USBD HID To EEPROM

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	An implementation of controlling the I2C transfer protocol of NUC240 from PC side via USB HID interface to access the EEPROM device
BSP Version	NUC230/240 Series BSP v3.01.002
Hardware	NuTiny–EVB–NUC240–LQFP100 V1.0

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 sample code demonstrates how to control the I2C data transfer of NUC240 via the USB HID (Human Interface Device) connection on the PC to access to the EEPROM device.

1.2 Principle

The operation flow of this sample code is to allow the user to execute commands from the PC, transfer the commands to the NUC240 via the interrupt transmission of the USB HID interface, and then access the EEPROM device. The NUC240 side waits for the USB connection and controls the transmission with the USB EP3 (End Point 3). When the EP3 issues a transmission interruption, that is, after the USB connection is successful, the NUC240 terminal starts to receive the command and performs the I2C read/write operation to access the EEPROM device. As shown in Figure 1-1:

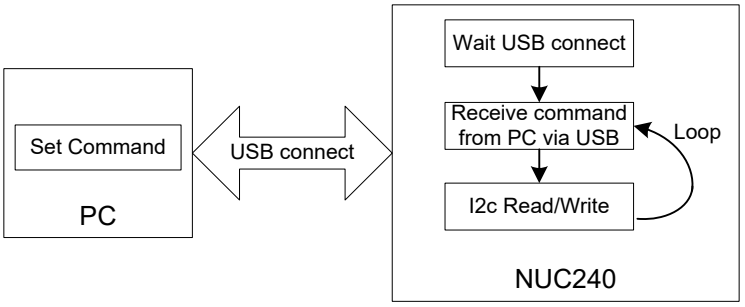

Figure 1-1 Connection and transmission flow control between PC and NUC240

1.3 Demo Result

In this sample code, user need to use two windows tools, HIDTransferTest and HID_I2C_TOOL. The following are links to get the two open source codes:

The link of HIDTransferTest:
https://github.com/OpenNuvoton/NUC472_442BSP/tree/master/SampleCode/StdDriver/USBD_HID_Transfer/WindowsTool/HIDTransferTest

The link of HID_I2C_TOOL:
http://bbs.21ic.com/icview-1103894-1-1.html

After connecting the PC to the USB port of the NUC240, execute this sample program and the initial message will show on the UART window of the PC. As shown in Figure 1-2:

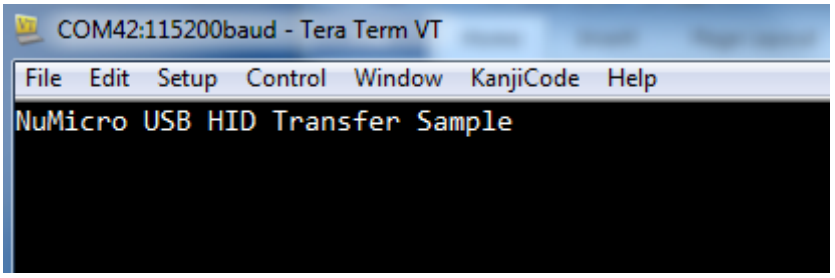

Figure 1-2 The initial message of the sample program

Then open the executable file HIDTransferTest.exe, a window will appear showing the connection is successful. As shown in Figure 1-3:

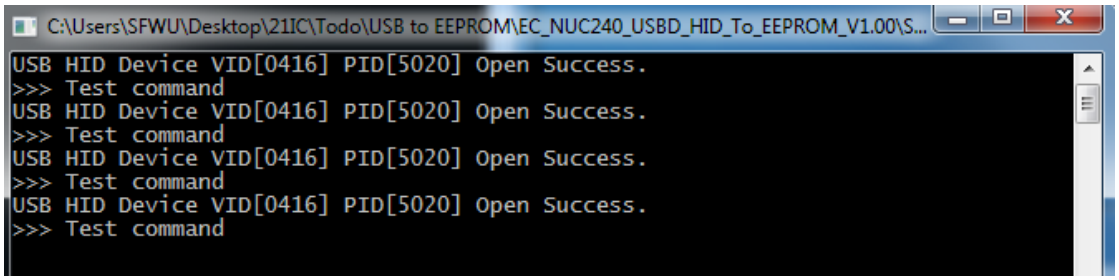

Figure 1-3 Detecting successful USB HID connection

After the successful USB HID connection being displayed, the message of I2C transmission data will be displayed on the UART window of the PC, indicating that I2C has started data transmission. As shown in Figure 1-4:

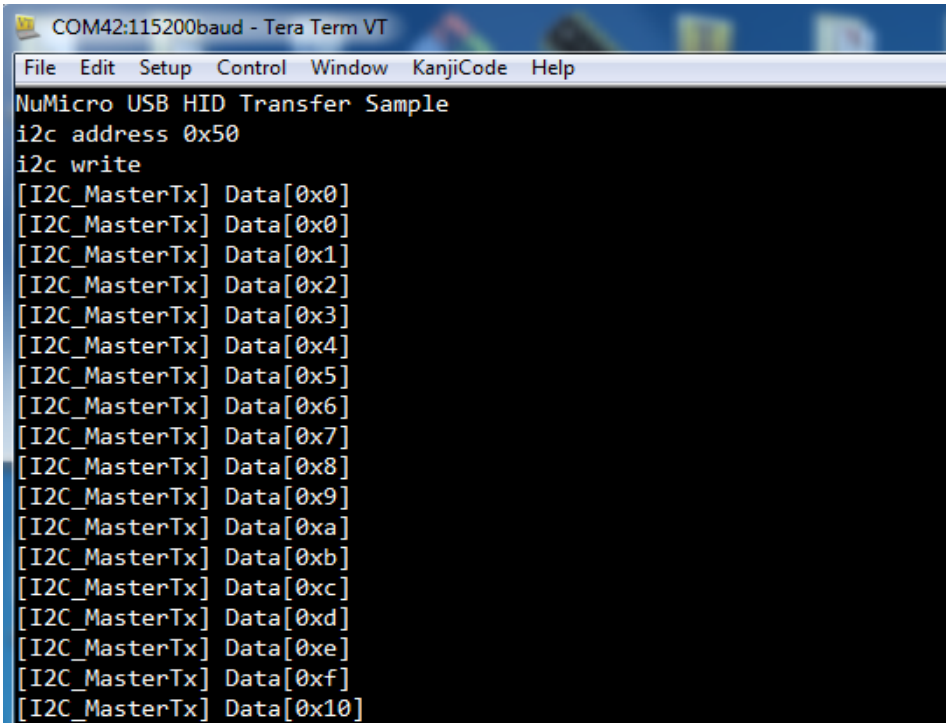

Figure 1-4 Messages transmitted with the EEPROM after the USB HID is connected

Then open the executable file HID_I2C_TOOL.exe, a window will appear as shown in Figure 1-5. User can set the starting position (OFFSET field), length (LENGTH field) and the value of the transmitted data (Value field on this interface.

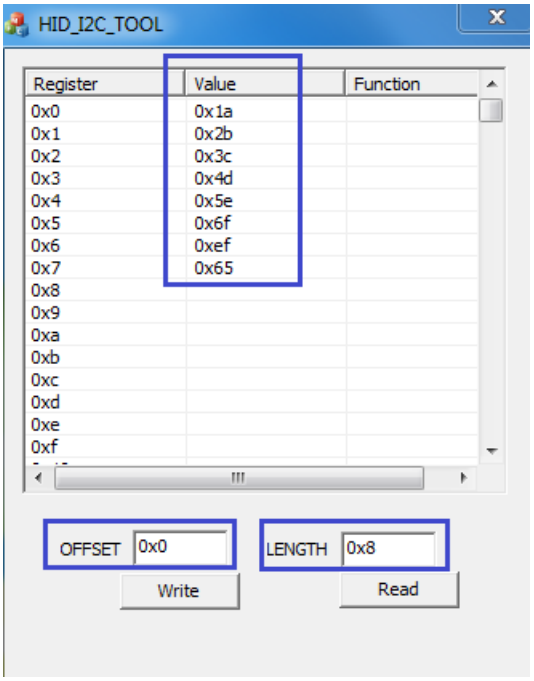

Figure 1-5 HID_I2C_TOOL interface and initial settings

After the initial setting is completed, user can select to execute the read (Read button) or write (Write button) operation. If user selects write operation, he can click “Write” button. As shown in Figure 1-6:

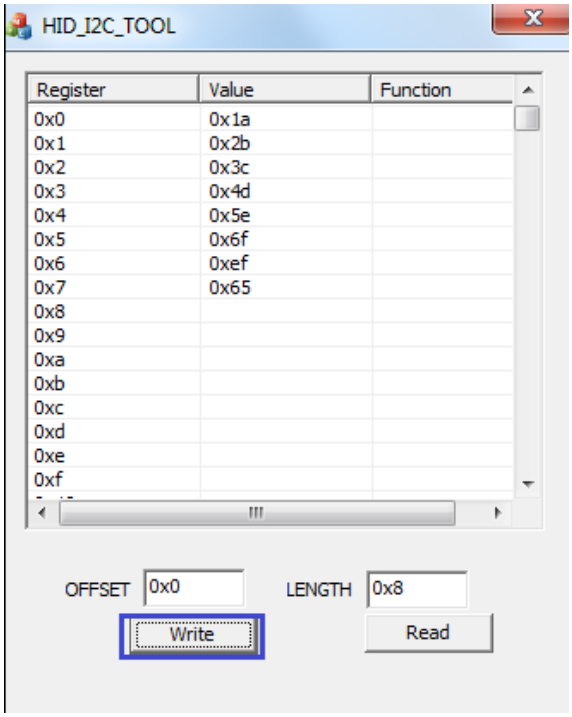

Figure 1-6 Performing I2C Write Data on the HID_I2C_TOOL Interface

Click the Write button and the UART window on the PC will display the data written to the EEPROM device. As shown in Figure 1-7:

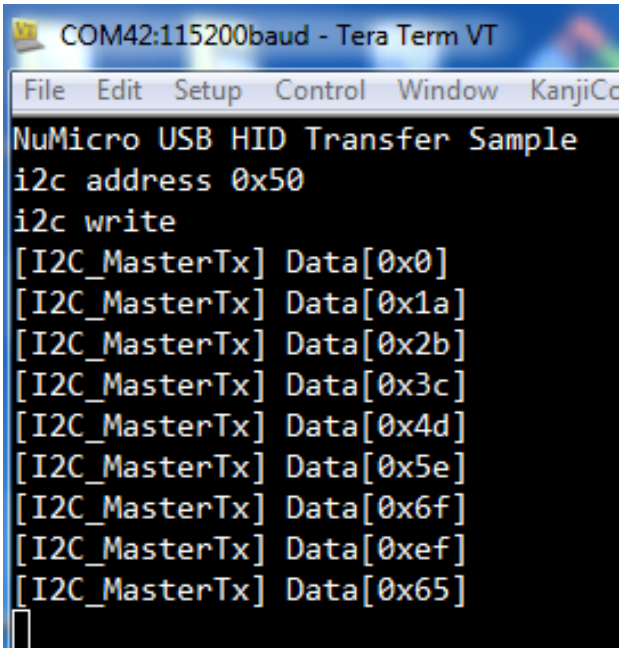

Figure 1-7 Displaying I2C written data on the UART window

If user selects read operation, he can click “Read” button, as shown in Figure 1-6:

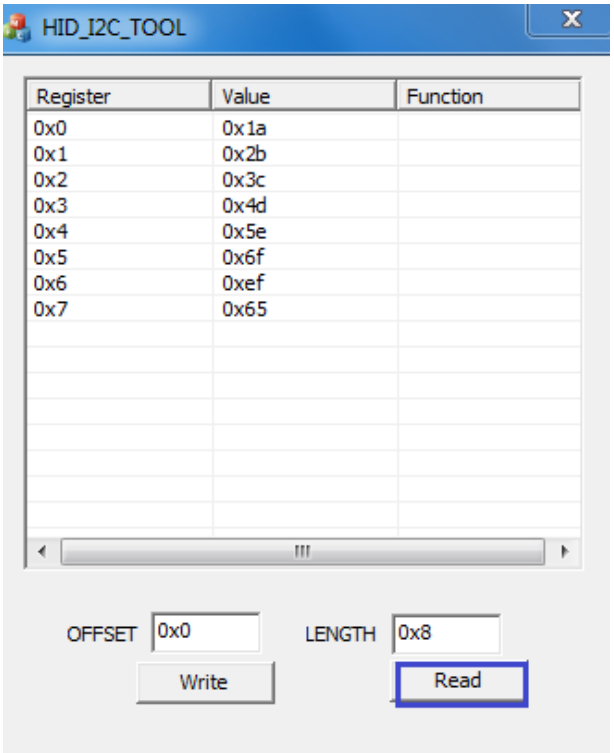

Figure 1-8 Performing I2C Read Data on the HID_I2C_TOOL Interface

Click the Read button and the UART window on the PC will display the data from the EEPROM device. As shown in Figure 1-9:

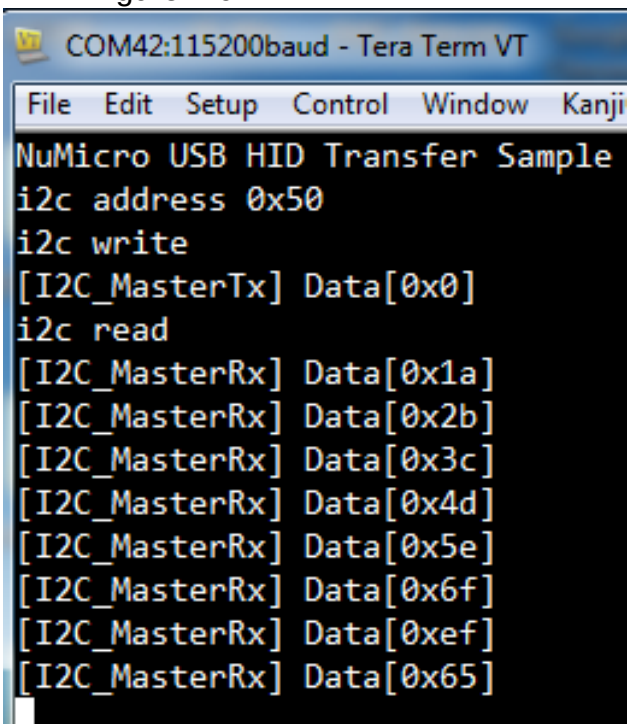

Figure 1-9 Displaying I2C read data on the UART window

2 Code Description

The function below is the initial setting of the EEPROM, which includes setting the SCL and SDA multi-function pin, the I2C transmission frequency, and the enabling of the I2C interrupt:

```
/*---------------------------------------------------------------------------*/
/*  EEPROM Initial Function                                                  */
/*---------------------------------------------------------------------------*/
void I2C_EEPROM_Init(void)
{
    /* Init I2C PIN setting */
    I2C_PIN_Init();

    /* Init I2C */
    I2C_Open(I2C_EEPROM, 100000);

    /* Enable I2C interrupt and set corresponding NVIC bit */
    I2C_EEPROM->I2CON |= I2C_I2CON_EI_Msk;
    NVIC_EnableIRQ(I2C_EEPROM_IRQn);
}
```

The function below is Write I2C EEPROM. The flow is to set the I2C Master TX callback function and start the I2C transmission:

```
/*---------------------------------------------------------------------------*/
/*  Write I2C EEPROM                                                         */
/*---------------------------------------------------------------------------*/
void I2C_EEPROM_Write(void)
{
    g_u8DataLen = 0;
    g_u8EndFlag = 0;

    /* I2C function to write data to slave */
    s_I2CHandlerFn = (I2C_FUNC)I2C_MasterTx;

    /* I2C as master sends START signal */
    I2C_SET_CONTROL_REG(I2C_EEPROM, I2C_I2CON_STA);

    /* Wait I2C Tx Finish */
    while (g_u8EndFlag == 0);
}
```

The function below is Read I2C EEPROM. The flow is to set the I2C Master RX callback function and start the I2C transfer:

```
/*---------------------------------------------------------------------------*/
/*  Read I2C EEPROM                                                          */
/*---------------------------------------------------------------------------*/
uint8_t I2C_EEPROM_Read(void)
{
    g_u8DataLen = 0;
    g_u8EndFlag = 0;

    /* I2C function to write data to slave */
    s_I2CHandlerFn = (I2C_FUNC)I2C_MasterRx;

    /* I2C as master sends START signal */
    I2C_SET_CONTROL_REG(I2C_EEPROM, I2C_I2CON_STA);

    /* Wait I2C Tx Finish */
    while (g_u8EndFlag == 0);

    return g_u8RxData;
}
```

The function below is the main program, the process is to wait for the USB connection, control the transmission with the endpoint 3 of the USB in NUC240. When the transmission interruption is triggered by the endpoint 3, the EEPRON device starts to receive the I2C transmission command (Read, Write) and executes it:

```
/*---------------------------------------------------------------------------*/
/*  Main Function                                                            *
/*---------------------------------------------------------------------------*/
int32_t main(void)
{
    uint8_t *pu8PtrOut, *pu8PtrIn;

    /* initial settings ... */
    /*..................... */


    /* Open USB controller */
    USBD_Open(&gsInfo, HID_ClassRequest, NULL);

    /*Init Endpoint configuration for HID */
    HID_Init();
```

```
/* Start USB device */
USBD_Start();

/* Enable USB device interrupt */
NVIC_EnableIRQ(USBD_IRQn);

while(1)
{
    if(g_u8EP3Ready==1)
    {
        g_u8EP3Ready=0;

        /* Interrupt OUT */
        pu8PtrOut = (uint8_t *)(USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP3));
        pu8PtrIn = (uint8_t *)(USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP2));

        /* Get command form HID and copy to command queue */
        USBD_MemCopy((uint8_t *)&g_tCmd, pu8PtrOut, 64);

        /* Set EEPROM device address */
        if(g_tCmd.u8Command==I2C_SET_ADDRESS)
        {
            g_u8DeviceAddr=g_tCmd.u8Size;
            printf("i2c address 0x%x\n\r",g_tCmd.u8Size);
        }

        /* Write data to EEPROM device */
        if(g_tCmd.u8Command==I2C_WRITE)
        {
            printf("i2c write\n\r");
            I2C_EEPROM_Write();
        }

        /* Read data from EEPROM device */
        if(g_tCmd.u8Command==I2C_READ)
        {
            printf("i2c read\n\r");
            I2C_EEPROM_Read();
        }
        g_tCmd.u8PID+=1;
```

```
            /* Write command to HID */
            USBD_MemCopy(pu8PtrIn, (void *)&g_tCmd, 64);
            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
  - NUC230/240 Series BSP CMSIS v3.01.002
- IDE version
  - Keil uVersion 5.28

Hardware Environment

- Circuit components
  - NuTiny–EVB–NUC240–LQFP100 V1.0
- Diagram
  - Connect the UART TX (PB.1) pin to the PC UART RX to display the execution result of the sample code on the PC.
  - Connect the NUC240 USB to the PC via the mini USB cable.
  - Connect I2C1_SDA and I2C1_SCL of NUC240 to SDA and SCL of EEPROM respectively.

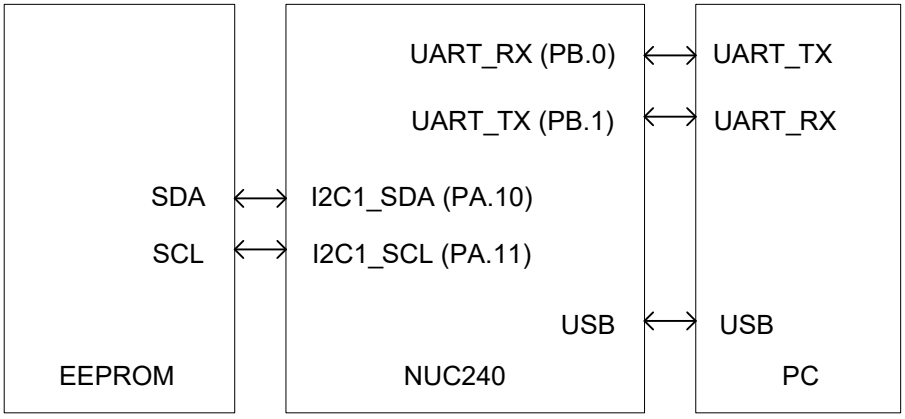

Figure 3-1 Connection of NUC240 and PC, EEPROM

4 Directory Information

- EC_NUC240_USBD_HID_To_EEPROM_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_To_EEPROM.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. 16, 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.