AN11985

LPC546xx SD Card with FATFS

Rev. 1.0 — 21 June 2017 **Application note**

Document information

Info	Content
Keywords	LPC546xx, SD/MMC card, FATFS, LPCXpresso54608 Eval Board Rev A, Keil MDK, IAR Embedded Workbench, MCUXpresso
Abstract	This application note introduces the SD/MMC card interface on LPC546xx devices and how to access the SD card with the SD/MMC interface based on SDK FATFS. It also shows how to create a directory and create, write and read a file. The source code is provided in LPC546xx SDK2.0.

Revision history

Rev	Date	Description
1.0	20170621	Initial version

Contact information

For more information, please visit: http://www.nxp.com

1. Overview

The LPC546xx is a family of ARM Cortex-M4 based microcontrollers running at frequencies of up to 180 MHz for embedded applications featuring a rich peripheral set, including SD/MMC interface with very low power consumption and enhanced debug features. The family includes up to 512 KB of flash, 200 KB of on-chip SRAM.

The SD/MMC card interface is available on all LPC546xx devices and supports the following features:
• Secure Digital memory, Secure Digital I/O, Multimedia Card and CE-ATA digital protocol commands.
• One SD (1.1) or MMC (4.4), CE-ATA (1.1), or eMMC (4.4) device.
• Internal (bus mastering) DMA.
• TX and RX FIFO (FIFO depth = 32 and FIFO data width = 32 bits).
• FIFO over-run and under-run prevention by stopping card clock.
• Block size of 1 to 65,535 bytes
• Supports up to a maximum of 52 MHz of interface frequency.

SD card is a kind of memory card used for storing information. There are different kinds of cards:
• Those labeled SD with capacities up to 2 GB.
• Those labeled SDHC with capacities between 4GB and 32GB.
• Those labeled SDXC with capacities of up to 2 TB (largest made is currently 512GB or 256 GB for MicroSD).
SD and SDHC are not compatible, but devices that accept SDHC also accept SD cards. The interface of SDHC and SDXC cards is the same, but SDXC uses a different file system. There are also different classes (Class 2/4 6…). These refer to the read and write speeds.

FATFS is a generic FAT/exFAT file system module for small embedded systems. The FATFS module is written in compliance with ANSI C (C89) and completely separated from the disk I/O layer. Therefore, it is independent of the platform. It can be incorporated into small microcontrollers with limited resource, such as 8051, PIC, AVR, ARM, Z80, 78K etc.

This application note introduces the SD/MMC card interface on LPC546xx device and how to access the SD card with the SD/MMC interface based on FATFS (R0.12b). It also shows how to create a directory and create, write and read a file.

2. SD/MMC interface

2.1 Block

The SD/MMC controller interface consists of the following main functional blocks:
• Bus Interface Unit (BIU) - Provides AHB and DMA interfaces for register and data read/writes.
• Card Interface Unit (CIU) - Handles the card protocols and provides clock management.
• Internal MCI DMA controller: AHB bus mastering DMA controller. See Fig 1.

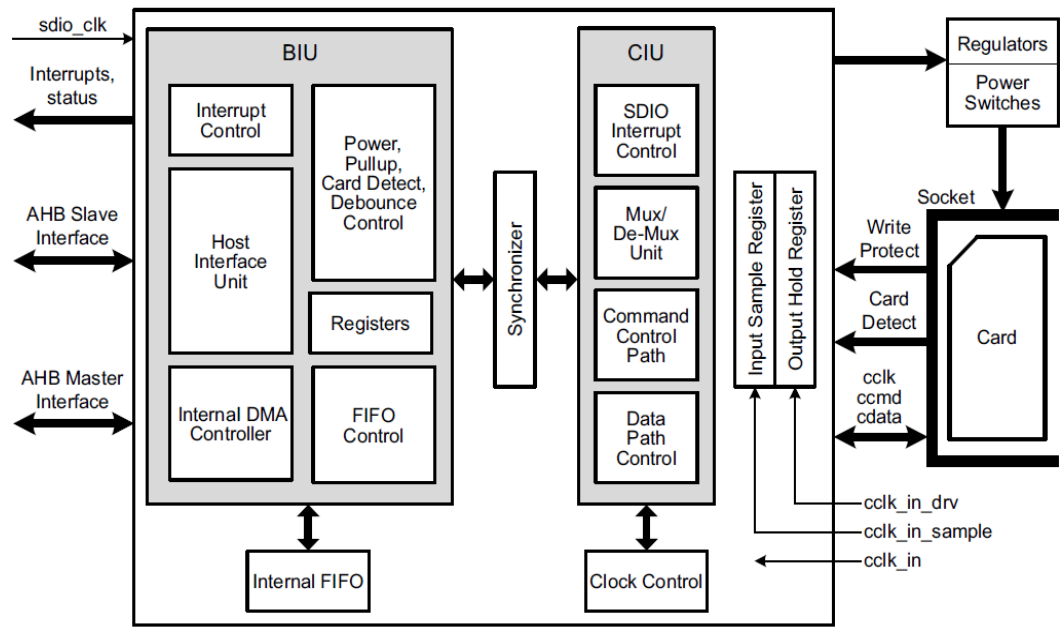

Fig 1. SD/MMC block diagram

Note: The Card Detect and Write Protect signals are from the SD/MMC card socket, not the SD/MMC card itself.

2.2 Pin description

Fig 2 describes the pins used for SD/MMC interface:

Pin function	Type	Description
SD_CLK	O	SD/SDIO/MMC clock.
SD_CARD_DET_N	I	SDIO card detect for single slot. A 0 represents the presence of a card.
SD_WR_PRT	I	SDIO card write protect. A 1 represents write is protected.
SD_CMD	I/O	Command input/output.
SD_D[7:0]	I/O	Data input/output for data lines DAT[7:0].
SD_VOLT[2:0]	O	SD/MMC bus voltage select output 2:0. SD/MMC General Purpose Output pins on pins SD_VOLT0, SD_VOLT1, and SD_VOLT2. These pins can be used to control an optional external regulator for the SD/MMC slot. If an external regulator is used to control the SD/MMC slot, voltage level translation will be needed between the IO pads and the SD/MMC slot.
SD_POW_EN	O	SD/SDIO/MMC slot power enable.
SD_BACKEND_PWR	O	Back-end power supply for embedded device. Controls back-end power supply for one embedded device; this bit does not control the VDDH of the host controller. A register bit enables software programming. The value on this register controls switching on and off of power to embedded device.
SD_CARD_INT_N	I	Card interrupt line. This pin is used to indicate a card interrupt, which is sampled even when the clock to the card is switched off. Connected to the eSDIO card interrupt line; it is defined only for eSDIO.

Fig 2. SD/MMC pin description

Typically, the pins of SD_CLK, SD_CARD_DET_N, SD_WR_PRT, SD_CMD, SD_D[7:0] and SD_POW_EN are used for interface to the SD card slot. For the pins settings, see the section "Hardware environment"

3. Implementation

This section introduces how to access a SD card with the SD/MMC interface on LPC546xx devices based on FATFS. Typically, 4-bit mode of SD interface is used to access a SD card with a single voltage (3.3V).

3.1 Hardware design

Fig 3 shows the schematic reference design of SD/MMC interface. It is from the LPCXpresso54608 Eval Board Rev A (OM13092). The link is:

http://www.nxp.com/products/software-and-tools/software-development-tools/software-tools/lpcxpresso-boards/lpcxpresso-board-for-lpc54608:OM13092

SHT 4 – connect to the pins of SD/MMC interface of LPC546xx

J7 – is the SDIO slot which can plug in the SD card

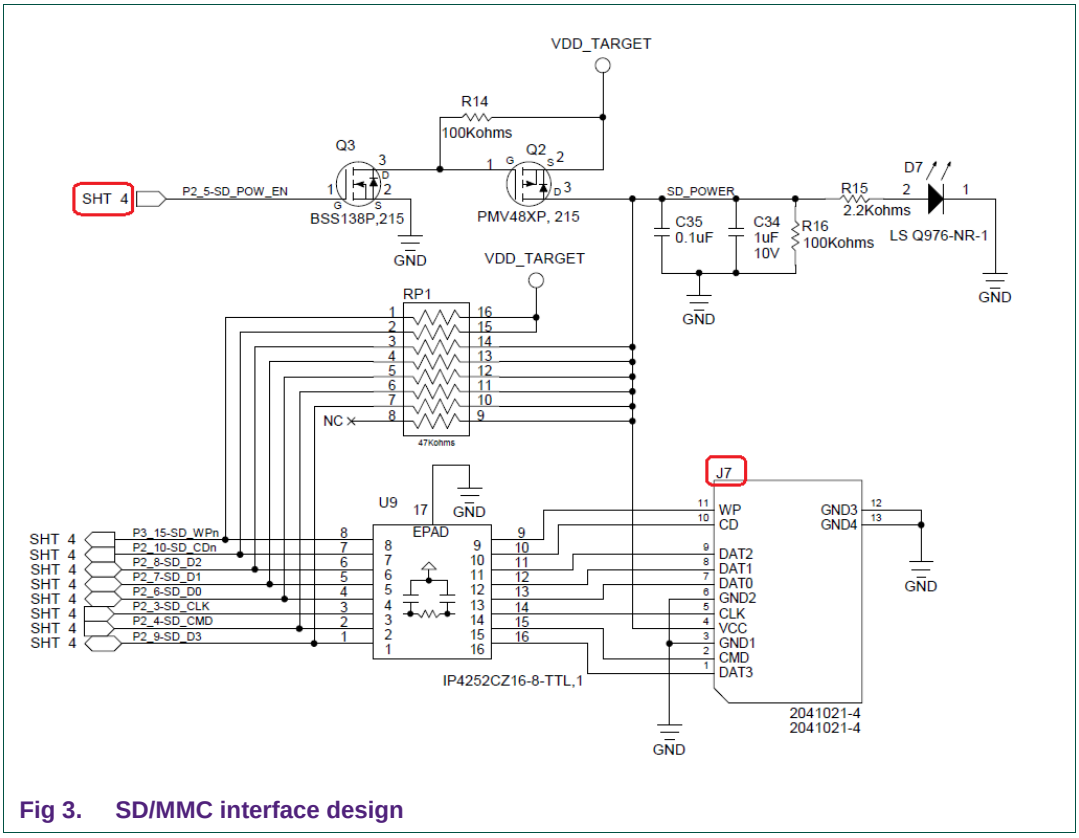

Fig 3. SD/MMC interface design

3.2 Software implementation

3.2.1 Software architecture

The software of SD/MMC interface is implemented in the *SDK_2.0_LPC54608J512 (downloading link:* https://mcuxpresso.nxp.com/en/welcome*)*. The software architecture of SD/MMC peripheral interface corresponds to the Peripheral Drivers layer, Stack/Middleware & Board Support layer of the SDK2.0 architecture. It expands to SD/MMC interface, Host interface, protocol commands interface, FATFS file system and board support. See the Fig 4:

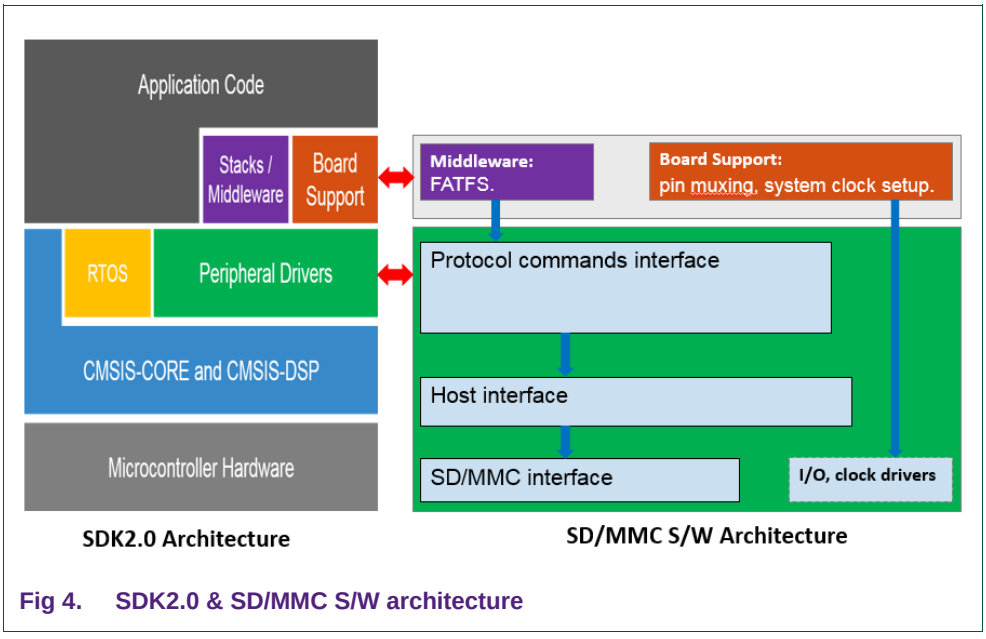

Fig 4. SDK2.0 & SD/MMC S/W architecture

Remark:

- Protocol commands interface relates to the files fsl_sdmmc.c, fsl_sd.c, fsl_mmc.c, fsl_sdio.c.
- Host interface relates to the file fsl_host.c
- SD/MMC interface relates to the file fsl_sdif.c

3.2.2 Pin settings

Fig 5 shows the suggested pin settings for SD_CLK, SD_CMD and SD_Dn.

IOCON bit(s)	Type D pin
11	OD: Set to 0 unless open-drain output is desired.
10	SLEW: Set to 1.
9	FILTEROFF: Set to 1.
8	DIGIMODE: Set to 1.
7	INVERT: Set to 0.
6	Not used, set to 0.
5:4	MODE: set to 0. Inactive(no pull-down/pull-up resistor enabled)
3:0	FUNC: Must select the correct function for this peripheral.

Fig 5. SD/MMC pin settings for SD_CLK, SD_CMD, SD_D[7:0]

Note: It is important to configure the other 4 data bus IOs, even if you do not use the 8-bit mode. If not configured, the 4-bit mode will not work properly

For SD_CARD_DET_N, SD_WR_PRT and SD_POW_EN pins, select the SD/MMC function, set DIGIMODE bits to digital mode, and set MODE bits to inactive.

3.2.3 Chip low level device control

This section introduces how to initialize the SD/MMC interface and transfer data using the peripheral.

3.2.3.1 SD/MMC initialization

In the initialization process of SD/MMC peripheral the following needs to be configured: need to enable clock, perform software reset, configure some settings for transfer and enable the interrupts, and clear interrupt status. Fig 6 shows the process.

```
/* enable SDIF clock */
CLOCK_EnableClock(kCLOCK_Sdio);

/* do software reset */
base->BMOD |= SDIF_BMOD_SWR_MASK;

/* reset all */
SDIF_Reset(base, kSDIF_ResetAll, SDIF_TIMEOUT_VALUE);

/*config timeout register */
timeout = base->TMOUT;
timeout &= ~(SDIF_TMOUT_RESPONSE_TIMEOUT_MASK | SDIF_TMOUT_DATA_TIMEOUT_MASK);
timeout |= SDIF_TMOUT_RESPONSE_TIMEOUT(config->responseTimeout)
          | SDIF_TMOUT_DATA_TIMEOUT(config->dataTimeout);

base->TMOUT = timeout;

/* config the card detect debounce clock count */
base->DEBNCE = SDIF_DEBNCE_DEBOUNCE_COUNT(config->cardDetDebounce_Clock);

/*config the watermark/burst transfer value */
base->FIFOTH = SDIF_FIFOTH_TX_WMARK(SDIF_TX_WATERMARK)
             | SDIF_FIFOTH_RX_WMARK(SDIF_RX_WATERMARK) | SDIF_FIFOTH_DMA_MTS(1U);

/* enable the interrupt status  */
SDIF_EnableInterrupt(base, kSDIF_AllInterruptStatus);

/* clear all interrupt/DMA status */
SDIF_ClearInterruptStatus(base, kSDIF_AllInterruptStatus);
SDIF_ClearInternalDMAStatus(base, kSDIF_DMAAllStatus);
```

Fig 6. SD/MMC initialization

3.2.3.2 Transfer

The data can be transferred with non-blocking using DMA. The major steps to do it are:

1. Configure the transfer parameter:
   - ✓ Configure the block size in the register BLKSIZ (bit 15:0) and byte count for transfer in the register BYTCNT(bit 31:0).
   - ✓ Configure the card command flags (see Fig 7) based on the transfer requests and response type for the SD card.

Symbol	Value	Description
RESPONSE_EXPECT		Response expect
	0	None. No response expected from card
	1	Expected. Response expected from card
RESPONSE_ LENGTH		Response length
	0	Short. Short response expected from card
	1	Long. Long response expected from card
CHECK_RESPONSE_CRC		Check response CRC.
	0	Do not check response CRC
	1	Check response CRC
DATA_EXPECTED		Data expected
	0	None. No data transfer expected (read/write)
	1	Data. Data transfer expected (read/write)
READ_WRITE		read/write. Don't care if no data expected from card.
	0	Read from card
	1	Write to card
TRANSFER_MODE		Transfer mode. Don't care if no data expected.
	0	Block data transfer command
	1	Stream data transfer command
SEND_AUTO_STOP		Send auto stop.
	0	No stop command sent at end of data transfer
	1	Send stop command at end of data transfer
WAIT PRVDATA COMPLETE		Wait prvdata complete.
	0	Send. Send command at once, even if previous data transfer has not completed.
	1	Wait. Wait for previous data transfer completion before sending command.
STOP ABORT CMD		Stop abort command.
	0	Disabled.
	1	Enabled.
UPDATE_CLOCK_REGISTERS_ ONLY		Update clock registers only.
	0	Normal. Normal command sequence
	1	No. Do not send commands, just update clock register value into card clock domain
USE_HOLD_REG		Use Hold Register.
	0	CMD and DATA sent to card bypassing HOLD register.
	1	CMD and DATA sent to card through the HOLD register.

Fig 7. Command flags

The related register CARDTHRCTL is used for configuring the card threshold size (bit 23:16) when the bit 0 in the register is set to enable the card read threshold.

For more details, see the function “SDIF_TransferConfig()” in fsl_sdif.c.

2. Use internal DMA mode to transfer between the card and host:

Configure the DMA descriptor for data transfer per the definitions of the selectable descriptor structure (See LPC546xx UM).

After configuring the descriptors, prepare the internal DMA controller for data transfer.

Finally, load the base address of the First Descriptor to the DBADDR register.

The process is shown in Fig 8:

```
/* use internal DMA interface */
base->CTRL |= SDIF_CTRL_USE_INTERNAL_DMAC_MASK;
/* enable the internal SD/MMC DMA */
base->BMOD |= SDIF_BMOD_DE_MASK;
/* enable DMA status check */
base->IDINTEN |= kSDIF_DMAAllStatus;
/* clear write/read FIFO request interrupt in DMA mode,
   DMA will handle the data transfer*/
SDIF_DisableInterrupt(base, kSDIF_WriteFIFORequest
                      | kSDIF_ReadFIFORequest | kSDIF_DataTransferOver);
/* load DMA descriptor buffer address */
base->DBADDR = (uint32_t)config->dmaDesBufferStartAddr;
```

Fig 8. Configure internal DMA controller

3. Send command to card

Load the command containing argument, index and control parameter configured in Step 1 to the related registers and wait for the command to be taken by CIU within timeout. See the function “SDIF_SendCommand()” of fsl_sdif.c in which the reference codes are shown in Fig 9:

```
base->CMDARG = cmd->argument;

base->CMD = SDIF_CMD_CMD_INDEX(cmd->index) |
            SDIF_CMD_START_CMD_MASK | (cmd->flags &
            (~SDIF_CMD_CMD_INDEX_MASK));
```

Fig 9. Load command to CMDARG & CMD registers

Remark:

(1) The “cmd->argument” set to the register CMDARG indicates the command argument to be passed to card, such as, block size, byte count.

(2) The “cmd->index” means the card commands. The commands used in SDK for SD/MMC card are listed in Fig 10:

```
/*! @brief SD card individual commands */
kSD_SendRelativeAddress = 3U,     /*!< Send Relative Address */
kSD_Switch = 6U,                  /*!< Switch Function */
kSD_SendInterfaceCondition = 8U, /*!< Send Interface Condition */
kSD_EraseWriteBlockStart = 32U,  /*!< Write Block Start */
kSD_EraseWriteBlockEnd = 33U,     /*!< Write Block End */
/*! @brief SD card individual application commands */
kSD_ApplicationSetBusWdith = 6U,              /*!< Set Bus Width */
kSD_ApplicationSendOperationCondition = 41U,  /*!< Send Operation Condition */
kSD_ApplicationSendScr = 51U,                 /*!< Send Scr */
/*! @brief SD/MMC card common commands */
kSDMMC_GoIdleState = 0U,          /*!< Go Idle State */
kSDMMC_AllSendCid = 2U,           /*!< All Send CID */
kSDMMC_SelectCard = 7U,           /*!< Select Card */
kSDMMC_SendCsd = 9U,              /*!< Send CSD */
kSDMMC_StopTransmission = 12U,    /*!< Stop Transmission */
kSDMMC_SendStatus = 13U,          /*!< Send Status */
kSDMMC_GoInactiveState = 15U,     /*!< Go Inactive State */
kSDMMC_SetBlockLength = 16U,      /*!< Set Block Length */
kSDMMC_ReadSingleBlock = 17U,     /*!< Read Single Block */
kSDMMC_ReadMultipleBlock = 18U,   /*!< Read Multiple Block */
kSDMMC_SendTuningBlock = 19U,     /*!< Send Tuning Block */
kSDMMC_SetBlockCount = 23U,       /*!< Set Block Count */
kSDMMC_WriteSingleBlock = 24U,    /*!< Write Single Block */
kSDMMC_WriteMultipleBlock = 25U, /*!< Write Multiple Block */
kSDMMC_Erase = 38U,               /*!< Erase */
kSDMMC_ApplicationCommand = 55U, /*!< Send Application Command */
```

Fig 10. SD/MMC card commands

For non-blocking data transfer mode, the result (success or error) is decided and handled by checking the interrupt status in the MINTSTS and IDSTS (for DMA) registers in the ISR.

3.2.4 File control and access

With the basic drivers of SD, file system (here is FATFS) can manage the SD card via SD/MMC interface and provide APIs to the application program for controlling and accessing the directory/files. The general process and operations to the directory/files in FATFS are as below:

(1) Mount a logical drive firstly.

(2) Then create FAT file system on the logical drive.

(3) After the step (1) & (2), user can access a directory and files, such as create and open a directory before reading it to list the files in it, create and open a file before reading and writing to it.

The related codes are shown in Fig 11:

```
/* Mount a Logical Drive */
f_mount(&g_fileSystem, driverNumberBuffer, 0U);
/* Create FAT file system on the logical drive */
f_mkfs(driverNumberBuffer, FM_ANY, 0U, work, sizeof work);
/* Create a Directory */
f_mkdir(_T("/dir_1"));
/*Create a File */
f_open(&g_fileObject, _T("/dir_1/f_1.dat"), (FA_WRITE | FA_READ | FA_CREATE_ALWAYS));
/* Open a Directory Object */
f_opendir(&directory, "/dir_1");
/* Read Directory for file information in it */
f_readdir(&directory, &fileInformation);
/* Write File */
f_write(&g_fileObject, g_bufferWrite, sizeof(g_bufferWrite), &bytesWritten);
/* Move the file pointer */
f_lseek(&g_fileObject, 0U);
/* Read File */
f_read(&g_fileObject, g_bufferRead, sizeof(g_bufferRead), &bytesRead);
```

Fig 11. Access a directory/file with FATFS

4. Demonstration

4.1 Hardware environment

- **Board**
  - ✓ LPCXpresso54608 Eval board Rev A (OM13092)

    (link: http://www.nxp.com/products/software-and-tools/software-development-tools/software-tools/lpcxpresso-boards/lpcxpresso-board-for-lpc54608:OM13092)
- **Debugger**
  - ✓ Integrated CMSIS-DAP debugger on the board
- **Miscellaneous**
  - ✓ 1 Micro USB cable
  - ✓ SD card (SDHC with Class 4, 8GB)
- **Board Setup**
  - ✓ Connect the Micro USB between PC and J8 printed “Debug Link” on the board for loading and running demo. This is also used for UART communication to a PC terminal. See Fig 12.

✓ Plug SD card in J7 (SD slot) on the board. See Fig 12.

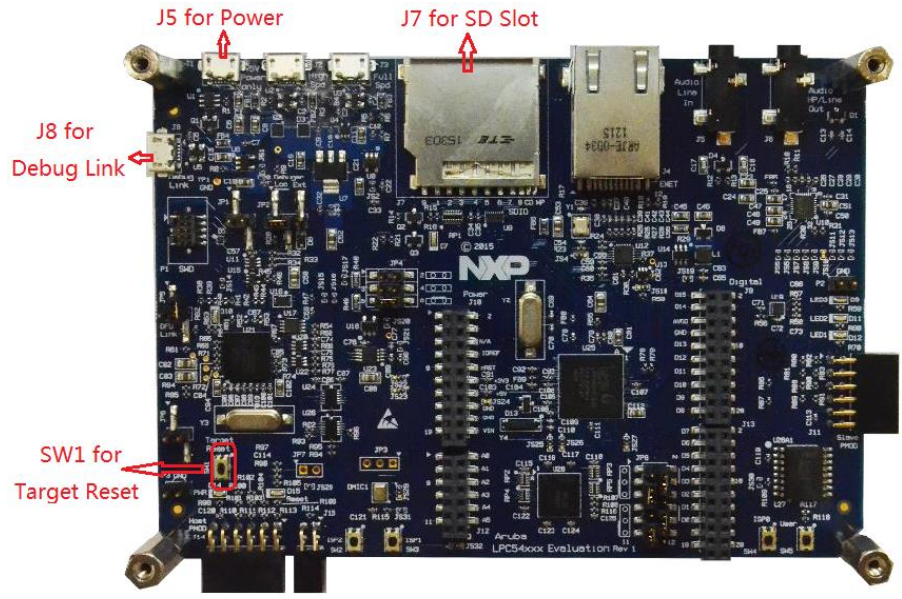

Fig 12. Board setup

4.2 Software environment

- **Tool chain**
  - ✓ MDK-ARM Professional V5.20 and above (IDE - uVision V5.20)
  - ✓ IAR embedded Workbench 7.70.2
  - ✓ MCUXpresso IDE
- **Software package**
  - ✓ SDK_2.0_LPC54608J512 (link: *https://mcuxpresso.nxp.com/en/welcome)*
- **UART terminal program**
  - ✓ Such as PuTTy

4.3 Steps and result

This Demo mounts a file system based on a SD card and then creates a directory, reads the directory, creates a file, writes to the file and reads the file. To demonstrate this, the major steps are:

1. **Build and download**

- Open and build the "*sdcard_fatfs*" project (before this step make sure the hardware environment has been setup as mentioned in the section "4.1 Hardware environment")

- Download the executable file using the debugger.

2. **Setup for UART terminal program**

- Check the COM number which is simulated as LPC-LinkII in “Device Manager” on your computer as shown below in Fig 13:

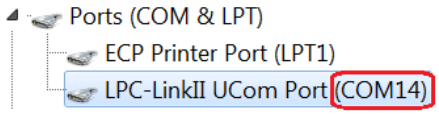

Fig 13. COM number

- Open the UART terminal program on your computer. Select the COM number as what is shown on the above step and configure the communication protocol as 115200+8+N+1.

3. **Run**

- Reset the board to run the board by pushing the SW1 button printed “Target Reset” on the board.

On the UART terminal, the following information will be displayed as shown in Fig 14.

```
Create directory......

Create a file in that directory......

Create a directory in that directory......

List the file in that directory......
General file : F_1.DAT.
Directory file : DIR_2.

Write/read file until encounters error.....

Write to above created file.
Read from above created file.
Compare the read/write content......
The read/write content is consistent.

Input 'q' to quit read/write.
Input other char to read/write file again.
q
The example will not read/write file again.
```

Fig 14. Result printed on UART terminal display

- Check the contents created in the SD card in the PC. To do it, input “q” to quit the demo. After inserting the SD card into SD card slot on the PC, the created directory and file can be seen in windows explorer. After opening the file with a proper tool on PC, strings of “a” is seen as written to the file. See Fig 15.

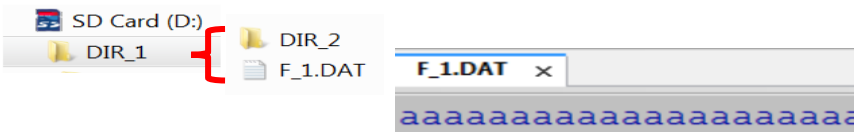

Fig 15. Directory, file and content created on SD card

5. Conclusion

The SD/MMC card interface is available on all LPC546xx devices and can support one SD (1.1) device with internal DMA controller. Based on LPCXpresso54608 Eval Board Rev A, accessing a SD card through the SD/MMC interface with FATFS (R0.12b) is implemented in LPC54608 SDK2.0.

This application note introduces how to design the hardware and software architecture with the SD/MMC interface. Typically, the internal DMA with non-blocking mode is used to transfer data between the SD/MMC interface and a SD card.

This application note demonstrates and verifies the function with a SDHC card (8GB, Class 4).

6. Legal information

6.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

6.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

6.3 Licenses

Purchase of NXP <xxx> components
<License statement text>

6.4 Patents

Notice is herewith given that the subject device uses one or more of the following patents and that each of these patents may have corresponding patents in other jurisdictions.

<Patent ID> — owned by <Company name>

6.5 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

<Name> — is a trademark of NXP Semiconductors N.V.

7. Index

No index entries found.

8. List of figures

9. List of tables

No table of figures entries found.

10. Contents

Date of release: 21 June 2017
Document identifier: AN11985