USB 轉 SPI 橋接器

NuMicro® 32位元系列微控制器範例代碼介紹

文件資訊

應用簡述	本範例使用 USB HID 將資料傳輸到 SPI Flash。模擬 USB 轉 SPI 橋接器。
BSP 版本	M480_Series_BSP_CMSIS_V3.05.001
開發平臺	NuMaker-PFM-M487 V3.0

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 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. 概述

本範例主要是模擬 USB 到 SPI 的橋接。 通過 USB 連接 PC 執行應用程式，並使用應用程式向 SPI 設備傳輸資料。本範例使用 BSP 中的 USB HID Transfer 和 SPI Flash 範例來實現此模擬。PC 端應用程式可以通過 USB 讀寫 SPI Flash。

本範例提供了一個簡單的應用程式 HIDTransferTest.exe 來測試 USB 轉 SPI 橋接器。 PC 通過 USB 向 M480 發出擦除/ 讀取/ 寫入命令，然後 SPI Flash 執行相應的擦除/ 讀取/ 寫入動作。示意圖如圖 1-1 所示。

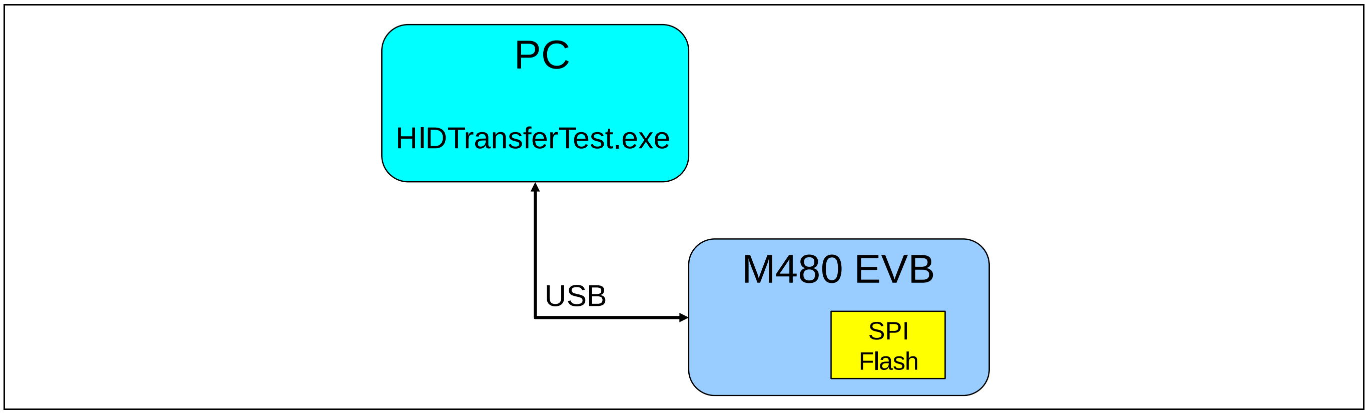
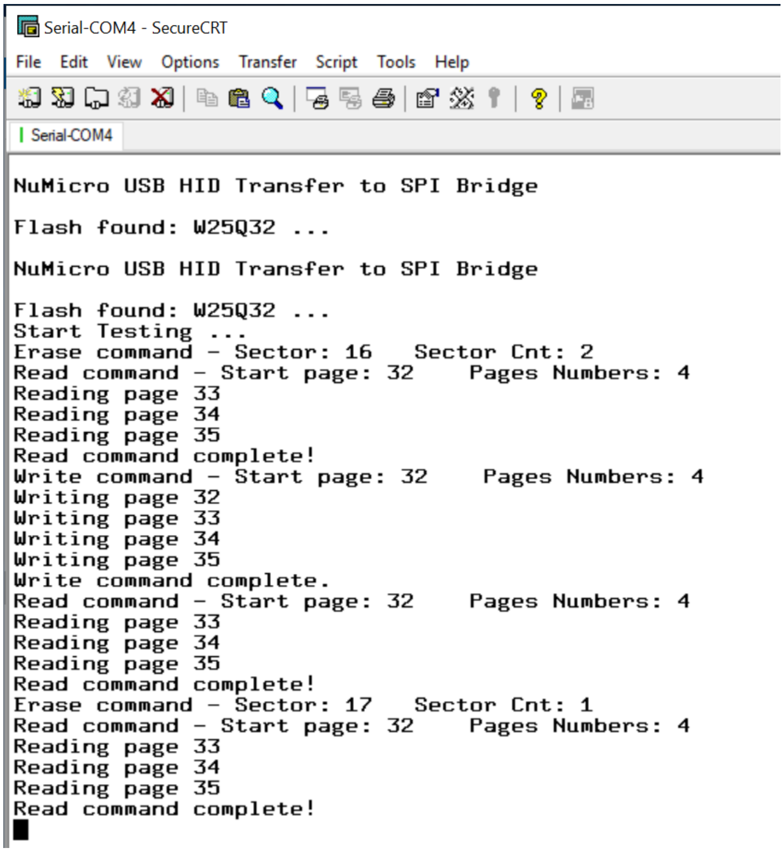


圖 1-1 PC 與 M480 EVB 的關係

1.1 執行結果

這個 SPI Flash 的磁區大小為 4096 位元組，而且頁的大小為 2048 位元組。HIDTransferTest.exe 會擦除兩個磁區，讀回擦除的資料並檢查是否為 0xFF；寫入四頁，讀回寫入資料並檢查是否正確；擦除一個磁區，讀回四頁數據並檢查是否正確。測試結果如下。



```
NuMicro USB HID Transfer to SPI Bridge

Flash found: W25Q32 ...

NuMicro USB HID Transfer to SPI Bridge

Flash found: W25Q32 ...
Start Testing ...
Erase command - Sector: 16    Sector Cnt: 2
Read command - Start page: 32     Pages Numbers: 4
Reading page 33
Reading page 34
Reading page 35
Read command complete!
Write command - Start page: 32     Pages Numbers: 4
Writing page 32
Writing page 33
Writing page 34
Writing page 35
Write command complete.
Read command - Start page: 32     Pages Numbers: 4
Reading page 33
Reading page 34
Reading page 35
Read command complete!
Erase command - Sector: 17    Sector Cnt: 1
Read command - Start page: 32     Pages Numbers: 4
Reading page 33
Reading page 34
Reading page 35
Read command complete!
```

圖 1-2 執行結果

2. 代碼介紹

SPI 相關函數。它包括讀取、寫入、擦除、讀取狀態和等待就緒功能。代碼位於 main.c 中。

```
uint8_t SpiFlash_ReadStatusReg(void)
{
    // /CS: active
    QSPI_SET_SS_LOW(QSPI_FLASH_PORT);

    // send Command: 0x05, Read status register
    QSPI_WRITE_TX(QSPI_FLASH_PORT, 0x05);

    // read status
    QSPI_WRITE_TX(QSPI_FLASH_PORT, 0x00);

    // wait tx finish
    while(QSPI_IS_BUSY(QSPI_FLASH_PORT));

    // /CS: de-active
    QSPI_SET_SS_HIGH(QSPI_FLASH_PORT);

    // skip first rx data
    QSPI_READ_RX(QSPI_FLASH_PORT);

    return (QSPI_READ_RX(QSPI_FLASH_PORT) & 0xff);
}

void SpiFlash_WaitReady(void)
{
    uint8_t volatile ReturnValue;

    do
    {
        ReturnValue = SpiFlash_ReadStatusReg();
        ReturnValue = ReturnValue & 1;
    }
    while(ReturnValue!=0);   // check the BUSY bit
}

void SpiFlash_NormalRead(uint32_t StartAddress, uint8_t *u8DataBuffer)
{
    uint32_t i;

    // /CS: active
    QSPI_SET_SS_LOW(QSPI_FLASH_PORT);

    // send Command: 0x03, Read data
    QSPI_WRITE_TX(QSPI_FLASH_PORT, 0x03);

    // send 24-bit start address
    QSPI_WRITE_TX(QSPI_FLASH_PORT, (StartAddress>>16) & 0xFF);
    QSPI_WRITE_TX(QSPI_FLASH_PORT, (StartAddress>>8)  & 0xFF);
    QSPI_WRITE_TX(QSPI_FLASH_PORT, StartAddress       & 0xFF);
    while(QSPI_IS_BUSY(QSPI_FLASH_PORT));
    // clear RX buffer
    QSPI_ClearRxFIFO(QSPI_FLASH_PORT);

    // read data
    for(i=0; i<256; i++)
    {
        QSPI_WRITE_TX(QSPI_FLASH_PORT, 0x00);
        while(QSPI_IS_BUSY(QSPI_FLASH_PORT));
        u8DataBuffer[i] = QSPI_READ_RX(QSPI_FLASH_PORT);
    }

    // wait tx finish
    while(QSPI_IS_BUSY(QSPI_FLASH_PORT));

    // /CS: de-active
    QSPI_SET_SS_HIGH(QSPI_FLASH_PORT);
}

void SpiFlash_NormalPageProgram(uint32_t StartAddress, uint8_t *u8DataBuffer)
{
    uint32_t i = 0;

    // /CS: active
    QSPI_SET_SS_LOW(QSPI_FLASH_PORT);

    // send Command: 0x06, Write enable
    QSPI_WRITE_TX(QSPI_FLASH_PORT, 0x06);

    // wait tx finish
    while(QSPI_IS_BUSY(QSPI_FLASH_PORT));

    // /CS: de-active
    QSPI_SET_SS_HIGH(QSPI_FLASH_PORT);

    // /CS: active
    QSPI_SET_SS_LOW(QSPI_FLASH_PORT);

    // send Command: 0x02, Page program
    QSPI_WRITE_TX(QSPI_FLASH_PORT, 0x02);

    // send 24-bit start address
    QSPI_WRITE_TX(QSPI_FLASH_PORT, (StartAddress>>16) & 0xFF);
    QSPI_WRITE_TX(QSPI_FLASH_PORT, (StartAddress>>8)  & 0xFF);
    QSPI_WRITE_TX(QSPI_FLASH_PORT, StartAddress       & 0xFF);

    // write data
    while(1)
    {
        if(!QSPI_GET_TX_FIFO_FULL_FLAG(QSPI_FLASH_PORT))
        {
            QSPI_WRITE_TX(QSPI_FLASH_PORT, u8DataBuffer[i++]);
            if(i > 255) break;
        }
    }
    // wait tx finish
    while(QSPI_IS_BUSY(QSPI_FLASH_PORT));

    // /CS: de-active
    QSPI_SET_SS_HIGH(QSPI_FLASH_PORT);

    QSPI_ClearRxFIFO(QSPI_FLASH_PORT);
}

/* one sector is 4KB */
void SpiFlash_SecotrErase(uint32_t StartSector, uint32_t EraseCount)
{
    uint32_t volatile i, offset;

    for (i=0; i<EraseCount; i++)
    {
        offset = (StartSector + i) * 0x1000;
        // /CS: active
        QSPI_SET_SS_LOW(QSPI_FLASH_PORT);

        // send Command: 0x06, Write enable
        QSPI_WRITE_TX(QSPI_FLASH_PORT, 0x06);

        // wait tx finish
        while(QSPI_IS_BUSY(QSPI_FLASH_PORT));

        // /CS: de-active
        QSPI_SET_SS_HIGH(QSPI_FLASH_PORT);

        // /CS: active
        QSPI_SET_SS_LOW(QSPI_FLASH_PORT);

        // send Command: 0x20, sector erase
        QSPI_WRITE_TX(QSPI_FLASH_PORT, 0x20);

        // send 24-bit start address
        QSPI_WRITE_TX(QSPI_FLASH_PORT, (offset>>16) & 0xFF);
        QSPI_WRITE_TX(QSPI_FLASH_PORT, (offset>>8)  & 0xFF);
        QSPI_WRITE_TX(QSPI_FLASH_PORT, offset       & 0xFF);

        // wait tx finish
        while(QSPI_IS_BUSY(QSPI_FLASH_PORT));

        // /CS: de-active
        QSPI_SET_SS_HIGH(QSPI_FLASH_PORT);

        QSPI_ClearRxFIFO(QSPI_FLASH_PORT);
    }
}
```

初始化系統後，首先配置 QSPI，然後設置 USB 設備。 代碼位於 main.c 中。

```
int32_t main(void)
{
    uint32_t u32TrimInit;
    uint16_t u16ID;

    /* Unlock protected registers */
    SYS_UnlockReg();

    SYS_Init();

    /* Configure UART0 and set UART0 Baudrate */
    UART_Open(UART0, 115200);

    /* Configure QSPI_FLASH_PORT as a master, MSB first, 8-bit transaction, QSPI Mode-0
timing, clock is 2MHz */
    QSPI_Open(QSPI_FLASH_PORT, QSPI_MASTER, QSPI_MODE_0, 8, 2000000);

    /* Enable the automatic hardware slave select function. Select the SS pin and
configure as low-active. */
    QSPI_EnableAutoSS(QSPI_FLASH_PORT, QSPI_SS, QSPI_SS_ACTIVE_LOW);

    printf("\nNuMicro USB HID Transfer to SPI Bridge\n\n");

    /* Read SPI Flash ID : W25Q32(0xEF15), W25Q16(0xEF14) */
    if((u16ID = SpiFlash_ReadMidDid()) != 0xEF15)
    {
        printf("Wrong ID, 0x%x\n", u16ID);
        while(1);
    }
    else
        printf("Flash found: W25Q32 ...\n");

    USBD_Open(&gsInfo, HID_ClassRequest, NULL);

    printf("Start Testing ...\n");
    /* Endpoint configuration */
    HID_Init();
    USBD_Start();

    ………

    NVIC_EnableIRQ(USBD_IRQn);

    while(1)
    {
    ………
    }
}
```

定義 USB HID 傳輸的命令和結構。代碼位於 *hid_to_spi_bridge.c* 中。

```
/* HID Transfer Commands */
#define HID_CMD_NONE     0x00
#define HID_CMD_ERASE    0x71
#define HID_CMD_READ     0xD2
#define HID_CMD_WRITE    0xC3
#define HID_CMD_TEST     0xB4

#define PAGE_SIZE        2048
#define TEST_PAGES       4
#define SECTOR_SIZE      4096

typedef struct __attribute__((__packed__))
{
    uint8_t u8Cmd;
    uint8_t u8Size;
    uint32_t u32Arg1;
    uint32_t u32Arg2;
    uint32_t u32Signature;
    uint32_t u32Checksum;
}
CMD_T;
```

定義 USB HID Transfer 對應的函數。代碼位於 hid_to_spi_bridge.c 中。

```
int32_t HID_CmdEraseSectors(CMD_T *pCmd)
{
    uint32_t u32StartSector;
    uint32_t u32Sectors;

    u32StartSector = pCmd->u32Arg1;
    u32Sectors = pCmd->u32Arg2;

    printf("Erase command - Sector: %d   Sector Cnt: %d\n", u32StartSector, u32Sectors);

    /* To erase the sector of storage */
    SpiFlash_SecotrErase(u32StartSector, u32Sectors);
    SpiFlash_WaitReady();

    /* To note the command has been done */
    pCmd->u8Cmd = HID_CMD_NONE;

    return 0;
}

int32_t HID_CmdReadPages(CMD_T *pCmd)
{
    uint32_t u32StartPage;
    uint32_t u32Pages;
    uint32_t volatile i, offset;

    u32StartPage = pCmd->u32Arg1;
    u32Pages     = pCmd->u32Arg2;
    printf("Read command - Start page: %d    Pages Numbers: %d\n", u32StartPage, 
u32Pages);

    if(u32Pages)
    {
        /* Update data to page buffer to upload */
        offset = u32StartPage * PAGE_SIZE;
        for (i=0; i<PAGE_SIZE/256; i++)
            SpiFlash_NormalRead(offset+i*256, g_u8PageBuff+i*256);
        g_u32BytesInPageBuf = PAGE_SIZE;

        /* The signature word is used as page counter */
        pCmd->u32Signature = 1;

        /* Trigger HID IN */
        USBD_MemCopy((uint8_t *)(USBD_BUF_BASE + USBD_GET_EP_BUF_ADDR(EP2)), (void 
*)g_u8PageBuff, EP2_MAX_PKT_SIZE);
        USBD_SET_PAYLOAD_LEN(EP2, EP2_MAX_PKT_SIZE);
        g_u32BytesInPageBuf -= EP2_MAX_PKT_SIZE;
    }

    return 0;
}

int32_t HID_CmdWritePages(CMD_T *pCmd)
{
    uint32_t u32StartPage;
    uint32_t u32Pages;

    u32StartPage = pCmd->u32Arg1;
    u32Pages     = pCmd->u32Arg2;

    printf("Write command - Start page: %d    Pages Numbers: %d\n", u32StartPage, 
u32Pages);
    g_u32BytesInPageBuf = 0;

    /* The signature is used to page counter */
    pCmd->u32Signature = 0;

    return 0;
}
```

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - M480_Series_BSP_CMSIS_V3.05.001
- IDE 版本
  - Keil uVersion 5.28

3.2 硬體需求

- 電路元件
  - NuMaker-PFM-M487 V3.0
- 線路示意圖
  - 將 UART0 TX (PB.13) pin 腳連接到 PC UART RX，以顯示範例代碼的執行結果。
  - 將 USB 連接到 PC。

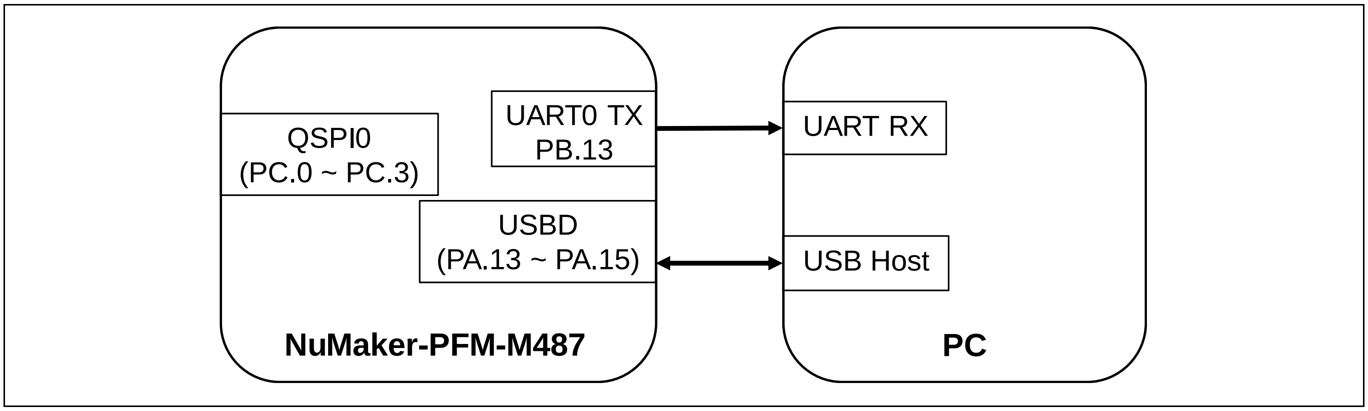


圖 3-1 線路示意圖

4. 目錄資訊

🗁 EC_M480_USBD_HID_to_SPI_Bridge_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

圖 4-1 目錄資訊

5. 範例程式執行

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，雙擊 *USBD_HID_to_SPI_bridge.uvproj*。
2. 進入編譯模式介面
   - 編譯
   - 下載代碼至記憶體
   - 進入 / 離開除錯模式
3. 進入除錯模式介面
   - 執行代碼

6. 修訂紀錄

Date	Revision	Description
2023.08.16	1.00	初始發佈。

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.