
...
        }
    }
    else
    {
// Host to device
        switch (buf[1])
        {
            case SET_REPORT:
            {
                if (buf[3] == 2)
                {
/* Request Type = Output */
                    USBD_SET_DATA1(EP1);
                    USBD_SET_PAYLOAD_LEN(EP1, buf[6]);
                    USBD_PrepareCtrlOut((uint8_t *)&SetReport, buf[6]);
/* Status stage */
                    USBD_PrepareCtrlIn(0, 0);
                }
                break;
            }

...

            case SET_PROTOCOL:
            {
                USBD_SET_DATA1(EP0);
                USBD_SET_PAYLOAD_LEN(EP0, 0);
                break;
            }
            default:
            {
                /* Setup error, stall the device */
                USBD_SetStall(EP0);
                USBD_SetStall(EP1);
                break;
            }
        }
    }
}
```

2.1.3 descriptor.c

HID_MouseReportDescriptor array includes the HID Report Descriptor for mouse function. It defines the data format in Table 2 and contains 3 buttons, X-axis, and Y-axis.

```
const uint8_t HID_MouseReportDescriptor[] =
{
    0x05, 0x01,     /* Usage Page(Generic Desktop Controls) */
    0x09, 0x02,     /* Usage(Mouse) */
    0xA1, 0x01,     /* Collection(Application) */
    0x09, 0x01,     /* Usage(Pointer) */
    0xA1, 0x00,     /* Collection(Physical) */
    0x05, 0x09,     /* Usage Page(Button) */
    0x19, 0x01,     /* Usage Minimum(0x1) */
    0x29, 0x03,     /* Usage Maximum(0x3) */
    0x15, 0x00,     /* Logical Minimum(0x0) */
    0x25, 0x01,     /* Logical Maximum(0x1) */
    0x95, 0x03,     /* Report Count(0x3) */
    0x75, 0x01,     /* Report Size(0x1) */
    0x81, 0x02,     /* Input(3 button bit) */
    0x95, 0x01,     /* Report Count(0x1) */
    0x75, 0x05,     /* Report Size(0x5) */
    0x81, 0x01,     /* Input(5 bit padding) */
    0x05, 0x01,     /* Usage Page(Generic Desktop Controls) */
    0x09, 0x30,     /* Usage(X) */
    0x09, 0x31,     /* Usage(Y) */
    0x15, 0x81,     /* Logical Minimum(0x81)(-127) */
    0x25, 0x7F,     /* Logical Maximum(0x7F)(127) */
    0x75, 0x08,     /* Report Size(0x8) */
    0x95, 0x02,     /* Report Count(0x2) */
    0x81, 0x06,     /* Input(1 byte) */
    0xC0,           /* End Collection */
    0xC0,           /* End Collection */
};
```

2.2 Configuration Descriptor for 2 Interfaces Composite Device

USB host requests configuration descriptor for device enumeration. gu8ConfigDescriptor is the configuration descriptor which contains description of 2 interfaces. Field “bNumInterfaces” be set as 2 to inform that there are 2 interfaces in the composite device. 2 sets of interface descriptor, HID descriptor and endpoint descriptor are listed sequentially in following.

```
/*!<USB Configure Descriptor */
const uint8_t gu8ConfigDescriptor[] =
{
    LEN_CONFIG,     /* bLength */
    DESC_CONFIG,    /* bDescriptorType */
    LEN_CONFIG_AND_SUBORDINATE & 0x00FF,/* wTotalLength */
    ((LEN_CONFIG_AND_SUBORDINATE & 0xFF00) >> 8),
    0x02,           /* bNumInterfaces */
    0x01,           /* bConfigurationValue */
    0x00,           /* iConfiguration */
    0x80 | (USBD_SELF_POWERED << 6) | (USBD_REMOTE_WAKEUP << 5),/* bmAttributes */
    USBD_MAX_POWER,         /* MaxPower */

/* HID Interface Descriptor*/
    LEN_INTERFACE,  /* bLength */
    DESC_INTERFACE, /* bDescriptorType */
    0x00,           /* bInterfaceNumber */
    0x00,           /* bAlternateSetting */
    0x01,           /* bNumEndpoints */
    0x03,           /* bInterfaceClass */
    0x01,           /* bInterfaceSubClass */
    HID_MOUSE,      /* bInterfaceProtocol */
    0x00,           /* iInterface */

/* HID Descriptor */
    LEN_HID,        /* bLength */
    DESC_HID,       /* bDescriptorType */
    0x10, 0x01,     /* bcdHID */
    0x00,           /* bCountryCode */
    0x01,           /* bNumDescriptors*/
    DESC_HID_RPT,   /* bDescriptorType. */
 /* Total length of report descriptor. */
    HID_MOUSE_REPORT_DESCRIPTOR_SIZE& 0x00FF,
    (HID_MOUSE_REPORT_DESCRIPTOR_SIZE& 0xFF00) >> 8),
```

```
    /* EP Descriptor: interrupt in. */
        LEN_ENDPOINT,  /* bLength */
        DESC_ENDPOINT, /* bDescriptorType */
        (INT_IN_EP_NUM_MS | EP_INPUT),/* bEndpointAddress */
        EP_INT,        /* bmAttributes */
        EP2_MAX_PKT_SIZE & 0x00FF,/* wMaxPacketSize */
        ((EP2_MAX_PKT_SIZE & 0xFF00) >> 8),
        HID_DEFAULT_INT_IN_INTERVAL,    /* bInterval */

    /* HID Interface Descriptor */
        LEN_INTERFACE,  /* bLength */
        DESC_INTERFACE, /* bDescriptorType */
        0x01,           /* bInterfaceNumber */
        0x00,           /* bAlternateSetting */
        0x01,           /* bNumEndpoints */
        0x03,           /* bInterfaceClass */
        0x01,           /* bInterfaceSubClass */
        HID_KEYBOARD,   /* bInterfaceProtocol */
        0x00,           /* iInterface */

    /* HID Descriptor */
        LEN_HID,        /* bLength */
        DESC_HID,       /* bDescriptorType */
        0x10, 0x01,     /* bcdHID */
        0x00,           /* bCountryCode*/
        0x01,           /* bNumDescriptors*/
        DESC_HID_RPT,   /* bDescriptorType */
    /* Total length of report descriptor. */
        HID_KEYBOARD_REPORT_DESCRIPTOR_SIZE& 0x00FF,
        ((HID_KEYBOARD_REPORT_DESCRIPTOR_SIZE& 0xFF00) >> 8),

    /* EP Descriptor: interrupt in. */
        LEN_ENDPOINT,   /* bLength */
        DESC_ENDPOINT,  /* bDescriptorType */
        (INT_IN_EP_NUM_KB | EP_INPUT), /* bEndpointAddress */
        EP_INT,         /* bmAttributes */
    /* wMaxPacketSize */
        EP3_MAX_PKT_SIZE & 0x00FF,
        ((EP3_MAX_PKT_SIZE & 0xFF00) >> 8),
        HID_DEFAULT_INT_IN_INTERVAL,     /* bInterval */
    };
```

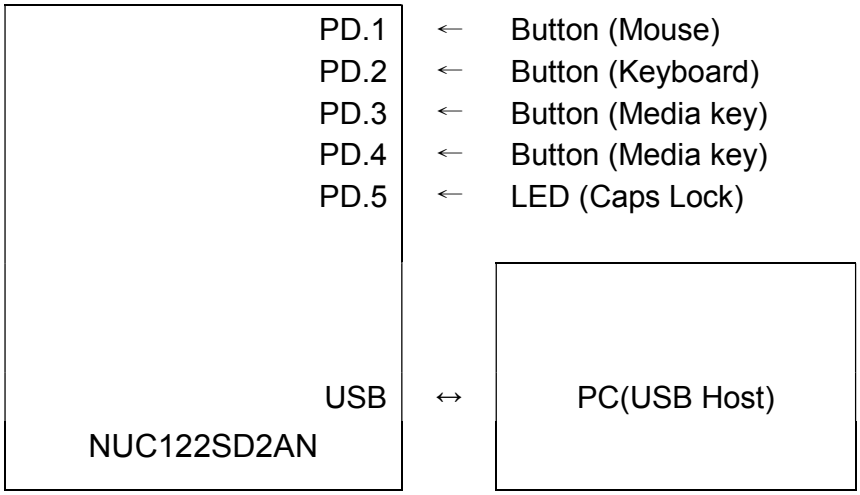
3 Software and Hardware Environment

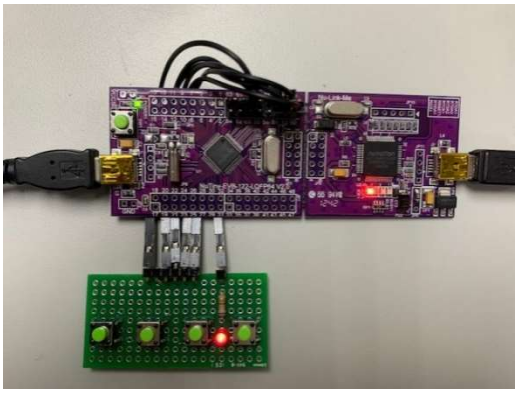
Software Environment

- BSP version
  - NUC122 Series BSP CMSIS v3.00.003
- IDE version
  - Keil uVersion5.26

Hardware Environment

- Circuit components
  - NuTiny-EVB-122-LQFP64 V2.0
  - USB mini USB cable
- Diagram





4 Directory Information

- EC_NUC122_USBD_HID_KB_MS_MMKey_LEDCtrl_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

5 How to Execute Example Code

1. This project supports Keil uVersion 5.26 or above.
2. Browsing into sample code folder by Directory Information (section 4) and double click USBD_HID_KB_MS_MMKey_LEDCtrl.uvproj.
3. Enter Keil compile mode
   a. Build
   b. Download
   c. Start/Stop debug session
4. Enter debug mode
   a. Run

6 Revision History

Date	Revision	Description
July.01, 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.