AN12777
Access dual SD cards on LPC55S6x

Rev. 0 — March 16 2020 Application Note

Contents

1 Introductions

1.1 Overview

The SDMMC card interface is available on all LPC55S6x/LPC55S2x/LPC552x devices.

The SDMMC card interface supports interface to two devices (SD0 and SD1) with the same SDMMC peripheral.

This application note introduces how to use the same SDMMC peripheral to access dual SD cards based on LPCXprsso55S69 SDK.

1.2 SDMMC blocks

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.

Figure 1 shows the block diagram.

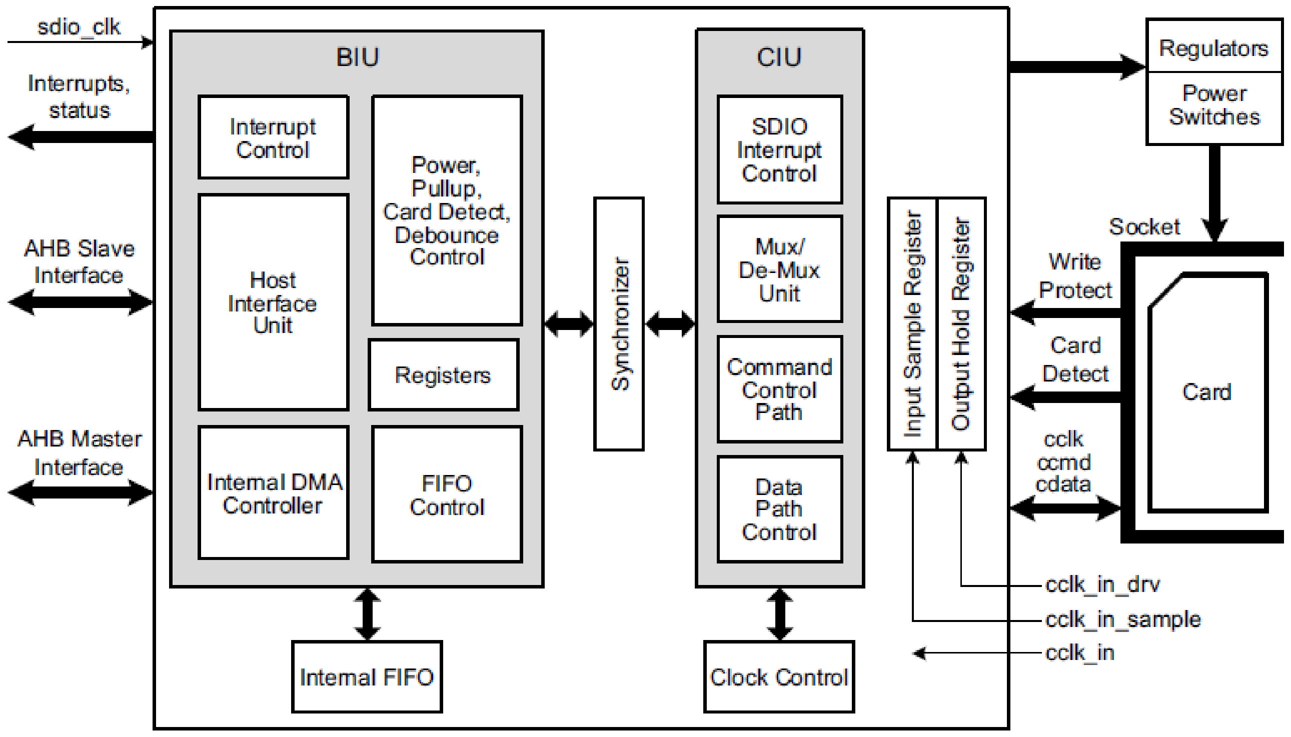


Figure 1. SDMMC block diagram

1.3 SDMMC pin description

Table 1 describes the available pins functions of SDMMC interface. Generally, `SDn_CLK`, `SDn_CARD_DET_N`, `SDn_CMD`, and `SDn_D` (n= 0, 1) pins are required to interface to a device and others are optional.

NOTE

`SD0_D` supports up to 8-bit data and `SD1_D` supports up to 4 bits. Both support 1-bit mode.

Table 1. SDMMC pin description

Pin function	Type	Description
`SD0_CLK`, `SD1_CLK`	O	SD/SDIO/MMC clock
`SD0_CARD_DET_N`, `SD1_CARD_DET_N`	I	SDIO card detect for signal slot. 0 indicates that there is a card.
`SD0_WR_PRT`	I	SDIO card writes protect 1 indicates that the write operation is protected.
`SD0_CMD`, `SD1_CMD`	O/I	Command input/output
`SD0_D[7:0]`, `SD1_D[3:0]`	O/I	Data input/output for data lines `DAT[7:0]`
`SD0_POW_EN`, `SD1_POW_EN`	O	SD/SDIO/MMC slot power enabled
`SD1_BACKEND_PWR`	O	Back-end power supply for embedded device

Table continues on the next page...

Table 1. SDMMC pin description (continued)

Pin function	Type	Description
		It controls back-end power supply for one embedded device. This bit does not control the VDDH of the host controller. A register bit enables the software programming. The value on this register controls to switch on and off the embedded device.
`SD0_CARD_INT_N`, `SD1_CARD_INT_N`	I	Card interrupt line This plan is used to indicate a card interrupt, which is sampled even when the closk to the card is switched off. It is connected to the eSDIO card interrupt line and defined only for eSDIO.

The SDMMC pins can be configured by I/O control registers (`IOCON`) like other peripherals. The pin settings for `SD0_CLK`, `SD0_CMD`, `SD0_Dn`, `SD1_CLK`, `SD1_CMD`, and `SD1_Dn` are suggested as Table 2.

Table 2. SDMMC suggested pin settings

IOCON bit(s)	Type D pin	Type A pin
10	Not used, set to 0.	Analog switch is open (disabled). Set to 0.
9	Controls open-drain mode. Set to 0	Same as Type D.
8	DIGIMODE: Set to 1.	Same as Type D.
7	INVERT: Set to 0	Same as Type D.
6	SLEW: Set to 1	Same as Type D.
5:4	Mode: Set to 0	Same as Type D.
3:0	FUNC: Must slect the correct function for this peripheral.	Same as Type D.
General comment	A good choice for SDIO functions.	A potential choice. The performance may be reduced by absence of the SLEW function.

NOTE

The pin function in Table 1 are configured by the value of bit 3:0 (FUNC field) in IOCON in Table 2. For the details, refer to the tables in **IOCON pin functions in relation to FUNC values** of *LPC55S6x/LPC55S2x/LPC552x User manual* (document UM11126).

2 Development and test environment

To show the feature of dual SD cards support, the software and hardware environments are built as below.

2.1 Software environment

- LPCXpresso55S69 SDK with middleware of SDMMC support (v2.63)
- KEIL MDK v5.27
- A serial terminal program (e.g. PuTTY) with the settings: 115200 + 8 + N + 1

2.2 Hardware environment and setup

- LPCXpresso55S69 EVK board Rev2.0
- Two SD cards
- Oone more board to support SD card with a card slot
- Personal computer
- Micro USB cable and some jump wires

The LPCXpresso55S69 EVK board is designed to support one SD card with one SD card slot connected to the SD0 pins. And it reserves the pins related to SD1 on Arduino header for connecting to another SD card. Figure 2 shows the screenshot of the schematic and the related pins are highlighted in yellow.

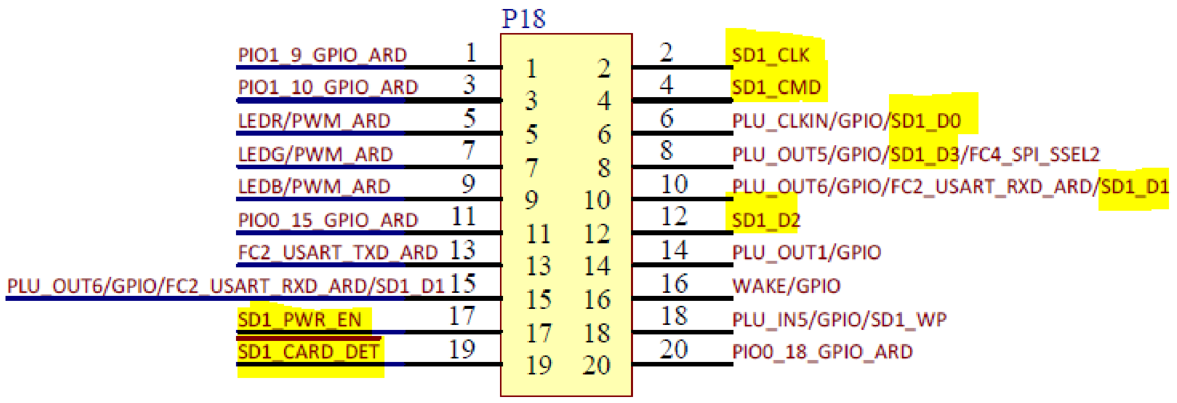


Figure 2. SD1 pins reserved on LPCXpresso55S69 EVK

To drive a SD card on another board with SD1 pins, power and ground should be shared as well. They are also reserved on an Arduino header of LPCXPrsso55S69 EVK, as shown in Figure 3, highlighted in yellow.

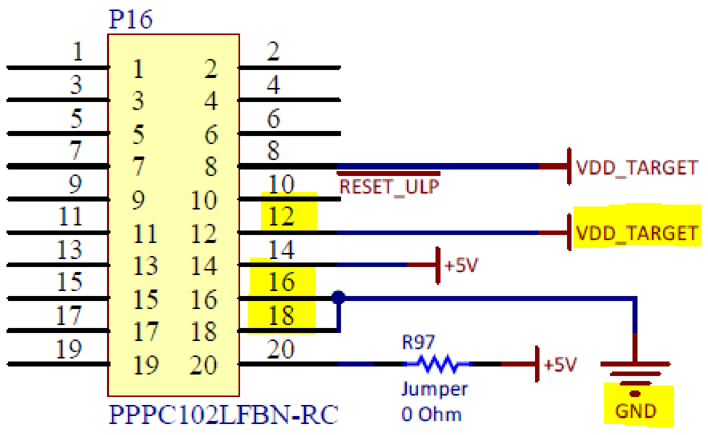


Figure 3. Power & GND pins reserved on LPCXpresso55S69 EVK

Figure 4 shows the screenshot of photo for actual SD1 connections on Arduino header on LPCXprsso55S69 EVK. With the jump wires, it is easy to connect to another board with a SD card slot for another card access.

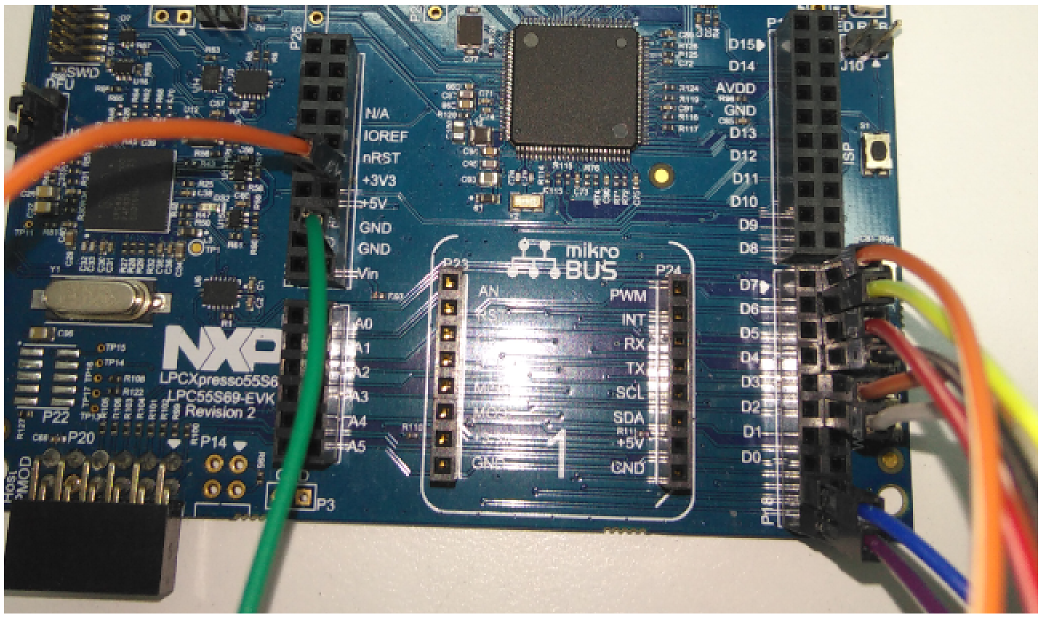


Figure 4. SD1 connections on Arduino header on LPCXpresso55S69 EVK

3 Software implementation

3.1 Basic ideas

The implementation of software is based on LPCXpresso55S69 SDK. The basic ideas for the SW implementation are as follows.

One SDMMC controller is shared with two devices although there are two sets of independent pins for them. So the access process to both devices must be serial not be parallel. It indicates that one transaction on one device should start only after the other is completed. Otherwise, two transactions could interfere to each other. This key point gives the restriction to the software implementation. The developer of application program should mind it as well.

Therefore, to show the feature more simply and clearly, the transactions on dual SD cards are implemented with polling and blocking mode in the reference codes.

Additionally, the current SDK only presents the functions to one SD card (SD0). To be better compatible to the SDK, as possible, the new functions to support another SD card (SD1) are created in new files instead of modifying the exist functions and files. Thus, these new files containing new functions can be added in SDK without interfering the existed files.

Lastly, the SDMMC peripheral are shared as mentioned, but there still presents some different configurations for SD0 and SD1 in registers:

- PWREN register for power enabling.
- CLKENA register for clock enabling.
- CDETECT register for card detection.
- CTYPE register for switching card type between 1-bit, 4-bit and 8-bit modes.
- When sending commands, bits 20-16 of the CMD register must be programmed with 0 or 1 for SD0 or SD1 respectively.

The controlling to `PWREN`, `CLKENA`, `CDETECT`, and `CTYPE` registers have been implemented at HAL level in SDK. For the introductions of the registers, refer to *LPC55S6x/LPC55S2x/LPC552x User manual* (document UM11126). In this software implementation for dual SD cards support, it is only required to call the APIs for SD1 support, e.g. SD1 initializations. However, sending a command to SD1 is not implemented in the transfer function of SDK. So it needs to add the transfer function of SD1. Table 3 describes related bit field of the CMD register.

Table 3. Command field for SD0 or SD1 in CMD register

But	Symbol	Value	Description	Reset value
20:16	`CARD_NUMBER`	0 or 1	Specifies the card number of SDCARD for which the current Command us being executed.	0

The new functions for SD1 enablement are added based on the above different registers configurations for both SD card devices (SD0 and SD1).

3.2 Adding functions on SDK for SD1 enablement

This section will introduce the points of how to add functions for SD1 enablement based on the SW levels of SDMMC in SDK: HAL level, Host level, protocol level and board level. For more details, refer to AN12777SW.

1. HAL level

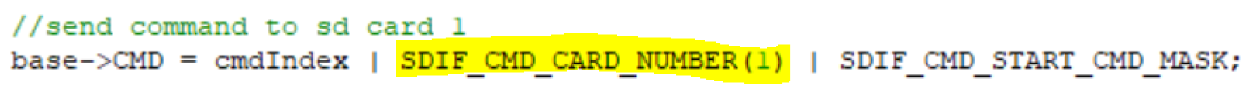
   HAL level of SDMMC interface in SDK is abstracted in the files of `fsl_sdif.c` and `fsl_sdif.h`, which contain the SDMMC registers settings.

   As mentioned in Basic ideas, except for sending command to SD1 via CMD register, all other different registers configurations for SD0 and SD1 have been implemented in SDK. Table 4 lists the function APIs for SD1.

Table 4. APIs of SD1 HAL in SDK

Function name	Description	Remark
`SDIF_EnableCard1Clock()`	Enable/disable SD card1 clock	Provided in `fsl_sdif.h`
`SDIF_EnableCard1Power()`	Enable/disable the SD card1 power	Provided in `fsl_sdif.h`
`SDIF_SetCard1BusWidth()`	Set SD card1 data bus width	Provided in `fsl_sdif.c`
`SDIF_DetectCard1Insert()`	Detect SD card1 insert status	Provided in `fsl_sdif.h`

   Create a function to send command to SD1 via CMD register. Refering to the existed function named `SDIF_SetCommandRegister()` for SD0, the new function is named as `SDIF1_SetCommandRegister()` where the SD1 is specified as the current card number (see Table 3 for the definition). The related code line is as shown in Figure 5. The highlighted in yellow is the settings of the bits.



```
//send command to sd card 1
base->CMD = cmdIndex | SDIF_CMD_CARD_NUMBER(1) | SDIF_CMD_START_CMD_MASK;
```

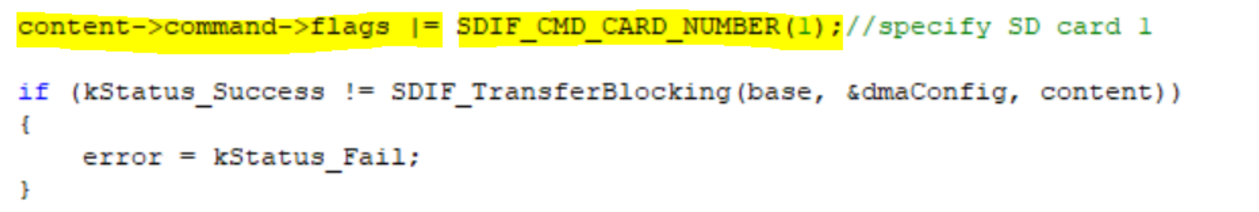
Figure 5. Setting command field for SD0 or SD1 in CMD register

   For more details, read the new file of `fsl_sdif1.c` including all the new functions of HAL level.

2. Host level

   Host level classified to middleware is the port between SDMMC HAL and protocol level. The basic system operations are done in this level, e.g. transfer function, card detection and card power switching. The reference code in this level is referring to the file of `fsl_sdmmc_host.c` with polling and blocking mode under the path of `\middleware\sdmmc\port\sdif \polling\`.

   A code line is added to specify the SD1 before transferring with blocking mode in the new transfer function for SD1 named `SDMMC1HOST_TransferFunction()`. See the highlighted in yellow in Figure 6.



```
content->command->flags |= SDIF_CMD_CARD_NUMBER(1);//specify SD card 1

if (kStatus_Success != SDIF_TransferBlocking(base, &dmaConfig, content))
{
    error = kStatus_Fail;
}
```

Figure 6. Setting Command field for SD0 or SD1 in CMD register

The function of `SDMMCHOST_DetectCard1InsertByHost()` for SD1 card detection and `SDMMCHOST_PowerOnCard1()`/ `SDMMCHOST_PowerOffCard1()` for SD1 power switching are to call individually the API `SDMMCHOST_CARD1_DETECT_INSERT_STATUS()` and `SDIF_EnableCard1Power()` which have been implemented in HAL level of SDK.

For more details, read the new file of `fsl_sdmmc_host1.c` including all the new functions of Host level.

3. **Protocol level**

   Protocol level classified to middleware implements the command protocol of SD card besides MMC and SDIO for being called by application program. Based on the different register configurations for SD0 and SD1 mentioned above, the new functions that need to be added for SD1 in this level are involved with SD1 initializations, SD1 clock & timing settings, SD1 power switching and SD1 detection which are implemented in a new file, `fsl_sd1.c`. They just call the APIs in HAL and Host level introduced above. Please read the file of `fsl_sd1.c` for the details.

   NOTE

   Because plenty of SD card protocol functions in the file of `fsl_sd.c` in SDK are defined as `static`. The new file of `fsl_sd1.c` will be included at the end of line of the file of `fsl_sd.c` in order to share the functions for reducing the code size.

4. **Board level**

   It needs to configure SD1 related pins as well since SD1 has its independent pins from SD0. This could be done according to the suggested settings in and referring to the configurations on SD0 related pins in SDMMC examples of SDK.

3.3 Updating SDK for dual SD cards support

The created source code and project information files for dual SD cards support are provided in package. They are required to merge into the SDK for working. Perform as below to update SDK for dual SD cards support.

- Copy the files of `fsl_sdif1.c` and `fsl_sdif1.h` under `\drivers\` in the attached SW package to `\devices\LPC55S69\drivers\` in the LPCXpresso55S69 SDK package.
- Copy the files of `fsl_sd1.h` and `fsl_sdmmc_host1.h` under `\sdmmc\inc\` in the attached SW package to `\middleware\sdmmc\inc\` in the LPCXpresso55S69 SDK package.
- Copy the files of `fsl_sdmmc_host1.c` under `\sdmmc\port\sdif\polling\` in the attached SW package to `\middleware\sdmmc\port\sdif\polling\` in the LPCXpresso55S69 SDK package.
- Copy the file of `fsl_sd1.c` under `\sdmmc\src` in the attached SW package to `\middleware\sdmmc\src\` in the LPCXpresso55S69 SDK package.

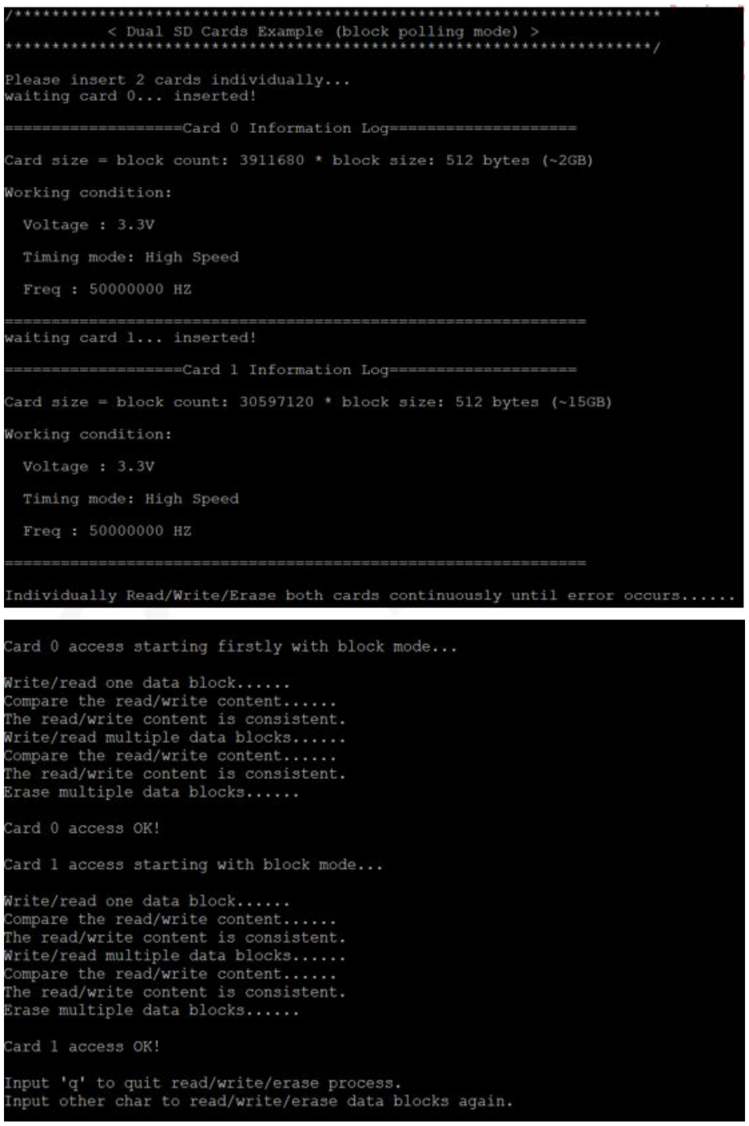
  Add `#include fsl_sd1.c` at the end of line of the file of `fsl_sd.c` in the LPCXpresso55S69 SDK package.
- Copy the folder of `dual_sdcards` containing keil MDK project information, example code and board level of codes in the attached SW package to `\boards\lpcxpresso55s69\demo_apps\` in the LPCXpresso55S69 SDK package.

3.4 Demonstrating dual SD cards access

Under `\boards\lpcxpresso55s69\demo_apps\dual_sdcards\`, there is a simple example implemented in the file of `dual_sdcards.c` to demonstrate the dual SD cards access.

In the example, after completing the basic configurations about system and peripheral of SDMMC, perform the following steps:

1. Keep detecting if one SD card (SD0) is inserted into the slot. Once detected, the card will be be powered on and initialized.
2. Print out some card information obtained in the initialization process.
3. Perform the same operations on the other SD card (SD1) on the other card slot. After both cards are initialized successfully, perform the access to SD0 card.
    a. Write one block of data to it and read it out.
    b. Compare if the data is consistent.
    c. Write/read/compare multiple block of data.
    d. If consistent, it indicates the access is successful. Or, it fails.
4. Perform the same access on SD1 card.
5. After setting up the hardware and software mentioned, open the project of `dual_sdcards.uvprojx` under the path `\boards\lpcxpresso55s69\demo_apps\dual_sdcards\cm33_core0\mdk\` in SDK. After building, downloading and running the demo successfully, the log for dual SD cards access will be seen on the serial terminal, as shown in Figure 7.



```
/************************************************************************
          < Dual SD Cards Example (block polling mode) >
************************************************************************/

Please insert 2 cards individually...
waiting card 0... inserted!

==================Card 0 Information Log====================

Card size = block count: 3911680 * block size: 512 bytes (~2GB)

Working condition:

  Voltage : 3.3V

  Timing mode: High Speed

  Freq : 50000000 HZ

==============================================================
waiting card 1... inserted!

==================Card 1 Information Log====================

Card size = block count: 30597120 * block size: 512 bytes (~15GB)

Working condition:

  Voltage : 3.3V

  Timing mode: High Speed

  Freq : 50000000 HZ

==============================================================

Individually Read/Write/Erase both cards continuously until error occurs......
```

```
Card 0 access starting firstly with block mode...

Write/read one data block......
Compare the read/write content......
The read/write content is consistent.
Write/read multiple data blocks......
Compare the read/write content......
The read/write content is consistent.
Erase multiple data blocks......

Card 0 access OK!

Card 1 access starting with block mode...

Write/read one data block......
Compare the read/write content......
The read/write content is consistent.
Write/read multiple data blocks......
Compare the read/write content......
The read/write content is consistent.
Erase multiple data blocks......

Card 1 access OK!

Input 'q' to quit read/write/erase process.
Input other char to read/write/erase data blocks again.
```

Figure 7. Logging for dual SD cards access

4 Summary

This application note elaborates on a way to implement dual SD cards support with the same one SDMMC interface on LPC55S6x family. It is presented in the reference codes based on LPCXpresso55S69 SDK with middleware of SDMMC support in which only one SD card is supported. Corresponding to the SDK, LPCXPresso55S69 EVK board is used for the enablement, as the other SD card’s pins are reserved for connecting easily.

- It introduces the SDMMC block and pin assignment for SD0 and SD1.
- It describes the software and hardware environment, especially hardware setup for the dual SD cards support.
- It explains how to add functions for the other SD cards support under the basic ideas, as one card has been supported in the SDK.
- It tells how to merge the software package into the SDK. Lastly, it shows the result with a simple example.

Date of release: March 16 2020

Document identifier: AN12777