USBD HID Keyboard SetReport

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	Demonstrate how to use keyboard to get data of set report request
BSP Version	NUC230/NUC240 Series BSP CMSIS v3.01.002
Hardware	NuTiny-EVB-NUC240 V1.2

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 example code implements a USB keyboard device and uses a GPIO to simulate key input. In the meanwhile, user can get data of Set_Report request from PC.

1.2 Principle

USB keyboard is a HID (Human Interface Devices) class device, which communicates with the HID class driver using either the control pipe or an interrupt pipe as shown in Figure 1-1. In this example code, the control pipe is used for receiving and responding to requests for USB control and class data, receiving data from the host, and the interrupt pipe is used for receiving asynchronous data from the device. Moreover, there is no interrupt OUT endpoint is declared in this example code, so the output report will be transmitted to device through the control endpoint using Set_Report request.

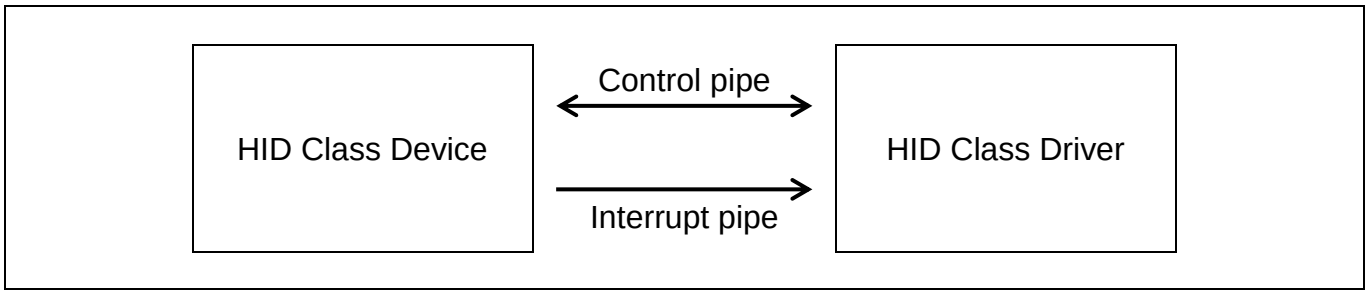

Figure 1-1 HID Class Communication Diagram

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 1.

Part	Offset/Size (Bytes)	Description
bmRequestType	0/1	00100001
bRequest	1/1	SET_REPORT (0x09)
wValue	2/2	Report Type and Report ID
wIndex	4/2	Interface
wLength	6/2	Report Length

Table 1 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 2. 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 2 Report Type

1.3 Demo Result

The following execution results will be output to the terminal tool with UART baud rate of 115200.
Once a GPIO pin, PB15, is wired to 0, the USB keyboard implemented in this example code will key in Caps Lock. At this time, you can check if the LED light of your own keyboard changes or not. Moreover, you can see the data of set report request you get as shown in Figure 1-2.

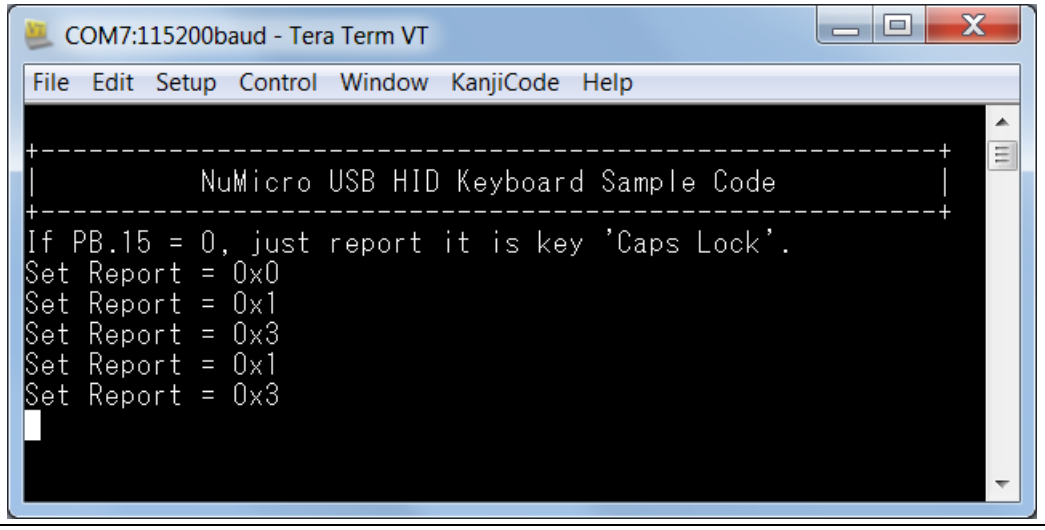

Figure 1-2 Execution Result

2 Code Description

In the main program, it is always polling the function of HID_UpdateKbData() to simulate a USB keyboard key input, and polling the flag of SetReport_flag to show the message on UART, what data we got when Set_Report request occurred.

```
    while (1)
    {
        if (SetReport_flag == 1)
        {
            SetReport_flag = 0;
            printf("Set Report = 0x%X\n\r", RcvBuffer[0]);
        }

        HID_UpdateKbData();
    }
```

When device receives IN token that is issued by PC to request data, g_u8EP2Ready will be set to 1 in endpoint 2 interrupt handler. If the g_u8EP2Ready is 1, the function of HID_UpdateKbData() will be executed to setup payload length and send data to PC. And if key is 1 when PB15 is wiring to low voltage, buf[2] will be set to 0x39 to simulate keying in “Caps Lock”.

```
void HID_UpdateKbData(void)
{
    int32_t i;
    uint8_t *buf;
    uint32_t key = 0xF;
    static uint32_t preKey;

    if (g_u8EP2Ready)
    {
        buf = (uint8_t *)(USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP2));

        /* If GPB15 = 0, just report it is key 'Caps Lock' */
        key = (PB->PIN & (1 << 15)) ? 0 : 1;

        if (key == 0)
        {
            for (i = 0; i < 8; i++)
            {
                buf[i] = 0;
```

```
            }

            if (key != preKey)
            {
                /* Trigger to note key release */
                USBD_SET_PAYLOAD_LEN(EP2, 8);
            }
        }
        else
        {
            preKey = key;
            buf[2] = 0x39; /* Key Caps Lock */
            USBD_SET_PAYLOAD_LEN(EP2, 8);
        }
    }
}
```

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 RcvBuffer and set a flag called SetReport_flag to 1.

```
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)
                {
                    USBD_PrepareCtrlOut((uint8_t *)&RcvBuffer, 1);

                    /* Request Type = Output */
                    USBD_SET_DATA1(EP1);
                    USBD_SET_PAYLOAD_LEN(EP1, buf[6]);

                    /* Status stage */
                    USBD_PrepareCtrlIn(0, 0);

                    SetReport_flag = 1;
                }
                break;
            }

            ...

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

3 Software and Hardware Environment

Software Environment

- BSP version
  - NUC230/NUC240 Series BSP CMSIS v3.01.002
- IDE version
  - Keil uVision V5.26

Hardware Environment

- Circuit components
  - NuTiny-EVB-NUC240
- Diagram
  - As shown in Figure 3-1, connect NUC240 USBD to PC via a mini USB cable; connect NUC240 pins: UART0_RX (PB.0) and UART0_TX (PB.1) to PC: UART_TX and UART_RX respectively for displaying the execution result on PC.

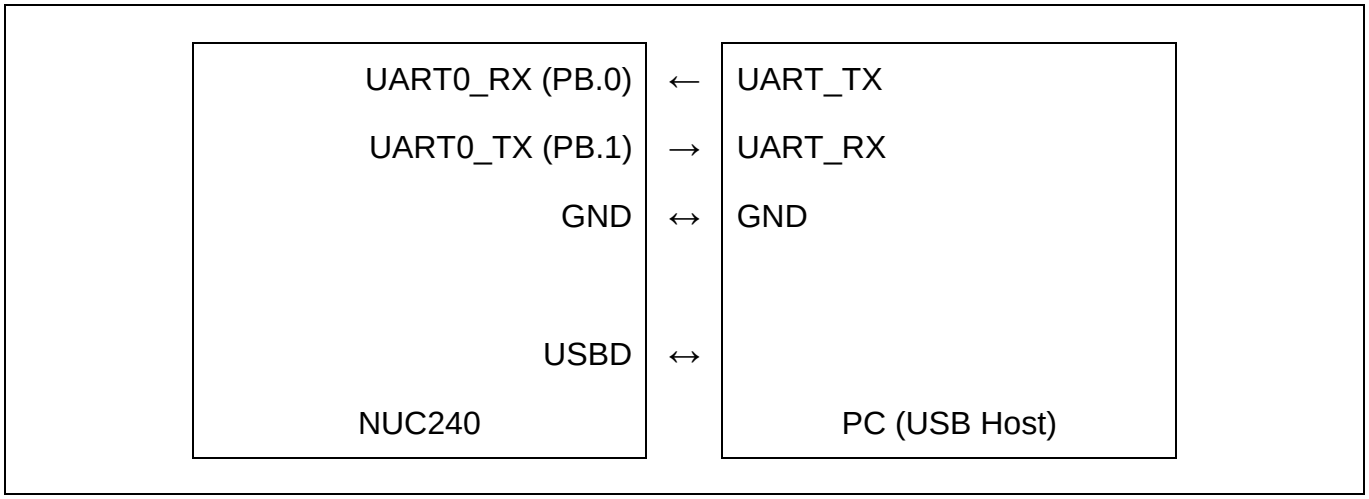

Figure 3-1 NUC240 and PC Connection

4 Directory Information

- EC_NUC240_USBD_HID_Keyboard_SetReport_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. Browsing into sample code folder by Directory Information (section 4) and double click NUC240_USBD_HID_Keyboard_SetReport.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
Aug. 26, 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.