USB Three-In-One HID Device

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	Demonstrate how to integrate HID joystick, HID mouse and HID keyboard in one USB device.
BSP Version	NUC230/240 Series BSP CMSIS V3.01.001
Hardware	NuTiny-EVB-NUC240 V1.2

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. In order to support gaming controller, this example code consists of HID joystick, HID mouse and HID keyboard in one USB device. The PB.5, PB.6 and PB.7 are used to select content of HID report for report test.

1.2 Principle

This example code includes 3 interfaces that follow HID (Human Interface Devices) specification. Based on USB spec, endpoint 0 for control transfer is used for USB enumeration. There are 3 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 single 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	NUC240 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
HID Joystick	Interface 2 endpoint 3: Interrupt IN	endpoint 4: 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 3 HID report descriptors to support HID report formats for joystick, mouse and keyboard that are listed in Table 2, Table 3 and Table 4.

Byte offset	Bit	Function
0	0~2	Button 1~3
0	3~7	Padding
1	0~7	X-axis
2	0~7	Y-axis
3	0~7	Wheel

Table 2 HID Mouse report format

Byte offset	Function
0	Modifier Keys
1	Reserved
2	Key code 1
3	Key code 2
4	Key code 3
5	Key code 4
6	Key code 5
7	Key code 6

Table 3 HID Keyboard report format

Byte offset	Bit	Function
0	0~7	Throttle
1	0~7	X-axis
2	0~7	Y-axis
3	0~3	Hat Switch
3	4~7	Button 1 ~ 4

Table 4 HID Mouse 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 3 interfaces to support HID mouse, keyboard and joystick function.

Figure 1 Device Tree

1.3.1 HID Mouse Report

In Figure 2, Data field of Y-axis is changed to 1 during the PB.5 is connected to ground(Button Pressed). At this time, the mouse pointer is moving down.

URB	2131	11:07:54.683	435.224...		Bulk or Interrupt Transfer	4 bytes buffer		in	01:00:
URB	2132-2129	11:07:54.700	435.240...	31.928 ...	Bulk or Interrupt Transfer	Input Report (Len 4)	00 00 00 00	in	01:00:
URB	2133	11:07:54.700	435.240...		Bulk or Interrupt Transfer	4 bytes buffer		in	01:00:
URB	2134-2131	11:07:54.715	435.256...	31.829 ...	Bulk or Interrupt Transfer	Input Report (Len 4)	00 00 00 00	in	01:00:
URB	2135	11:07:54.715	435.256...		Bulk or Interrupt Transfer	4 bytes buffer		in	01:00:
URB	2136-2133	11:07:54.731	435.272...	31.835 ...	Bulk or Interrupt Transfer	Input Report (Len 4)	00 00 01 00	in	01:00:
URB	2137	11:07:54.731	435.272...		Bulk or Interrupt Transfer	4 bytes buffer		in	01:00:
URB	2138-2135	11:07:54.747	435.288...	31.963 ...	Bulk or Interrupt Transfer	Input Report (Len 4)	00 00 01 00	in	01:00:
URB	2139	11:07:54.747	435.288...		Bulk or Interrupt Transfer	4 bytes buffer		in	01:00:
URB	2140-2137	11:07:54.763	435.304...	31.911 ...	Bulk or Interrupt Transfer	Input Report (Len 4)	00 00 01 00	in	01:00:
URB	2141	11:07:54.763	435.304...		Bulk or Interrupt Transfer	4 bytes buffer		in	01:00:
URB	2142-2139	11:07:54.779	435.320...	31.898 ...	Bulk or Interrupt Transfer	Input Report (Len 4)	00 00 01 00	in	01:00:
URB	2143	11:07:54.779	435.320...		Bulk or Interrupt Transfer	4 bytes buffer		in	01:00:

Figure 2 Mouse Report Packets

1.3.2 HID Keyboard Report

In Figure 3, Data field of key code 1 is changed to 0x1E during the PB.6 is connected to ground(button pressed). At this time, the character of keyboard input is '1' and showed at input area of notepad displayed in Figure 4. The key code, 0x1E, defined in "HID USAGE TABLE V1.12" is character '1'.

URB	2963	15:30:46.879	16207.3...		Bulk or Interrupt Transfer	8 bytes buffer		in
URB	2964-2961	15:30:46.895	16207.3...	31.829 ...	Bulk or Interrupt Transfer	Input Report (Len 8)	00 00 00 00 00 00 00 00	in
URB	2965	15:30:46.895	16207.3...		Bulk or Interrupt Transfer	8 bytes buffer		in
URB	2966-2963	15:30:46.911	16207.3...	31.920 ...	Bulk or Interrupt Transfer	Input Report (Len 8)	00 00 00 00 00 00 00 00	in
URB	2967	15:30:46.911	16207.3...		Bulk or Interrupt Transfer	8 bytes buffer		in
URB	2968-2965	15:30:46.927	16207.3...	31.967 ...	Bulk or Interrupt Transfer	Input Report (Len 8)	00 00 1E 00 00 00 00 00	in
URB	2969	15:30:46.927	16207.3...		Bulk or Interrupt Transfer	8 bytes buffer		in
URB	2970-2967	15:30:46.943	16207.4...	31.974 ...	Bulk or Interrupt Transfer	Input Report (Len 8)	00 00 1E 00 00 00 00 00	in
URB	2971	15:30:46.943	16207.4...		Bulk or Interrupt Transfer	8 bytes buffer		in
URB	2972-2969	15:30:46.959	16207.4...	31.979 ...	Bulk or Interrupt Transfer	Input Report (Len 8)	00 00 1E 00 00 00 00 00	in

Figure 3 Keyboard interface packets

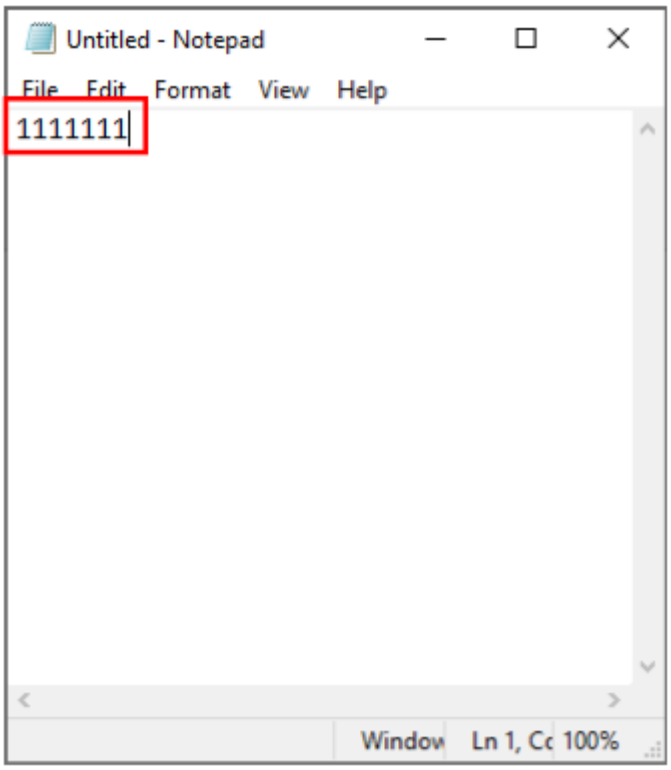


Figure 4 Keyboard input to notepad

1.3.3 HID joystick

In Figure 5, data fields is changed during the PB.7 is connected to ground. At this time, the difference of joystick data is shown in Figure 5. The behavior of joystick can be observed by the built-in tool, “Game Controllers”, in Windows as shown in Figure 6.

URB	1320-1317	11:00:49.165	9.70444...	32.024 ...	Bulk or Interrupt Transfer	Input Report (Len 4)	00 00 00 00
URB	1321	11:00:49.165	9.70448...		Bulk or Interrupt Transfer	4 bytes buffer	
URB	1322-1319	11:00:49.181	9.72049...	31.986 ...	Bulk or Interrupt Transfer	Input Report (Len 4)	00 00 00 00
URB	1323	11:00:49.181	9.72052...		Bulk or Interrupt Transfer	4 bytes buffer	
URB	1324-1321	11:00:49.197	9.73635...	31.865 ...	Bulk or Interrupt Transfer	Input Report (Len 4)	7F 7F 7F 63
URB	1325	11:00:49.197	9.73637...		Bulk or Interrupt Transfer	4 bytes buffer	
URB	1326-1323	11:00:49.213	9.75263...	32.113 ...	Bulk or Interrupt Transfer	Input Report (Len 4)	7F 7F 7F 63
URB	1327	11:00:49.213	9.75267...		Bulk or Interrupt Transfer	4 bytes buffer	
URB	1328-1325	11:00:49.230	9.76855...	32.171 ...	Bulk or Interrupt Transfer	Input Report (Len 4)	7F 7F 7F 63
URB	1329	11:00:49.230	9.76859...		Bulk or Interrupt Transfer	4 bytes buffer	

Figure 5 Joystick interface packets

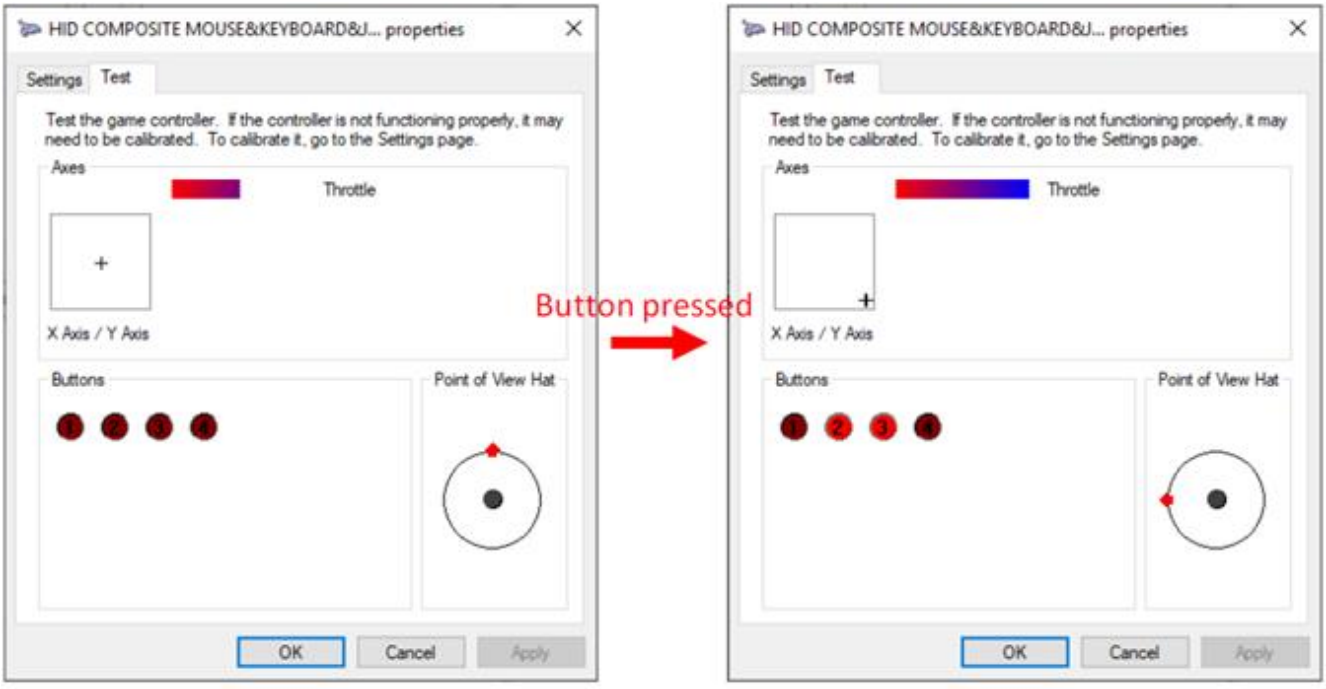


Figure 6 Joystick test function

To go to the properties window in Figure 6, please open “Game Controllers” window, find this device and then press the “property” button in Figure 7.

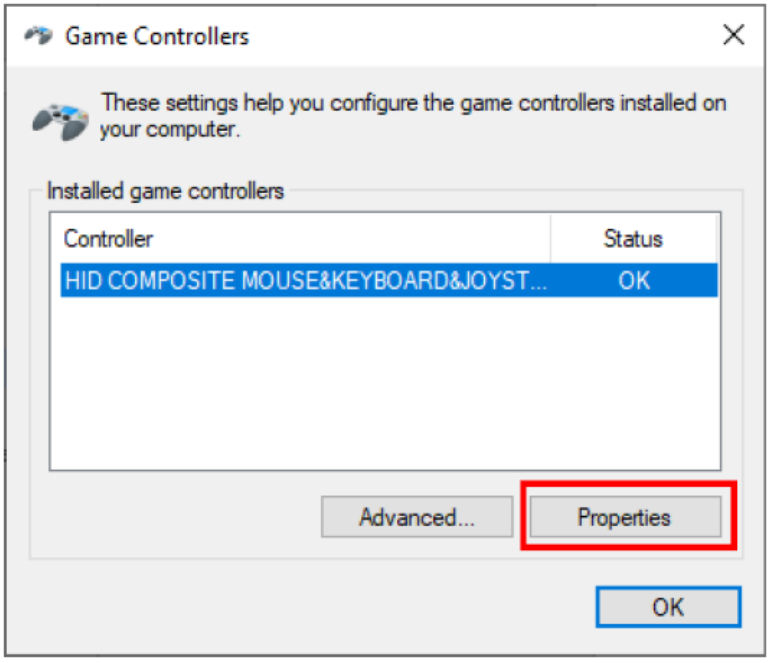


Figure 7 Game Controllers window

2 Code Description

This example code supports HID mouse, HID keyboard and HID joystick function that follows the HID specification. Therefore, the implementation and transfer flow of the three interfaces are similar. Each function has an interrupt IN endpoint. In Figure 8, 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	Time Stamp
98	S	0x96	19	1	0	4 bytes	0x4B	11.282 us	1 . 466 081 700
Transaction	F	IN	ADDR	ENDP	T	Data	ACK	Time	Time Stamp
99	S	0x96	19	2	1	8 bytes	0x4B	13.968 us	1 . 466 092 982
Transaction	F	IN	ADDR	ENDP	T	Data	ACK	Time	Time Stamp
100	S	0x96	19	3	1	4 bytes	0x4B	15.975 ms	1 . 466 106 950

Figure 8 Interrupt IN Transaction for three endpoints

2.1 Mouse, Keyboard and Joystick

The below 3 functions in main.c are used to report HID data for test. User has to follow hardware design and control flow to modify the HID report content.

```
    while (1) {
        HID_UpdateMouseData();
        HID_UpdateKbData();
        HID_UpdateJoyStickData();
    }
```

2.1.1 hid_mousekeyboardjoystick.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_UpdateMouseData() will be executed to setup the payload and then send data if g_u8EP2Ready is 1.

Implementation of HID_UpdateKeyboardData() and HID_UpdateJoystickData() are almost the same with HID_UpdateMouseData(). The few differences are the settings of buffer and endpoint number. For keyboard, interrupt IN buffer is 8 bytes and NUC240 endpoint number is 3. For joystick, the buffer is 4 bytes and the endpoint number is 4. The summary is listed in Table 1.

```
    void HID_UpdateMouseData(void)
    {
        uint8_t *buf;
        if (g_u8EP2Ready)
        {
            buf = (uint8_t *)(USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP2));
            /* Clear buf */
            buf[0] = buf[1] = buf[2] = buf[3] = 0;
            /* If PB.5 = 0, just mouse move */
```

```
            if (PB5 == 0) {
                buf[0] = 0x00; /* Button*/
                buf[1] = 0x00; /* X */
                buf[2] = 0x01; /* Y */
                buf[3] = 0x00; /* wheel */
            } /* Set transfer length and trigger IN transfer */
            /* Clear flag */
            g_u8EP2Ready = 0;
            /* Set transfer length and trigger IN transfer for EP2*/
            USBD_SET_PAYLOAD_LEN(EP2, 4);
        }
    }
```

2.1.2 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, Y-axis and wheel.

```
    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) */
        0x75, 0x01,         /* Report Size(0x1) */
        0x95, 0x03,         /* Report Count(0x3) */
        0x81, 0x02,         /* Input(3 button bit) */
        0x75, 0x05,         /* Report Size(0x5) */
        0x95, 0x01,         /* Report Count(0x1) */
        0x81, 0x01,         /* Input(5 bit padding) */
        0x05, 0x01,         /* Usage Page(Generic Desktop Controls) */
        0x09, 0x30,         /* Usage(X) */
        0x09, 0x31,         /* Usage(Y) */
        0x09, 0x38,         /* Usage(Wheel) */
        0x15, 0x81,         /* Logical Minimum(0x81)(-127) */
        0x25, 0x7F,         /* Logical Maximum(0x7F)(127) */
        0x75, 0x08,         /* Report Size(0x8) */
        0x95, 0x03,         /* Report Count(0x3) */
```

```
        0x81, 0x06,         /* Input(1 byte wheel) */
        0xC0,               /* End Collection */
        0xC0                /* End Collection */
    };
```

2.2 Configuration Descriptor for 3 Interfaces Composite Device

USB host requests configuration descriptor for device enumeration. gu8ConfigDescriptor is the configuration descriptor which contains description of 3 interfaces. Field “bNumInterfaces” be set as 3 to inform that there are 3 interfaces in the composite device. 3 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 */
        /* wTotalLength */
        LEN_CONFIG_AND_SUBORDINATE & 0x00FF,
        ((LEN_CONFIG_AND_SUBORDINATE & 0xFF00) >> 8),
        0x03,           /* bNumInterfaces */
        0x01,           /* bConfigurationValue */
        0x00,           /* iConfiguration */
        0x80 | (USBD_SELF_POWERED << 6) | (USBD_REMOTE_WAKEUP << 5),/* bmAttributes */
        USBD_MAX_POWER,         /* MaxPower */

        /* I/F descr: HID - Mouse */
        LEN_INTERFACE,  /* bLength */
        DESC_INTERFACE, /* bDescriptorType */
        0x00,           /* bInterfaceNumber */
        0x00,           /* bAlternateSetting */
        0x01,           /* bNumEndpoints */
        0x03,           /* bInterfaceClass */
        0x00,           /* bInterfaceSubClass */
        HID_MOUSE,      /* bInterfaceProtocol */
        0x00,           /* iInterface */

        /* HID Descriptor */
        LEN_HID,        /* Size of this descriptor in UINT8s. */
        DESC_HID,       /* HID descriptor type. */
        0x10, 0x01,     /* HID Class Spec. release number. */
        0x00,           /* H/W target country. */
        0x01,           /* Number of HID class descriptors to follow. */
```

```
        DESC_HID_RPT,   /* Descriptor type. */
        /* Total length of report descriptor. */
        sizeof(HID_MouseReportDescriptor) & 0x00FF,
        ((sizeof(HID_MouseReportDescriptor) & 0xFF00) >> 8),

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

        HID_DEFAULT_INT_IN_INTERVAL,     /* bInterval */

        /* I/F descr: HID - Keyboard */
        LEN_INTERFACE,  /* bLength */
        DESC_INTERFACE, /* bDescriptorType */
        0x01,           /* bInterfaceNumber */
        0x00,           /* bAlternateSetting */
        0x01,           /* bNumEndpoints */
        0x03,           /* bInterfaceClass */
        0x00,           /* bInterfaceSubClass */
        HID_KEYBOARD,   /* bInterfaceProtocol */
        0x00,           /* iInterface */

        /* HID Descriptor */
        LEN_HID,        /* Size of this descriptor in UINT8s. */
        DESC_HID,       /* HID descriptor type. */
        0x10, 0x01,     /* HID Class Spec. release number. */
        0x00,           /* H/W target country. */
        0x01,           /* Number of HID class descriptors to follow. */
        DESC_HID_RPT,   /* Descriptor type. */
        /* Total length of report descriptor. */
        sizeof(HID_KeyboardReportDescriptor) & 0x00FF,
        ((sizeof(HID_KeyboardReportDescriptor) & 0xFF00) >> 8),

        /* EP Descriptor: interrupt in. */
        LEN_ENDPOINT,   /* bLength */
        DESC_ENDPOINT,  /* bDescriptorType */
```

```
        (HID_KB_EP_NUM | EP_INPUT), /* bEndpointAddress */
        EP_INT,         /* bmAttributes */
        /* wMaxPacketSize */
        EP3_MAX_PKT_SIZE & 0x00FF,
        ((EP3_MAX_PKT_SIZE & 0xFF00) >> 8),
        HID_DEFAULT_INT_IN_INTERVAL,     /* bInterval */

        /* I/F descr: HID - Joystick */
        LEN_INTERFACE,  /* bLength */
        DESC_INTERFACE, /* bDescriptorType */
        0x02,           /* bInterfaceNumber */
        0x00,           /* bAlternateSetting */
        0x01,           /* bNumEndpoints */
        0x03,           /* bInterfaceClass */
        0x00,           /* bInterfaceSubClass */
        HID_NONE,       /* bInterfaceProtocol */
        0x00,           /* iInterface */

        /* HID Descriptor */
        LEN_HID,        /* Size of this descriptor in UINT8s. */
        DESC_HID,       /* HID descriptor type. */
        0x10, 0x01,     /* HID Class Spec. release number. */
        0x00,           /* H/W target country. */
        0x01,           /* Number of HID class descriptors to follow. */
        DESC_HID_RPT,   /* Descriptor type. */
        /* Total length of report descriptor. */
        sizeof(HID_JoystickReportDescriptor) & 0x00FF,
        ((sizeof(HID_JoystickReportDescriptor) & 0xFF00) >> 8),

        /* EP Descriptor: interrupt in. */
        LEN_ENDPOINT,   /* bLength */
        DESC_ENDPOINT,  /* bDescriptorType */
        (HID_JOYSTICK_EP_NUM | EP_INPUT), /* bEndpointAddress */
        EP_INT,         /* bmAttributes */
        /* wMaxPacketSize */
        EP4_MAX_PKT_SIZE & 0x00FF,
        ((EP4_MAX_PKT_SIZE & 0xFF00) >> 8),
        HID_DEFAULT_INT_IN_INTERVAL,     /* bInterval */
    };
```

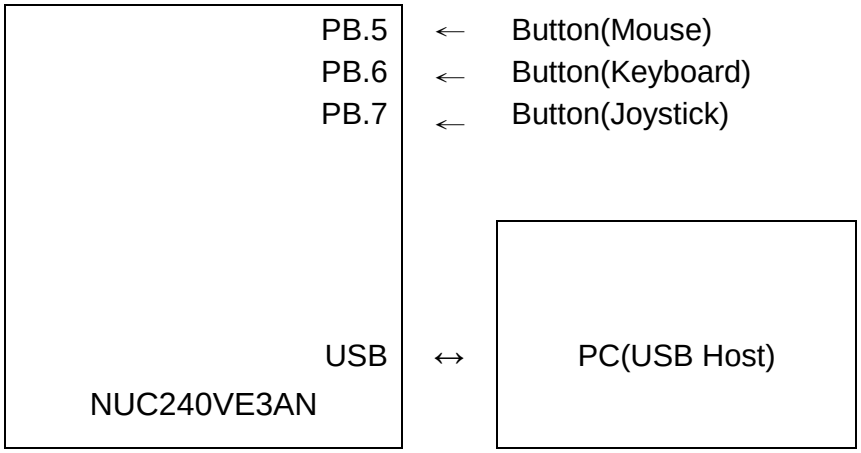
3 Software and Hardware Environment

Software Environment

- BSP version
  - NUC230/240 Series BSP CMSIS v3.01.001
- IDE version
  - Keil uVersion 5.26
- Tool for USB packet sniffing and device tree display
  - USBLyzer (http://www.usblyzer.com/)

Hardware Environment

- Circuit components
  - NuTiny-EVB-NUC240 V1.2
  - USB mini USB cable
- Diagram



4 Directory Information

- EC_NUC240_USB_Mouse_Keyboard_Joystick_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 USBD_HID_MouseKeyboardJoystick.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
July 9, 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.