USBD Bulk Communication

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	Demonstrate how to do USB communication in bulk transfer type
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 bulk device and a PC-side application used to communicate with that device. PC-side program implements read and write control to USB bulk device by calling bulk APIs. USB bulk device gets the data from PC and returns it to PC, and the data will be printed on window command processor simultaneously.

1.2 Principle

When the USB bulk device connects to the host, the host needs to install driver software located in Driver_BulkUSB folder to recognize the USB device. The driver software can make corresponding device operation and let CPU carry on various applications and operations. When device and host connect correctly, they can start to transfer data and communicate to each other.
This example code provides PC-side with bulk application to let user easily access the IN/OUT endpoint of USB bulk device. The APIs provided by this application are listed in Table 1. For more information about the function description, incoming parameter and return value of each API, please refer to the source file, BulkUSB, located in App_BulkUSB folder.

Bulk APIs
BOOL CBulkUSB::OpenDevice(void)
BOOL CBulkUSB::WriteUSB(LPCVOID lpBuffer, DWORD nNumberOfBytesToWrite)
BOOL CBulkUSB::ReadUSB(LPVOID lpBuffer, DWORD nNumberOfBytesToRead)
void CBulkUSB::CloseDevice(void)

Table 1 Bulk APIs

The driver software and bulk application mentioned above are placed in the directory of EC_NUC240_USBD_Bulk_Communication_V1.00\SampleCode\ExampleCode\NUC240_USBD_Bulk_Communication.

1.3 Demo Result

The following execution results will be output to window command processor.
After you build the bulk application, you can open BulkUSB.exe located in Debug folder via window command processor. When you execute the execution file, PC reads VID and PID of the USB bulk device firstly, then writes 64-byte data to the USB bulk device, and reads 64-byte data from it finally. The execution result as shown in Figure 1-1.

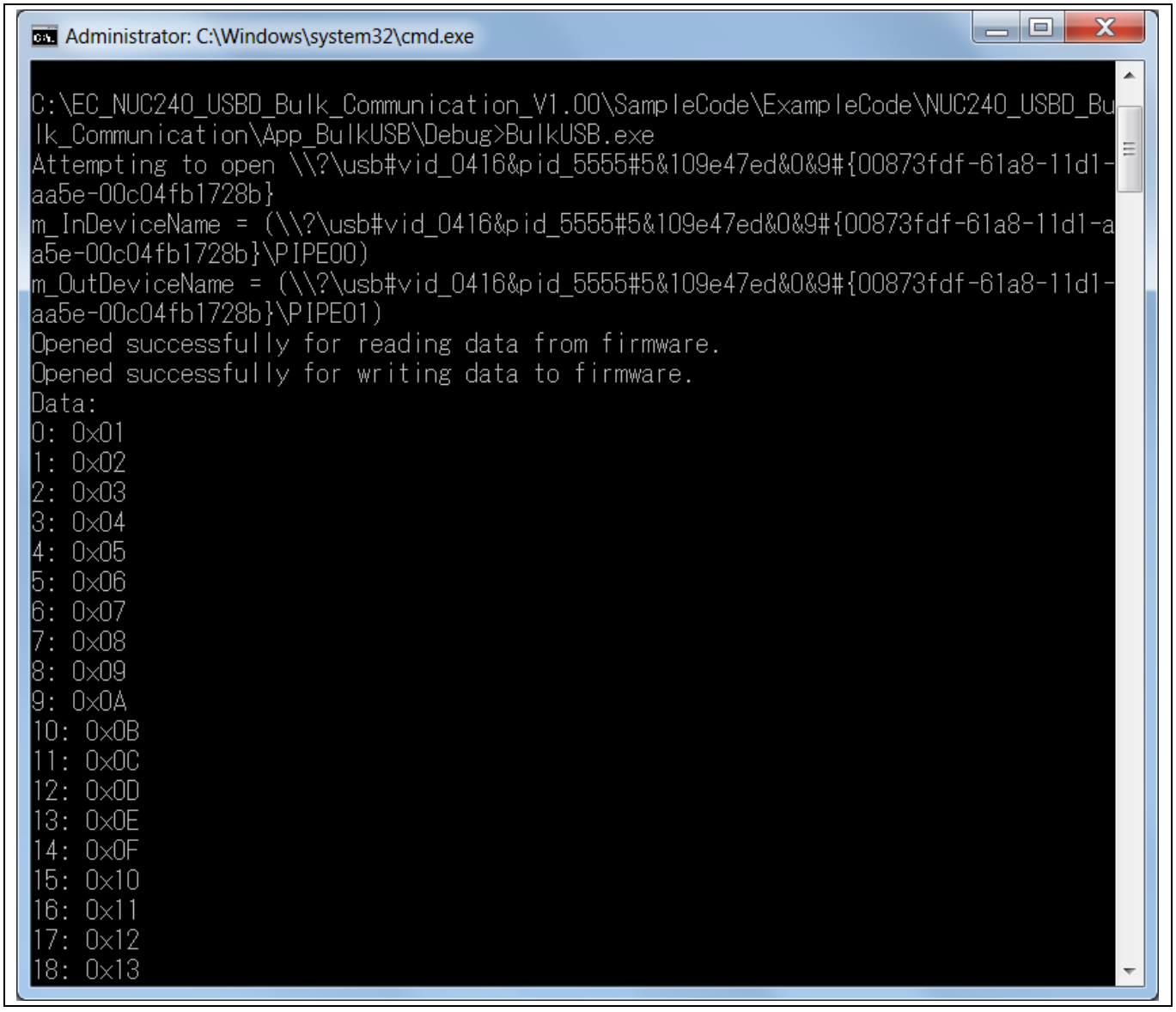

Figure 1-1 Execution Result

2 Code Description

The function of VCOM_Init() is used to do endpoint configuration including endpoint type, endpoint buffer, etc. In this example code, endpoint 2 is configured as bulk IN endpoint and endpoint 3 is configured as bulk OUT endpoint.

```
void VCOM_Init(void)
{
    ...

    /* EP2 ==> Bulk IN endpoint, address 1 */
    USBD_CONFIG_EP(EP2, USBD_CFG_EPMODE_IN | BULK_IN_EP_NUM);
    /* Buffer offset for EP2 */
    USBD_SET_EP_BUF_ADDR(EP2, EP2_BUF_BASE);

    /* EP3 ==> Bulk Out endpoint, address 2 */
    USBD_CONFIG_EP(EP3, USBD_CFG_EPMODE_OUT | BULK_OUT_EP_NUM);
    /* Buffer offset for EP3 */
    USBD_SET_EP_BUF_ADDR(EP3, EP3_BUF_BASE);
    /* trigger receive OUT data */
    USBD_SET_PAYLOAD_LEN(EP3, EP3_MAX_PKT_SIZE);
}
```

When device receives OUT token that is issued by PC to send data, the function of EP3_Handler() will be executed in endpoint 3 interrupt handler. In the function of EP3_Handler(), device will prepare data according to the data received from PC for endpoint 2 buffer. After that, when device receives IN token, the data stored in endpoint 2 buffer will be sent back to PC.

```
void EP3_Handler(void)
{
    int i;

    /* Bulk OUT */
    gu32RxSize = USBD_GET_PAYLOAD_LEN(EP3);
    gpu8TxBuf = (uint8_t *)(USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP2));
    gpu8RxBuf = (uint8_t *)(USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP3));

    for (i = 0; i < 64; i++)
        gpu8TxBuf[i] = gpu8RxBuf[i] + 1;

    /* Ready to get next BULK out */
```

```
    USBD_SET_PAYLOAD_LEN(EP3, EP3_MAX_PKT_SIZE);
    USBD_SET_PAYLOAD_LEN(EP2, EP2_MAX_PKT_SIZE);
}
```

3 Software and Hardware Environment

Software Environment

- BSP version
  - NUC230/NUC240 Series BSP CMSIS v3.01.002
- IDE version
  - Keil uVision V5.28

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.

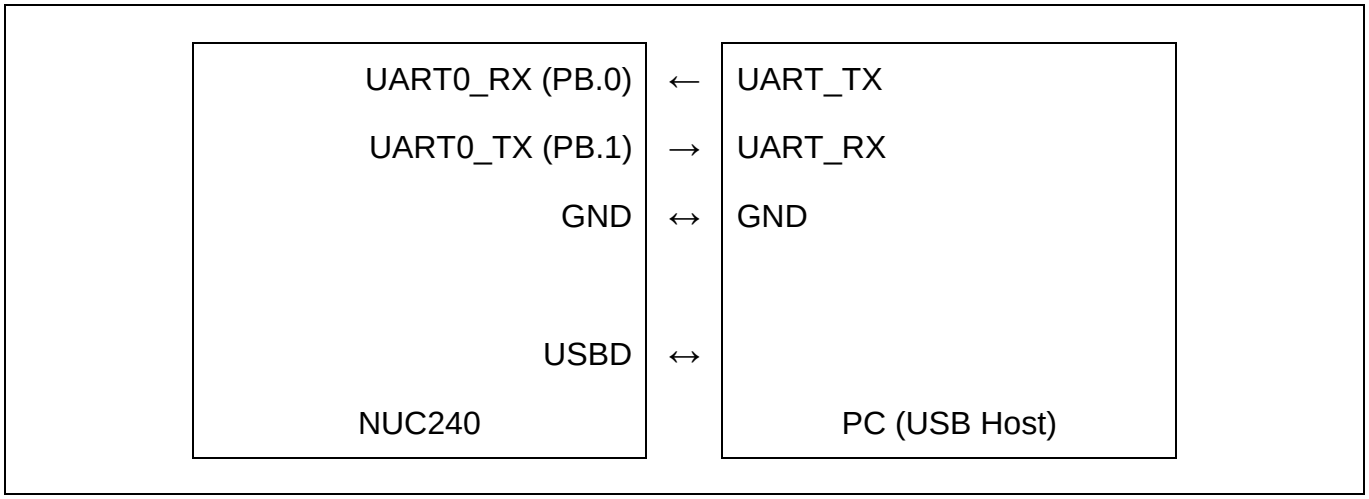

Figure 3-1 NUC240 and PC Connection

4 Directory Information

- EC_NUC240_USBD_Bulk_Communication_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_Bulk_Communication.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
Sep. 19, 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.