SD Card with FATFS

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	SD Card with FAT File System.
BSP Version	NUC230/240 Series BSP v3.01.002
Hardware	NUC240VE3AN NuEdu Board V2.0 + NuEdu-Advance02 V1.1

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

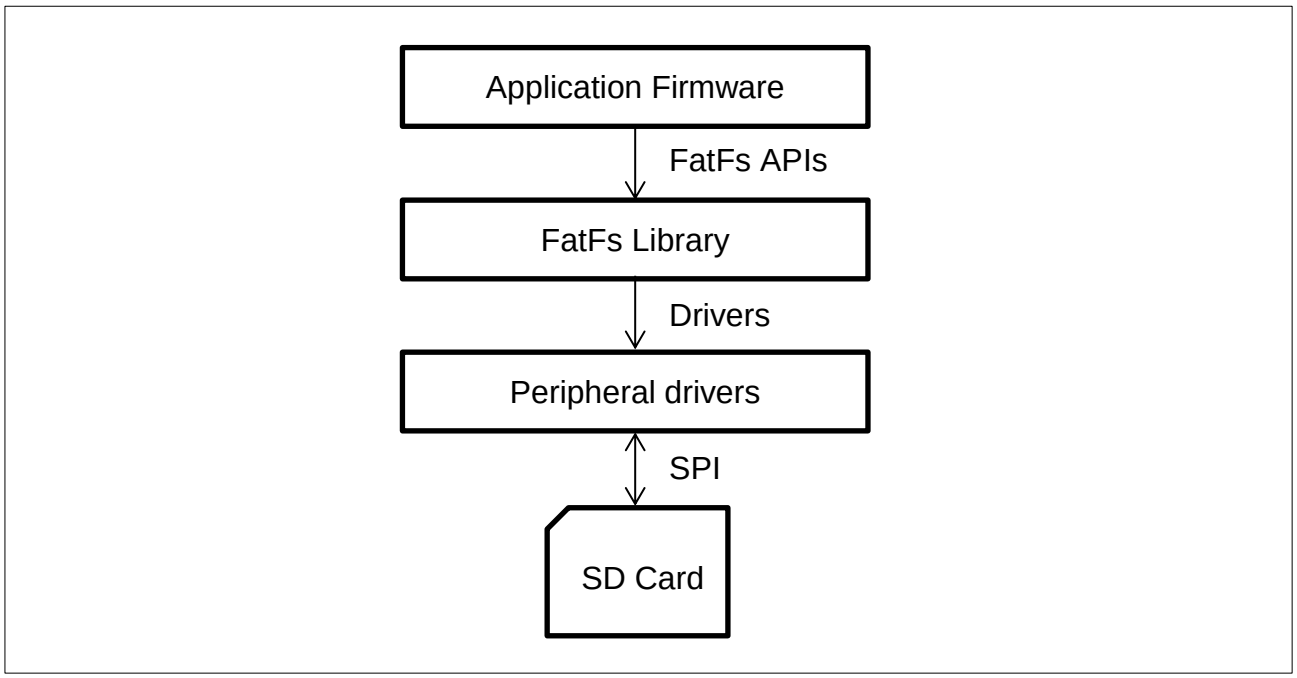
1 Function Description

1.1 Introduction

The SD memory card system supports two kinds of communication mode: SD mode and SPI mode. This example uses SPI mode to communication with an SD card, and access SD card in FAT format. The file system layer adopts the third-part library FatFs, which is a generic FAT/exFAT file system module for small embedded systems.

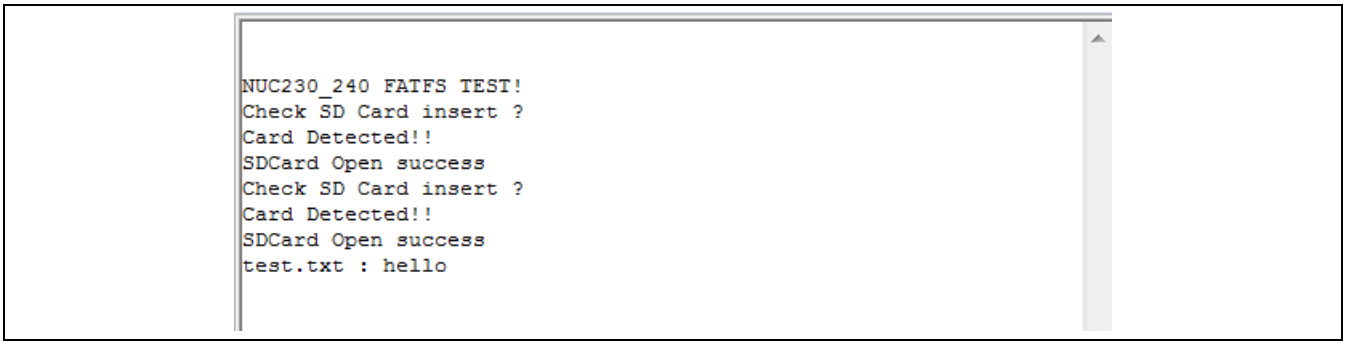
1.2 Principle

The architecture of SD card access by SPI mode is shown as below. The basic access flow is application firmware calls FatFs APIs to read/write a file, and then FatFs APIs calls SPI drivers to get/send data with SD card.



1.3 Demo Result

User can enter debug mode of Keil MDK and open UART#1 window. The execution result shown as below figure.



2 Code Description

The FatFs library provides file system functions. Some APIs of this example used are shown as below.

FUNCTION NAME	FUNCTION DESCRIPTION
f_open	Open or create a file
f_close	Close an open file object
f_read	Read data from the file
f_write	Write data to the file
f_lseek	Move file pointer of the file object
f_mount	Mount/Unmount a logical drive

There are five FatFs APIs need to be porting in EC_NUC230_240_FATFS_SDCard_V1.0 \SampleCode\ExampleCode\NUC230_240_FATFS_SDCard\diskio.c.

FUNCTION NAME	FUNCTION DESCRIPTION
disk_initialize	Initialize device
disk_status	Get device status
disk_read	Read sector(s)
disk_write	Write sector(s)
disk_ioctl	Control device dependent functions

Some FatFs APIs need calls SD card driver to access SD card. These SD card driver that implemented by SPI in sdcard.c are list as below. The path of sdcard.c is located at EC_NUC230_240_FATFS_SDCard_V1.0\SampleCode\ExampleCode\NUC230_240_FATFS_SDCard.

FUNCTION NAME	FUNCTION DESCRIPTION
SDCARD_Open	Open GPIO function and initial SD card
SDCARD_Close	Close SD card
MMC_Command_Exec	Send SDCARD CMD and Receive Response
GetLogicSector	Get logic sector size
SDCARD_GetCardSize	Get card total sector after SDCARD is opened
SpiRead	Get data from SD card
SpiWrite	Store data into SD card

The sample code is as follows.
Enable crystals and PLL. Enable UART0 module and SPI1 module clock, select its clock

source. Configure the MFP function of used pins in SYS_Init().
Configure the MFP function of pins as follows:

- PB.0 as UART0_RXD
- PB.1 as UART0_TXD
- PC.9 as SPI1_CLK
- PC.10 as SPI1_MISO
- PC.11 as SPI1_MOSI

```
void SYS_Init(void)
{
    /* Unlock protected registers */
    SYS_UnlockReg();

    /*---------------------------------------------------------------------------------------------------------*/
    /* Init System Clock                                                                                       */
    /*---------------------------------------------------------------------------------------------------------*/

    /* Enable Internal RC 22.1184MHz clock */
    CLK_EnableXtalRC(CLK_PWRCON_OSC22M_EN_Msk);

    /* Waiting for Internal RC 22M clock ready */
    CLK_WaitClockReady(CLK_CLKSTATUS_OSC22M_STB_Msk);

    /* Switch HCLK clock source to Internal RC and HCLK source divide 1 */
    CLK_SetHCLK(CLK_CLKSEL0_HCLK_S_HIRC, CLK_CLKDIV_HCLK(1));

    /* Enable external XTAL 12MHz clock */
    CLK_EnableXtalRC(CLK_PWRCON_XTL12M_EN_Msk);

    /* Waiting for external XTAL clock ready */
    CLK_WaitClockReady(CLK_CLKSTATUS_XTL12M_STB_Msk);

    /* Set core clock as PLL_CLOCK from PLL */
    CLK_SetCoreClock(PLL_CLOCK);

    /* Enable UART module clock */
    CLK_EnableModuleClock(UART0_MODULE);

    /* Select UART module clock source */
    CLK_SetModuleClock(UART0_MODULE, CLK_CLKSEL1_UART_S_PLL, CLK_CLKDIV_UART(1));
```

```
    /* Enable SPI1 peripheral clock */
    CLK_EnableModuleClock(SPI1_MODULE);

    /* Select HCLK as the clock source of SPI1 */
    CLK_SetModuleClock(SPI1_MODULE, CLK_CLKSEL1_SPI1_S_HCLK, MODULE_NoMsk);

    /*---------------------------------------------------------------------------------------*/
    /* Init I/O Multi-function                                                               */
    /*---------------------------------------------------------------------------------------*/

    /* Set GPB multi-function pins for UART0 RXD and TXD */
    SYS->GPB_MFP &= ~(SYS_GPB_MFP_PB0_Msk | SYS_GPB_MFP_PB1_Msk);
    SYS->GPB_MFP |= SYS_GPB_MFP_PB0_UART0_RXD | SYS_GPB_MFP_PB1_UART0_TXD;

    /* Setup SPI1 multi-function pins */
    #ifdef DEFINE_SS
    SYS->GPC_MFP = SYS_GPC_MFP_PC9_SPI1_CLK | SYS_GPC_MFP_PC10_SPI1_MISO0 |
SYS_GPC_MFP_PC11_SPI1_MOSI0;
    SYS->ALT_MFP = SYS_ALT_MFP_PC9_SPI1_CLK | SYS_ALT_MFP_PC10_SPI1_MISO0 |
SYS_ALT_MFP_PC11_SPI1_MOSI0;
    #else
    SYS->GPC_MFP = SYS_GPC_MFP_PC8_SPI1_SS0 | SYS_GPC_MFP_PC9_SPI1_CLK |
SYS_GPC_MFP_PC10_SPI1_MISO0 | SYS_GPC_MFP_PC11_SPI1_MOSI0;
    SYS->ALT_MFP = SYS_ALT_MFP_PC8_SPI1_SS0 | SYS_ALT_MFP_PC9_SPI1_CLK |
SYS_ALT_MFP_PC10_SPI1_MISO0 | SYS_ALT_MFP_PC11_SPI1_MOSI0;
    #endif

    /* Lock protected registers */
    SYS_LockReg();
}
```

Initialize a drive by called disk_initialize(), mount a logical drive by called f_mount(), open a file by called f_open(), write data by called f_write(), read data by f_read(), and close a file by called f_close() in main().

```
int32_t main(void)
{
    uint32_t u32Count;
    FIL file;       /* File objects */
    char achBufRead[512];
    FRESULT res;

    /* Init System, IP clock and multi-function I/O */
    SYS_Init();
```

```
/* Init UART to 115200-8n1 for print message */
UART_Open(UART0, 115200);

printf("\n\nNUC230_240 FATFS TEST!\n");

/* SD card initializetion */
if (disk_initialize(0) != RES_OK)
{
    printf("SDCard initialization failed\n");
}

disk_read(0, g_aBuff, 2, 1);

/* Mount a logical drive */
res = f_mount(&g_aFatFs[0], g_aSD_Path, 1);
if (res)
{
    put_rc(res);
    printf("[ERROR] Mount file system is failed!\n");
    while(1);
}

/* Open test.txt file */
f_open(&file, "test.txt", FA_OPEN_ALWAYS | FA_READ |FA_WRITE);

/* Move the file pointer to end of file */
f_lseek(&file, file.obj.objsize);

/* Write hello text into test.txt */
f_write(&file, "hello ", 6, &u32Count);

/* Move the file pointer to start of file */
f_lseek(&file, 0);

/* Read content in test.txt */
f_read(&file, achBufRead, file.obj.objsize, &u32Count);

printf("test.txt : %s \n", achBufRead);

/* Close file */
```

```
    f_close(&file);

    while(1);
}
```

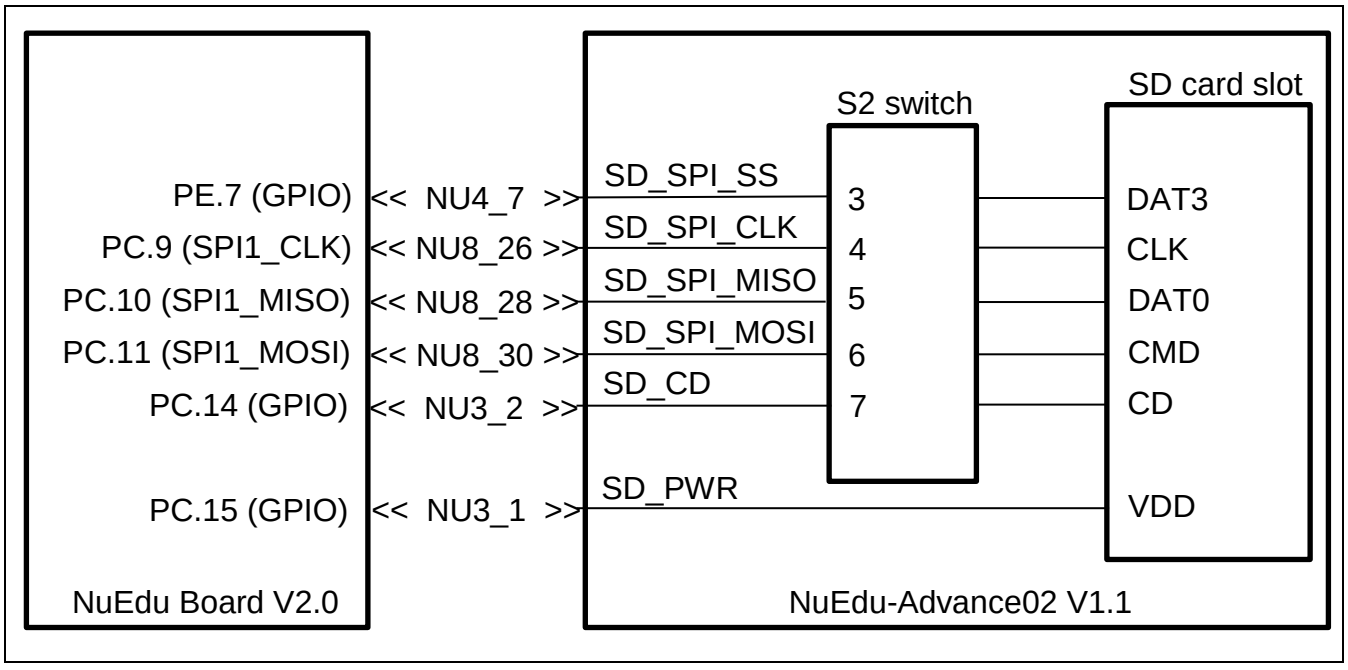
3 Software and Hardware Environment

Software Environment

- BSP version
  - NUC230/240 Series BSP v3.01.002
- IDE version
  - Keil uVersion 5.26

Hardware Environment

- Circuit components
  - NUC240VE3AN NuEdu Board V2.0
  - NuEdu-Advance02 V1.1(SD card interface through SPI, to set 1, 2, 3, 4, 5, 6, 7 ON on S2 switch)
- Diagram



- PE.7: A GPIO pin sets as output mode, it is slave select pin and connects to DAT3 of SD card slot.
- PC.9: This is the SPI1_CLK pin and connects to CLK of SD card slot.

- PC.10: This is the SPI1_MISO pin and connects to DAT0 of SD card slot.
- PC.11: This is the SPI1_MOSI pin and connects to CMD of SD card slot.
- PC.14: A GPIO pin sets as input mode, it is card detect pin and connects to CD of SD card slot.
- PC.15: A GPIO pin sets as output mode, it connects to VDD of SD card slot.

4 Directory Information

- EC_ NUC230_240_FATFS_SDCard _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 NUC230_240_FATFS_SDCard.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. 21, 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.