USB Device Remote Wakeup

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	Demonstrate how to Implement USB Device Remote Wakeup Function.
BSP Version	M031 Series BSP CMSIS V3.01.000
Hardware	NuMaker-M032SE V1.3

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 contains the USB suspend/resume and remote wakeup knowledge. All USB devices must support suspend mode to keep low power consumption that is less than 2.5 mA after the USB devices observe no bus traffic for more than 3 mS. To exit suspend mode, a USB device can support remote wake-up capability to actively wake up suspended USB host or passively wait for the USB host to transfer the resume signal. The following context describes the implementation of suspend/resume and remote wake-up function on a USB mouse using the M032 microcontroller.

1.2 USB Suspend

The SUSPEND(USBD_ATTR[1]) interrupt flag of USB device is asserted when start-of-frame(SOF) signal isn't received for three consecutive frames (3 mS). When entering suspend state, the application software running in the M032 must take action to reduce power consumption based on USB current limitation. In suspend state, the USB Devices are able to respond to reset and resume commands.

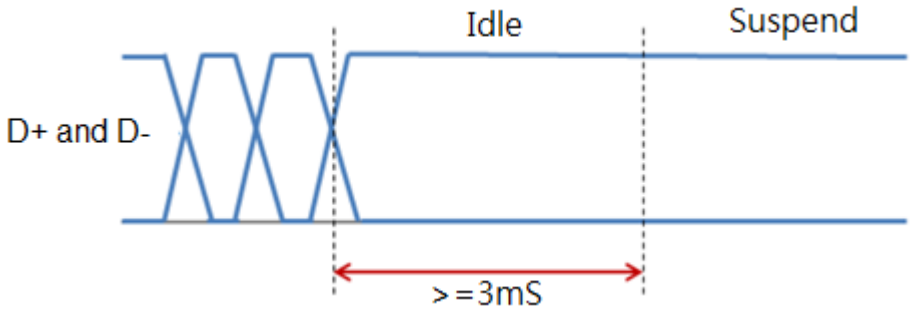


Figure 1 USB suspend signal

1.3 USB Resume

The suspended USB device can be resumed after USB host asserts a K-state on D+ and D- for 20 mS. (The K-state is defined in the USB specification based on USB speed). There are two ways to generate resume flow to a suspended USB device.

1. The USB Host can actively drive the K-state on D+ and D- for 20mS to resume devices. After suspended USB device detects resume flow, the RESUME(USBD_ATTR[2]) interrupt flag of USB device is asserted. The application software running in the M032 has to return to normal operation mode.
2. If a USB device supports the remote resume feature, this USB device can be applied to initiate the process of resuming normal operation. To begin resume flow, the application software running in the M032 has to set the RWAKEUP(USBD_ATTR[5]) to 1 to assert the K-state on D+ and D- for between 1 and 15mS. After USB host detects this state, USB host will take over controlling D+ and D- to keep K-state for the remainder of the 20 mS. This suspended USB

device detects resume flow and the RESUME(USBD_ATTR[2]) interrupt flag of USB device is asserted.

1.4 Demo Result

In Figure 2, when PC enters sleep mode in Windows, USB host will stop sending any package and then USB device enters suspend state. If one key of PC0 ~ PC5 is pressed, M032 USB device will wake PC from sleep using remote wake flow.

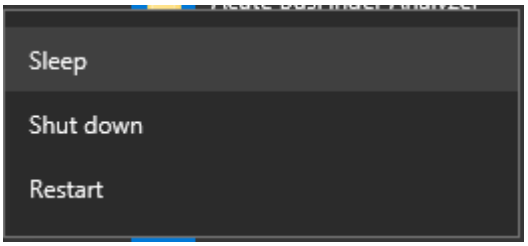


Figure 2 Power mode selection of Windows 10

2 Code Description

When USB suspend interrupt happens, g_u8Suspend is set to 1 to indicate that USB device is suspend state and then USB PHY is turned off to reduce power consumption. When USB reset or resume interrupt happens, g_u8Suspend is cleared to 0 to return from suspend to normal operation state.

```
void USBD_IRQHandler(void)
{
    uint32_t volatile u32IntSts = USBD_GET_INT_FLAG();
    uint32_t volatile u32State = USBD_GET_BUS_STATE();

    if (u32IntSts & USBD_INTSTS_FLDET)
    {
        …
    }

    if (u32IntSts & USBD_INTSTS_BUS)
    {
        /* Clear event flag */
        USBD_CLR_INT_FLAG(USBD_INTSTS_BUS);
        if (u32State & USBD_STATE_USBRST)
        {
            /* Bus reset */
            USBD_ENABLE_USB();
            USBD_SwReset();
            g_u8Suspend = 0;
        }
        if (u32State & USBD_STATE_SUSPEND)
        {
            /* Enter power down to wait USB attached */
            g_u8Suspend = 1;

            /* Enable USB but disable PHY */
            USBD_DISABLE_PHY();
        }
        if (u32State & USBD_STATE_RESUME)
        {
            /* Enable USB and enable PHY */
            USBD_ENABLE_USB();
            g_u8Suspend = 0;
        }
```

```
        }
        …
    }
```

The below functions in main.c are used to report HID mouse data or to enter power down mode when USB device is suspend state.

```
    while (1) {
            /* Enter power down when USB suspend */
            if(g_u8Suspend)
            {
                PowerDown();
                /* Waiting for key release */
                while((GPIO_GET_IN_DATA(PC) & 0x3F) != 0x3F);
            }
            HID_UpdateMouseData();
    }
```

In order to arrive minimum power consumption in suspend state, M032 executes power down command to stop running and then wait for USB interrupt or GPIO interrupt. Once M032 receives GPIO and g_usbd_RemoteWakeupEn is set to 1 by USB host feature command, M032 wakes up and then asserts K-state for 1 mS to do remove wakeup flow.

```
    void PowerDown()
    {
        /* Unlock protected registers */
        SYS_UnlockReg();

        printf("Enter power down ...\n");
        while(!IsDebugFifoEmpty());

        CLK_PowerDown();

        /* Note HOST to resume USB tree if it is suspended and remote wakeup enabled */
        if(g_usbd_RemoteWakeupEn && g_u8Suspend)
        {
            /* Enable PHY before sending Resume('K') state */
            USBD->ATTR |= USBD_ATTR_PHYEN_Msk;

            /* Keep remote wakeup for 1 ms */
            USBD->ATTR |= USBD_ATTR_RWAKEUP_Msk;
            CLK_SysTickDelay(1000); /* Delay 1ms */
            USBD->ATTR ^= USBD_ATTR_RWAKEUP_Msk;
            printf("Remote Wakeup!!\n");
        }
```

```
        printf("device wakeup!\n");

        /* Lock protected registers */
        SYS_LockReg();
    }
```

2.1 hid_mouse.c

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.

```
    void HID_UpdateMouseData(void)
    {
        uint8_t *buf;
        uint32_t u32Reg;
        static int32_t x = 0, y = 0;
        uint32_t u32MouseKey;
    /*
       Key definition:
            PC0 Down
            PC1 right
            PC2 up
            PC3 right key
            PC4 left
            PC5 left key
    */
        if (g_u8EP2Ready)
        {
            buf = (uint8_t *)(USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP2));
            u32Reg = PC->PIN & 0x3F;
            /* To control Y axis */
            if((u32Reg & 1) == 0)
                 y += 1;
            else if((u32Reg & 4) == 0)
                 y += -1;
            else
                 y = 0;
            if(y > 48) y = 48;
            if(y < -48) y = -48;

            /* To control X axis */
```

```
            if((u32Reg & 2) == 0)
                x += 1;
            else if((u32Reg & 0x10) == 0)
                x += -1;
            else
                x = 0;
            if(x > 48) x = 48;
            if(x < -48) x = -48;
            /* Mouse key */
            u32MouseKey = 0;
            if((u32Reg & 0x20) == 0)
                u32MouseKey |= 1; /* Left key */
            if((u32Reg & 0x8) == 0)
                u32MouseKey |= 2; /* Right key */

            /* Update new report data */
            buf[0] = u32MouseKey;
            buf[1] = x >> 2;
            buf[2] = y >> 2;
            buf[3] = 0x00;
            g_u8EP2Ready = 0;
            /* Set transfer length and trigger IN transfer */
            USBD_SET_PAYLOAD_LEN(EP2, 4);
        }
    }
```

2.2 Descriptor.c

HID_MouseReportDescriptor array includes the HID Report Descriptor for mouse function. It defines the data format 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 */
    };
```

2.3 Configuration Descriptor

USB host requests configuration descriptor for device enumeration. gu8ConfigDescriptor is the configuration descriptor that includes interface descriptor, HID descriptor and endpoint descriptor in following. The USBD_REMOTE_WAKEUP must be set to inform USB host that USB device supports remove wakeup function.

```
    /*!<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),
        0x01,           /* bNumInterfaces */
        0x01,           /* bConfigurationValue */
        0x00,           /* iConfiguration */
        0x80 | (USBD_SELF_POWERED << 6) | (USBD_REMOTE_WAKEUP << 5),/* bmAttributes */
        USBD_MAX_POWER,         /* MaxPower */

        /* I/F descr: HID */
        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,        /* 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 */
        (INT_IN_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 */
    };
```

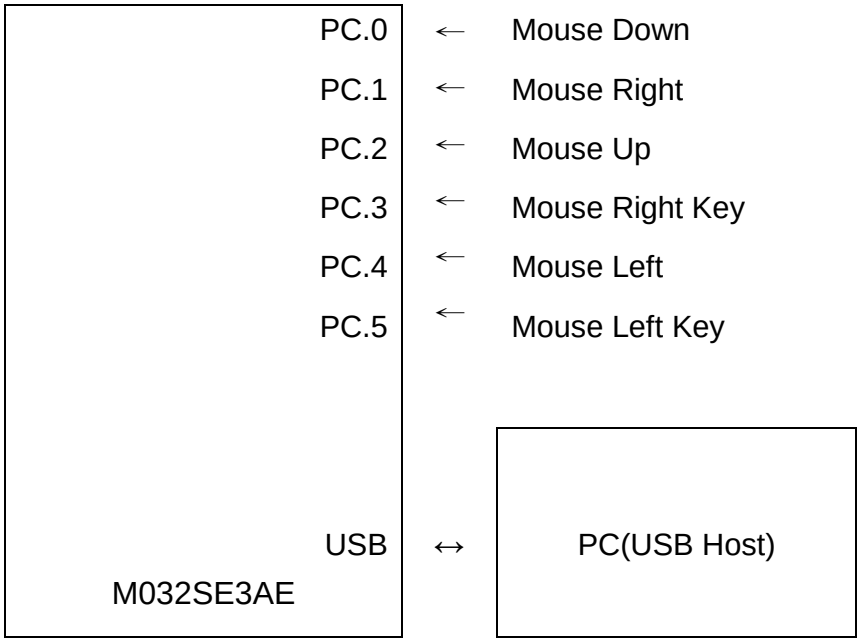
3 Software and Hardware Environment

Software Environment

- BSP version
  - M031 Series BSP CMSIS v3.01.000
- IDE version
  - Keil uVersion 5.26
- Tool for USB packet sniffing and device tree display
  - USBLyzer (http://www.usblyzer.com/)

Hardware Environment

- Circuit components
  - NuMaker-M032SE V1.3
  - USB mini USB cable
- Diagram



4 Directory Information

- EC_M032_USBD_HID_Remote_Wakeup_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_RemoteWakeup.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 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.

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.