WebUSB HID Mouse

Example Code Introduction for 32-bit NuMicro® Family

Document Information

Application	WebUSB provides a way to access nearby USB devices through a web page. This means that users can interact with the device by browsing the device manufacturer's web page without the need for additional software installation. This example code demonstrates how to add support for the WebUSB interface to an existing HID Mouse device and provides a web page example to illustrate how to access the device through a web page.
BSP Version	M2351 BSP v3.00.007
Hardware	NuMaker-PFM-M2351

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

USB has become the standard interface for many devices, and USB devices connected to computers are typically categorized into various device classes, such as keyboards, mouse, cameras, and more. These devices utilize the operating system's class drivers for support. In addition to these standardized devices, there are numerous devices that do not belong to any specific class. These devices require custom drivers, and utilizing them often involves installing specific software. Installing these devices typically involves searching for drivers and software on the manufacturer's website, downloading, and installing them. If you want to use the device on another computer, you may need to repeat this process again.

WebUSB provides a method for servicing these non-standard USB devices, allowing them to be operated through a web browser. This means that hardware manufacturers can offer a way to access their devices from a webpage without the need for their own applications and driver software. When connecting a new device that supports WebUSB, the browser displays a notification with a link to the manufacturer's website. The webpage provides services corresponding to the device, and through appropriate user actions, the browser will prompt whether to allow connection to the device. Once the connection is successful, the device can be operated through the webpage provided by the manufacturer. Throughout the entire process of connection and usage, there is no need to download and install drivers, making it convenient and quick for users to start using the device.

This example code demonstrates how to integrate a standard HID mouse device with a device supporting WebUSB into a composite device. This allows the device to support standard mouse functions while also supporting non-standard USB device functions through WebUSB. Developers can extend this example to implement interesting applications, such as mouse products that support WebUSB. In addition to standard USB mouse functions, it can also download new LED flashing modes and settings on the mouse via WebUSB, modify macro tasks, and more.

2. USB Description

2.1 USB Device Descriptor

Value	Field	Description
0x12	bLength	Size of this descriptor
0x01	bDescriptorType	Device descriptor
0x0300	bcdUSB	USB Specification Release Number
0x00	bDeviceClass	Class code. Setting the field to zero enables each interface in a configuration to have its own class information, allowing independent operation of the interfaces.
0x00	bDeviceSubClass	Subclass code. If the bDeviceClass field is reset to zero, this field must also be reset to zero.
0x00	bDeviceProtocol	Protocol code. Reset to zero, the device does not use class-specific protocols on a device basis.
0x08	bMaxPacketSize0	Maximum packet size for endpoint zero
0x0416	idVendor	Vendor ID (Nuvoton)
0xB0F3	idProduct	Product ID (WebUSB HID Mouse)
0x0100	bcdDevice	Device release number in binary-coded Decimal
0x01	iManufacturer	Index of string descriptor describing manufacturer
0x02	iProduct	Index of string descriptor describing product
0x00	iSerialNumber	Index of string descriptor describing the device's serial number
0x01	bNumConfigurations	Number of possible configurations

Here, the general information of the device is described, including the ID, version, product name, serial number, etc. The ID, in particular, will be used to filter out WebUSB devices that can be connected on the webpage.

2.2 Configuration Descriptor

Value	Field	Description
Configuration descriptor		
0x09	bLength	Size of this descriptor
0x02	bDescriptorType	CONFIGURATION Descriptor
0x0039	wTotalLength	Total descriptor length
0x02	bNumInterfaces	Number of interfaces supported by this configuration
0x01	bConfigurationValue	CONFIGURATION 1
0x00	iConfiguration	Index of string descriptor describing this configuration (None)
0x80	bmAttributes	Bus powered and no remote wakeup
0x32	bMaxPower	Maximum power consumption of the USB device from the bus. (50 * 2mA = 100mA)
HID Interface descriptor		
0x09	bLength	Size of this descriptor
0x04	bDescriptorType	Interface descriptor
0x00	bInterfaceNumber	Interface 0
0x00	bAlternateSetting	Alternate setting 0 (default)
0x01	bNumEndpoints	1 endpoint
0x03	bInterfaceClass	HID interface class
0x01	bInterfaceSubClass	Boot interface subclass
0x02	bInterfaceProtocol	Mouse
0x00	iInterface	Interface name
HID descriptor		
0x09	bLength	Size of this descriptor
0x21	bDescriptorType	HID descriptor
0x0101	bcdHID	HID version 1.1
0x00	bCountryCode	Hardware target country
0x01	bNumDescriptors	Number of HID class descriptors to follow
0x22	bDescriptorType	Report descriptor type
0x0034	wDescriptorLength	Total length of the Report descriptor
Endpoint descriptor		
0x07	bLength	Size of this descriptor
0x05	bDescriptorType	Endpoint descriptor
0x81	bEndpointAddress	Endpoint 1 (IN)
0x03	bmAttributes	Interrupt
0x0008	wMaxPacketSize	8 byte packets
0x14	bInterval	20ms interval
Interface descriptor for WebUSB		
0x09	bLength	Size of this descriptor

0x04	bDescriptorType	Interface descriptor
0x01	bInterfaceNumber	Interface 1
0x00	bAlternateSetting	Alternate setting 0 (default)
0x02	bNumEndpoints	2 endpoints
0xFF	bInterfaceClass	Vendor-specific interface class
0x00	bInterfaceSubClass	
0x00	bInterfaceProtocol	
0x00	iInterface	Interface name (none)
Endpoint descriptor		
0x07	bLength	Size of this descriptor
0x05	bDescriptorType	Endpoint descriptor
0x82	bEndpointAddress	Endpoint 2 (IN)
0x02	bmAttributes	Bulk
0x0040	wMaxPacketSize	64 byte packets
0x00	bInterval	N/A for bulk endpoints
Endpoint descriptor		
0x07	bLength	Size of this descriptor
0x05	bDescriptorType	Endpoint descriptor
0x03	bEndpointAddress	Endpoint 3 (OUT)
0x02	bmAttributes	Bulk
0x0040	wMaxPacketSize	64 byte packets
0x00	bInterval	N/A for bulk endpoints

The USB configuration descriptor is composed of multiple descriptors. Each descriptor begins with the bLength and bDescriptorType fields to identify the descriptor and its size. The first interface is one with the corresponding HID descriptor and an HID interface with a single endpoint used to pass input events to the operating system, which is the HID Mouse function interface. The second interface is a vendor-specific interface with two endpoints used to send commands to the device and receive responses, known as the WebUSB interface.

2.3 WebUSB Descriptor

To add WebUSB support with minimal impact on the original device functionality, BOS (Binary Object Store) is used to provide WebUSB-related function descriptors.

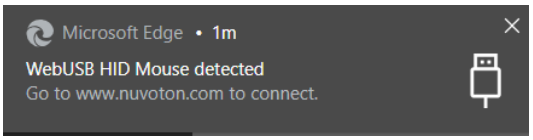
Binary Object Store (BOS) is a concept introduced in USB 3.0 but has also been incorporated into USB 2.0 devices as part of USB version 2.1. Through BOS, support for WebUSB is declared, and this declaration content follows the BOS's own configuration descriptor block. The actual content is as follows:

Value	Field	Description
Binary device Object Store descriptor		
0x05	bLength	Size of this descriptor
0x0F	bDescriptorType	Binary device Object Store descriptor
0x0039	wTotalLength	Total length of this series of descriptors
0x01	bNumDeviceCaps	Number of device capability descriptors in the BOS
WebUSB platform capability descriptor		
0x18	bLength	Size of this descriptor
0x10	bDescriptorType	Device capability descriptor
0x05	bDevCapabilityType	Platform capability descriptor
0x00	bReserved	
--	PlatformCapablityUUID	WebUSB platform capability descriptor GUID in little-endian format {3408b638-09a9-47a0-8bfd-a0768815b665}
0x0100	bcdVersion	WebUSB descriptor version 1.0
0x01	bVendorCode	bRequest value for WebUSB
0x01	iLandingPage	URL for landing page
Microsoft OS 2.0 platform capability descriptor		
0x1C	bLength	Size of this descriptor
0x10	bDescriptorType	Device capability descriptor
0x05	bDevCapabilityType	Platform capability descriptor
0x00	bReserved	
	PlatformCapablityUUID	Microsoft OS 2.0 platform compatibility descriptor GUID in little-endian format {D8DD60DF-4589-4CC7-9CD2-659D9E648A9F}
0x06030000	dwWindowsVersion	Minimum compatible Windows version (Windows 8.1)
0x00B2	wMSOSDescriptorSetTotalLength	Total length of the descriptor set
0x02	bMS_VendorCode	bRequest value for retrieving further Microsoft descriptors
0x00	bAltEnumCode	Device does not support alternate enumeration

To use WebUSB on the Windows operating system, it is necessary to load a specific driver called WinUSB, which provides the interface for applications to access the device. This can be achieved by either installing a custom driver information file (INF) on the system or by providing Microsoft OS compatibility descriptors. Since using INF files requires users to take additional

steps to install the driver, here we only introduce the method of providing Microsoft OS 2.0 platform descriptors. This approach helps manufacturers avoid the need for signature certification for their custom drivers and eliminates the inconvenience for users to download and install drivers separately.

From the content of the BOS descriptor in the above table, it can be observed that it includes both the WebUSB platform descriptor and the Microsoft OS 2.0 platform descriptor. It defines the UUIDs for the WebUSB platform and Microsoft OS 2.0 platform, along with the corresponding function command codes bVendorCode (0x01) and bMS_VendorCode (0x02). The iLandingPage parameter specifies the URL of the webpage for WebUSB. After the device successfully connects to the operating system, the web browser displays a notification. Users can click on this notification to directly access the specified URL, as shown below.



2.4 URL Descriptor

When a WebUSB device is connected to the operating system, the browser requests the device to return the webpage description using the Vendor Code command defined in BOS. The returned content is as follows:

Value	Field	Description
URL descriptor		
0x12	bLength	Size of this descriptor
0x03	bDescriptorType	URL descriptor
0x01	bScheme	https://
--	URL	UTF-8 encoded URL content “www.nuvoton.com”

2.5 Microsoft OS 2.0 Descriptor

Value	Field	Description
Microsoft OS 2.0 descriptor set header		
0x000A	wLength	Size of this descriptor
0x0000	wDescriptorType	Descriptor set header descriptor
0x06030000	dwWindowsVersion	Minimum compatible Windows version (Windows 8.1)
0x00B2	wTotalLength	Total length of the descriptor set
Microsoft OS 2.0 configuration subset header		
0x0008	wLength	Size of this descriptor
0x0001	wDescriptorType	Configuration subset header desc.
0x00	bConfigurationValue	Applies to configuration 1 (indexed from 0 despite configurations normally indexed from 1)

0x00	bReserved	Must be set to 0
0x00A8	wTotalLength	Total length of the subset including this header
Microsoft OS 2.0 function subset header		
0x0008	wLength	Size of this descriptor
0x0002	wDescriptorType	Function subset header descriptor
0x01	bFirstInterface	First interface of the function
0x00	bReserved	Must be set to 0
0x00A0	wSubsetLength	Total length of the subset including this header
Microsoft OS 2.0 compatible ID descriptor		
0x0014	wLength	Size of this descriptor
0x0003	wDescriptorType	Compatible ID descriptor
"WINUSB\0\0"	CompatibileID	ASCII string padded to 8 bytes
"\0\0\0\0\0\0\0\0"	SubCompatibleID	ASCII string padded to 8 bytes
Microsoft OS 2.0 registry property descriptor		
0x0084	wLength	Size of this descriptor
0x0004	wDescriptorType	Registry property descriptor
0x0007	wPropertyDataType	REG_MULTI_SZ
0x002A	wPropertyNameLength	Length of the property name
--	PropertyName	Property name with null terminator encoded in UTF-16LE. "DeviceInterfaceGUIDs\0"
0x0050	wPropertyDataLength	Length of the property value
--	PropertyData	GUID plus two null terminators encoded in UTF-16LE. "{fd45b4d3-9e41-4e1d-9ba8-cc03726b9db8}\0\0" Note: The GUID here is not fixed but generated by “New-Guid” command of PowerShell.

Please note that Windows will only obtain this information during the initial query of the device. If the device does not respond with valid descriptors, it will not query this descriptor again the next time the device is connected.

Microsoft provides a list of USB device registry entries describing the registry entries created when enumerating a device. When testing, you can delete the entries created for a specific device in this registry list to force Windows to attempt reading the descriptors again. The registry list is located at:

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\usbflags]

3. Firmware

To implement a WebUSB device, in addition to the necessary device descriptors, handling the Vendor commands corresponding to the WebUSB device descriptors is crucial. Additionally, you need to manage the corresponding Endpoint processing.

3.1 Vendor Command

The Vendor Commands that need to be handled for WebUSB include commands to retrieve the URL and to obtain the MS OS 2.0 Descriptor Set. These command codes are specified in the BOS configuration descriptor.

Get URL:

bmRequestType	bRequest	wValue	wIndex	wLength	Data (response)
0b11000000	0x01	0x0001	0x0002	*	The URL descriptor

Get MS OS 2.0 Descriptor:

bmRequestType	bRequest	wValue	wIndex	wLength	Data (response)
0b11000000	0x02	0x0000	0x0007	*	The URL descriptor

3.2 Implementation

- The function for setting up Vendor Command

```
    printf("+--------------------------------------------------------+\n");
    printf("|          NuMicro USB HID Mouse Sample Code             |\n");
    printf("+--------------------------------------------------------+\n");

    USBD_Open(&gsInfo, HID_ClassRequest, NULL);

    /* Set handler for Vendor requests */
    USBD_SetVendorRequest(WebUSB_VendorRequest);

    /* Endpoint configuration */
    HID_Init();
    USBD_Start();
```

Since Vendor Commands are requested by the operating system during device enumeration, they must be set up during USB device initialization.

- Vendor Command Handler:

```
void WebUSB_VendorRequest()
{
    uint8_t au8Buf[8];
    uint32_t u32Len;
    uint8_t bRequest;
```

```
    uint16_t wValue;
    uint16_t wIndex;
    uint16_t wLength;

    USBD_GetSetupPacket(au8Buf);

    /*
        bmRequestType   bRequest    wValue  wIndex  wLength Data (response)
        0b11000000      0x02        0x0000  0x0007  *       MS OS 2.0 Descriptor Set

        bRequest is the vendor code defined in BOS descriptor
        wIndex = 7 is to retrieve the Microsoft OS 2.0 Descriptor Set from the device.

    */
    bRequest = au8Buf[1];
    wValue = au8Buf[2] + ((uint16_t)au8Buf[3] << 8);
    wIndex = au8Buf[4] + ((uint16_t)au8Buf[5] << 8);
    wLength= au8Buf[6] + ((uint16_t)au8Buf[7] << 8);

    switch(bRequest)
    {
        case 2: // VendorCode defined in BOS
        {
            // Stall if not supported command
            if((wIndex != 0x07) || (g_USBD_sInfo->gu8MSDesc == 0))
            {
                USBD_SET_EP_STALL(EP0);
                USBD_SET_EP_STALL(EP1);
            }

            /* To retrieve the Microsoft OS 2.0 Descriptor Set from the device */
            u32Len = USBD_Minimum(wLength, g_USBD_sInfo->gu8MSDesc[8]);
            USBD_PrepareCtrlIn((uint8_t *)g_USBD_sInfo->gu8MSDesc, u32Len);
            USBD_PrepareCtrlOut(0, 0UL); /* For status stage */
            break;
        }
        case 1: // VendorCode defined in BOS
        {
            // Stall if not supported command
            if((wIndex != 0x02) || (wValue != 0x01))
            {
                USBD_SET_EP_STALL(EP0);
                USBD_SET_EP_STALL(EP1);
            }

            /* To retrieve URL */
            u32Len = USBD_Minimum(wLength, g_USBD_sInfo->gu8StringDesc[3][0]);
            USBD_PrepareCtrlIn((uint8_t *)g_USBD_sInfo->gu8StringDesc[3], u32Len);
            USBD_PrepareCtrlOut(0, 0UL); /* For status stage */

            break;
        }
        default:
        {
            /* Unsupported request */
            USBD_SET_EP_STALL(EP0);
```

```
            USBD_SET_EP_STALL(EP1);
            break;
        }
    }
}
```

This function primarily handles the functionalities of Get URL and Get MS OS 2.0 Descriptor Set. If it receives a command other than these two, it will return a STALL response. However, if it receives the correct command, it will respond with either the URL descriptor or the MS OS 2.0 Descriptor Set based on the received command.

4. Software and Hardware Requirements

4.1 Software Requirements

- BSP Version
  - M2351 BSP v3.00.007
- Applications
  - Google Chrome or Microsoft Edge

4.2 Hardware Requirements

- Circuit components
  - NuMaker-PFM-M2351

5. Directory Information

The directory structure is shown below:

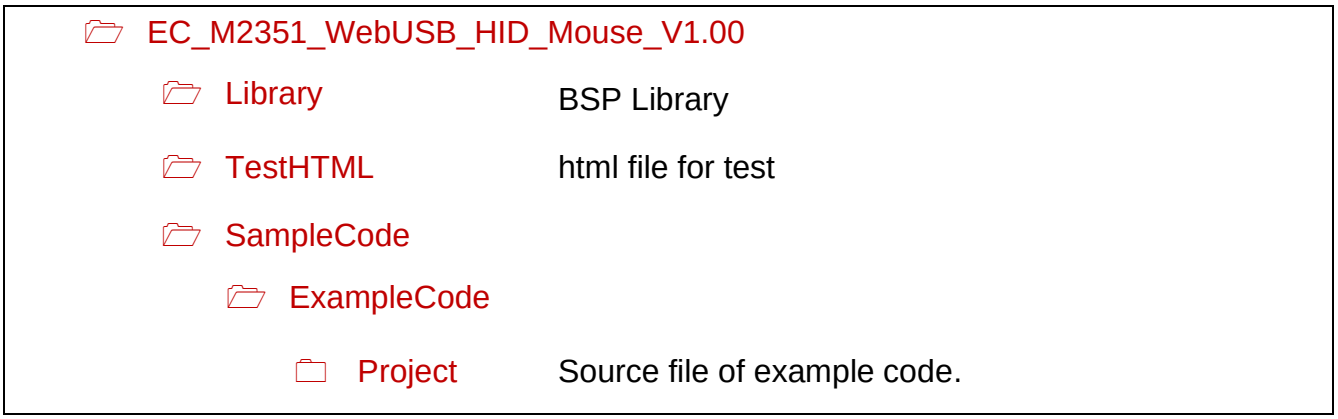


Figure 5-1 Directory Structure

6. Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click *USBD_HID_Mouse.uvprojx*.
2. Open Keil compile to build and download the code to target board.
3. Reboot the board to execute the firmware.
4. Plug-in the board with USB to PC (Windows 8 or later)
5. Browse to TestHTML folder and open *test_webusb.html* by Google Chrome or Microsoft Edge.
6. Click '**Connect To WebUSB Device**' button and select WebUSB HID Mouse device to connect.

Figure 6-1 Connect to WebUSB Device

7. Click '**Send**' button to send message in Sender block. WebUSB will echo the message from the Receiver block.

Figure 6-2 Send and Receive Message

7. Revision History

Date	Revision	Description
2024.03.08	1.00	Initial version.

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.