M480 SPI NAND Data Logger

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This sample code demonstrates accessing files in external SPI NAND Flash.
BSP Version	M480 BSP CMSIS V3.03.000
Hardware	M483 Nand Flash 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 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 sample code demonstrates accessing files in external Winbond W25N02JW SPI NAND Flash. The file system is FAT32. User can store data in SPI NAND Flash as a data logger.

1.1 Demo Result

The demo result is shown below.

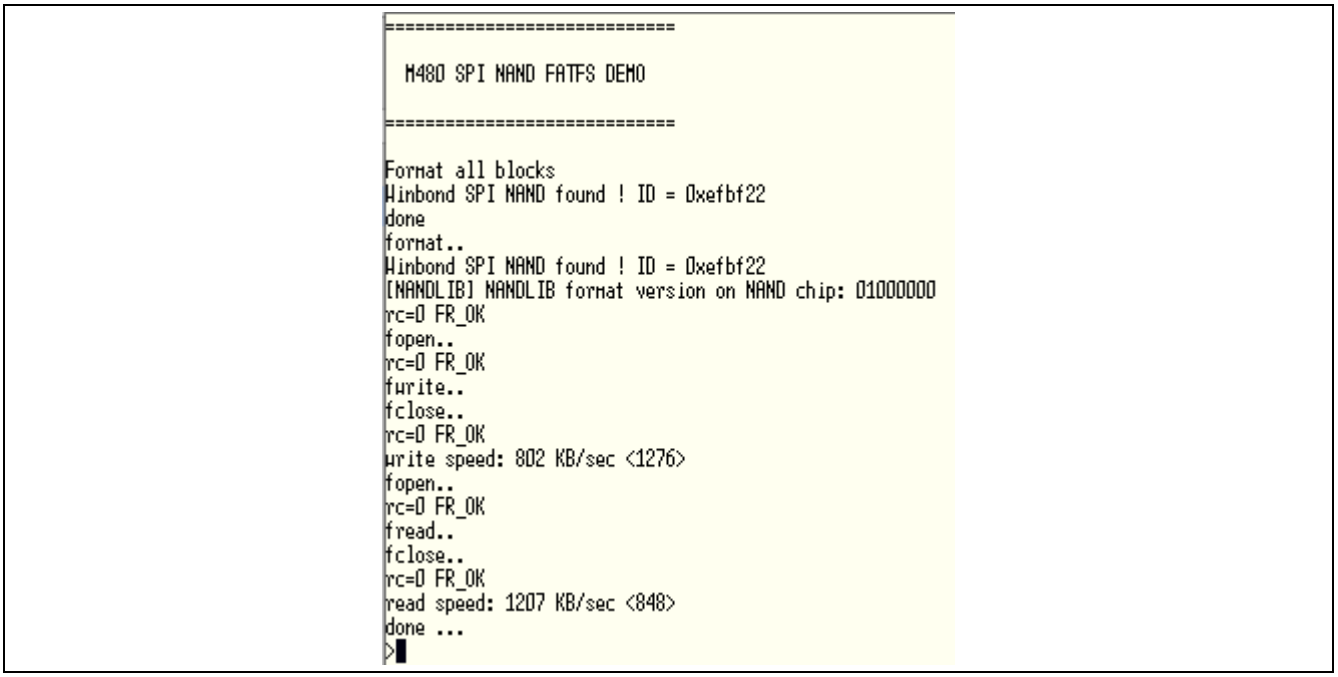

Figure 1-1 Demo Result

2 Code Description

First, initialize the M480 in main.c, by calling SYS_Init() that initializes UART0, Timer0 and SPI3.

```
void SYS_Init(void)
{

    /* Set core clock as PLL_CLOCK from PLL and SysTick source to HCLK/2*/
    CLK_SetCoreClock(PLL_CLOCK);

    /* Set PCLK0/PCLK1 to HCLK/2 */
    CLK->PCLKDIV = (CLK_PCLKDIV_PCLK0DIV2 | CLK_PCLKDIV_PCLK1DIV2);

    /* Enable peripheral clock */
    CLK_EnableModuleClock(UART0_MODULE);
    CLK_EnableModuleClock(TMR0_MODULE);
    CLK_EnableModuleClock(SPI3_MODULE);

    /* Peripheral clock source */
    CLK_SetModuleClock(UART0_MODULE, CLK_CLKSEL1_UART0SEL_HIRC, CLK_CLKDIV0_UART0(1));
    CLK_SetModuleClock(TMR0_MODULE, CLK_CLKSEL1_TMR0SEL_HIRC, 0);
    CLK_SetModuleClock(SPI3_MODULE, CLK_CLKSEL2_SPI3SEL_PCLK0, MODULE_NoMsk);

    /* Set GPB multi-function pins for UART0 RXD and TXD */
    SYS->GPB_MFPH &= ~(SYS_GPB_MFPH_PB12MFP_Msk | SYS_GPB_MFPH_PB13MFP_Msk);
    SYS->GPB_MFPH |= (SYS_GPB_MFPH_PB12MFP_UART0_RXD | SYS_GPB_MFPH_PB13MFP_UART0_TXD);

    /* Setup SPI3 multi-function pins */
    SYS->GPB_MFPH &= ~(SYS_GPB_MFPH_PB8MFP_Msk | SYS_GPB_MFPH_PB9MFP_Msk |
SYS_GPB_MFPH_PB10MFP_Msk | SYS_GPB_MFPH_PB11MFP_Msk);
    SYS->GPB_MFPH |= (SYS_GPB_MFPH_PB8MFP_SPI3_MOSI | SYS_GPB_MFPH_PB9MFP_SPI3_MISO |
SYS_GPB_MFPH_PB10MFP_SPI3_SS | SYS_GPB_MFPH_PB11MFP_SPI3_CLK);

    /* Enable SPI3 clock pin (PB11) schmitt trigger */
    PB->SMTEN |= GPIO_SMTEN_SMTEN11_Msk;

    SysTick->LOAD = CyclesPerUs;
    SysTick->CTRL |= SysTick_CTRL_ENABLE_Msk;
}
```

SPI NAND Flash configuration. Hook SPI NAND information and drivers.

```
NDRV_T NandDiskDriver = {
    .vid = 0xef,        /* Verder ID */
    .did = 0xbf22,      /* Devide ID */
    .mcycle = 1,        /* 5 address cycles */
    .pblock = 1024,     /* total physical block count */
    .lblock = 1000,     /* total logical block count */
    .ppb = 64,          /* page per block */
    .pagesize = Page_size,/* page size */
    .oobsize = 64,      /* spare area size */
    .ecc = 8,           /* correctable error bit */
    .eccbyte = 13,      /* ECC byte per 512 byte data */
    .init = Init,
    .pread = ReadPage,
    .pwrite = WritePage,
    .isDirtyPage = IsDirtyPage,
    .isValidBlock = IsValidBlock,
    .erase = EraseBlock,
    .markBad=MarkBadBlock
};
```

In main(), do file access (open/read/write/close).

```
int main(void)
{
    long        p1;
    TCHAR       nand_path[] = { '0', ':', 0 };    /* NAND drive started from 0 */

    UINT s1, s2, cnt;


    /* Mount SPI NAND disk */
    f_mount(&gFatfsVol, nand_path, 0);
    printf("format..\n");
    put_rc(f_mkfs("", FM_FAT, 0, Buff, FF_MAX_SS));
    f_mount(&gFatfsVol, nand_path, 1);

    /* Write file test: f_open -> f_write -> f_close */
    printf("fopen..\n");
    put_rc(f_open(&file1, "hello.txt", FA_OPEN_APPEND | FA_WRITE));
    p1 = 10 * 1024 * 1024;
    printf("fwrite..\n");
    s1 = gSec;
    while(p1 > 0)
    {
        f_write(&file1, Buff, 8192, &cnt);
        p1 -= cnt;
    }
    s2 = gSec;

    printf("fclose..\n");
    put_rc(f_close(&file1));
    printf("write speed: %d KB/sec <%d>\n", 10*1024*100/(s2-s1), s2-s1);

    /* Read file test: f_open -> f_read -> f_close */
    printf("fopen..\n");
    put_rc(f_open(&file1, "hello.txt", FA_READ));
    p1 = 10 * 1024 * 1024;
    printf("fread..\n");
    s1 = gSec;
    while(p1 > 0)
    {
        f_read(&file1, Buff, 8192, &cnt);
        p1 -= cnt;
    }
    s2 = gSec;

    printf("fclose..\n");
    put_rc(f_close(&file1));
    printf("read speed: %d KB/sec <%d>\n", 10*1024*100/(s2-s1), s2-s1);

    printf("done ...\n");
```

3 Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - M480 Series BSP CMSIS v3.03.000
- IDE version
  - Keil uVersion 5.26

3.2 Hardware Requirements

- Circuit components
  - M483 Nand Flash V1.0
  - SPI NAND Flash W25N02JW

- Pin Connect

V_{DD}	↔	V_{DD}
GND	↔	GND
PB.11	↔	SPI_NAND_FLASH_CLK
PB.10	↔	SPI_NAND_FLASH_SS
PB.9	↔	SPI_NAND_FLASH_D1
PB.8	↔	SPI_NAND_FLASH_D0
M483 SPI3		W25N02JW

Figure 3-1 Pin Connect

4 Directory Information

The directory structure is shown below.

EC_M480_SPI_Nand_Data_Logger_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
NandLib	NAND Flash logic to physical mapping
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	Source file of example code
ThirdParty	
bch	Bose-Chaudhuri-Hocquenghem (BCH) codes used by NAND Error Correction Code
FatFs	Generic FAT File system

Figure 4-1 Directory Structure

5 Example Code Execution

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

6 Revision History

Date	Revision	Description
2021.09.14	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.