Application note

Getting started with STM32H7 Series SDMMC host controller

Introduction

The SDMMC (secure digital multimedia card) host interface in the STM32H7 Series provides an interface between the AHB bus and SD (secure digital) memory cards, SDIO (secure digital input / output) cards and MMC (multimedia card) devices.

This application note describes as an example the SDMMC host interface specific to STM32H743/753 microcontrollers, and explains how to use the module to transfer data from/to SD, MMC and e-MMC memory cards in multiple configurations.

This document describes the SDMMC interaction with other internal peripherals, and presents typical examples that highlight the SDMMC host interface features that facilitate its configuration.

1 STM32H743/753 SDMMC host interface

The STM32H7 Series are Arm® based devices.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

STM32H7 Series include below SDMMC features:
- Supports SD, SDIO, MMC and e-MMC memory types.
- Supports data transfer in block(s) mode, SDIO multibyte mode and MMC stream mode.
- Full compliance with MultiMediaCard system specification version 4.51.
- Full compatibility with previous versions of MultiMediaCards (backward compatibility).
- Full compliance with SD memory card specification version 4.1 (SPI mode and UHS-II mode not supported).
- Full compliance with SDIO card specification version 4.0.
- Support data wide bus 1-bit , 4-bit and 8-bit modes.
- Data transfer up to 208 MHz depending on maximum allowed I/O speed (refer to product datasheet for more details).
- Have its own internal DMA (IDMA) for burst data transfer in single buffer or double buffer mode.

The table below presents an overview of the supported speed modes of the SDMMC host interface.

Table 1. **SDMMC supported speed modes**

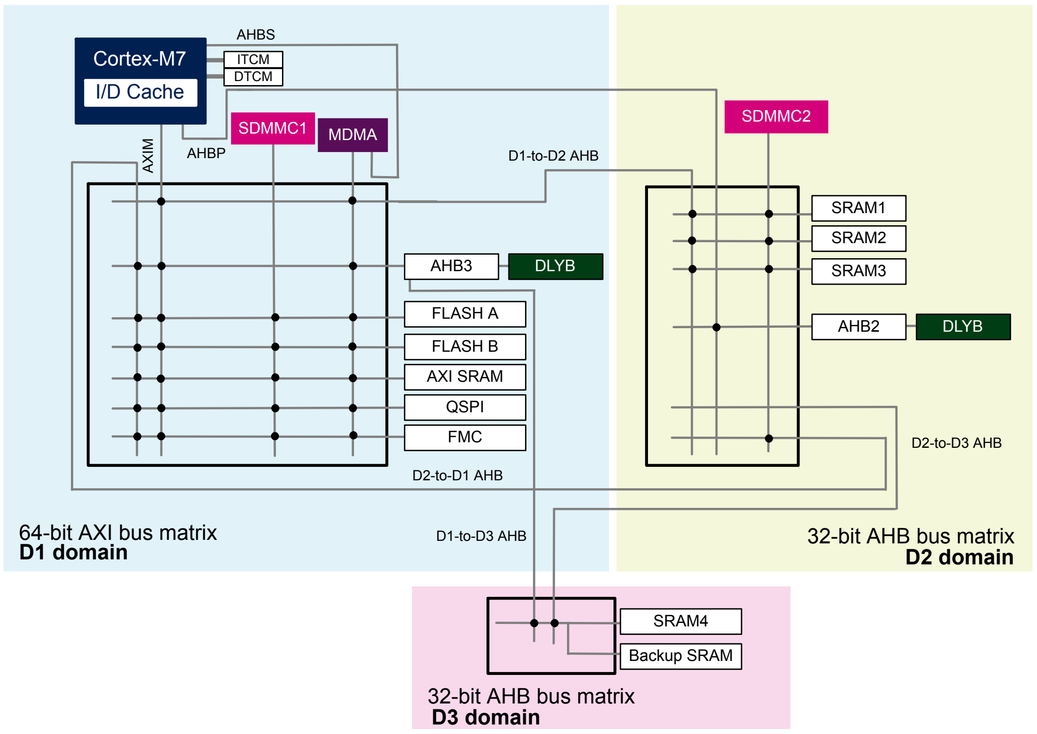
SD & SDIO	Max bus speed [Mbyte/s][1]	Max clock frequency [MHz][2]	Signal voltage (V)
DS (default speed)	12.5	25	3.3
HS (high speed)	25	50	3.3
SDR12	12.5	25	1.8
SDR25	25	50	1.8
SDR50	50	50	1.8
DDR50	50	100	1.8
SDR104	104	208	1.8
MMC cards			
Legacy compatible	26	26	3/1.8/1.2
High speed SDR	52	52	3/1.8/1.2
High speed DDR	104	104	3/1.8/1.2
High speed HS200	200	200	1.8/1.2

1. *Maximum bus speed in 4-bit mode for SD& SDIO and 8-bit mode for MMC cards.*
2. *The maximum data transfer depends on the maximum allowed I/O speed.*

1.1 SDMMC host interface in STM32H743/H753 architecture

All STM32H7 Series devices provide two SDMMC host interfaces: SDMMC1 and SDMMC2. Each interface has its own features. The figure below presents an extract from the STM32H7 Series architecture showing the SDMMC host interface integration.

Figure 1. SDMMC1 and SDMMC2 internal connections

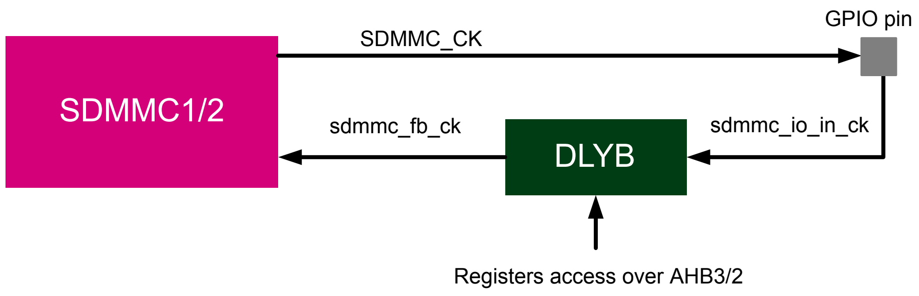


The SDMMC1 is in the D1 domain and the SDMMC2 is in the D2 domain, each one of them have a master interface connected respectively to the 64-bit AXI bus matrix and the 32-bit AHB bus matrix over an AHB master bus which can access different memories.

The SDMMC1 and SDMMC2 registers are accessible through their slave interface connected respectively to AHB3 and AHB2.

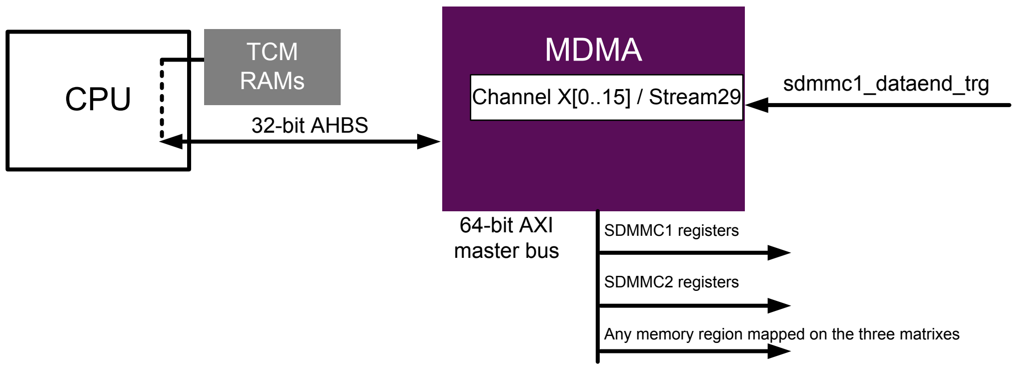
Each SDMMC have his own delay block (DLYB) accessible over AHB3 for SDMMC1 and over AHB2 for SDMMC2. The DLYB can be used to align the sampling clock on the data received (see figure below).

Figure 2. SDMMC and the delay block (DLYB)



The master DMA (MDMA) provides a channel for the SDMMC1 to enable successive data transfer from/to TCM RAMs and any memory region mapped on the three matrixes without any CPU action. The MDMA can also access the SDMMC1 and SDMMC2 registers and enable a new data transfer using Linked-list mode without any CPU action.

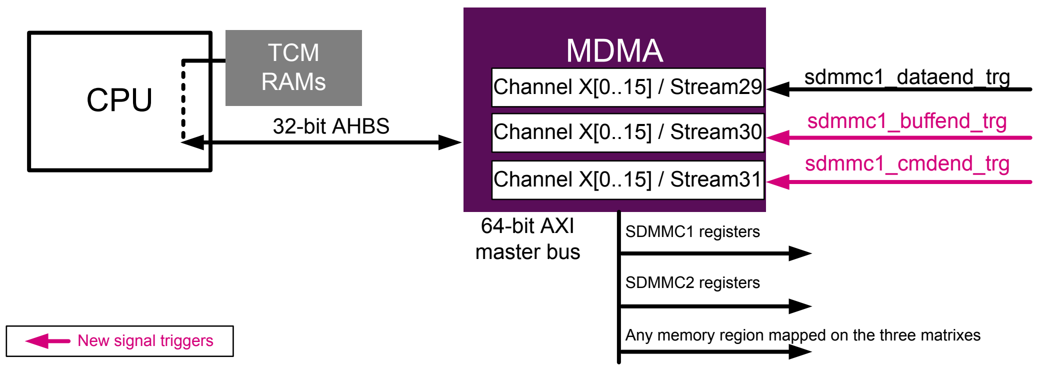
Figure 3. Master DMA (MDMA)



For the next STM32H7 Series revision, two signal triggers from the SDMMC to the MDMA will be added so the SDMMC will have three signals trigger for the MDMA:

- The data end trigger : sdmmc1_dataend_trg
- The buffer end trigger : sdmmc1_buffend_trg
- The command end trigger : sdmmc1_cmdend_trg

Figure 4. New signal triggers for MDMA and SDMMC



The SDMMC1 generate control signals for the external voltage switch transceiver to support the UHS-I mode.

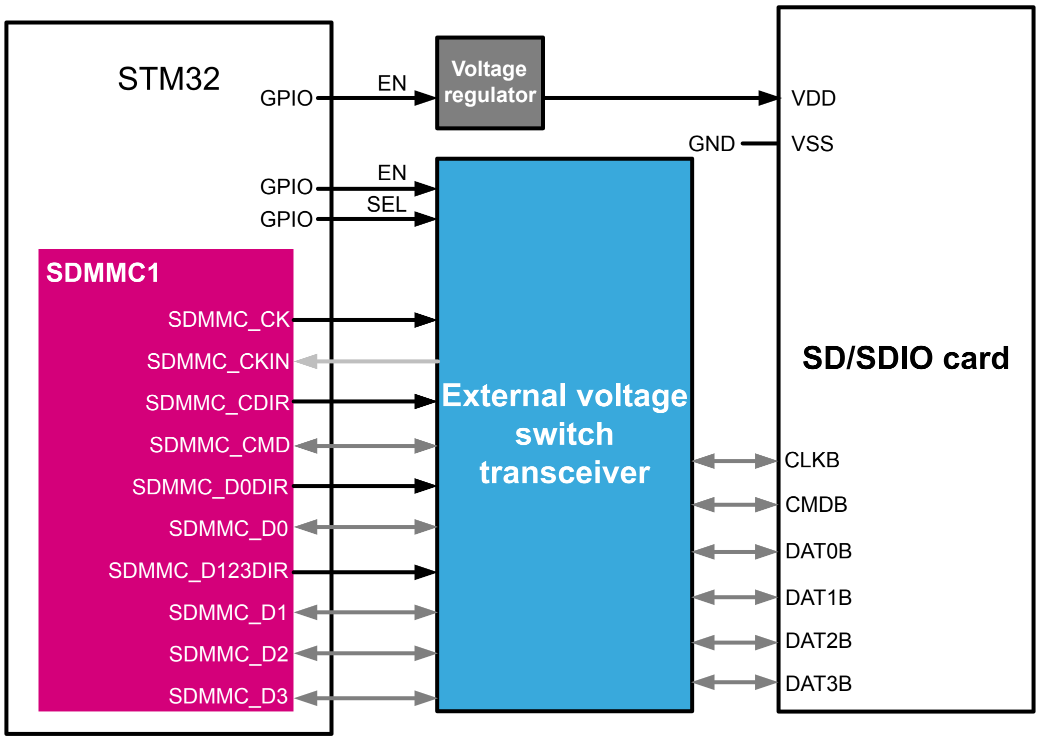


Figure 5. Voltage switch transceiver

The table below presents the main features for SDMMC1 and SDMMC2.

Table 2. SDMMC1 and SDMMC2 main features

Features	SDMMC1	SDMMC2
SDMMC master interface	Connected to the 64-bit AXI bus matrix in D1 domain through a 32-bit AHB master bus	Connected to the 32-bit AHB bus matrix in D2 domain through a 32-bit AHB master bus
SDMMC slave interface	Operating on the AHB3 bus	Operating on the AHB2 bus
Memory access	**Domain D1** • AXI SRAM • Quad-SPI • FMC	**Domain D1** • AXI SRAM • Quad-SPI • FMC **Domain D2** • SRAM1/SRAM2/SRAM3 **Domain D3** • SRAM4 • Backup SRAM
SDMMC delay block	Operating on the AHB3 bus	Operating on the SHB2 bus
MDMA	• The MDMA provides a channel for succesful end of data transfer • The MDMA can configure the SDMMC registers to enable a new data transfer	• The MDMA can configure the SDMMC registers to enable a new data transfer
UHS-I	Generates control signals to control the external voltage switch transceiver	No control signals generation for external transceiver(1)

1. *Can support external voltage switch transceiver with bus direction sensing (transceiver does not need direction control signals).*

1.2 SDMMC host interface block diagram

This section presents the SDMMC block diagram. The SDMMC internal input/output signals table and the SDMMC are described in the tables following the figure.

Figure 6. **SDMMC block diagram**

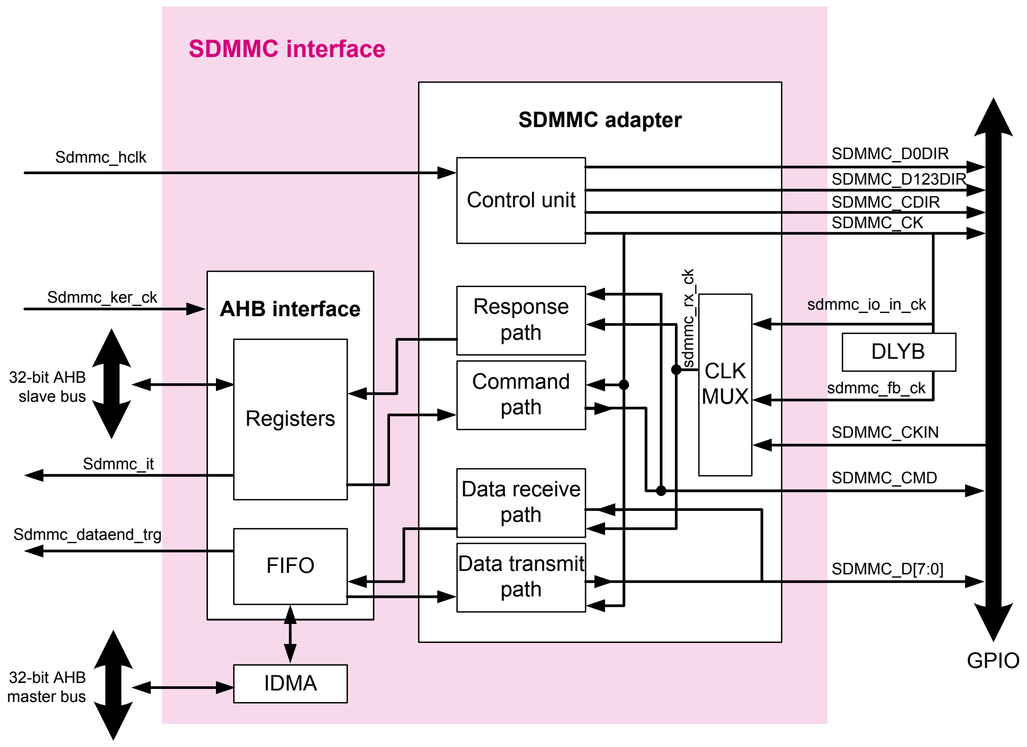


Table 3. **SDMMC internal input/output signals**

Signal name	Signal type	Description
Sdmmc_ker_ck	Digital input	SDMMC kernel clock
Sdmmc_hclk	Digital input	AHB clock
Sdmmc_it	Digital output	SDMMC global output
Sdmmc_dataend_trg	Digital output	SDMMC data end trigger
Sdmmc_rx_ck	Digital input	Selected clock for received data and responses
Sdmmc_io_in_ck	Digital input	Feedback clock Internally connected to SDMMC_CK
Sdmmc_fb_ck	Digital input	Feedback clock from the delay block (DLYB)

Table 4. **SDMMC pins**

Signal name	Signal type	Description
SDMMC_D0DIR	Digital output	Indicates the direction of SDMMC_D0 signal
SDMMC_D123DIR	Digital output	Indicates the direction of SDMMC_D[1:3] signals
SDMMC_CDIR	Digital output	Indicates the direction of SDMMC_CMD signal
SDMMC_CK	Digital output	Clock for the SD/SDIO/MMC card
SDMMC_CKIN	Digital input	Feedback clock from the external voltage transceiver
SDMMC_CMD	Digital input/output	Command and response line
SDMMC_D [7:0]	Digital input/output	Data transmission lines

The SDMMC host interface contains two main interfaces: the adapter interface and the AHB interface. These interfaces are described in the following sections.

1.2.1 Adapter interface

The Adapater interface uses the **sdmmc_ker_ck** domain and provides connection between the external card and the AHB interface. This interface contains:

- **Control unit**

  It groups the power management functions and the clock management with dividers. It generates the control signals for the external voltage switch transceiver.
- **Command/response path units**

  They transfer command and responses signals on the **SDMMC_CMD** line. The *command* path is clocked by the **SDMMC_CK** and the *response* path is clocked by the **sdmmc_rx_ck**.
- **Data receive/transmit path units**

  They transfer data on the SDMMC_D [7:0] lines. The *transmit* data is clocked by the **SDMMC_CK** and the *receive* data is clocked by the **sdmmc_rx_ck**.
- **CLK MUX unit**

  It selects the source clock for the *response* path and the *data receive* path.

1.2.2 AHB interface

This interface uses the **SDMMC_hclk** domain, it contains the AHB slave interface and the AHB master interface.

- **AHB slave interface**

  Provides access to the SDMMC registers and the FIFO, it also generates the interrupt requests and the data end trigger for the MDMA. The FIFO size is 32-bit wide, 16 words deep.
- **AHB master interface**

  Contains the internal direct memory access (IDMA) to provide high-speed data transfer between the FIFO and the memory.

IDMA (internal direct memory access)

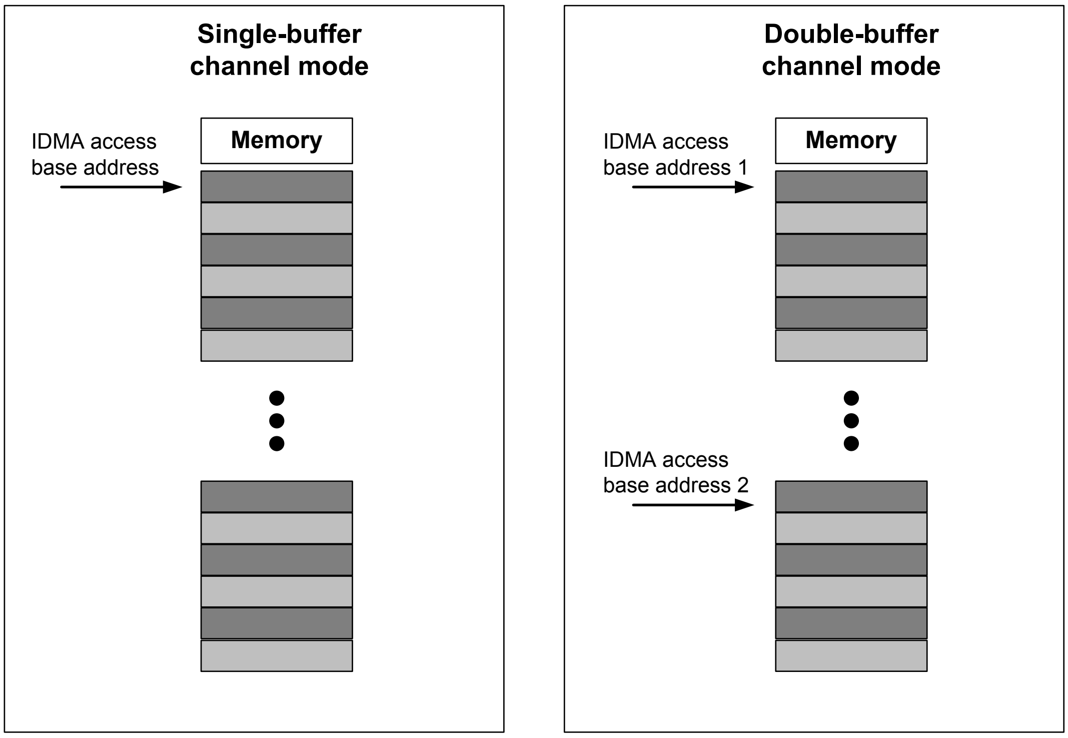
The SDMMC have its own IDMA which can be configured over the SDMMC registers. The IDMA provides high-speed data transfer between the FIFO and the memory. It supports the following features:

- Burst transfer of 8 beats
- Easy configuration through the SDMMC registers
- Two modes of operation: Single-buffer transfer or Double-buffer transfer.

In Single-buffer configuration, the IDMA access one base address and starts the data transfer.

In Double-buffer configuration, the IDMA access one base address and starts the data transfer until all data has been transferred, then it moves to the second base address and starts another transfer. This allow the firmware to process the first data buffer while the IDMA is accessing the second memory buffer.

Figure 7. **IDMA Single-buffer and Double-buffer channels modes**


Single-buffer channel mode
IDMA access base address
Memory
Double-buffer channel mode
IDMA access base address 1
Memory
IDMA access base address 2

1.3 SDMMC host interface differences between STM32F7 Series and STM32H7 Series

The new SDMMC interface embedded within the STM32H7 Series enhances the SDMMC host capabilities compared to the previous version of the peripheral embedded in the STM32F7 Series.

The following table contains the main differences between the SDMMC in STM32F7 Series and the SDMMC in STM32H7 Series.

Table 5. SDMMC differences between STM32F7 Series and STM32H7 Series

SDMMC feature(1)	STM32F7 Series	STM32H7 Series
Compatibility version	SD(2.0) / MMC(4.2) / SDIO(2.0)	SD(4.1) / MMC(4.51) / SDIO(4.0)
Data transfer clock frequency	0-50 MHz (for 3.3 V)	0-50 MHz (for 3.3 V) 0-208 MHz (for 1.8 V)
Operating buses	Slave interface: 32-bit APB bus	Slave interface: 32-bit AHB bus Master interface: 32-bit AHB bus
DMA transfer	Request an external DMA (DMA2) Bust of 4 bits	Supports an internal DMA burst
Supported speed modes for SD and SDIO cards	Default speed / high speed (3.3 V) UHS-I not supported	Default speed / high speed (3.3 V) UHS-I modes: : SDR12 / SDR25 / SDR50 / DDR50 / SDR104
Supported speed modes for MMC cards	Legacy compatible High speed SDR	Legacy compatible High speed SDR High speed DDR High speed HS200
Boot operation	Supports only alternative boot	Supports normal boot and alternative boot
DLYB	Not available	Available
MDMA features	Not available	Available

1. *Maximum frequency depends on the maximum allowed IO speed.*

2 SDMMC host interface initialization with SD and MMC cards

2.1 Initiate SDMMC host interface configuration using STM32CubeMx

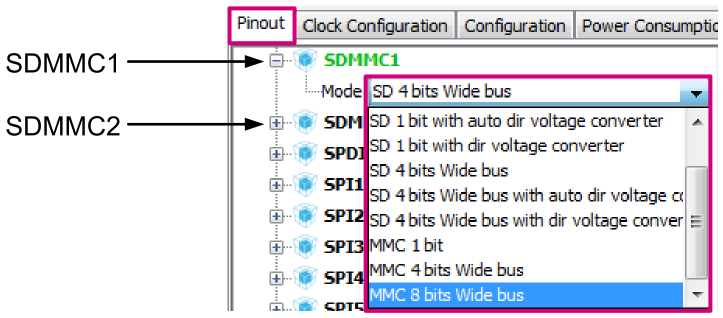
This section presents an example on how to configure the SDMMC using the STM32CubeMx tool.

Using the STM32CubeMX tool is a very simple, easy and rapid way to configure the SDMMC peripheral and its GPIOs as it permits the generation of a project with preconfigured SDMMC.

2.1.1 SDMMC host interface GPIO configuration using STM32CubeMx

Once the STM32CubeMX project is created, in the *Pinout* tab choose one of the listed hardware configurations. The figure below shows how to select the SDMMC hardware configuration with the STM32CubeMX.

Figure 8. **SDMMC pins configuration with STM32CubeMx**


SDMMC1
SDMMC2
Pinout
Clock Configuration
Configuration
Power Consumptic
SDMMC1
SDM
SPDI
SPI1
SPI2
SPI3
SPI4
SD 1 bit with auto dir voltage converter
SD 1 bit with dir voltage converter
SD 4 bits Wide bus
SD 4 bits Wide bus with auto dir voltage cc
SD 4 bits Wide bus with dir voltage conver
MMC 1 bit
MMC 4 bits Wide bus
MMC 8 bits Wide bus

Mode SD 4 bits Wide bus

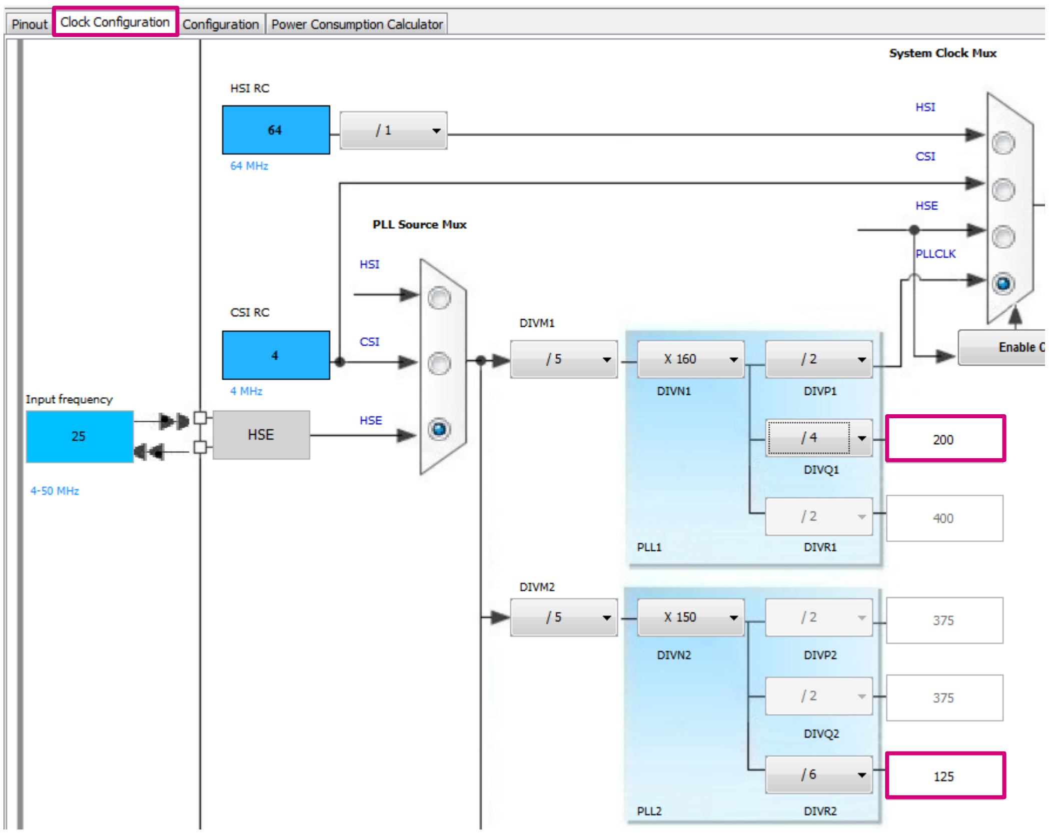
In the pinout panel, the user can chose between SDMMC1 and SDMMC2. The user selects the configuration mode depending on the desired application and below configurations are created automatically:

- Type of card : SD/SDIO/MMC card
- Bus width: 1-bit , 4-bit , 8-bit mode
- External transceiver: enable/disable command signals for external transceiver.

2.1.2 SDMMC host interface clock configuration using STM32CubeMx

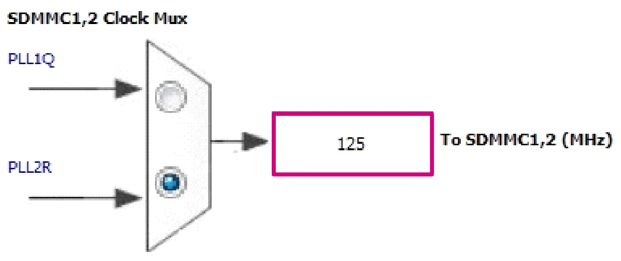
The first step is to select the source clock for the sdmmc_ker_ck as show in the figure below.

Figure 9. sdmmc_ker_ck configuration with STM32CubeMx



In the clock configuration panel the user can select the source clock for the sdmmc_ker_ck1 and sdmmc_ker_ck2 either from DIVQ1 or DIVR2. To make the selection, the user scrolls down the panel and finds a figure like the one below:

Figure 10. sdmmc_ker_ck source clock selection with STM32CubeMx



In the SDMMC1 and SDMCC2, Clock Mux selects the source clock for sdmmc_ker_ck1 or sdmmc_ker_ck2.

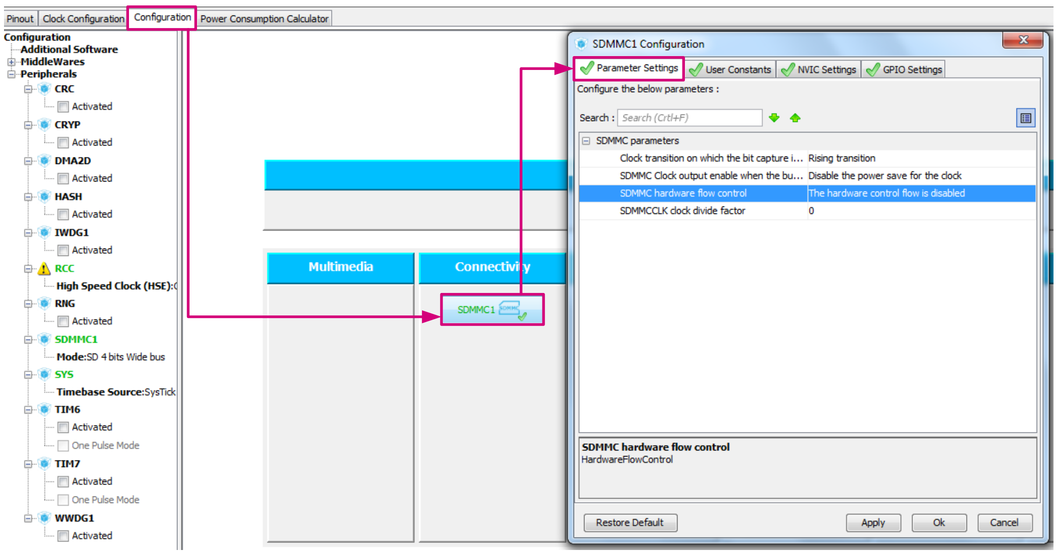
2.1.3 SDMMC host interface parameters configuration using STM32CubeMx

The first step is to go to the *Configuration* panel, then in the *Connectivity* section the user finds the SDMMC1 configuration panel when he can select the parameter settings:

- Select the clock edge

- Enable the Power-save mode (disable the clock when the SDMMC is in idle state and there is no data or command transfer)
- Enable or disable the hardware flow control (refer to Section 5 SDMMC host interface and hardware flow control)
- Select the divide factor for the SDMMC_CK (SDMMC_CK = sdmmc_ker_ck / [CLKDIV*2])

Figure 11. **SDMMC parameters configuration with STM32CubeMx**



Configure the below parameters :

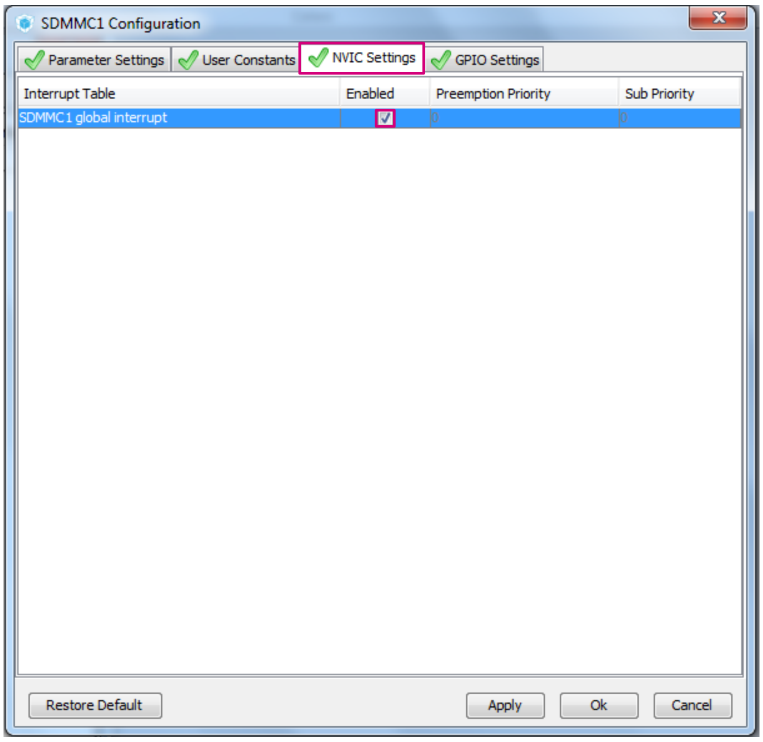
SDMMC parameters

Clock transition on which the bit capture i…	Rising transition
SDMMC Clock output enable when the bu…	Disable the power save for the clock
SDMMC hardware flow control	The hardware control flow is disabled
SDMMCCLK clock divide factor	0

SDMMC hardware flow control
HardwareFlowControl

In the *NVIC Settings* tab, the user enables the SDMMC global interrupt then click on the OK button (see figure below).

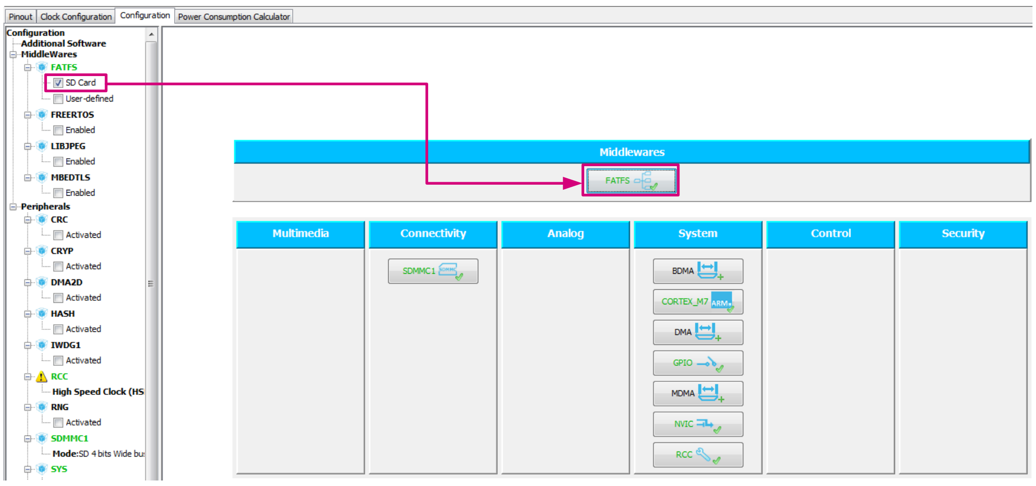
Figure 12. **SDMMC global interrupt configuration with STM32CubeMx**



Interrupt Table	Enabled	Preemption Priority	Sub Priority
SDMMC1 global interrupt	✓	0	0

The file system FATFS drivers can be added by selecting the middleware configuration as presented in the next figure:

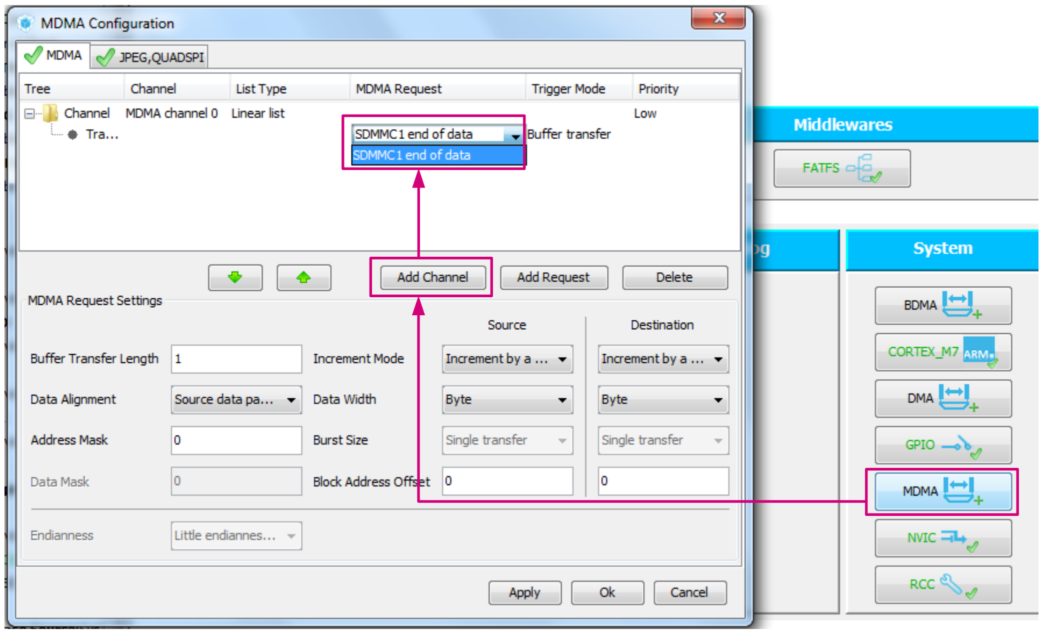
Figure 13. **SDMMC FATFS configuration with STM32CubeMx**



Finally, in order to configure the MDMA end of data transfer trigger from the SDMMC1, the user selects the MDMA in the *System* configuration section.

Once the dialog box is open, the user clicks on the *Add a channel* button and selects *SDMMC1 end of data*. Now the user can configure the MDMA for the data transfer (buffer transfer length, data alignment, source and destination). See the figure below for an example of the dialog boxes that the user finds when doing this configuration.

Figure 14. **MDMA end of data configuration with STM32CubeMx**



Tree	Channel	List Type	MDMA Request	Trigger Mode	Priority
Channel	MDMA channel 0	Linear list			Low
Tra...			SDMMC1 end of data	Buffer transfer	

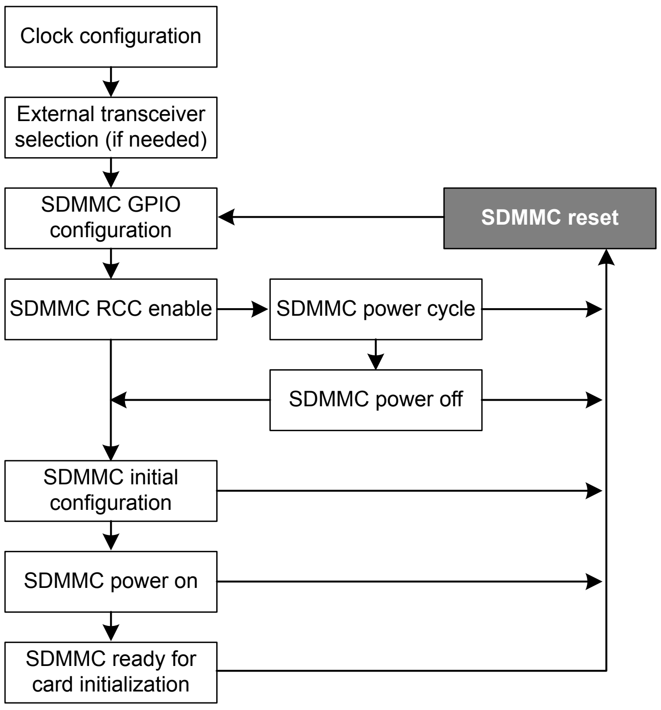
MDMA Request Settings

			Source	Destination
Buffer Transfer Length	1	Increment Mode	Increment by a ...	Increment by a ...
Data Alignment	Source data pa...	Data Width	Byte	Byte
Address Mask	0	Burst Size	Single transfer	Single transfer
Data Mask	0	Block Address Offset	0	0
Endianness	Little endiannes...			

2.2 Initiate SDMMC host interface configuration

This section presents an example on how to initiate the SDMMC host interface until the command path and data path are ready to initiate the SD card or the MMC card. See a synopsis on the process below:

Figure 15. **Synoptic diagram of SDMMC initialization**

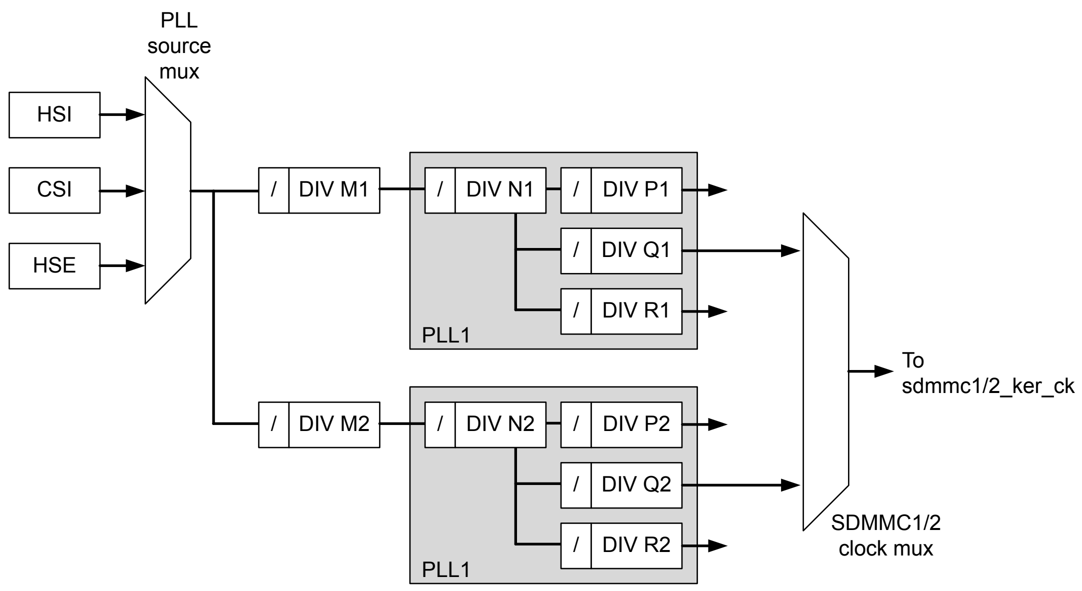


Clock configuration

The RCC registers should be configured to select the clock for sdmmc_ker_ck and SDMMC_hclk.

The figure below illustrates two source clocks for sdmmc_ker_ck for both sdmmc1 and sdmmc2.

Figure 16. **Source clock for sdmmc1/2_ker_ck**



With a duty cycle close to 50%, the DIV[P/Q/R]x values shall be even.

For SDMMCx, the duty cycle shall be 50% when supporting DDR.

The configuration of the AHB3/2 clock domain to select the SDMMC_hclk must be done and should respect the following relation: SDMMC_hclk > ((3x bus width / 32) x SDMMC_CK).

The *clock configuration* is now completed, sdmmc_ker_ck and SDMMC_hclk are configured but they are not enabled.

External transceiver selection (if needed)

This configuration is optional, and it is needed when using UHS-I modes. The external transceiver must be selected. A GPIO pin should be configured as digital output and the transceiver must be selected.

SDMMC GPIO configuration

The SDMMC GPIO pins for shall be configured for command, data, clock and if needed for the control signals to the external transceiver. Refer to the specific device's datasheet for SDMMC GPIO configuration.

SDMMC RCC enable

The RCC registers shall be configured to enable the clock for SDMMC1 or SDMMC2, this action enables both sdmmc_ker_ck and sdmmc_hclk for the selected peripheral.

SDMMC power cycle

It is advised to perform a power cycle before initializing the external card. Refer to the specific device's reference manual for more information on how to perform a power cycle sequence.

SDMMC power off

SDMMC disabled and SDMMC signals drive "1".

SDMMC initial configuration

When using the external transceiver the direction signal polarity must be set as *high*.

For the SDMMC initial configuration, the SDMMC_CK must be in the range of 100 to 400 KHz, the DDR mode and the bus speed must be disabled.

SDMMC power on

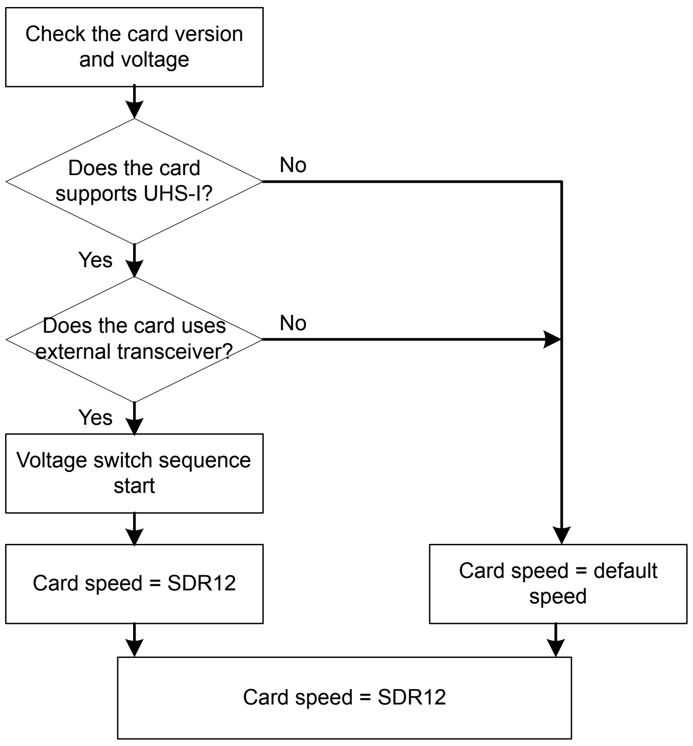
The power on of the SDMMC, must be set. Once it is done, the SDMMC is ready for card initialization.

2.3 SD card initialization

2.3.1 SD power on and initialization

The SD card power on and initialization process is illustrated in the following flow chart:

Figure 17. **SD card power on and initialization diagram**


Check the card version and voltage
Does the card supports UHS-I?
No
Yes
Does the card uses external transceiver?
No
Yes
Voltage switch sequence start
Card speed = SDR12
Card speed = default speed
Card speed = SDR12

The user checks the card version and verify if the card supports UHS-I and if an external voltage switch transceiver is supported. If yes, the user starts the voltage sequence and defines the card speed as SDR12 (else define it as *normal* speed) then move to the card initialization. The support of UHS-I and external transceiver is optional and it depends on user application.

Check the card version and voltage

The host sends CMD8, and depending on the result:

- If the card returns a response, it means that it is version 2.00 or later
  - If the response is not valid, it means that the voltage range is not compatible
  - If the response is valid, it means that the card voltage range is compatible
- If the card does not return a response, it means that it is a version 2.00 or later with a mismatch voltage or a version 1.X SD card or not an SD card.

Does the card supports UHS-I?

To check if the card support UHS-I the host sends ACMD41 repetitively and checks the response. If the card support UHS-I the host needs to check if an external transceiver is supported then start the voltage switch sequence.

Voltage switch sequence

The host needs to enable the voltage switch from SDMMC registers for the voltage switch sequence.

- Before sending CMD11 check the SDMMC_CK. If it stopped and then *busy signal* = asserted. If done, enable the external transceiver, otherwise the SD card do not support that feature.
- Wait for the voltage switch critical timing section completion flag to be set, then check the busy signal. If done, the voltage switch succeeded and the SD card speed is set to SDR12, otherwise it failed.
- Clear the voltage switch flag and the status flags.

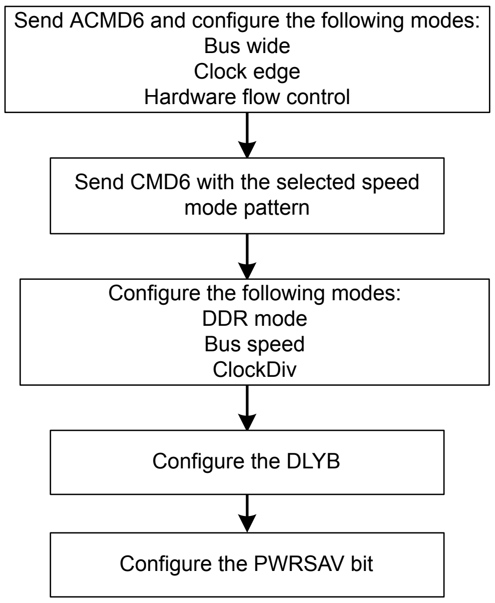
Card initialization

- Send CMD2 to get all cards unique identification number.
- Send CMD3 to set the related card address (RCA).
- Send CMD9 to get the cards specific data (CSD).

2.3.2 Configure the wide bus operation for SD card

The flowchart below presents the synoptic diagram of the SD card bus wide configuration.

Figure 18. **SD card bus wide configuration diagram**


Send ACMD6 and configure the following modes:
Bus wide
Clock edge
Hardware flow control
Send CMD6 with the selected speed mode pattern
Configure the following modes:
DDR mode
Bus speed
ClockDiv
Configure the DLYB
Configure the PWRSAV bit

Note: CMD6 is not supported by SD cards version 1.01.

The detailed steps to configure the wide bus operation are the following:

1. Send ACMD6 then configure the SDMMC to switch the bus wide for 1-bit mode or 4-bit mode (the 1-bit mode is not supported with UHS-I speeds).
2. Set the clock edge as rising or falling (for SDMMC_CK > 50 MHz it is advised to set the clock edge as falling).
3. Enable or disable the hardware flow control.
4. Send CMD6, the SD card returns the contained 64-bytes of data if the card supports that speed mode, then it switches to the selected speed mode (if supported).

Table 6. **CMD6 data pattern for speed mode selection**

Card operating speed after voltage switch sequence (succeed or failed)	CMD6 argument for the selected speed mode
DS , SDR12	0x80FF FF00
HS,SDR25	0x80FF FF01
SDR50	0x80FF FF02

Card operating speed after voltage switch sequence (succeed or failed)	CMD6 argument for the selected speed mode
SDR104	0x80FF FF03
DDR50	0x80FF FF04

5. DDR mode, Bus speed mode and ClockDiv mode must be configured after sending CMD6 because the SD card switches to the selected speed mode after sending the 64-bytes of data. Changing the clock and bus mode before sending the command may generate a data read error.
6. Configure the DLYB (if needed) using CMD19 for tuning.
7. Configure the PWRSAV bit. PWRSAV bit stops the clock when the CPSM and DPSM are in Idle state (no command or data transfer over the bus). Enabling PWRSAV bit before configuring the DLYB causes an error because the SDMMC_CK is stopped.

Example of CMD6 switch with SDR104:

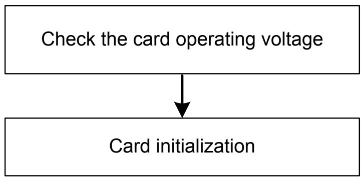
```
errorstate = SDMMC_CmdSwitch(hsd->Instance, SDMMC_SDR104_SWITCH_PATTERN);
    if(errorstate != HAL_OK)
    {
      return errorstate;
    }
```

2.4 MMC card initialization

2.4.1 MMC power on and initialization

Find below the synoptic diagram of the MMC card initialization:

Figure 19. Synoptic diagram of MMC initialization



Check the card operating voltage

1. Put the MMC card into idle state by sending CMD0.
2. Check if the card operating voltage matches by sending CMD1.

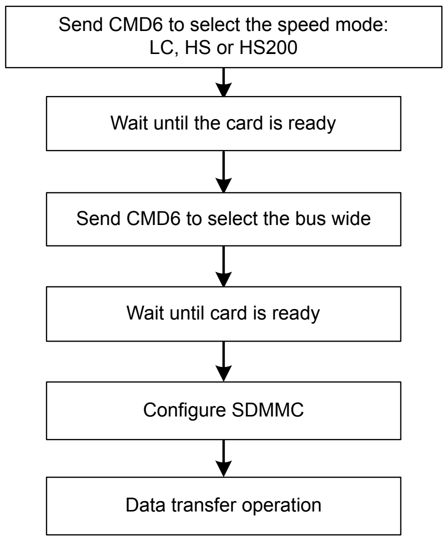
Card initialization

1. Send CMD2 to get all cards unique identification number.
2. Send CMD3 to set the related card address (RCA).
3. Send CMD9 to get the card the card specific data (CSD).
4. Send CMD13 to check if card is ready.
5. Send CMD8 to get the extended card specific data (EXTCSD).

2.4.2 Configure the wide bus operation for MMC card

The flowchart below presents the synoptic diagram of the MMC card bus wide configuration.

Figure 20. **MMC card bus wide configuration diagram**


Send CMD6 to select the speed mode: LC, HS or HS200
Wait until the card is ready
Send CMD6 to select the bus wide
Wait until card is ready
Configure SDMMC
Data transfer operation

The detailed steps to configure the wide bus operation are the following:

1. Send CMD6 to select the speed mode (LC, HS or HS200) and wait until the card is ready using CMD13.

Table 7. **CMD6 data pattern for speed mode selection**

Card operating speed after voltage switch sequence (succeed or failed)	CMD6 argument for the selected speed mode
Legacy compatible	0x03B9 0000
HS SDR / HS DDR	0x03B9 0100
HS200	0x03B9 0200

2. Send CMD6 to select the bus wide (1-bit, 4-bit, 8-bit) and wait until the card is ready.

Table 8. **CMD6 data pattern for speed mode selection**

Bus wide	CMD6 argument for the selected bus wide
1-bit data bus	0x03B7 0000
4-bit data bus	0x03B7 0100
8-bit data bus	0x03B7 0200
4-bit data bus (HS DDR)	0x03B7 0500
8-bit data bus (HS DDR)	0x03B7 0800

3. Configure the SDMMC registers to select:
   – The clock edge: rising or falling.
   – Power save: enable or disable.
   – Bis width: 1-bit, 4-bit or 8-bit.
   – Hardware flow control: enable or disable.
   – CLK div.
   – Bus speed: enable or disable.
   – DDR mode: enable or disable.

Now the card is ready for data transfer operations.

3 How to read / write with SDMMC host interface in single / multiple block(s) mode

The configurations illustrated below are the same for Interrupt, IDMA single buffer and IDMA double buffer modes.

Figure 21. Data transfer initialization

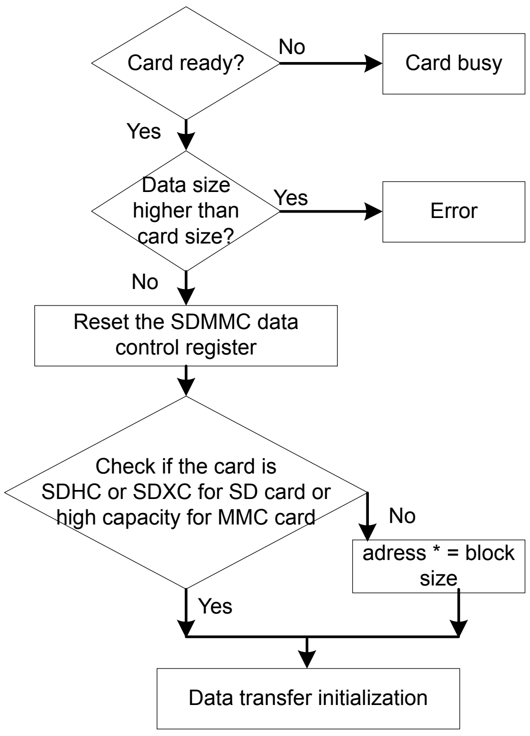
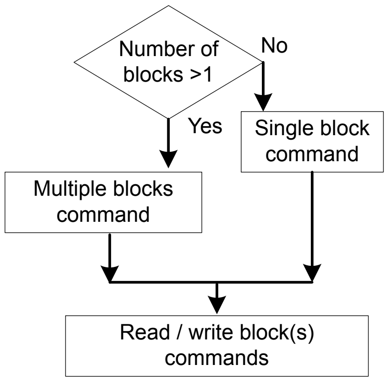

Card ready?
No
Card busy
Yes
Data size higher than card size?
Yes
Error
No
Reset the SDMMC data control register
Check if the card is SDHC or SDXC for SD card or high capacity for MMC card
No
adress * = block size
Yes
Data transfer initialization

Figure 22. Read/write block(s) commands



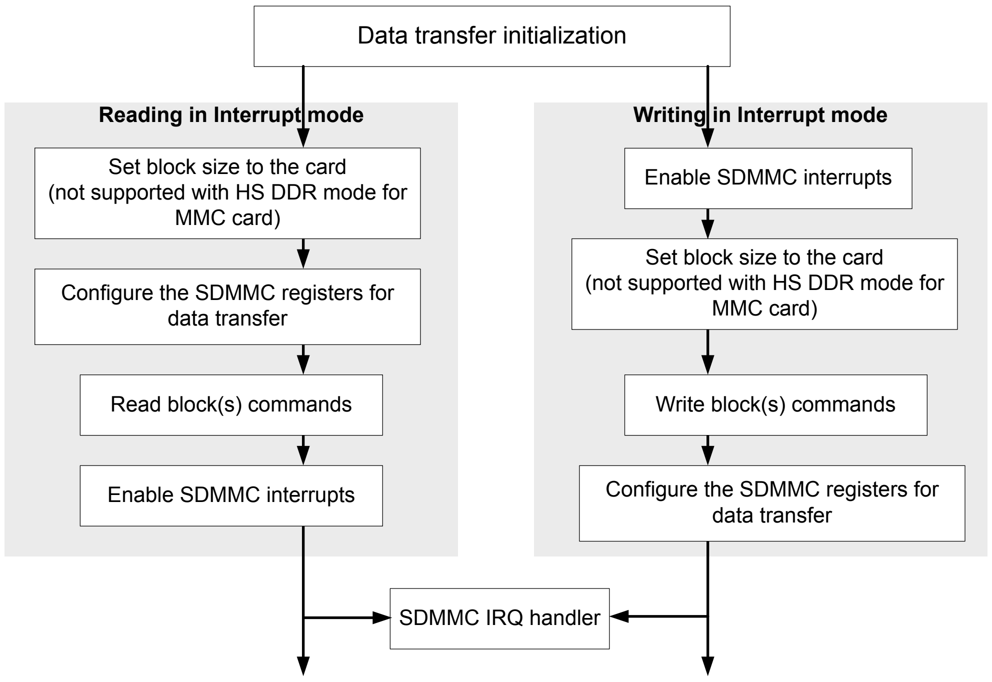
The following commands are used for read/write multiple/single block

- CMD17: read single block.
- CMD18: read multiple blocks.
- CMD23: write single block.
- CMD24: write multiple blocks.

3.1 Interrupt mode

The read/write flow in Interrupt mode is illustrated below.

Figure 23. **Read/write data in Interrupt mode**


Data transfer initialization
Reading in Interrupt mode
Set block size to the card (not supported with HS DDR mode for MMC card)
Configure the SDMMC registers for data transfer
Read block(s) commands
Enable SDMMC interrupts
Writing in Interrupt mode
Enable SDMMC interrupts
Set block size to the card (not supported with HS DDR mode for MMC card)
Write block(s) commands
Configure the SDMMC registers for data transfer
SDMMC IRQ handler

Reading in Interrupt mode

1. Send the block size to the card using CFinMD16 then configure the SDMMC registers for data transfer:
   – Set the data timeout.
   – Set the data length.
   – Set the data block size.
   – Configure the data transfer direction (from SD/MMC card to host).
   – Configure the data transfer mode block(s).
   – Enable the DPSM.
2. Send the Read-command block(s) then enable the interrupts listed in the following table:

Table 9. **Interrupts to enable to read in Interrupt mode**

SDMMC interrupts enabled	Description
DCRCFAIL	Data CRC check failed
DTIMEOUT	Data timeout
RXOVERR	Received FIFO overrun
DATAEND	Successful end of data transfer
RXFIFOHF	Received FIFO half full

Writing in Interrupt mode

1. Enable the interrupts listed in the following table:

Table 10. **Interrupts to enable to write in Interrupt mode**

SDMMC interrupts enabled	Description
DCRCFAIL	Data CRC check failed
DTIMEOUT	Data timeout
TXUNDERR	Transfer FIFO underrun
DATAEND	Successful end of data transfer
RXFIFOHF	Transfer FIFO half empty

2. Send block size to the card using CMD16 then send the Write-command block(s).
3. Configure the SDMMC registers for data transfer:
   – Set the data timeout.
   – Set the data length.
   – Set the data block size.
   – Configure the data transfer direction (from host to SD/MMC card).
   – Configure the data transfer mode block(s).
   – Enable the DPSM.

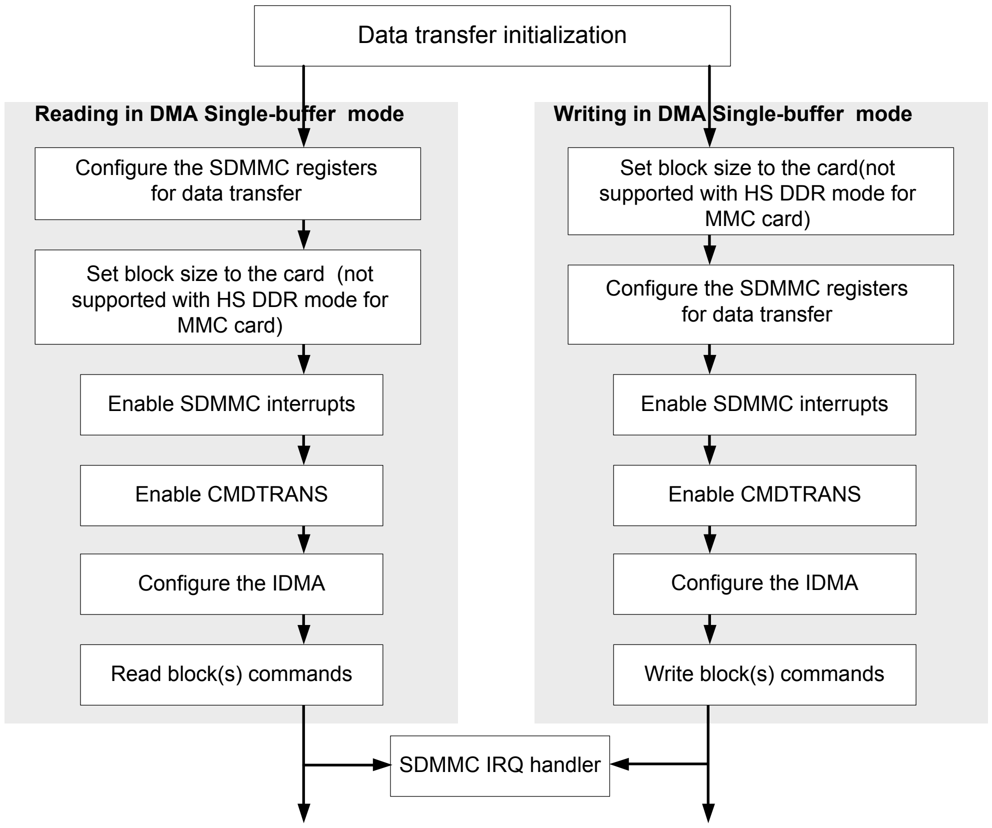
SDMMC IRQ handler

The IRQ handler reads/writes data from the FIFO whenever the FIFO is half empty or half full, calls the callback functions and manages the data read/write errors. Also it sends the end of data transfer command CMD12 after successful read or write.

3.2 DMA Single-buffer mode

The read/write flow in DMA Single-buffer mode is illustrated below.

Figure 24. **Read/write data in DMA Single-buffer mode**


Data transfer initialization
Reading in DMA Single-buffer mode
Configure the SDMMC registers for data transfer
Set block size to the card (not supported with HS DDR mode for MMC card)
Enable SDMMC interrupts
Enable CMDTRANS
Configure the IDMA
Read block(s) commands
Writing in DMA Single-buffer mode
Set block size to the card(not supported with HS DDR mode for MMC card)
Configure the SDMMC registers for data transfer
Enable SDMMC interrupts
Enable CMDTRANS
Configure the IDMA
Write block(s) commands
SDMMC IRQ handler

Reading in DMA Single-buffer mode

1. Configure the SDMMC registers for data transfer:
   – Set the data timeout.
   – Set the data length.
   – Set the data block size.
   – Configure the data transfer direction (from SD/MMC card to host).
   – Configure the data transfer mode block(s).
   – Disable the DPSM.
2. Send the block size to the card using CMD16 then enable the interrupts listed in the following table:

Table 11. **Interrupts to enable to read in DMA Single-buffer mode**

SDMMC interrupts enabled	Description
DCRCFAIL	Data CRC check failed
DTIMEOUT	Data timeout
RXOVERR	Received FIFO overrun
DATAEND	Successful en dof data transfer

3. Enable CMDTRANS. The CPSM treats the command as a data transfer command, stops the interrupt period and signals data enable to the DPSM.
4. Configure the SDMMC registers to set the buffer base address for the IDMA and enable the IDMA as Single-buffer mode.
5. Send the read command block(s).

Writing in DMA Single-buffer mode

1. Send block size to the card using CMD16 then configure the SDMMC registers for data transfer:
   – Set the data timeout.
   – Set the data length.
   – Set the data block size.
   – Configure the data transfer direction (from host to SD/MMC card).
   – Configure the data transfer mode block(s).
   – Disable the DPSM.
2. Enable the interrupts listed in the following table:

Table 12. **Interrupts to enable to write in DMA Single-buffer mode**

SDMMC interrupts enabled	Description
DCRCFAIL	Data CRC check failed
DTIMEOUT	Data timeout
TXUNDERR	Transfer FIFO underrun
DATAEND	Successful end of data transfer

3. Enable CMDTRANS.
4. Configure the SDMMC registers to set the buffer base address for the IDMA and enable the IDMA as Single-buffer mode.
5. Send the write command block(s).

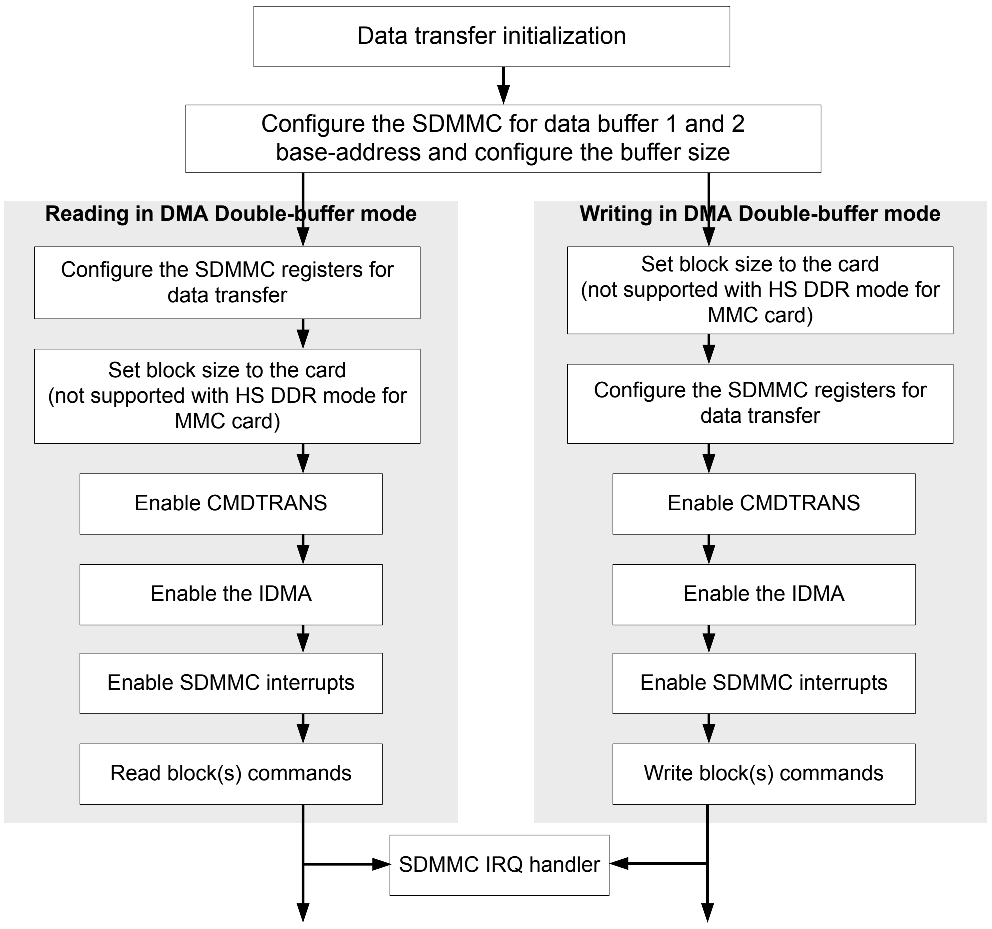
SDMMC IRQ handler

The IRQ handler calls the callback functions and manages the data read/write errors. It also sends the end of data transfer command CMD12 after successful read or write.

3.3 DMA Double-buffer mode

The read/write flow in DMA Double-buffer mode is illustrated below.

Figure 25. Read/write data in DMA Double-buffer mode


Data transfer initialization
Configure the SDMMC for data buffer 1 and 2 base-address and configure the buffer size
Reading in DMA Double-buffer mode
Configure the SDMMC registers for data transfer
Set block size to the card (not supported with HS DDR mode for MMC card)
Enable CMDTRANS
Enable the IDMA
Enable SDMMC interrupts
Read block(s) commands
Writing in DMA Double-buffer mode
Set block size to the card (not supported with HS DDR mode for MMC card)
Configure the SDMMC registers for data transfer
Enable CMDTRANS
Enable the IDMA
Enable SDMMC interrupts
Write block(s) commands
SDMMC IRQ handler

Reading DMA Double-buffer mode

1. Configure the SDMMC register for data transfer:
   - Set the data timeout.
   - Set the data length.
   - Set the data block size.
   - Configure the data transfer direction (from SD/MMC card to host).
   - Configure the data transfer mode block(s).
   - Disable the DPSM.
2. Send block size to the card using CMD16.
3. Enable CMDTRANS.
4. Configure the SDMMC register to enable the IDMA as Double-buffer mode.
5. Enable the interrupts listed in the following table:

Table 13. **Interrupts to enable to read in DMA Double-buffer mode**

SDMMC interrupts enabled	Description
DCRCFAIL	Data CRC check failed
DTIMEOUT	Data timeout
RXOVERR	Received FIFO overrun
DATAEND	Successful end of data transfer
IDMATE	IDMA transfer error
IDMABTC	IDMA buffer transfer complete

6. Send the read command block(s).

Writing in DMA Double-buffer mode

1. Send block size to the card using CMD16.
2. Configure the SDMMC registers for data transfer:
   - Set the data timeout.
   - Set the data length.
   - Set the data block size.
   - Configure the data transfer direction (from host to SD/MMC card).
   - Configure the data transfer mode block(s).
   - Disable the DPSM.
3. Enable CMDTRANS.
4. Configure the SDMMC registers to enable the IDMA as Double-buffer mode.
5. Enable the interrupts listed in the following table:

Table 14. **Interrupts to enable to write in DMA Double-buffer mode**

SDMMC interrupts enabled	Description
DCRCFAIL	Data CRC check failed
DTIMEOUT	Data timeout
TXUNDERR	Transfer FIFO underrun
DATAEND	Successful end of data transfer
IDMATE	IDMA transfer error
IDMABTC	IDMA buffer transfer complete

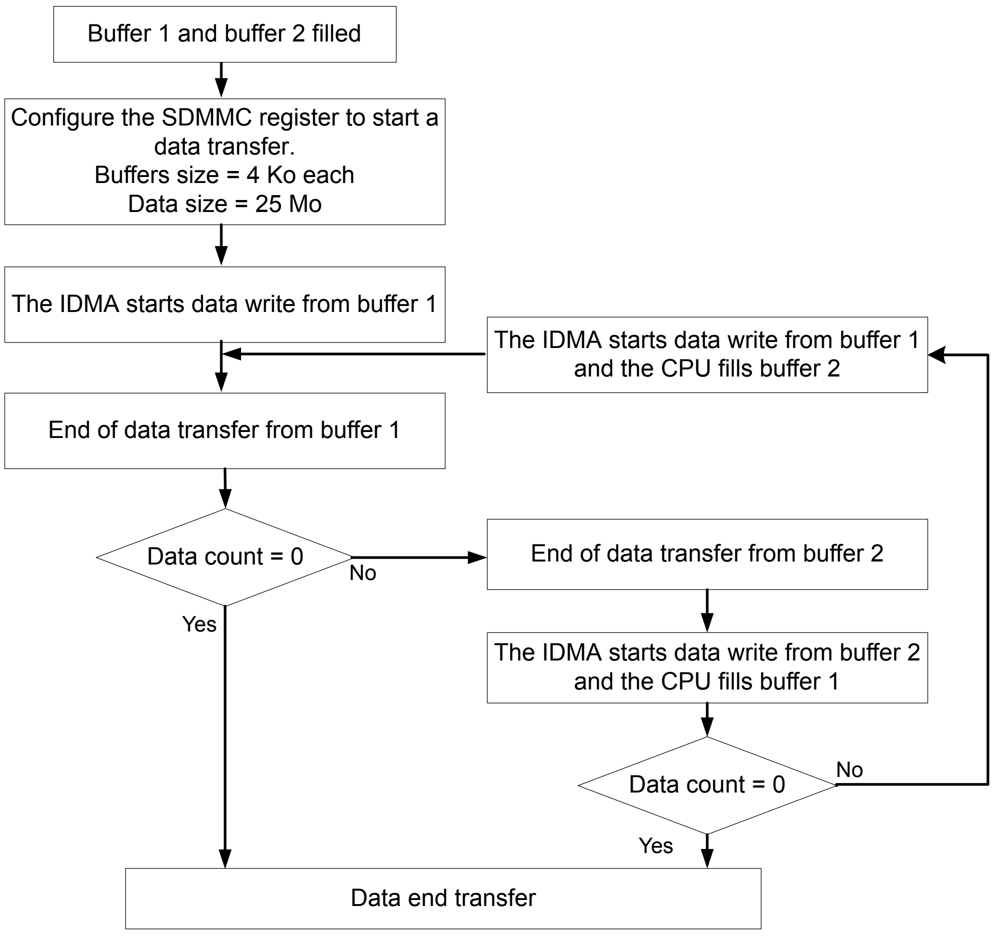
6. Send the write command block(s).

SDMMC IRQ handler

The IRQ handler calls the callback functions and manages the data read/write errors. It also sends the end of data transfer command CMD12 after successful read or write.

Th next figure illustrates an IDMA Double-buffer mode example of writing 25 Mo of data in one transaction with IDMA Double-buffer mode. The full application example is available in STM32Cube_FW_H7_V1.2.0

Figure 26. **IDMA Double-buffer mode example**


Buffer 1 and buffer 2 filled
Configure the SDMMC register to start a data transfer. Buffers size = 4 Ko each Data size = 25 Mo
The IDMA starts data write from buffer 1
The IDMA starts data write from buffer 1 and the CPU fills buffer 2
End of data transfer from buffer 1
Data count = 0
No
Yes
End of data transfer from buffer 2
The IDMA starts data write from buffer 2 and the CPU fills buffer 1
Data count = 0
No
Yes
Data end transfer

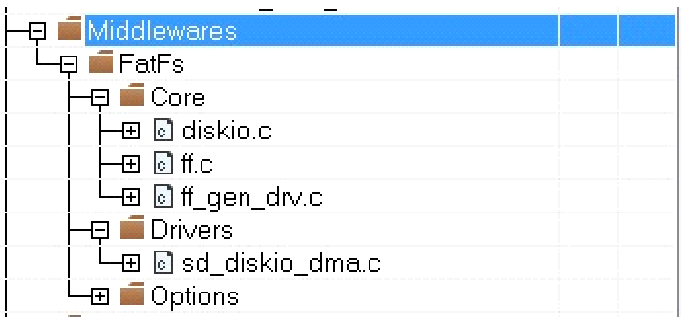
4 How to read/write with SDMMC host interface using file system FATFS

Data can be transferred from SD card using file system FATFS in Polling, Interrupt and DMA modes. The following example (available in **STM32Cube_FW_H7_V1.2.0**) describes how to create, write and then read a text document in DMA Single-buffer mode with SDMMC on STM32H743I-EVAL board.

4.1 Imported project files for file system management with SDMMC

In this Example, middleware and BSP files are added to the project for FATFS functions and in order to facilitate the configuration.

Figure 27. **Middleware files tree**

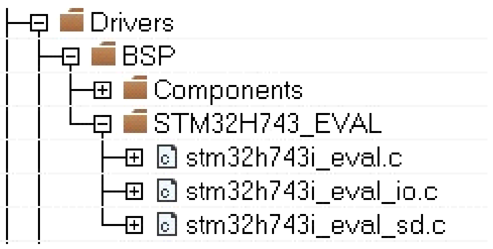


```
├⊟ Middlewares
│ └⊟ FatFs
│   ├⊟ Core
│   │ ├⊞ diskio.c
│   │ ├⊞ ff.c
│   │ └⊞ ff_gen_drv.c
│   ├⊟ Drivers
│   │ └⊞ sd_diskio_dma.c
│   └⊞ Options
```

The middleware files contain the FATFS files **diskio.c**, **ff.c** and **ff_gen_drv.c**; and the driver file **sd_diskio_dma.c** .

The user can chose the data transfer mode (Interrupt, DMA, and Polling) by modifying or changing the **sd_diskio_dma.c** file. It contains the functions for SD initialization, read and write called from the BSP files.

Figure 28. **BSP files tree**



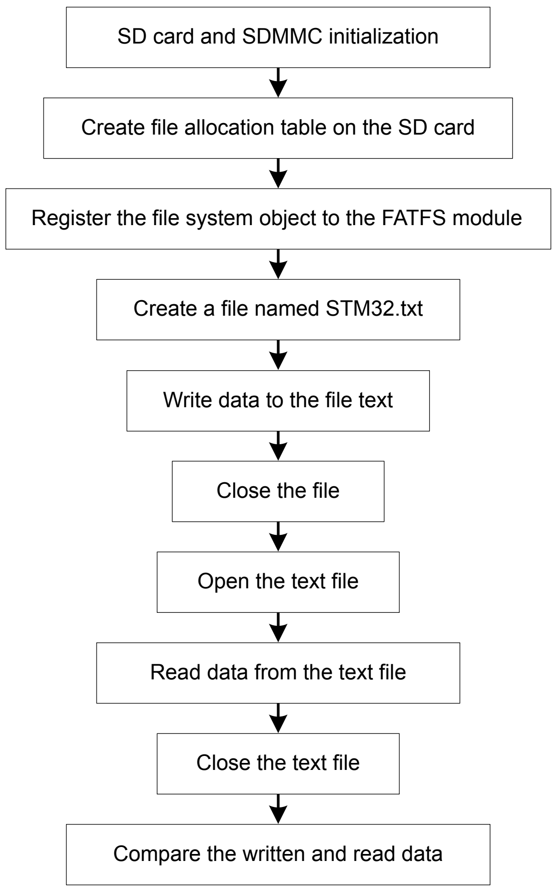
```
├⊟ Drivers
│ ├⊟ BSP
│ │ ├⊞ Components
│ │ └⊟ STM32H743_EVAL
│ │   ├⊞ stm32h743i_eval.c
│ │   ├⊞ stm32h743i_eval_io.c
│ │   └⊞ stm32h743i_eval_sd.c
```

The user needs to select the BSP files depending on the board (as an example here the STM32H743I-EVAL board) the BSP files contain a set of functions needed to manage the IO pins. User can also integrate his own BSP files.

4.2 File system example

The following chart illustrates the description of a file system example.

Figure 29. File system example


SD card and SDMMC initialization
Create file allocation table on the SD card
Register the file system object to the FATFS module
Create a file named STM32.txt
Write data to the file text
Close the file
Open the text file
Read data from the text file
Close the text file
Compare the written and read data

5 SDMMC host interface and hardware flow control

The hardware flow control functionality is used to avoid FIFO underrun (TX mode) and overrun (RX mode) errors.

The SDMMC_hclk must respect the following relation:

SDMMC_hclk > ((3x bus width / 32) x SDMMC_CK).

When enabling the hardware flow control, SDMMC_hclk can be a slightly reduced, this action does not cause a data error but it reducse the SDMMC data speed transfer.

As an example of transferring data in DMA mode with *CPU clock frequency = SDMMC_hclk = 37.5 MHz*, when enabling the hardware flow control the CPU clock frequency can reach *SDMMC_hclk = 20 MHz* but the data speed transfer is reduced by 33%.

Note: Hardware flow control shall only be used when the SDMMC_Dn data is cycle-aligned with the SDMMC_CK. Whenever the sdmmc_fb_ck from the DLYB delay block is used (such as in the case of SDR104 mode with a t_{OP} *and* D_{tOP} *delay > 1 cycle), hardware flow control can* ***not*** *be used. Refer to the device's datasheet for more information.*

6 How to enable the DLYB

6.1 DLYB enabling use case presentation

The DLYB is used to tune the sampling clock on the received data. When transferring data in high speed, a data read error may be prevented by enabling the DLYB.

6.2 DLYB enabling use case configuration

The DLYB can be configured with two different methods; each method has its own advantages compared to the other:

- Without tuning command: the advantage is that it take less time to be configured.
- With tuning command: the advantage is that there are less chances to face a data transfer error.

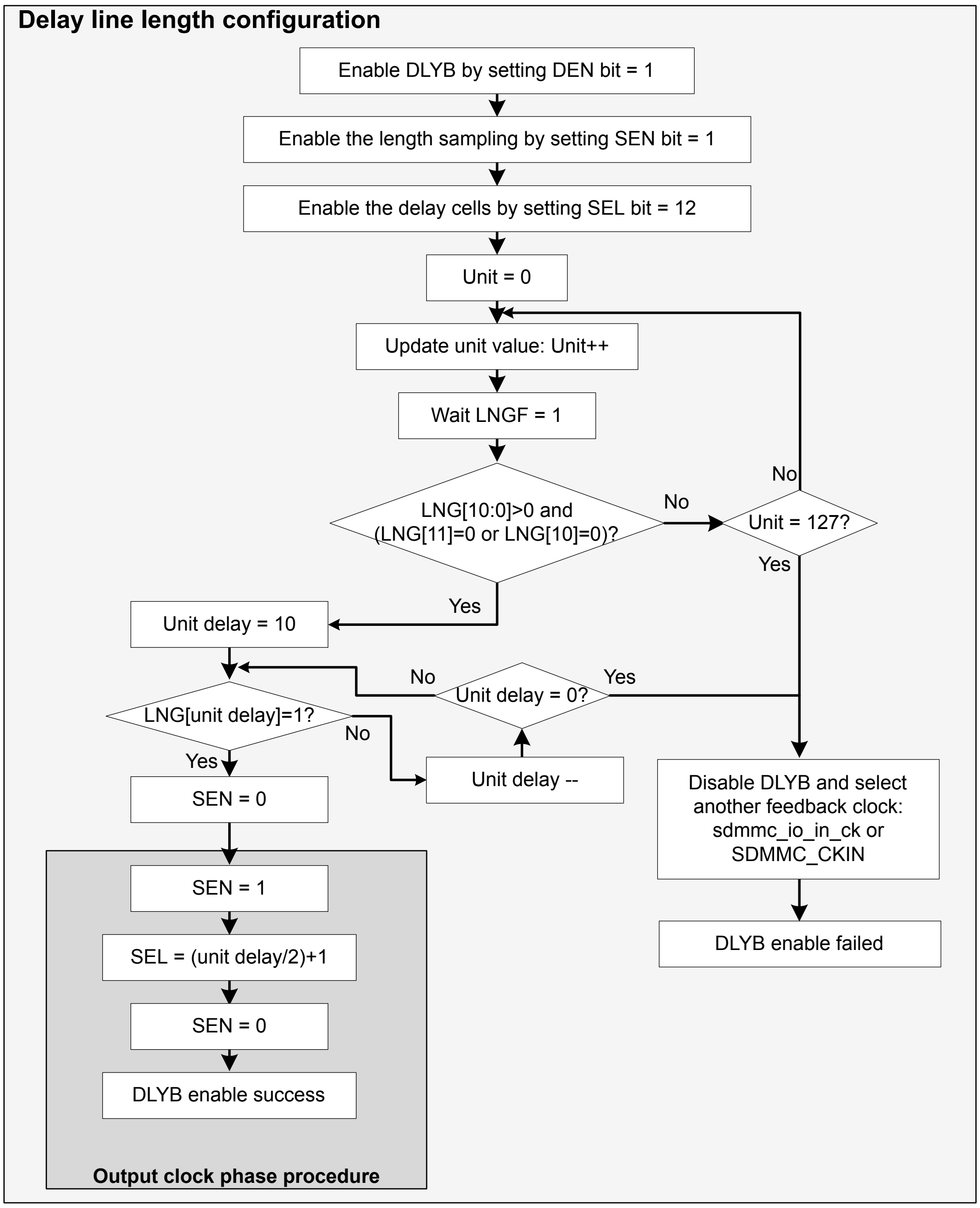
6.2.1 DLYB configuration without tuning command

To configure DLYB without tuning command, select the DLYB feedback clock as input clock and then enable the delay block. See an example below:

```
/*select the DLYB feedback clock as input clock*/
     MODIFY_REG(hsd->Instance->CLKCR, SDMMC_CLKCR_SELCLKRX, SDMMC_CLKCR_SELCLKRX_1);
     /*Enable the DLYB*/
     DelayBlock_Enable(DLYB_SDMMC1);
```

The following diagram shows the steps to enable the delay block function:

Figure 30. Delay block configuration


Delay line length configuration
Enable DLYB by setting DEN bit = 1
Enable the length sampling by setting SEN bit = 1
Enable the delay cells by setting SEL bit = 12
Unit = 0
Update unit value: Unit++
Wait LNGF = 1
LNG[10:0]>0 and (LNG[11]=0 or LNG[10]=0)?
No
Unit = 127?
No
Yes
Yes
Unit delay = 10
LNG[unit delay]=1?
No
Unit delay = 0?
Yes
No
Unit delay --
Yes
SEN = 0
Disable DLYB and select another feedback clock: sdmmc_io_in_ck or SDMMC_CKIN
DLYB enable failed
SEN = 1
SEL = (unit delay/2)+1
SEN = 0
DLYB enable success
Output clock phase procedure

6.2.2 DLYB configuration with tuning command

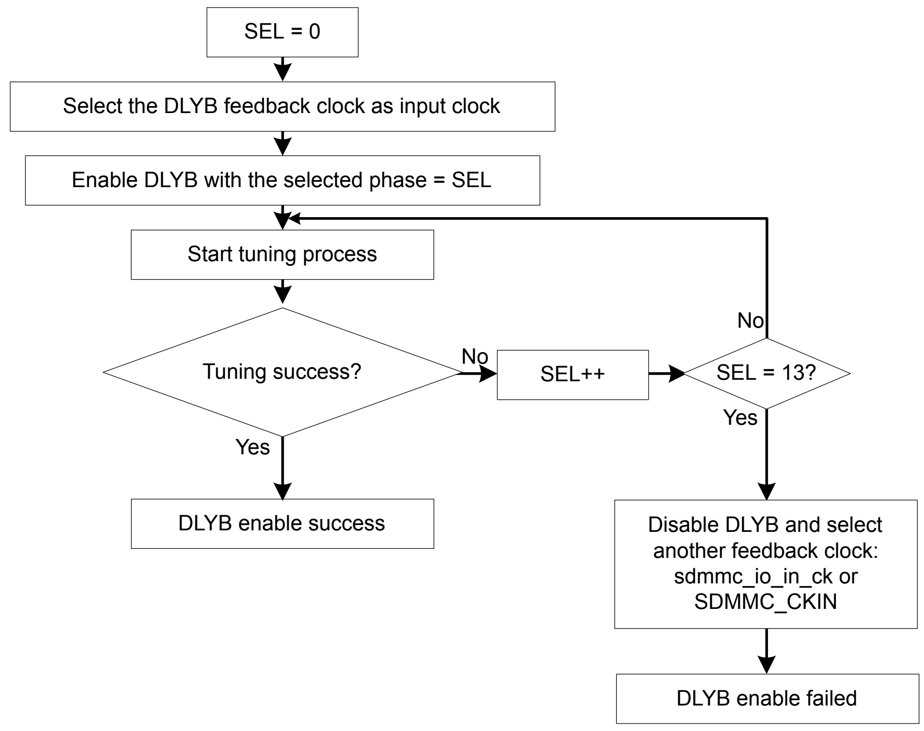
See below an example on how to configure DLYB with tuning command:

```
uint32_t sel=0;
      /*select the DLYB feedback clock as input clock*/
      MODIFY_REG(hsd->Instance->CLKCR, SDMMC_CLKCR_SELCLKRX, SDMMC_CLKCR_SELCLKRX_1 /*SDMMC_C
LKCR_SELCLKRX_1*/);
      /*Enable DLYB with sel = 0 and start tuning proceedure.repeat with updating the value o
f sel until sel = 13 or tuning succeeded */
      while (sel!=13)
      {
        DelayBlock_Enable(DLYB_SDMMC1,sel);
        errorstate = tuning(hsd);
      if(errorstate == HAL_OK)
      {
      return errorstate;
      }
        sel++;

      }
      /* in case tuning didnt succeed with each selected phase (sel = 13) disable dlyb and se
lect the SDMMC_CKIN */
      DelayBlock_Disable(DLYB_SDMMC1);
     MODIFY_REG(hsd->Instance->CLKCR, SDMMC_CLKCR_SELCLKRX, SDMMC_CLKCR_SELCLKRX_0/*SDMMC_CLK
CR_SELCLKRX_0*/);
```

The following diagram shows the steps to enable DLYB with tuning command:

Figure 31. **DLYB configuration with tuning command**


SEL = 0
Select the DLYB feedback clock as input clock
Enable DLYB with the selected phase = SEL
Start tuning process
Tuning success?
No
SEL++
SEL = 13?
No
Yes
Yes
DLYB enable success
Disable DLYB and select another feedback clock: sdmmc_io_in_ck or SDMMC_CKIN
DLYB enable failed

The process starts with SEL = 0, then follow the next steps:

1. Select the sdmmc_fb_ck as input clock.
2. Start the DLYB line length configuration procedure , when the delay line length is configured to one input clock period, select the output clock phase value SEL.

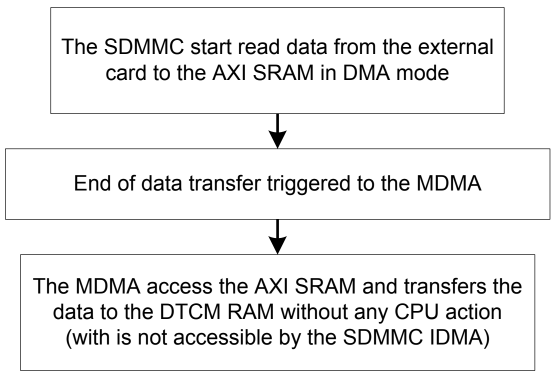
3. Start the tuning process: send CMD19, for tuning it returns 64 bytes of data.
4. Check data CRC error:
   – If no data CRC error occurred, DLYB was enabled successfully
   – If a data CRC error occurred, reconfigure the DLYB with another clock phase value until the tuning process succeed or until tuning was tested with all clock phases selection value [0..12] then disable the DLYB and keep the old configuration of input clock.

7 SDMMC host interface and MDMA

7.1 Transfer data from SD card to DTCM memory with SDMMC and MDMA

The following example describes how to configure the MDMA to transfer the data received successfully by the SDMMC to DTCM memory using end of data transfer trigger.

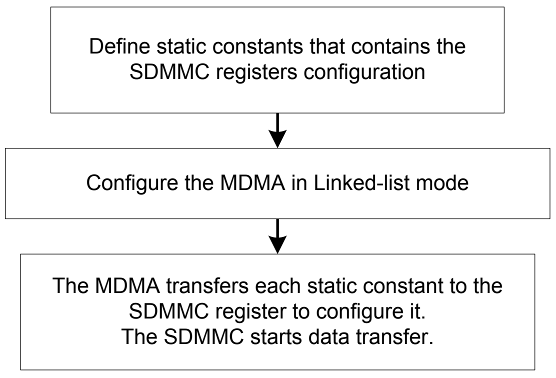
Figure 32. MDMA and SDMMC end of data trigger example


The SDMMC start read data from the external card to the AXI SRAM in DMA mode
End of data transfer triggered to the MDMA
The MDMA access the AXI SRAM and transfers the data to the DTCM RAM without any CPU action (with is not accessible by the SDMMC IDMA)

7.2 Configure the MDMA to enable data transfer with the SDMMC

The following example describes how to configure the MDMA to configure the SDMMC registers to enable data transfer in DMA Single-buffer mode without any CPU action and using the Linked-list mode.

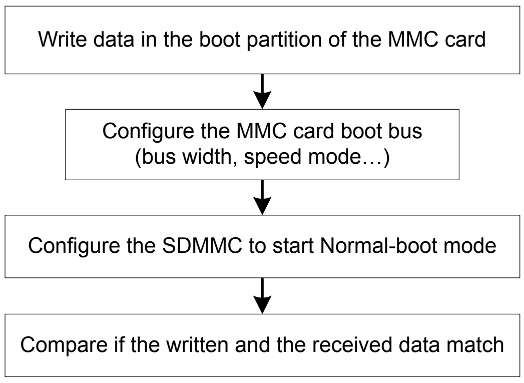
Figure 33. MDMA and SDMMC Linked-list mode example


Define static constants that contains the SDMMC registers configuration
Configure the MDMA in Linked-list mode
The MDMA transfers each static constant to the SDMMC register to configure it. The SDMMC starts data transfer.

8 How to enable Normal-boot mode from an MMC card

The following example describes how to execute Normal-boot mode from a multimedia card.

Figure 34. **MMC card Normal-boot mode example**


Write data in the boot partition of the MMC card
Configure the MMC card boot bus (bus width, speed mode…)
Configure the SDMMC to start Normal-boot mode
Compare if the written and the received data match

8.1 Write data in the boot partition of the MMC card

After initializing the MMC card, access the boot partition by sending CMD6 with one of the arguments shown in the table below to write boot code to it. CMD6 is used to configure the MMC EXTCSD.

Table 15. **Partitions selected to write into and their CMD6 arguments**

Partition selected to write into boot code	Argument for CMD6
User partition	0x03B3 0000
Boot partition 1	0x03B3 0100
Boot partition 2	0x03B3 0200

For this example's purposes we chose boot partition 1.

So in order to write data in the boot partition of the MMC card:

1. Access the boot partition and wait until the card is ready.
2. Erase the boot partition and write data to it and wait until the card is ready.

8.2 Configure the MMC card boot bus

1. Enable boot acknowledgment and configure the MMC card to boot from the desired boot partition (boot partition 1) and wait until MMC card is ready.
   – CMD 6 boot acknowledgment argument selection : 0x03B3 4000

Table 16. **Partitions selected for boot and their CMD6 arguments**

Partition selection for boot	Argument for CMD6
User partition	0x03B3 0800
Boot partition 1	0x03B3 1000
Boot partition 2	0x03B3 3000

2. Configure the bus width for boot mode:

Table 17. **MMC boot bus widths and their CMD6 arguments**

Bus width selected for boot	Argument for CMD6
1-bit	0x03B1 0000
4-bit	0x03B1 0100
8-bit	0x03B1 0200

3. Configure the speed mode for boot.

Table 18. **MMC speed modes selected for boot and their CMD6 arguments**

Bus width selected for boot(1)	Argument for CMD6
Legacy compatible	0x03B1 0000
HS SDR	0x03B1 0800
HS DDR	0x03B1 1000

4. Preserve or reset bus configuration after boot.

Table 19. **Bus configuration after boot and their CMD6 arguments**

Bus configuration after boot	Argument for CMD6
Preserve	0x03B1 0000
Reset	0x03B1 0400

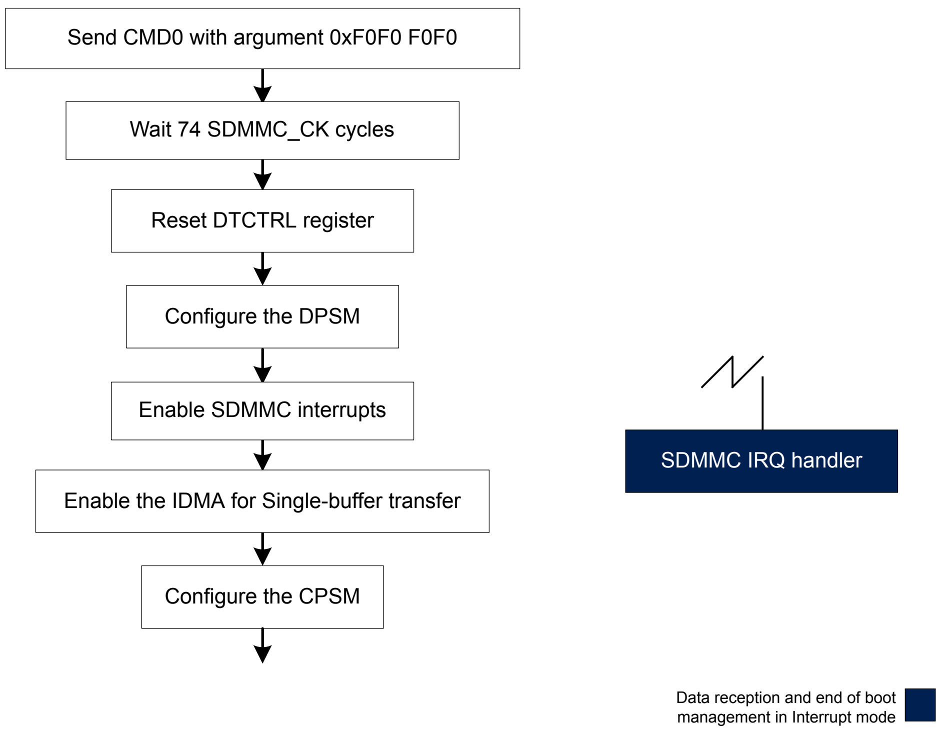
1. HS200 is not supported for boot mode

After all those steps, the MMC card is configured and ready for boot mode.

8.3 Configure the SDMMC to start Normal-boot mode

The following procedure is an example of SDMMC configuration for Normal-boot mode using the Single-buffer DMA mode.

Figure 35. **SDMMC configuration for Normal-boot mode**


Send CMD0 with argument 0xF0F0 F0F0
Wait 74 SDMMC_CK cycles
Reset DTCTRL register
Configure the DPSM
Enable SDMMC interrupts
Enable the IDMA for Single-buffer transfer
Configure the CPSM
SDMMC IRQ handler
Data reception and end of boot management in Interrupt mode

1. Send CMD0 with argument = 0xF0F0F0F0 to reset the MMC card to pre-idle state.
2. Wait 74 SDMMC_CK cycles.
3. Reset the DTCTRL register.
4. Configure the DPSM:
   - Enable BOOTACKEN.
   - Set the acknowledgement timeout and the data timeout (SDMMC_ACKTIMER and SDMMC_DTIMER).
   - Set the data block size and the data length.
   - Set the transfer direction (from MMC card to SDMMC).
   - Set the transfer mode as block mode.
   - Disable the DPSM.
5. Enable the following interrupts:

Table 20. **Interrupts to enable to configure SDMMC to start Normal-boot mode**

SDMMC interrupts enabled	Description
ACKTIMEOUT	Boot acknowledgement timeout
ACKFAIL	Boot acknowledgement check fail
DCRCFAIL	Data CRC check failed
CMDSENT	Command sent
RXOVERR	Received FIFO overrun

SDMMC interrupts enabled	Description
DATAEND	Successful end of data transfer

6. Enable the IDMA for Single-buffer transfer mode then configure the CPSM:
   – Select the Normal-boot mode.
   – Enable boot mode (BOOTEN = 1).
   – Enable the CPSM.

SDMMC IRQ handler

The IRQ handler manages the errors. It disables the BOOTEN bit after the data end flag is set.
If CMDSENT flag is set and the MMC card has exited boot mode, it clears the flag and calls the callback function.

8.3.1 Compare the written and the received data

After receiving data with Normal-boot mode, compare both of the written and the received data to confirm that they match.

9 Conclusion

The STM32H743/753 MCUs offer two new flexible SDMMCs, each one of them offer multiple different features allowing user to select the best configuration for his application. The SDMMC host interface supports multiple different memory devices with a very high-speed data transfer. This application note demonstrates the STM32H7 Series SDMMC host interface performances and its flexibility. The user of this interface allows lower development costs and fasten the time to market.

Revision history

Table 21. Document revision history

Date	Version	Changes
26-Nov-2018	1	Initial release.

Contents

List of tables

List of figures