# NUC240 SPI FATFS

Example Code Introduction for 32-bit NuMicro® Family

## Information

| | |
|---|---|
| Application | Demonstrate how to access SPI Flash using FatFs |
| 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 plans SPI Flash as a storage device. In order to manage the data structure of the file when accessing the SPI Flash, this example code uses FatFs to do the file system management.

## 1.2 Principle

### 1.2.1 FatFs

FatFs is a generic FAT (File Allocation Table) file system module designed for small embedded systems. The FatFs module is written in compliance with ANSI C and is completely separated from the disk I/O layer. Therefore, it is independent of the hardware platform. Moreover, the FatFs module can be incorporated into small microcontrollers with limited resources without any modifications, such as 8051, ARM, etc.

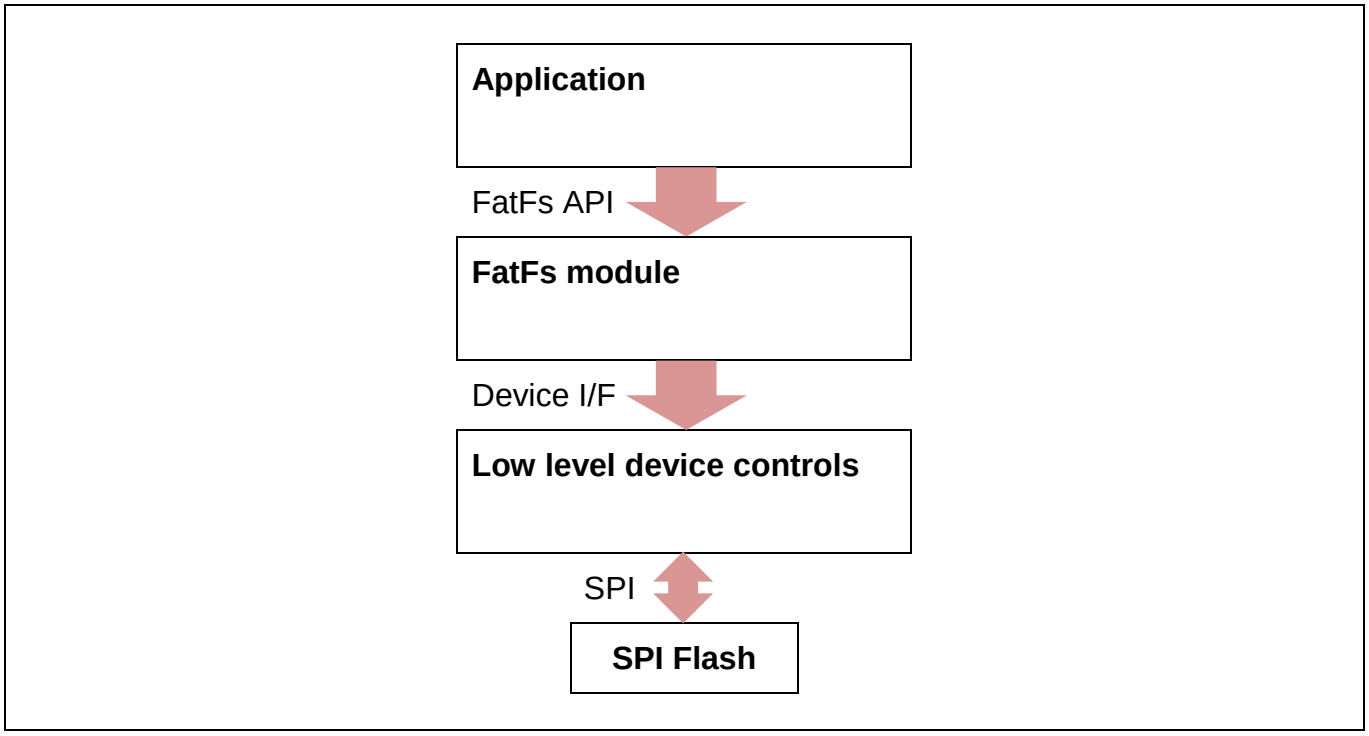


Figure 1-1 FatFs Control Block Diagram

Figure 1-1 shows the FatFs control block diagram. FatFs API provides various file system function for the application, the APIs are used in this example code are listed in Table 1.

| FatFs API | Description |
|---|---|
| f_mount | Register/Unregister the work area of the volume |
| f_mkfs | Create a FAT volume on the logical drive |

| | |
|---|---|
| f_opendir | Open a directory |
| f_readdir | Read a directory item |
| f_getfree | Get total size and free size on the volume |

Table 1 Application Interface Used in Example Code

Since FatFs module is the file system layer independent of platforms and storage media, it is completely separated from the physical devices, such as SPI Flash. As shown in Figure 1-1, the low level device control module is not any part of FatFs module, so device interface needs to be implemented by this example code for FatFs accessing the SPI Flash. Table 2 lists media access interface and its function provided by this example code. For more information about application interface and media access interface, please refer to official website: http://elm-chan.org/fsw/ff/00index_e.html.

| Device I/F | Description |
|---|---|
| disk_status | Get device status |
| disk_initialize | Initialize device |
| disk_read | Read sector(s) |
| disk_write | Write sector(s) |
| disk_ioctl | Control device dependent functions |

Table 2 Media Access Interface Implemented in Example Code

### 1.2.2 SPI Flash

This example code uses W25X16A Serial Flash memory as a storage device. The W25X16A is accessed through a SPI compatible bus consisting of four signals: Serial Clock (CLK), Chip Select (/CS), Serial Data Input (DI) and Serial Data Output (DO). Figure 1-2 shows its pin configuration.

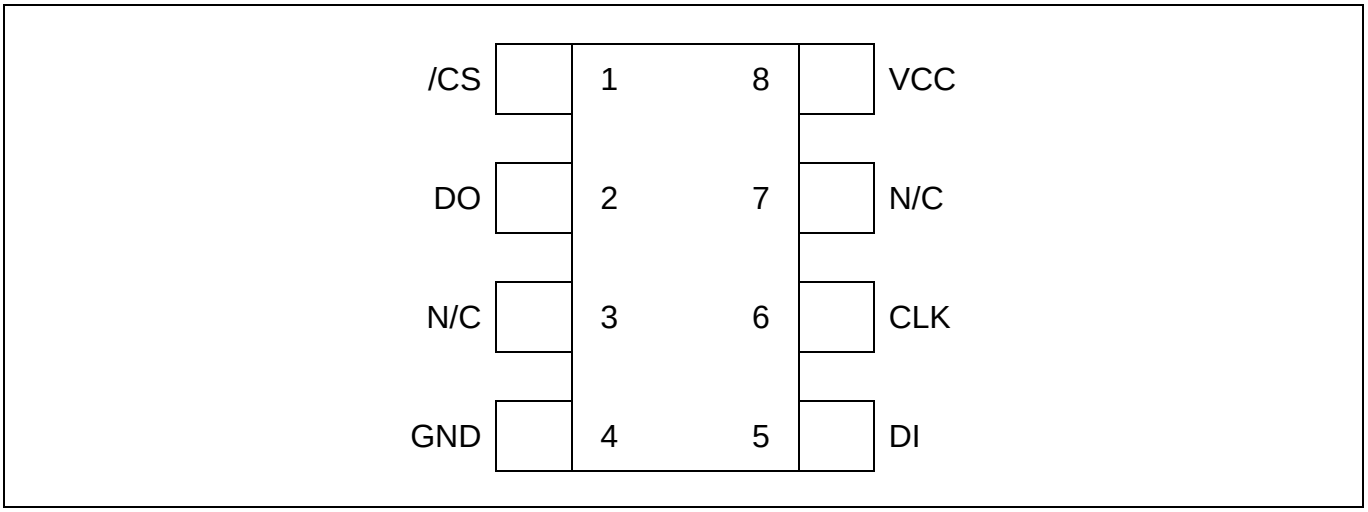


Figure 1-2 W25X16A Pin Configuration

Standard SPI instructions use the DI input pin to serially write instructions, addresses or data to the device on the rising edge of CLK. The DO output pin is used to read data or status from the device on the falling edge of CLK. Table 3 lists the instructions of the W25X16A used in this example code. Moreover, manufacturer and device identification are listed in Table 4.

| Instruction Name | Command | Description |
|---|---|---|
| DRVSPIFLASH_WRITE_ENABLE | 0x06 | Write enable |
| DRVSPIFLASH_WRITE_DISABLE | 0x04 | Write disable |
| DRVSPIFLASH_READ_STATUS | 0x05 | Read status register - 1 |
| DRVSPIFLASH_FAST_READ | 0x0B | Fast read |
| DRVSPIFLASH_PAGE_PROGRAM | 0x02 | Page program |
| DRVSPIFLASH_SECTOR_ERASE | 0x20 | Sector erase (4KB) |
| DRVSPIFLASH_DEVICE_ID | 0x90 | Manufacturer/Device ID |

Table 3 Instruction Table Used in Example Code

| Manufacturer ID | |
|---|---|
| Winbond Serial Flash | 0xEF |
| Device ID | |
| W25X16A | 0x14 |

Table 4 Manufacturer and Device Identification

## 1.3 Demo Result

The following execution results will be output to the terminal tool with UART baud rate of 115200.
When “FF_USE_MKFS” is defined to 1, the file system will be formatted. Figure 1-3 shows the directory information after formatting the file system. You can see how many available sizes can be stored in the SPI Flash.

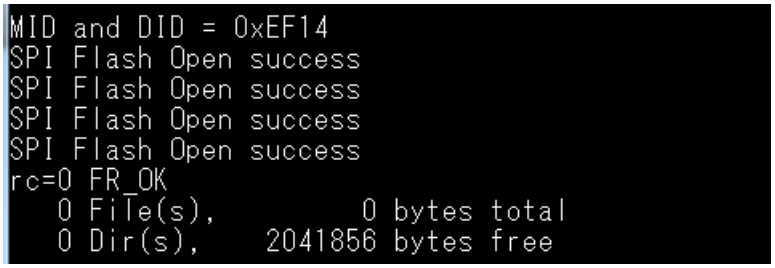


Figure 1-3 Directory Information without File

After you store the file in the SPI Flash, the directory information including the file attributes, modification date, modification time, size, name and how many remaining sizes can be used will be listed as shown in Figure 1-4.

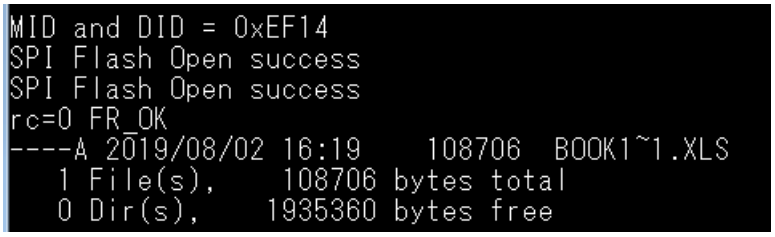


Figure 1-4 Directory Information with File

# 2 Code Description

Initialize SPI and configure SPI multi-function pins.

```
    /* Setup SPI0 multi-function pins */
    SYS->GPC_MFP &= ~(SYS_GPC_MFP_PC0_Msk | SYS_GPC_MFP_PC1_Msk | SYS_GPC_MFP_PC2_Msk |
                      SYS_GPC_MFP_PC3_Msk);
    SYS->GPC_MFP |= SYS_GPC_MFP_PC0_SPI0_SS0 | SYS_GPC_MFP_PC1_SPI0_CLK |
                    SYS_GPC_MFP_PC2_SPI0_MISO0 | SYS_GPC_MFP_PC3_SPI0_MOSI0;
    SYS->ALT_MFP &= ~(SYS_ALT_MFP_PC0_Msk | SYS_ALT_MFP_PC1_Msk | SYS_ALT_MFP_PC2_Msk |
                      SYS_ALT_MFP_PC3_Msk);
    SYS->ALT_MFP |= SYS_ALT_MFP_PC0_SPI0_SS0 | SYS_ALT_MFP_PC1_SPI0_CLK |
                    SYS_ALT_MFP_PC2_SPI0_MISO0 | SYS_ALT_MFP_PC3_SPI0_MOSI0;


    /* Select HCLK as the clock source of SPI0 */
    CLK_SetModuleClock(SPI0_MODULE, CLK_CLKSEL1_SPI0_S_HCLK, MODULE_NoMsk);


    /* Enable SPI0 peripheral clock */
    CLK_EnableModuleClock(SPI0_MODULE);


    /* Configure SPI0 as a master, SPI clock rate 12 MHz, clock idle low, 32-bit
       transaction, drive output on falling clock edge and latch input on rising edge. */
    SPI_Open(SPI_Flash, SPI_MASTER, SPI_MODE_0, 32, 12000000);


    /* Disable automatic slave selection function and set SPIn_SS0 pin to high state. */
    SPI_DisableAutoSS(SPI_Flash);
    SPI_SET_SS0_HIGH(SPI_Flash);
```

Read SPI Flash MID and DID to check whether SPI Flash open success or not.

```
    /* Read MID & DID */
    u32SpiMidDid = SpiReadMidDid();


    if ((u32SpiMidDid & 0xFFFF) == 0xEF14)
    {
        stat = RES_OK;
        printf("SPI Flash Open success\n");
    }
    else
    {
        stat = STA_NOINIT;
        printf("SPI Flash Open failed\n");
    }
```

Mount a logical drive.

```
    /* Mount a logical drive */
    res = f_mount(&FatFs[0], ptr, 1);

    if (res)
    {
        put_rc(res);
        printf("Don't mount file system\n");
    }
```

Create a FAT volume. User can define “FF_USE_MKFS” to 1 in ffconf.h to format the file system.

```
#if (FF_USE_MKFS)
    /* Create a FAT volume */
    res = f_mkfs(ptr, FM_FAT, 0, Buff, sizeof(Buff));

    if (res)
    {
        put_rc(res);
        printf("Don't format\n");
    }

#endif
```

List the directory information. User can get it from UART log.

```
    /* List directory information */
    put_rc(f_opendir(&dir, ptr));
    p1 = s1 = s2 = 0;

    for (;;)
    {
        /* Read directory entries in sequence */
        res = f_readdir(&dir, &Finfo);

        if ((res != FR_OK) || !Finfo.fname[0])
            break;

        if (Finfo.fattrib & AM_DIR)
        {
            s2++;
```

```
        }
        else
        {
            s1++;
            p1 += Finfo.fsize;
        }

        printf("%c%c%c%c%c %u/%02u/%02u %02u:%02u %9lu  %s\n",
               (Finfo.fattrib & AM_DIR) ? 'D' : '-',
               (Finfo.fattrib & AM_RDO) ? 'R' : '-',
               (Finfo.fattrib & AM_HID) ? 'H' : '-',
               (Finfo.fattrib & AM_SYS) ? 'S' : '-',
               (Finfo.fattrib & AM_ARC) ? 'A' : '-',
               (Finfo.fdate >> 9) + 1980, (Finfo.fdate >> 5) & 15, Finfo.fdate & 31,
               (Finfo.ftime >> 11), (Finfo.ftime >> 5) & 63, Finfo.fsize, Finfo.fname);
    }

    printf("%4u File(s),%10u bytes total\n%4u Dir(s)", s1, p1, s2);

    /* Get number of free clusters */
    if (f_getfree(ptr, (DWORD *)&p1, &fs) == FR_OK)
        printf(", %10u bytes free\n", p1 * fs->csize * 512);
```

# 3 Software and Hardware Environment

## Software Environment

- BSP version
  - NUC230/NUC240 Series BSP CMSIS v3.01.002
- IDE version
  - Keil uVision V5.26

## Hardware Environment

- Circuit components
  - NuTiny-EVB-NUC240
  - SPI Flash (W25X16A)
- Diagram
  - As shown in Figure 3-1, connect NUC240 pins: SPI0_SS0 (PC.0), SPI0_CLK (PC.1), SPI0_MISO0 (PC.2) and SPI0_MOSI0 (PC.3) to W25X16A pins: /CS, CLK, DO and DI respectively for accessing SPI Flash by SPI.

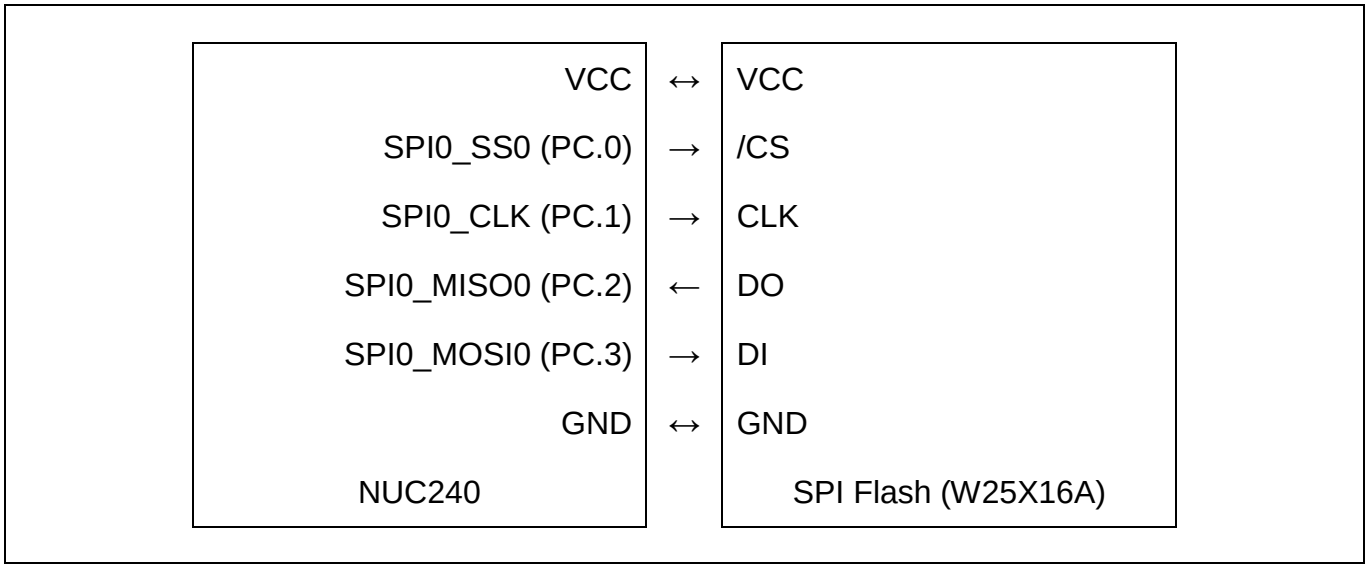


Figure 3-1 NUC240 and W25X16A Pin Connection

# 4 Directory Information

- EC_NUC240_SPI_FATFS_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
  - ThirdParty
    - FatFs — A generic FAT file system module for small embedded systems. Its official website is: http://elm-chan.org/fsw/ff/00index_e.html.

# 5 How to Execute Example Code

1. Browsing into sample code folder by Directory Information (section 4) and double click NUC240_SPI_FATFS.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 |
|---|---|---|
| Aug. 8, 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.*