DFU Dual Interface ISP

Example Code Introduction for 32-bit NuMicro® Family

Document Information

Application	This example code uses M252 USBD to implement a dual-interface DFU device to update APROM and SPI Flash.
BSP Version	M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.008
Hardware	NuMaker-M252KG V1.4 and external SPI Flash

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

This example code demonstrates a USB DFU (Device Firmware Upgrade) ISP implementation executed on the internal LDROM of the NuMicro M252 series microcontroller. It enables firmware update for both the internal APROM and external SPI Flash memory through a standard USB DFU interface.

This example code requires additional hardware and software modules:

- Standard SPI Flash with four signals: /SS, CLK, MOSI and MISO.

  e.g. Winbond W25Q32 SPI Flash.
- DFU utility for firmware download/upload operations.

  https://dfu-util.sourceforge.net/
- Windows driver provides USB drivers for DFU device recognition.

1.1 Principle

USB DFU is a USB standard that lets users update device firmware over a USB connection. This example code supports dual-interface with two storage backends (internal APROM and external SPI Flash), while staying compatible with standard DFU tools like dfu-util.

This example code boots from the internal LDROM and can then be used with dfu-util to download (write) or upload (read) data from APROM and SPI Flash. To meet the LDROM size (4 KB) constraints, this example only implements the minimal USB DFU functions needed to operate as a DFU-compatible device and does not fully implement the entire protocol.

Figure 1-1 shows a simplified block diagram of the example code and detailed pin connections are described in Section 3.2.

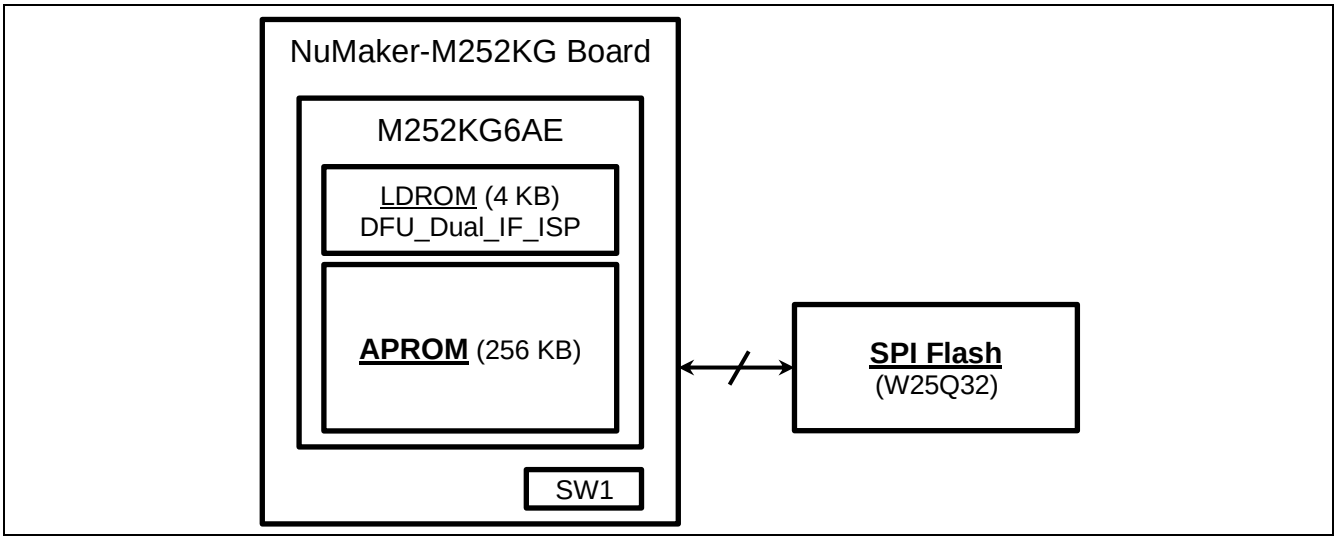


Figure 1-1 Block Diagram

1.2 Demo Result

Check Figure 3-1 to connect **SW1** to VCC and power on to boot into DFU ISP mode. Then connect the USB device port of NuMaker-M252KG to the Windows PC's USB host port to establish a DFU connection.

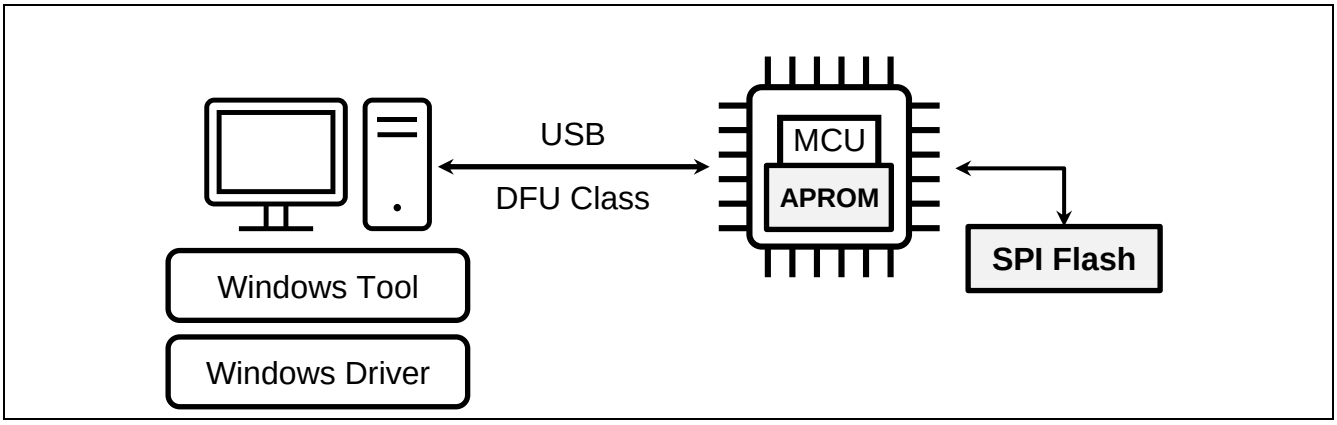


Figure 1-2 USB DFU Connection

If this is the first time to connect a Nuvoton DFU device to this PC, Windows may not recognize it and will mark it as an unknown device. Please follow Section 1.2.1 to install Nuvoton DFU driver before using the DFU device.

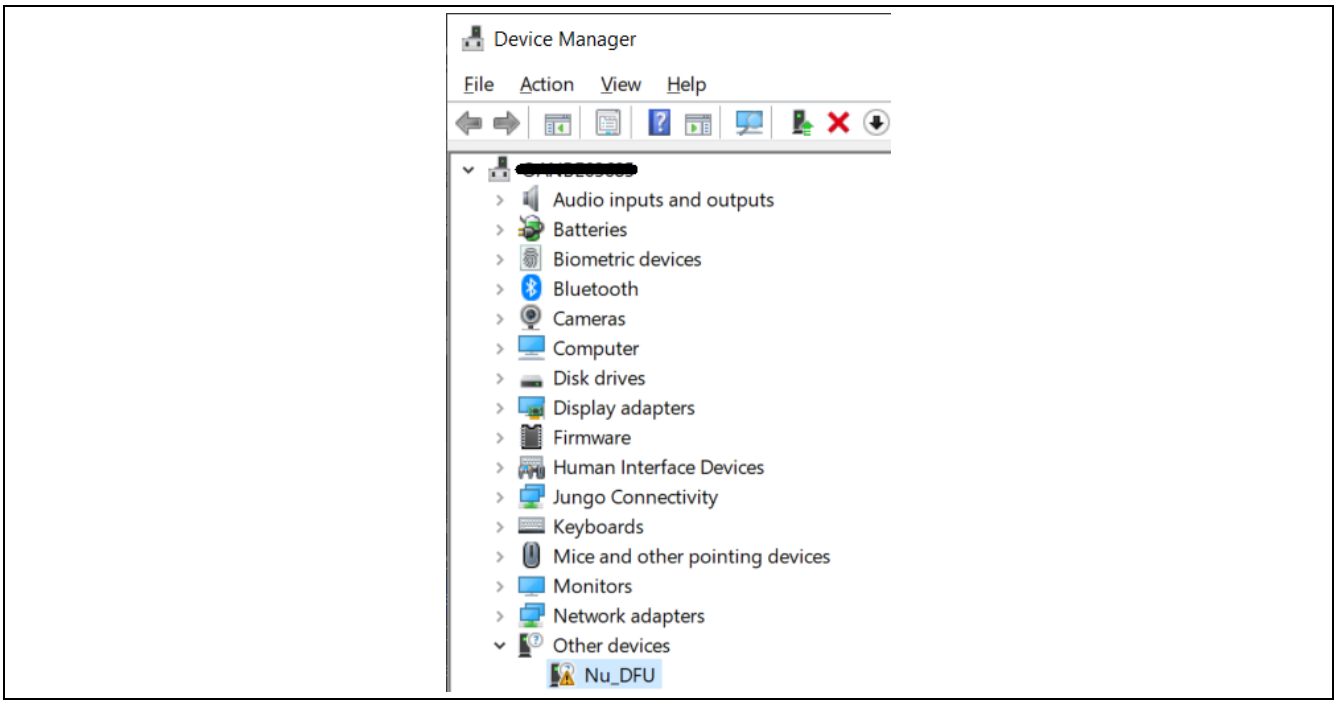
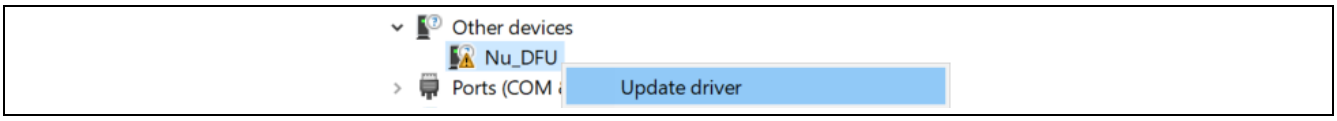


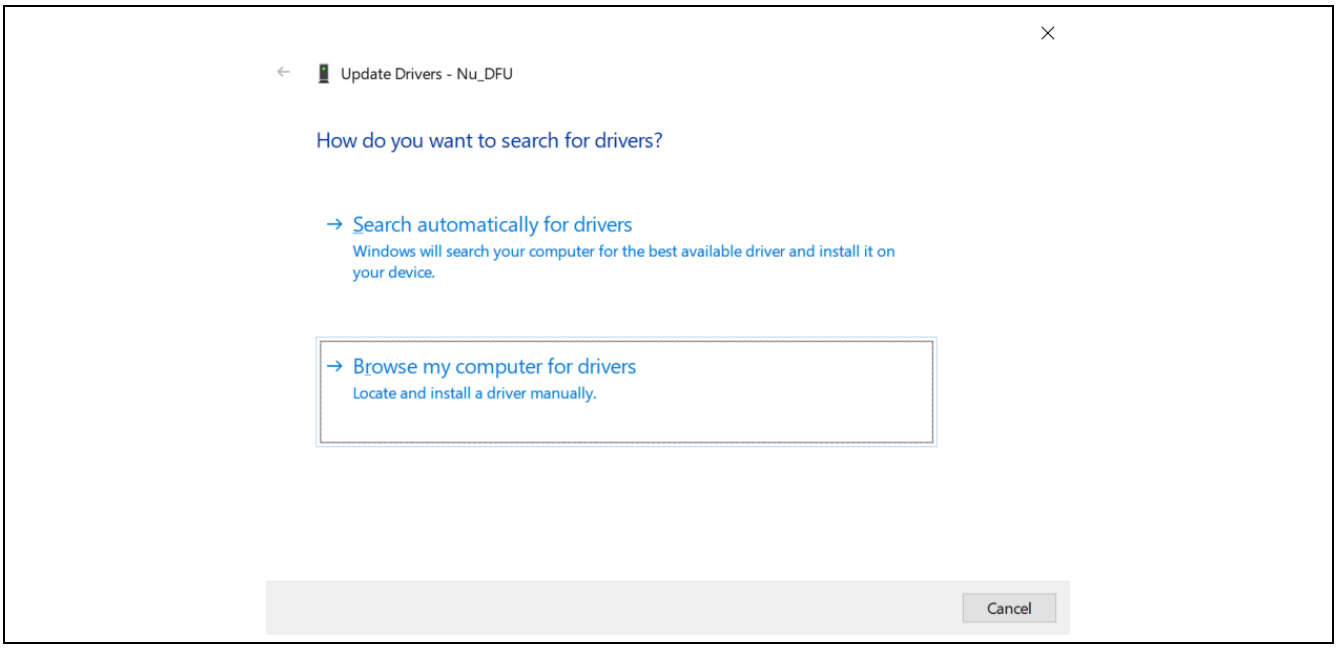
Figure 1-3 Unknown Device in Device Manager

1.2.1 Install Nuvoton DFU Driver

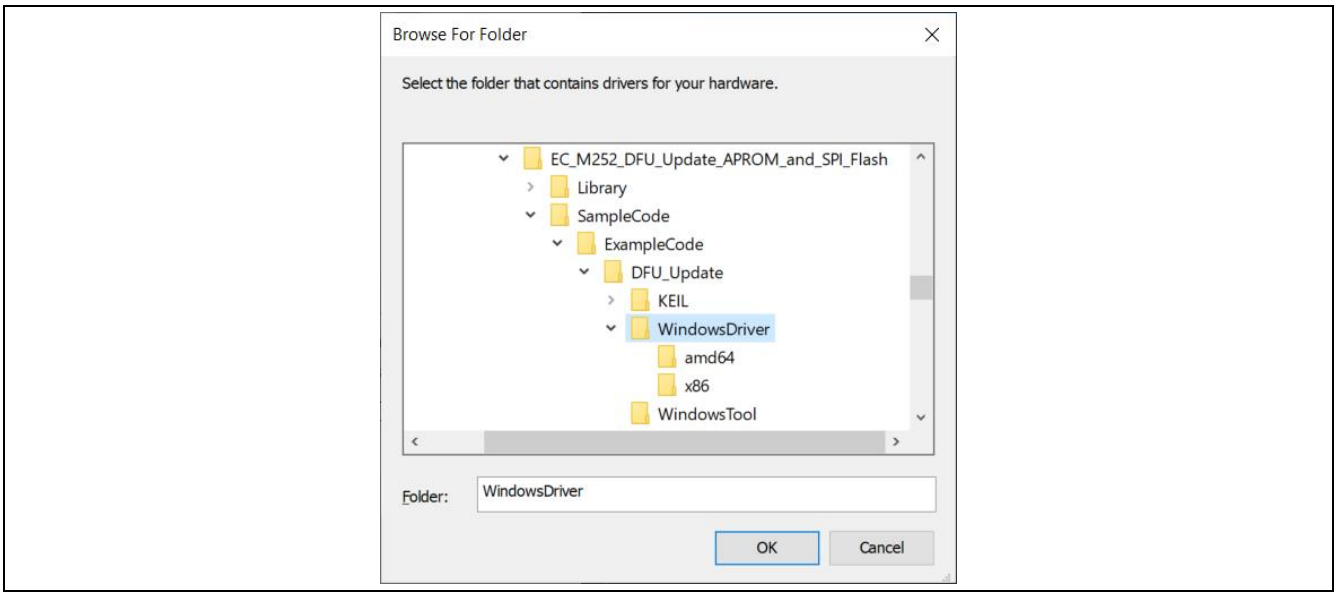
1. Right click on Nu_DFU and select “**Update driver**”.



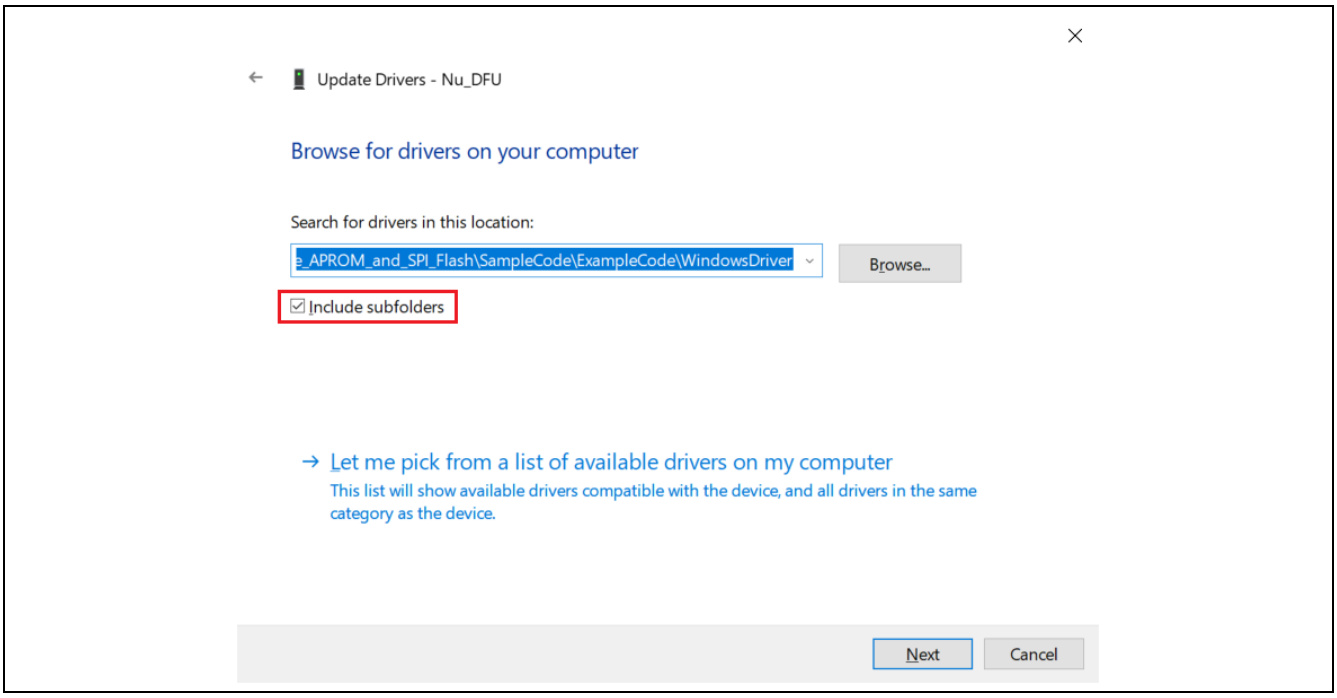
2. Select “Browse my computer for drivers”.



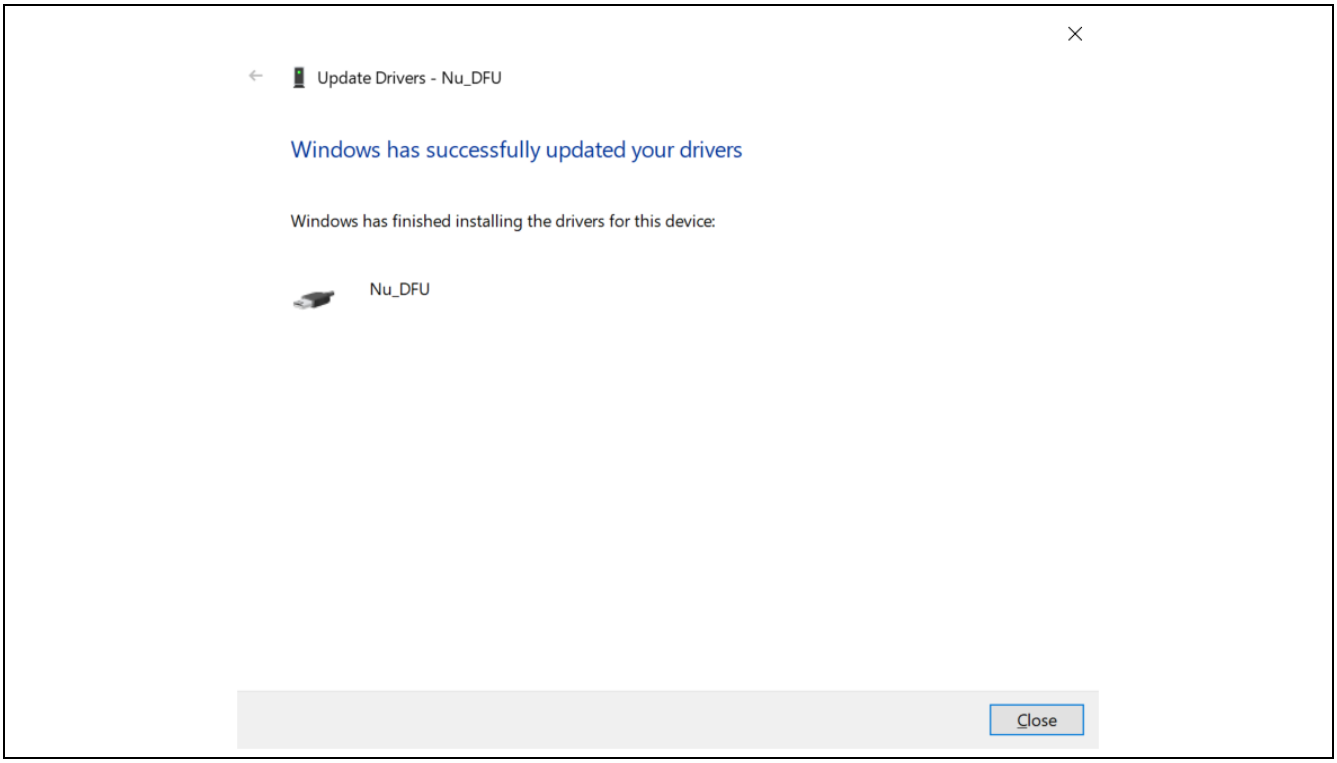
3. Check the Directory Information to select **WindowsDriver** folder and click “**OK**”.



4. Check “**Include subfolders**” and click “**Next**”.



5. Check if driver is installed successfully and click “**Close**” to finish installation.



6. Check if **Nu_DFU** is installed successfully in Windows Device Manager.

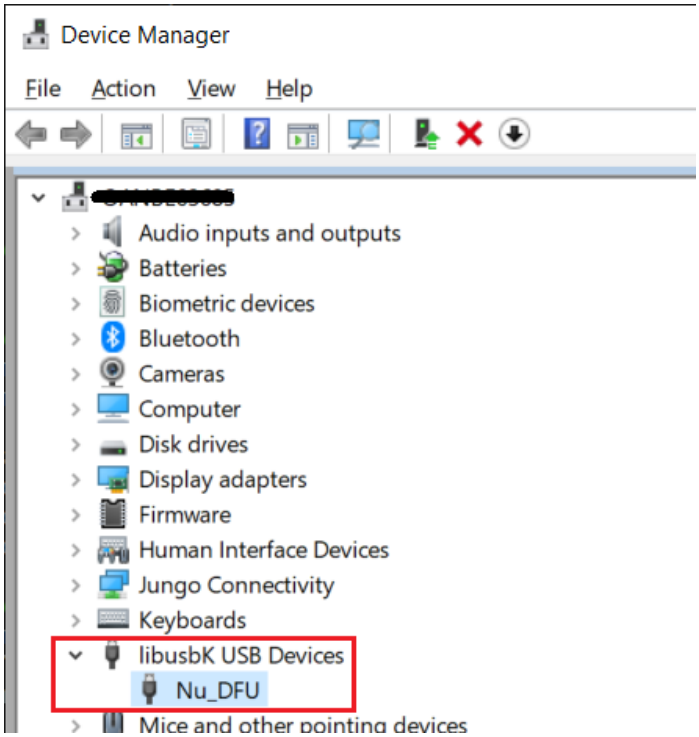
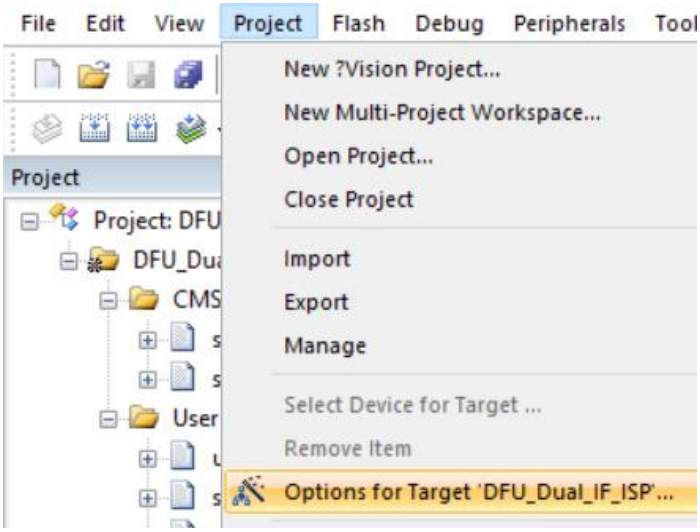


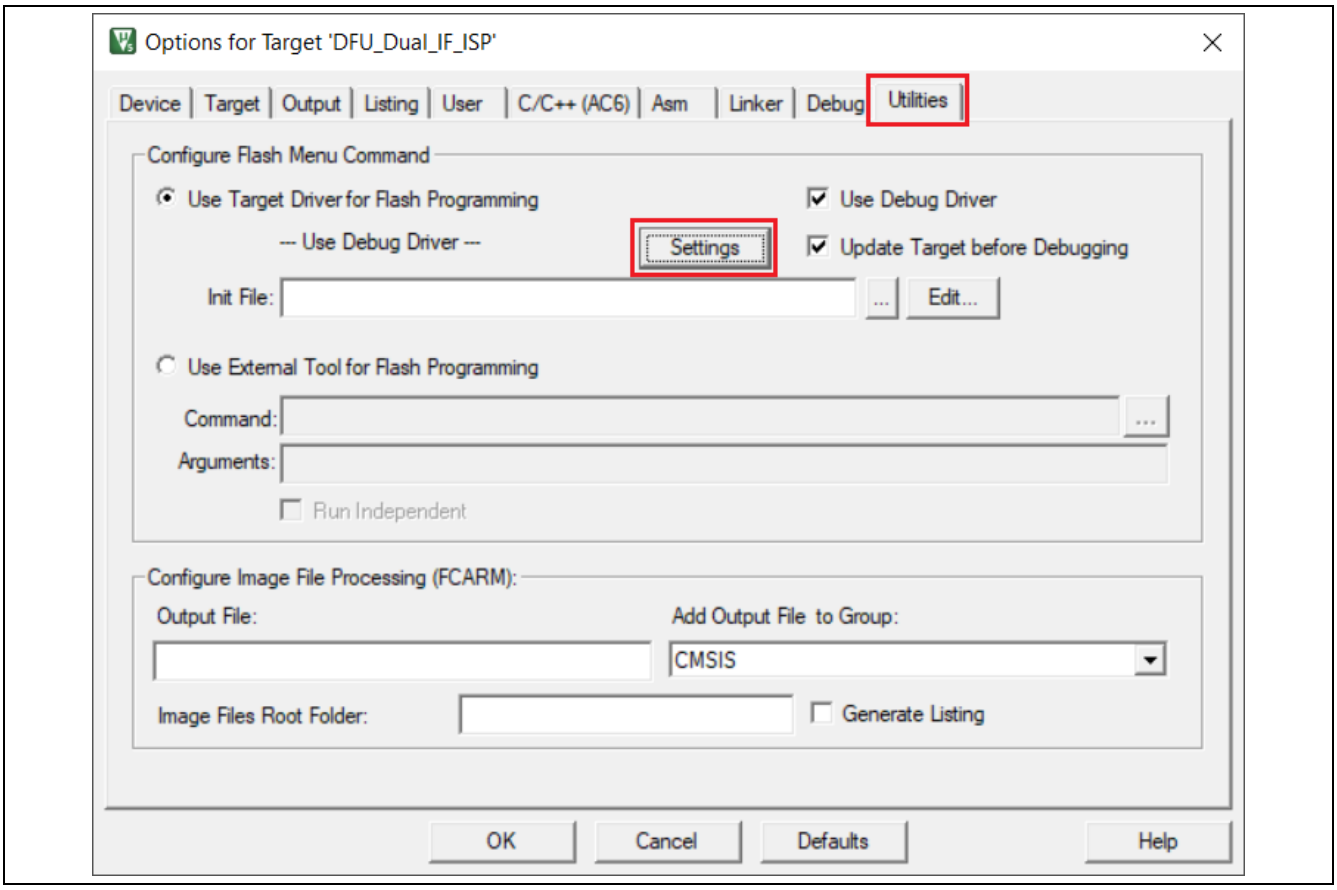
Figure 1-4 DFU Device in Device Manager

1.2.2 Configure Boot from LDROM

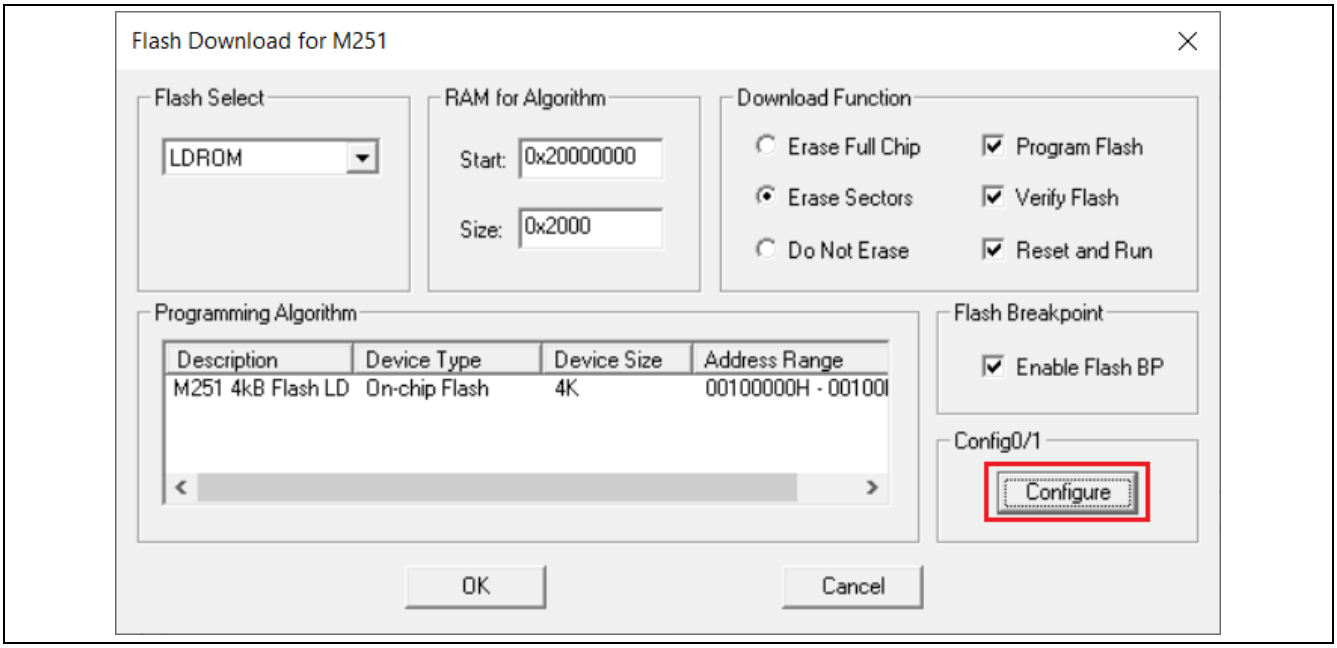
1. Go to the "**Project**" menu and select "**Options for Target**".



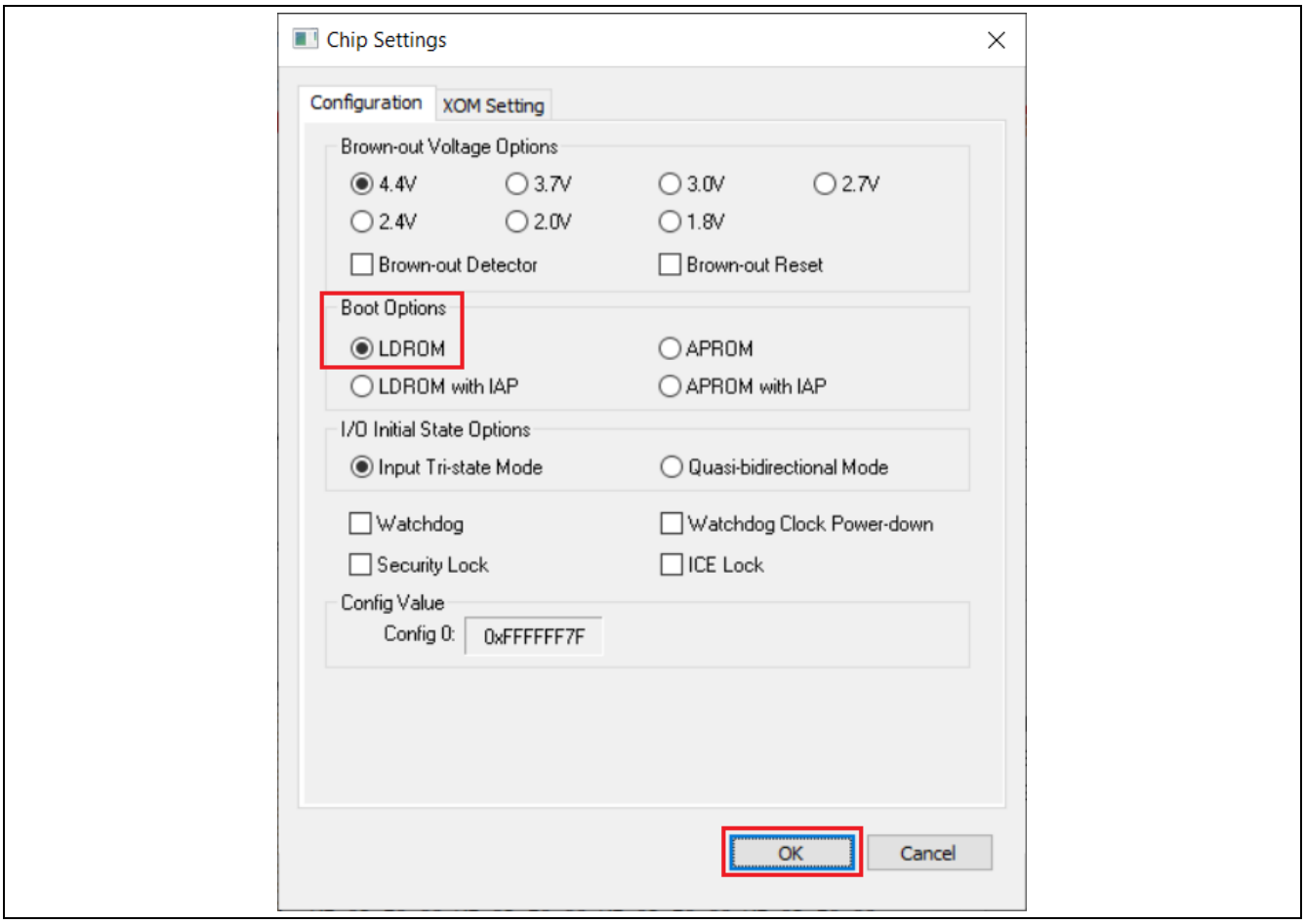
2. Navigate to “**Utilities**” tab and click “**Settings**” button.



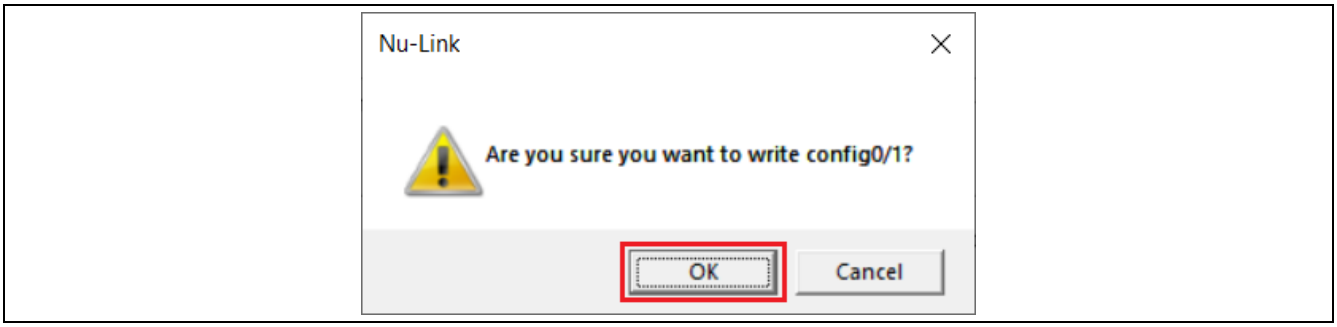
3. Click “**Configure**” button to open “**Chip Settings**” dialog.



4. Select “**LDROM**” under Boot Options and click “**OK**.”



5. Click “**OK**” to write configuration.



6. Build and download code to LDROM.

1.2.3 DFU Utility

dfu-util is an open-source host-side tool for the USB DFU protocol. It can list DFU-capable devices, select targets (alternate settings), and download/upload firmware images to various storage regions on embedded devices.

Check the Directory Information section to get the *dfu-util.exe* under WindowsTool folder.

1.2.3.1 List DFU Devices and Targets

Use “dfu-util.exe -l” to list all available devices.

Check if the DFU device is ready with this command before downloading and uploading firmware image.

```
.\dfu-util.exe -l
dfu-util 0.11

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2021 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to http://sourceforge.net/p/dfu-util/tickets/

Found DFU: [0416:bdf0] ver=0100, devnum=11, cfg=1, intf=0, path="1-1.1.1.3", alt=1,
name="@SPI Flash/0x0/1024*4Ke", serial="NVT_DFU"
Found DFU: [0416:bdf0] ver=0100, devnum=11, cfg=1, intf=0, path="1-1.1.1.3", alt=0,
name="@APROM/0x0/512*512Be", serial="NVT_DFU"
```

The device provides two DFU interfaces through alternate settings (alt=0 and alt=1).

	APROM	**SPI Flash**
	name="@APROM/ 0x0/512*512Be"	name="@SPI Flash/ 0x0/1024*4Ke"
Alternate Setting	0	1
Base Address	0x0	0x0
Layout	512 x 512 B	1024 x 4 KB
Program/Erase Unit	512 B	4 KB
Capacity	256 KB	4 MB

1.2.3.2 APROM Download and Upload

Use “dfu-util.exe -D” to download binary data to the specified storage.

Below command uses “-a 0” to select APROM and “-D .\aprom.bin” to download *aprom.bin* to the device via DFU. The test image *aprom.bin* is in the WindowsTool folder and can be replaced with your own image as needed.

“-d 0x0416:0xBDF0” is used to specify vendor/product ID(s) of the dedicated DFU device. If there is only one DFU device connected to the PC, -d option can be omitted.

```
.\dfu-util.exe -d 0x0416:0xBDF0 -a 0 -D .\aprom.bin
dfu-util 0.11

...

Opening DFU capable USB device...
Device ID 0416:bdf0
Device DFU version 0110
Claiming USB DFU Interface...
```

```
Setting Alternate Interface #0 ...
Determining device status...
DFU state(2) = dfuIDLE, status(0) = No error condition is present
DFU mode device DFU version 0110
Device returned transfer size 1024
Copying data from PC to DFU device
Download        [=========================] 100%       262144 bytes
Download done.
DFU state(2) = dfuIDLE, status(0) = No error condition is present
Done!
```

Use “dfu-util.exe -U” to upload binary data from the specified storage.

Below command uses “-a 0” to select APROM and “-U .\read_aprom.bin” to upload the full APROM contents from the device via DFU and save them to *read_aprom.bin*.

```
.\dfu-util.exe -d 0x0416:0xBDF0 -a 0 -U .\read_aprom.bin
dfu-util 0.11

...

Opening DFU capable USB device...
Device ID 0416:bdf0
Device DFU version 0110
Claiming USB DFU Interface...
Setting Alternate Interface #0 ...
Determining device status...
DFU state(2) = dfuIDLE, status(0) = No error condition is present
DFU mode device DFU version 0110
Device returned transfer size 1024
Copying data from DFU device to PC
Upload  [=========================] 100%       262144 bytes
Upload done.
Received a total of 262144 bytes
```

1.2.3.3 SPI Flash Download and Upload

Below command uses “-a 1” to select SPI Flash and “-D .\spi.bin” to download *spi.bin* to the device via DFU. The test image *spi.bin* is in the WindowsTool folder and can be replaced with your own image as needed.

```
.\dfu-util.exe -d 0x0416:0xBDF0 -a 1 -D .\spi.bin
dfu-util 0.11

...

Opening DFU capable USB device...
Device ID 0416:bdf0
Device DFU version 0110
Claiming USB DFU Interface...
Setting Alternate Interface #1 ...
Determining device status...
DFU state(2) = dfuIDLE, status(0) = No error condition is present
DFU mode device DFU version 0110
Device returned transfer size 1024
```

```
Copying data from PC to DFU device
Download        [=========================] 100%      4194304 bytes
Download done.
DFU state(2) = dfuIDLE, status(0) = No error condition is present
Done!
```

Below command uses “-a 1” to select SPI Flash and “-U .\read_spi.bin” to upload the full SPI Flash contents from the device via DFU and save them to *read_spi.bin*.

```
.\dfu-util.exe -d 0x0416:0xBDF0 -a 1 -U .\read_spi.bin
dfu-util 0.11

...

Opening DFU capable USB device...
Device ID 0416:bdf0
Device DFU version 0110
Claiming USB DFU Interface...
Setting Alternate Interface #1 ...
Determining device status...
DFU state(2) = dfuIDLE, status(0) = No error condition is present
DFU mode device DFU version 0110
Device returned transfer size 1024
Copying data from DFU device to PC
Upload  [=========================] 100%      4194304 bytes
Upload done.
Received a total of 4194304 bytes
```

1.2.3.4 Partial Download and Upload

Use “dfu-util.exe -s offset:size” to specify start offset and data size in bytes.

Below command downloads *spi_1024KB.bin* to SPI Flash at offset 0x10000; omitting the size means the whole binary is downloaded.

```
.\dfu-util.exe -d 0x0416:0xBDF0 -a 1 -s 0x10000 -D .\spi_1024KB.bin
dfu-util 0.11

...

Opening DFU capable USB device...
Device ID 0416:bdf0
Device DFU version 0110
Claiming USB DFU Interface...
Setting Alternate Interface #1 ...
Determining device status...
DFU state(2) = dfuIDLE, status(0) = No error condition is present
DFU mode device DFU version 0110
Warning: DfuSe option used on non-DfuSe device
Device returned transfer size 1024
DfuSe interface name: "SPI Flash"
Downloading element to address = 0x00010000, size = 1048576
Erase           [=========================] 100%      1048576 bytes
Erase    done.
Download        [=========================] 100%      1048576 bytes
```

```
Download done.
File downloaded successfully
```

Below command uploads a partial data block from the SPI Flash; upload 0x100000 bytes starting at offset 0x10000 and save data to *read_spi_1024KB.bin*.

```
.\dfu-util.exe -d 0x0416:0xBDF0 -a 1 -s 0x10000:0x100000 -U .\read_spi_1024KB.bin
dfu-util 0.11

...

Opening DFU capable USB device...
Device ID 0416:bdf0
Device DFU version 0110
Claiming USB DFU Interface...
Setting Alternate Interface #1 ...
Determining device status...
DFU state(2) = dfuIDLE, status(0) = No error condition is present
DFU mode device DFU version 0110
Warning: DfuSe option used on non-DfuSe device
Device returned transfer size 1024
DfuSe interface name: "SPI Flash"
Upload  [=========================] 100%      1048576 bytes
Upload done.
```

1.2.3.5 Download Code to APROM and Execute

To jump to execute code on APROM, please follow Section 1.2.2 to configure boot from LDROM.

The WindowsTool folder contains the test image *Dump_SPI_Flash.bin* to dump data from the SPI Flash. The Dump_SPI_Flash project can be found in the ExampleCode folder.

Use below command to download *Dump_SPI_Flash.bin* to APROM.

```
.\dfu-util.exe -d 0x0416:0xBDF0 -a 0 -D .\Dump_SPI_Flash.bin
```

After the download is finished, connect **SW1** to GND to exit DFU mode. Then system will jump to execute the code on the APROM and dump raw data on the SPI Flash through UART console.

```
+----------------------------+
|        Dump SPI Flash      |
+----------------------------+
Flash found: W25Q32 ...

[0x00000000] 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A
[0x00000020] 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A
[0x00000040] 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A
[0x00000060] 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A
[0x00000080] 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A
[0x000000A0] 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A
[0x000000C0] 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A
[0x000000E0] 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A 0x5A5A5A5A
Continue ? q: Quit, p: Jump to next 64 KB, others: Continue
```

Figure 1-5 Raw Data Dump of SPI Flash

2. Code Description

Configure the USBD and SPI0 module clock and multi-function pin settings in SYS_Init() in *main.c.*

```
void SYS_Init(void)
{
    /*---------------------------------------------------------------------------------------*/
    /* Init System Clock                                                                     */
    /*---------------------------------------------------------------------------------------*/
    /* Enable Internal HIRC 48 MHz clock */
    CLK->PWRCTL |= CLK_PWRCTL_HIRCEN_Msk;

    /* Waiting for Internal RC clock ready */
    while (!(CLK->STATUS & CLK_STATUS_HIRCSTB_Msk));

    /* Switch HCLK clock source to HIRC and HCLK source divided by (HCLK_DIV + 1),
       and USB clock source is from HIRC and divided by (USBD_DIV + 1) */
    CLK->CLKSEL0 = (CLK->CLKSEL0 & ~CLK_CLKSEL0_HCLKSEL_Msk) | CLK_CLKSEL0_HCLKSEL_HIRC;
    CLK->CLKDIV0 = (CLK->CLKDIV0 & ~CLK_CLKDIV0_HCLKDIV_Msk) | CLK_CLKDIV0_HCLK(HCLK_DIV);
    CLK->CLKDIV0 = (CLK->CLKDIV0 & ~CLK_CLKDIV0_USBDIV_Msk)  | CLK_CLKDIV0_USB(USBD_DIV);
```

This example uses PCLK1 as SPI clock source and PA0 as MOSI, PA1 as MISO, PA2 as CLK and PA3 as /SS to connect the SPI Flash.

```
    /* Select PCLK1 as the clock source of SPI0 */
    CLK->CLKSEL2 = (CLK->CLKSEL2 & ~CLK_CLKSEL2_SPI0SEL_Msk) | CLK_CLKSEL2_SPI0SEL_PCLK1;

    /* Enable module clock */
    CLK->APBCLK0 |= CLK_APBCLK0_USBDCKEN_Msk | CLK_APBCLK0_SPI0CKEN_Msk;
    CLK->AHBCLK  |= CLK_AHBCLK_ISPCKEN_Msk;

    /* Setup SPI0 multi-function pins */
    SYS->GPA_MFPL &= ~(SYS_GPA_MFPL_PA0MFP_Msk | SYS_GPA_MFPL_PA1MFP_Msk |
                       SYS_GPA_MFPL_PA2MFP_Msk | SYS_GPA_MFPL_PA3MFP_Msk);
    SYS->GPA_MFPL  = SYS_GPA_MFPL_PA0MFP_SPI0_MOSI | SYS_GPA_MFPL_PA1MFP_SPI0_MISO |
                     SYS_GPA_MFPL_PA2MFP_SPI0_CLK  | SYS_GPA_MFPL_PA3MFP_SPI0_SS;

    /* Enable SPI0 clock pin (PA2) schmitt trigger */
    PA->SMTEN |= GPIO_SMTEN_SMTEN2_Msk;
}
```

Use PB4 as detect pin to control whether the system stays in DFU ISP mode or boots to APROM.

```
#define DetectPin           PB4

...

int32_t main(void)
{
    /* The system stays in DFU ISP mode when DetectPin is high and
       boots to APROM when DetectPin is low. */
    while (DetectPin != 0)
    {
        ...
```

```
    }
}
```

Refer to the Datasheet of target MCU and SPI Flash to specify the size and erase unit of APROM and SPI Flash in descriptors: g_au8Alt0StringDesc and g_au8Alt1StringDesc in *descriptors.c.*

Note: USB string descriptors use UTF-16LE (Little Endian) encoding for their string data and the high byte is typically 0 for standard ASCII characters.

```
/* APROM
 *   Base address: 0x0
 *   Erase unit: 512B
 *   Size: 512 * 512B = 256KB
 */
const uint8_t g_au8Alt0StringDesc[] = {
    42, DESC_STRING,
    '@', 0, 'A', 0, 'P', 0, 'R', 0, 'O', 0, 'M', 0, '/', 0,
    '0', 0, 'x', 0, '0', 0, '/', 0,
    '5', 0, '1', 0, '2', 0, '*', 0, '5', 0, '1', 0, '2', 0, 'B', 0,
    // e: Readable and Writeable
    'e', 0
};

/* SPI Flash
 *   Base address: 0x0
 *   Erase unit: 4KB
 *   Size: 1024 * 4KB = 4MB
 */
const uint8_t g_au8Alt1StringDesc[] = {
    48, DESC_STRING,
    '@', 0, 'S', 0, 'P', 0, 'I', 0, ' ', 0, 'F', 0, 'l', 0, 'a', 0, 's', 0, 'h', 0, '/',
0,
    '0', 0, 'x', 0, '0', 0, '/', 0,
    '1', 0, '0', 0, '2', 0, '4', 0, '*', 0, '4', 0, 'K', 0,
    // e: Readable and Writeable
    'e', 0
};
```

Follow the SPI Flash Datasheet to define the SPI information in *cmd_handle.h.*

```
#define SPI_FLASH_PORT          SPI0
// SPI information depends on target SPI Flash.
#define SPI_FLASH_ID            0xEF15
#define SPI_FLASH_SECTOR_SIZE   4096
#define SPI_FLASH_PAGE_SIZE     256
#define SPI_FLASH_SIZE          0x400000
```

The DFU download and upload requests are handled by the WriteData and ReadData functions in *cmd_handle.c.*

USBD_GetAltSetting() is used to determine the target storage for the data transfer.

```
// Read data from APROM or SPI Flash
int ReadData(uint32_t u32StartAddr, uint32_t u32EndAddr, uint32_t *pu32Data)
{
    // Get the target storage
    uint32_t u32AltSetting = USBD_GetAltSetting();
```

```
    if (u32AltSetting == DFU_INTF_APROM)
    {
        return FMC_Proc(FMC_ISPCMD_READ, u32StartAddr, u32EndAddr, pu32Data);
    }
    else if (u32AltSetting == DFU_INTF_SPI_FLASH)
    {
        return SPI_Proc(FMC_ISPCMD_READ, u32StartAddr, u32EndAddr, pu32Data);
    }

    return -1;
}

// Write data into APROM or SPI Flash
int WriteData(uint32_t u32StartAddr, uint32_t u32EndAddr, uint32_t *pu32Data)
{
    // Get the target storage
    uint32_t u32AltSetting = USBD_GetAltSetting();

    if (u32AltSetting == DFU_INTF_APROM)
    {
        return FMC_Proc(FMC_ISPCMD_PROGRAM, u32StartAddr, u32EndAddr, pu32Data);
    }
    else if (u32AltSetting == DFU_INTF_SPI_FLASH)
    {
        return SPI_Proc(FMC_ISPCMD_PROGRAM, u32StartAddr, u32EndAddr, pu32Data);
    }

    return -1;
}
```

SPI_Proc() handles SPI Flash read, write, and erase operations.

```
int SPI_Proc(uint32_t u32Cmd, uint32_t u32StartAddr, uint32_t u32EndAddr,
             uint32_t *pu32Data)
{
    uint32_t u32Addr;

    if (u32StartAddr >= SPI_FLASH_SIZE)
        return -1;
```

SPI communication parameters and clock frequency is specified when SPI_Open() is invoked.

```
    /* Configure SPI_FLASH_PORT as a master, MSB first, 8-bit transaction,
       SPI Mode-0 timing, clock is 4 MHz */
    SPI_Open(SPI_FLASH_PORT, SPI_MASTER, SPI_MODE_0, 8, FREQ_4MHZ);
```

Check if the SPI Flash device ID is as expected; otherwise, report an error.

```
    /* Check SPI Flash ID */
    if (SpiFlash_ReadMidDid() != SPI_FLASH_ID)
    {
        return -1;
    }
```

Handle SPI Flash operations based on u32Cmd:

- Read data page by page from the start to the end address.
- Write data page by page with automatic sector-erase handling.

```
    if (u32Cmd == FMC_ISPCMD_READ)
    {
        for (u32Addr = u32StartAddr; u32Addr < u32EndAddr; u32Addr += SPI_FLASH_PAGE_SIZE)
        {
            SpiFlash_ReadPage(u32Addr, (uint8_t *)pu32Data);
            pu32Data = (uint32_t *)((uint32_t)pu32Data + SPI_FLASH_PAGE_SIZE);
        }
    }
    else
    {
        for (u32Addr = u32StartAddr; u32Addr < u32EndAddr; u32Addr += SPI_FLASH_PAGE_SIZE)
        {
            /* Erase flash by sector before write data */
            if ((u32Addr % SPI_FLASH_SECTOR_SIZE) == 0)
            {
                SpiFlash_EraseSector(u32Addr);
                SpiFlash_WaitReady();
            }

            SpiFlash_WritePage(u32Addr, (uint8_t *)pu32Data);
            SpiFlash_WaitReady();
            pu32Data = (uint32_t *)((uint32_t)pu32Data + SPI_FLASH_PAGE_SIZE);
        }
    }
}
```

Support basic SPI Flash read/write/erase instructions are defined in spi_user.h.

```
#define SPI_FLASH_WRITE_STATUS  0x01
#define SPI_FLASH_PROG_PAGE     0x02
#define SPI_FLASH_READ          0x03
#define SPI_FLASH_READ_STATUS   0x05
#define SPI_FLASH_WRITE_ENABLE  0x06
#define SPI_FLASH_ERASE_SECTOR  0x20
#define SPI_FLASH_READ_ID       0x90
#define SPI_FLASH_ERASE_CHIP    0xC7
```

DFU_ClassRequest() in dfu_transfer.c handles the minimal requirements of DFU state transition.

WriteData() and ReadData() are invoked on host data transfer requests and report an error if the transfer fails.

```
void DFU_ClassRequest(void)
{
    ...

    if (buf[0] & 0x80)
    {
        /* Device-to-Host */
        switch (buf[1])
        {
            case DFU_GETSTATUS:
```

```
            {
                if (dfu_status.bState == STATE_dfuDNLOAD_SYNC)
                {
                    dfu_status.bState = STATE_dfuDNLOAD_IDLE;

                    if ((prog_struct.data_len > 0) &&
                        /* Write data to request offset of target storage */
                        WriteData(
                            prog_struct.u32Offset + prog_struct.block_num * TRANSFER_SIZE,
                            prog_struct.data_len,
                            (uint32_t *)prog_struct.buf) != 0
                    )
                    {
                        dfu_status.bStatus = STATUS_errWRITE;
                        dfu_status.bState  = STATE_dfuERROR;
                    }
                }
        ...
    }
    else
    {
        // Host-to-Device
        switch (buf[1])
        {
            ...

            case DFU_UPLOAD:
            {
                if (dfu_status.bState == STATE_dfuIDLE || dfu_status.bState ==
STATE_dfuUPLOAD_IDLE)
                {
                    if (wLength == 0)
                    {
                        dfu_status.bState = STATE_dfuIDLE;
                        return;
                    }
                    else
                    {
                        if (dfu_status.bState == STATE_dfuIDLE)
                        {
                            dfu_status.bState = STATE_dfuUPLOAD_IDLE;
                        }

                        /* Read data from request offset of target storage */
                        if (ReadData(
                                prog_struct.u32Offset + wValue * TRANSFER_SIZE,
                                wLength,
                                (uint32_t *)prog_struct.buf) < 0
                        )
                        {
                            dfu_status.bState = STATE_dfuIDLE;
                            USBD_PrepareCtrlIn(0, 0);
                        }

        ...
    }
```

```
}
```

3. Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.008
- IDE version
  - Keil uVision 5.43.0
- DFU utility
  - dfu-util.exe
- Windows driver

3.2 Hardware Requirements

- Circuit components
  - NuMaker-M252KG V1.4
  - Winbond W25Q32 SPI Flash (Or compatible SPI Flash)
- Pin Connect
  - Connect the external SPI Flash to the NuMaker-M252KG SPI0 as Figure 3-1.
  - Connect SW1 (PB.4) to VCC to boot into DFU ISP mode or connect SW1 to GND to boot from APROM.
  - Connect the USB device port of NuMaker-M252KG to the PC USB host to establish a DFU connection.
  - Connect UART0 TX (PB.13) pin to PC UART RX to show the dump result of the SPI Flash and UART0 RX (PB.12) pin to PC UART TX to select the menu item.

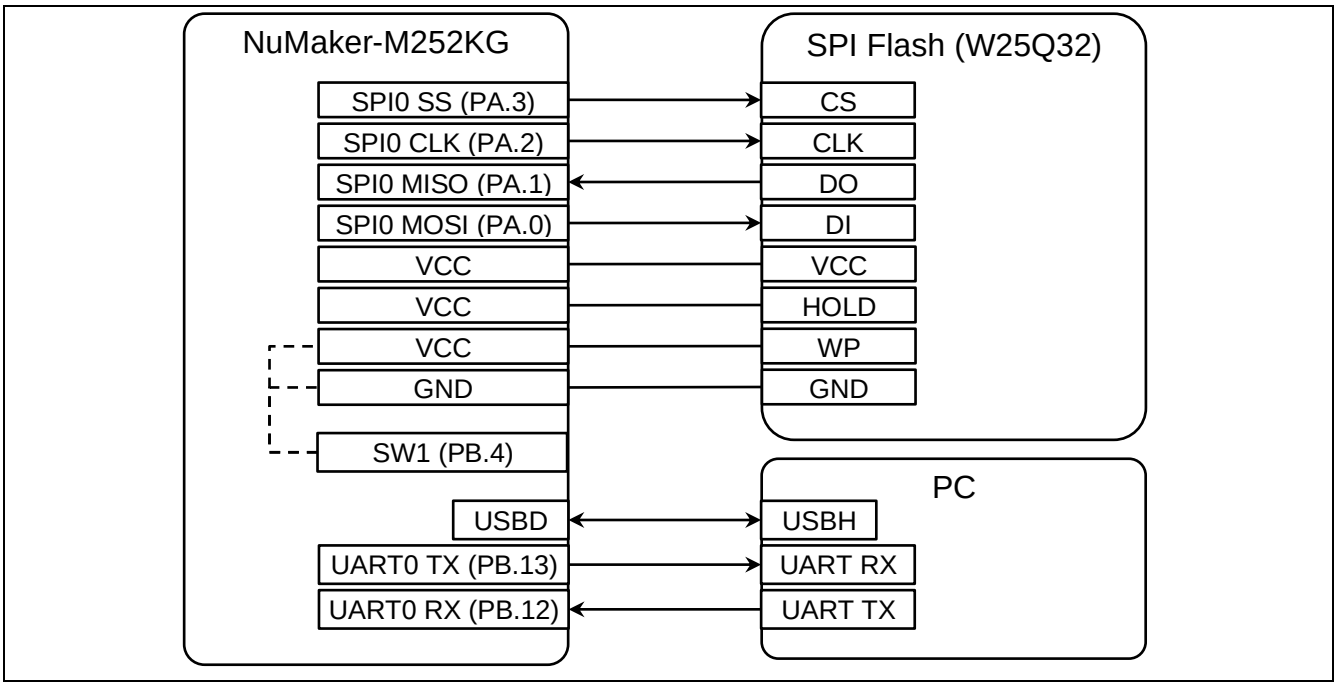


Figure 3-1 Pin Connect

4. Directory Information

The directory structure is shown below.

EC_M252_DFU_Dual_Interface_ISP	
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	
DFU_Dual_IF_ISP	Source files of example code
WindowsDriver	USB drivers for Nuvoton DFU device
WindowsTool	dfu-util.exe for firmware download/upload operations
Dump_SPI_Flash	Example code to dump the SPI Flash

Figure 4-1 Directory Structure

5. Example Code Execution

1. Browse the DFU_Dual_IF_ISP folder as described in the Directory Information section and double-click *DFU_Dual_IF_ISP.uvprojx*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6. Revision History

Date	Revision	Description
2025.10.15	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.