Application Note

Multimedia Processor for Mobile Applications

EMMC Interface

EMMA Mobile1

Document No. S19904EJ1V0AN00
Date Published Aug. 2009

PREFACE

Purpose The purpose of this document is to specify the usage of EMMA Mobile1 EMMC (SDM) interface.

Organization This document includes the following:

- Introduction
- Usage of EMMC(SDM) Interface
- Example of EMMC Operation
- EMMC Driver Function

Notation Here explains the meaning of following words in text:

Note	Explanation of item indicated in the text
Caution	Information to which user should afford special attention
Remark	Supplementary information

Related document The following tables list related documents.

Reference Document

Document Name	Version/date	Author	Description
S19265EJ1V0UM00_ASMUGIO.pdf	1st Edition	NECEL	SMU&GPIO user’s manual
S19268EJ1V0UM00_1chip.pdf	1st Edition	NECEL	1 chip user’s manual
S19361JJ2V0UM00_SDI.pdf	2nd Edition	NECEL	SDM interface user’s manual
S19907EJ1V0AN00_GD.pdf	1st Edition	NECEC	GD Spec
KMCEG0000A-S9980(4GB moviNAND_8Gb MLC Based)_0.0.pdf	V0.0	SAMAUNG	EMMC chip user’s manual

Disclaimers

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1 Introduction

1.1 Outline

This document will show users how to operate EMMC chip on EMMA Mobile1 evaluation board. The EMMC operation will use SDM interface. More details about EMMC feature please refer to" **KMCEG0000A-S9980 (4GB moviNAND_8Gb MLC Based)_0.0.pdf**" (EMMC chip user's manual), please contact "Samsung Electronics " to get the EMMC chip user's manual.

1.2 Development Environment

- Hardware environment of this project is listed as below.

Table 1-1 Hardware Environment

Name	Version	Maker
EMMA Mobile 1 evaluation board (PSKCH2Y-S-0016-01)	-	NEC Electronics
PARTNER-Jet ICE ARM	M20	Kyoto Microcomputer Co. Ltd

- Software used in this project is listed as below.

Table 1-2 Software Environment

Name	Version	Maker
GNUARM Toolchain	V4.3.2	GNU
WJETSET-ARM	V5.10a	Kyoto Microcomputer Co. Ltd

Chapter 2 Usage of EMMC(SDM) Interface

According to the hardware feature, the EMMA Mobile 1 EMMC (SDM) interface has the following main function:

1. Initialization
2. Data Transfer
3. Erase
4. Write Protect Management
5. Lock/Unlock Operation

2.1 Initialization

Following figure shows EMMA Mobile 1 EMMC initialize progress:

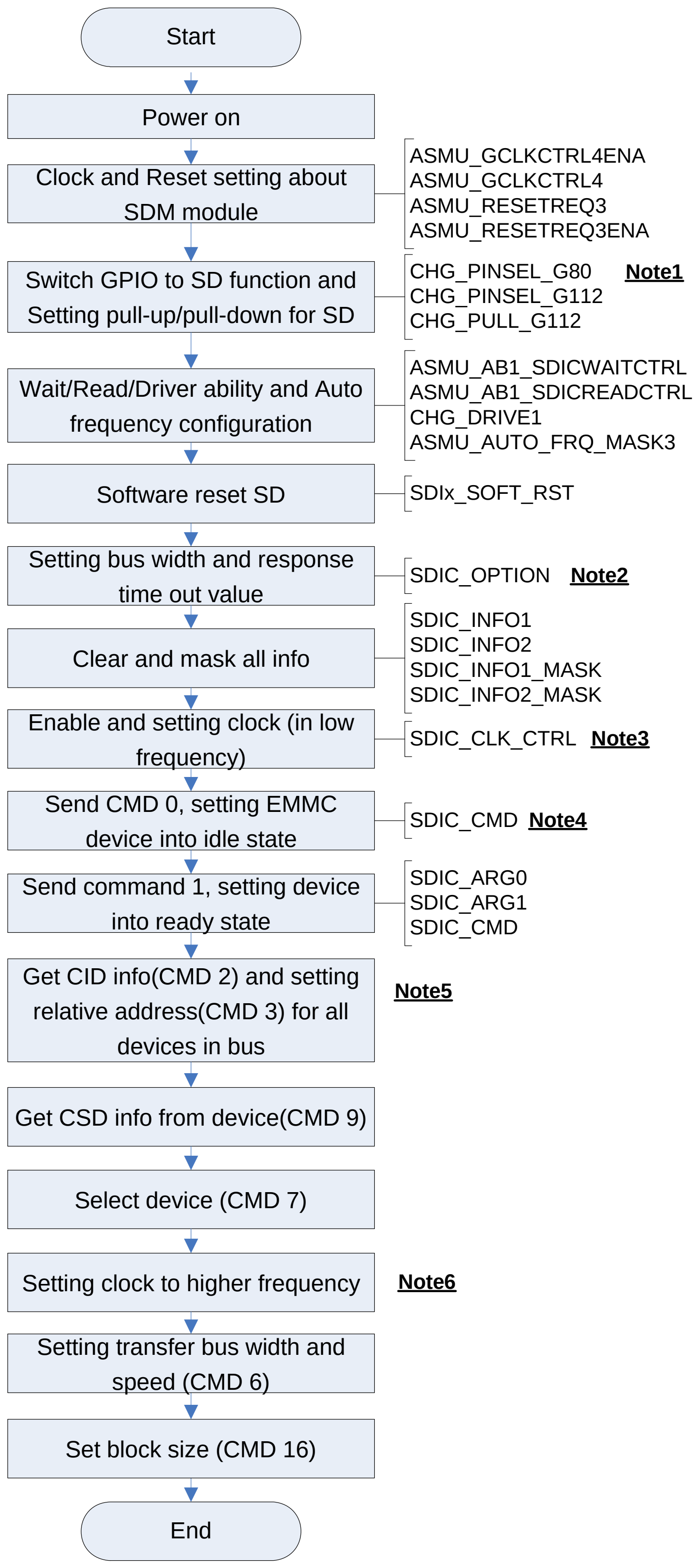
Start
Power on
Clock and Reset setting about SDM module
ASMU_GCLKCTRL4ENA
ASMU_GCLKCTRL4
ASMU_RESETREQ3
ASMU_RESETREQ3ENA
Switch GPIO to SD function and Setting pull-up/pull-down for SD
CHG_PINSEL_G80 Note1
CHG_PINSEL_G112
CHG_PULL_G112
Wait/Read/Driver ability and Auto frequency configuration
ASMU_AB1_SDICWAITCTRL
ASMU_AB1_SDICREADCTRL
CHG_DRIVE1
ASMU_AUTO_FRQ_MASK3
Software reset SD
SDIx_SOFT_RST
Setting bus width and response time out value
SDIC_OPTION Note2
Clear and mask all info
SDIC_INFO1
SDIC_INFO2
SDIC_INFO1_MASK
SDIC_INFO2_MASK
Enable and setting clock (in low frequency)
SDIC_CLK_CTRL Note3
Send CMD 0, setting EMMC device into idle state
SDIC_CMD Note4
Send command 1, setting device into ready state
SDIC_ARG0
SDIC_ARG1
SDIC_CMD
Get CID info(CMD 2) and setting relative address(CMD 3) for all devices in bus
Note5
Get CSD info from device(CMD 9)
Select device (CMD 7)
Setting clock to higher frequency
Note6
Setting transfer bus width and speed (CMD 6)
Set block size (CMD 16)
End

Figure 2-1 EMMA Mobile 1 EMMC Initialization

Note:

1) Switch pins to SD function, in this document, use SD2 as the EMMC interface, users should operate this step according the actual hardware connection, for example: if users connect the EMMC with SD1 interface, please switch the related GPIO to SD1 function.

2) Before the initialization of EMMC device, set the bus width to be 1bit and the response time out value to be maximum, reason is: in EMMC identification progress, bus width will use 1bit and the clock frequency should in 10-400 KHz range.

3) After setting the clock for SDM transfer, please wait 1ms for stability.

4) After send command, user should check the command response, if error occurred (except when response time out for CMD 2), the initialization will be ended abnormally.
Commands simple description please refers to the "**APPENDIX B COMMANDS**".
More details about the commands format and function please refer to "**Chapter4.6 Commands**" of the EMMC chip user's manual.

5) CID: Card Identification
CSD: Card Specific Data
More details about CID and CSD information please refer to "**Chapter 5.0 REGISTERS**" of the EMMC chip user's manual

6) After initialization, users can configure the bus width and clock frequency for transfer speed according the CSD parameter of EMMC chip, the transfer speed defines in CSD register just clock frequency not in high speed mode, so the actual transfer speed is related to EMMC chip specification version.

7) More details about SDM registers and related bits please refer to SDM interface user's manual of EMMA Mobile 1.

2.2 Data Transfer

EMMC has two kinds of data transfer mode: DMA mode and CPU mode. In realize operation, mainly use DMA mode to transfer data, so in this chapter, introduce the DMA mode operation, about EMMC single block read/write in CPU mode; please refer to **"APPENDIX A EMMC Driver Function".** Following figure shows EMMA Mobile 1 EMMC DMA read/write progress:

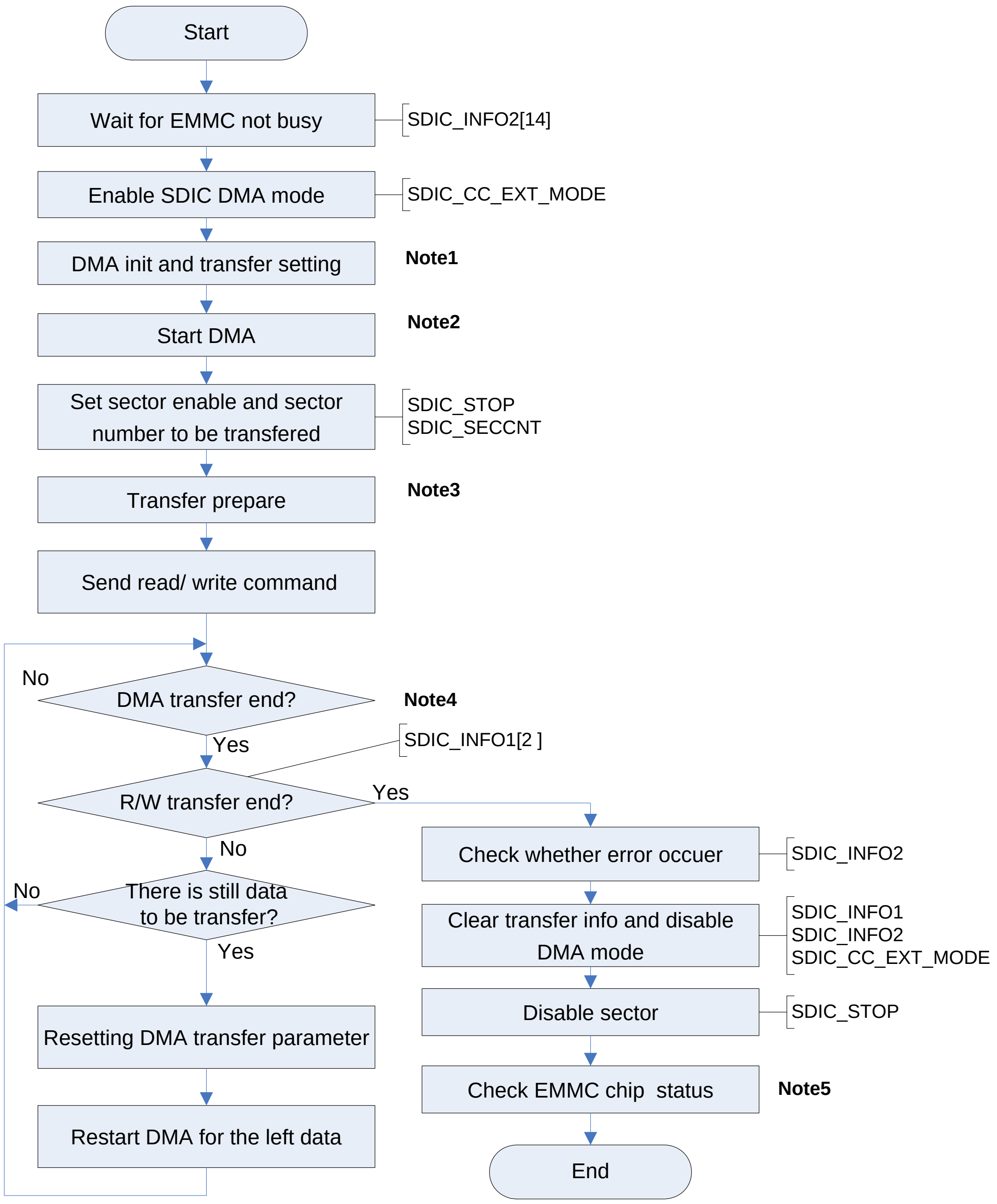

Figure 2-2 EMMA Mobile 1 EMMC Data Transfer in DMA Mode

Note:

1) In DMA init and transfer setting step, user should reset DMA channel, open DMA clock, clear DMA interrupt source, set transfer parameter for DMA.

DMA reset setting related register:
    ASMU_RESETREQ0ENA
    ASMU_RESETREQ0
DMA clock setting related register:
    ASMU_GCLKCTRL0
    ASMU_GCLKCTRL0ENA

P2M Clear DMA interrupt source related register:
    DMA_P2M_PE0_LCH4LCH7_INT_REQ_CL
    DMA_P2M_DSP_LCH4LCH7_INT_REQ_CL

M2P Clear DMA interrupt source related register:
    DMA_M2P_PE0_LCH4LCH7_INT_REQ_CL
    DMA_M2P_DSP_LCH4LCH7_INT_REQ_CL

P2M DMA transfer setting related register:
    DMA_P2M_LCH5_AADD
    DMA_P2M_LCH5_BADD
    DMA_P2M_LCH5_BOFF
    DMA_P2M_LCH5_BSIZE
    DMA_P2M_LCH5_BSIZE_COUNT
    DMA_P2M_LCH5_LENG
    DMA_P2M_LCH5_MODE

M2P DMA transfer setting related register:
    DMA_M2P_LCH5_BADD
    DMA_M2P_LCH5_AADD
    DMA_M2P_LCH5_AOFF
    DMA_M2P_LCH5_ASIZE
    DMA_M2P_LCH5_ASIZE_COUNT
    DMA_M2P_LCH5_LENG
    DMA_M2P_LCH5_MODE

2) Start DMA transfer.
P2M start register
    DMA_P2M_CONT

M2P start register
    DMA_M2P_CONT

3) Before data transfer, clear all information register, make sure data transfer end, enable the related interrupt.

Related register:

SDIC_INFO1
SDIC_INFO2
SDIC_INFO1_MASK
SDIC_INFO2_MASK
SDIC_STOP

4) Check whether DMA transfer has ended

P2M transfer related register:

DMA_P2M_CONTSTATUS

M2P transfer related register:

DMA_M2P_CONTSTATUS

5) After data transfer, send CMD 13(SEND_STATUS) to read EMMC status register info, check whether error occurred.

Related register:

SDIC_ARG0
SDIC_ARG1
SDIC_CMD
SDIC_RSP0
SDIC_RSP1
SDIC_INFO1
SDIC_INFO2
SDIC_INFO1_MASK
SDIC_INFO2_MASK

More details about EMMC status register information please refer to “**Chapter 4.9 moviNAND Status**” of the EMMC chip user’s manual.

2.3 Erase

Following figure shows EMMA Mobile 1 EMMC erase progress:

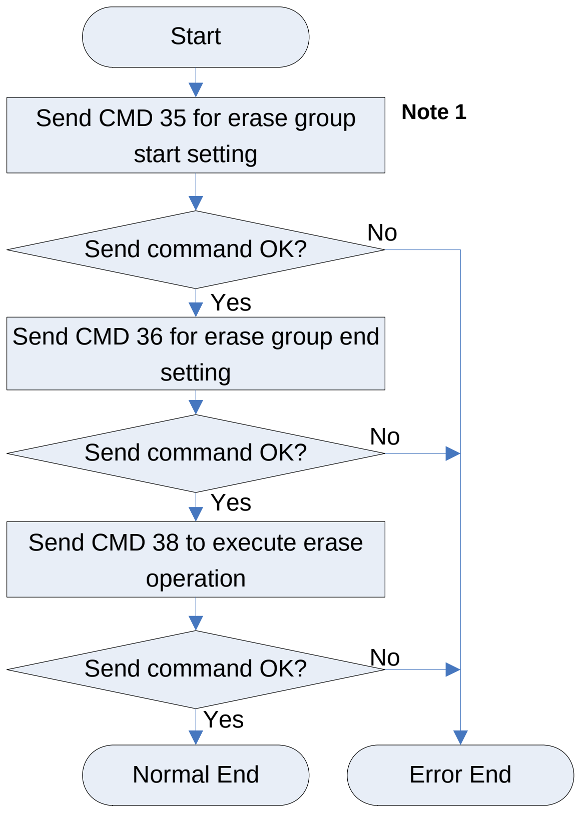
Start
Send CMD 35 for erase group start setting
Note 1
Send command OK?
No
Yes
Send CMD 36 for erase group end setting
Send command OK?
No
Yes
Send CMD 38 to execute erase operation
Send command OK?
No
Yes
Normal End
Error End

Figure 2-3 EMMA Mobile 1 EMMC Erase Operation

Note:

1) Parameter about CMD 35, CMD36 will be the address to start and end, and the address will be in Group erase unit, more details about erase operation and meaning of erase group unit please refer to “**Chapter 4.2.9 Erase**” and “**Chapter 4.10 Memory Array Partitioning**” of EMMC chip user’s manual.

2.4 Write Protect Management

Write protect (WP) management including there command: set write protect (CMD 28), clear write protect (CMD 29) and send write protect info (CMD30). To realize these functions, users just need to send command to EMMC device.

The write protect argument is WP address, it based on unit of WP_GRP_SIZE, which is defined in CSD register.

More details about write protect function and meaning of WP_GRP_SIZE please refer to "**Chapter 4.2.10 Write Protect Management**" and "**Chapter 4.10 Memory Array Partitioning**" of EMMC chip user's manual.

2.5 Lock/Unlock Operation

Lock/Unlock operation including password setting and cancel lock/unlock EMMC device and enforce erase. To complete these function, please follow the fixed data structure format as followed.

Table 2-1 Lock/Unlock EMMC Data Structure

Byte #	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit2**	**Bit 1**	**Bit 0**
0	Reserved				ERASE	LOCK_UNLOCK	CLR_PWD	SET_PWD
1	PWD_LEN							
2	Password data							
...								
PWD_LEN + 1								

ERASE: ‘1’ = Forced Erase Operation (all other bits shall be ‘0’) and only command byte is sent.
LOCK/UNLOCK: ‘1’ = Locks the EMMC. ‘0’ = Unlock the EMMC (note that it is valid to set this bit together with SET_PWD but it is not allowed to set it together with CLR_PWD).
CLR_PWD: ‘1’ = Clears password (PWD).
SET_PWD: ‘1’ = Set new password to PWD
PWD_LEN: Defines the following password length (in bytes). Valid password length is 1 to 16 bytes.
PWD: The password (new or currently used depending on the command).

Following figure shows the lock/unlock operation flow:

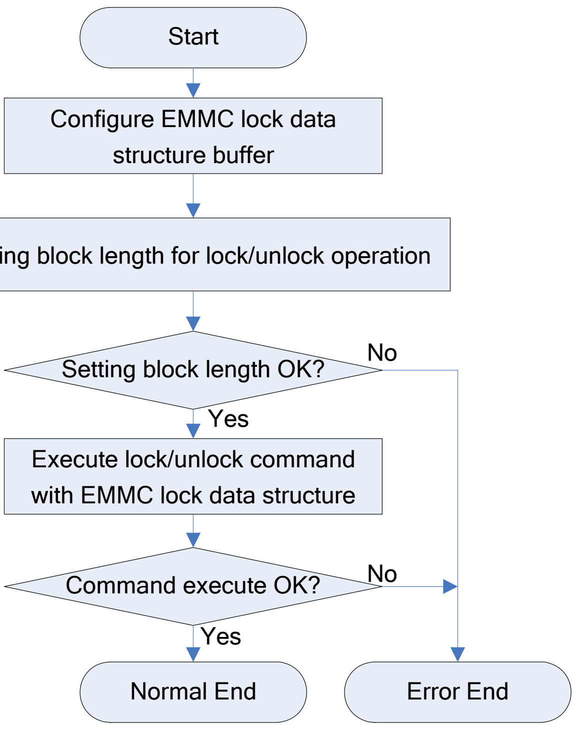
Start
Configure EMMC lock data structure buffer
Setting block length for lock/unlock operation
Setting block length OK?
No
Yes
Execute lock/unlock command with EMMC lock data structure
Command execute OK?
No
Yes
Normal End
Error End

Figure 2-4 EMMA Mobile 1 EMMC Lock/Unlock Operation

More details about lock/unlock operation and fixed data structure please refer to “**Chapter 4.2.11 moviNAND Lock/Unlock Operation**” of EMMC chip user’s manual.

Chapter 3 Example of EMMC Operation

3.1 Outline of EMMC Operation

This chapter will show users how to operate EMMC chip using SDM interface.

On EMMA Mobile 1 evaluation board (PSKCH2Y-S-0016-01), the external EMMC chip is KMCEG0000M (Manufacture: Samsung). Its capacity is 4GB, and it has multimedia card system specification Ver4.2 compatible feature. More details about the KMCEG0000A please refer to its user’s manual.

Figure 3-1 shows the connection of EMMA Mobile 1 EMMC interface and KMCEG0000M.

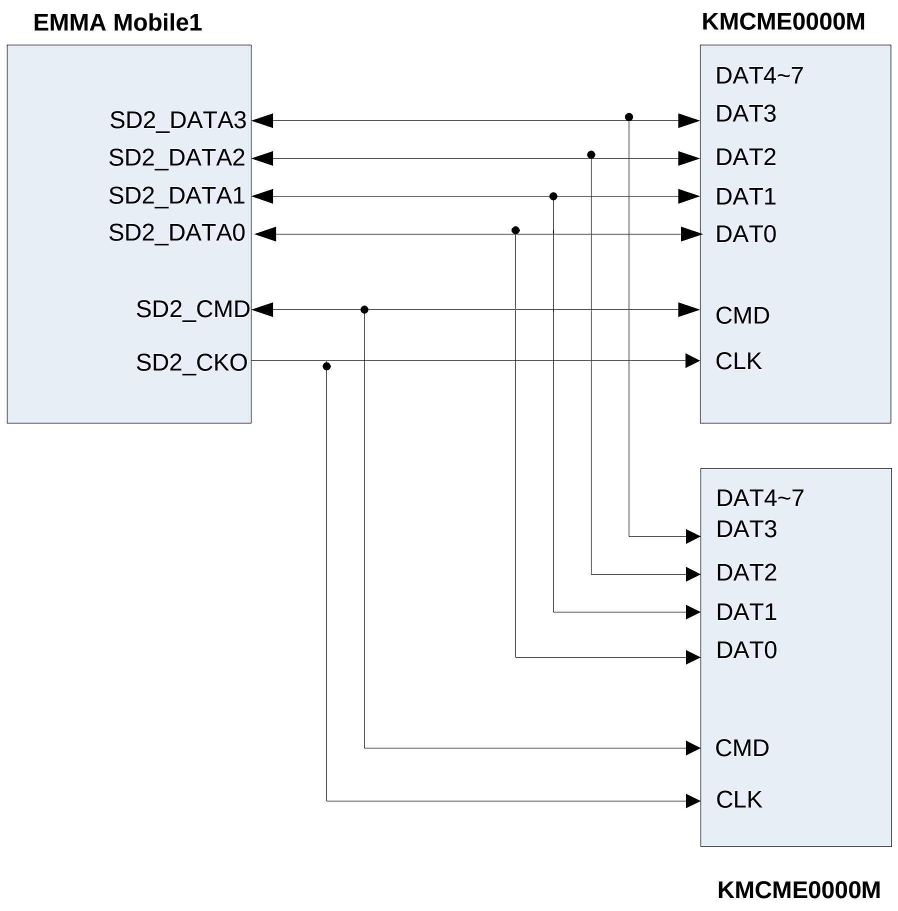

Figure 3-1 Connection between EMMA Mobile 1 and KMCEG0000A

3.2 Initialization

Before EMMC operation, initialization should be executed at first.

3.2.1 Operation Flow

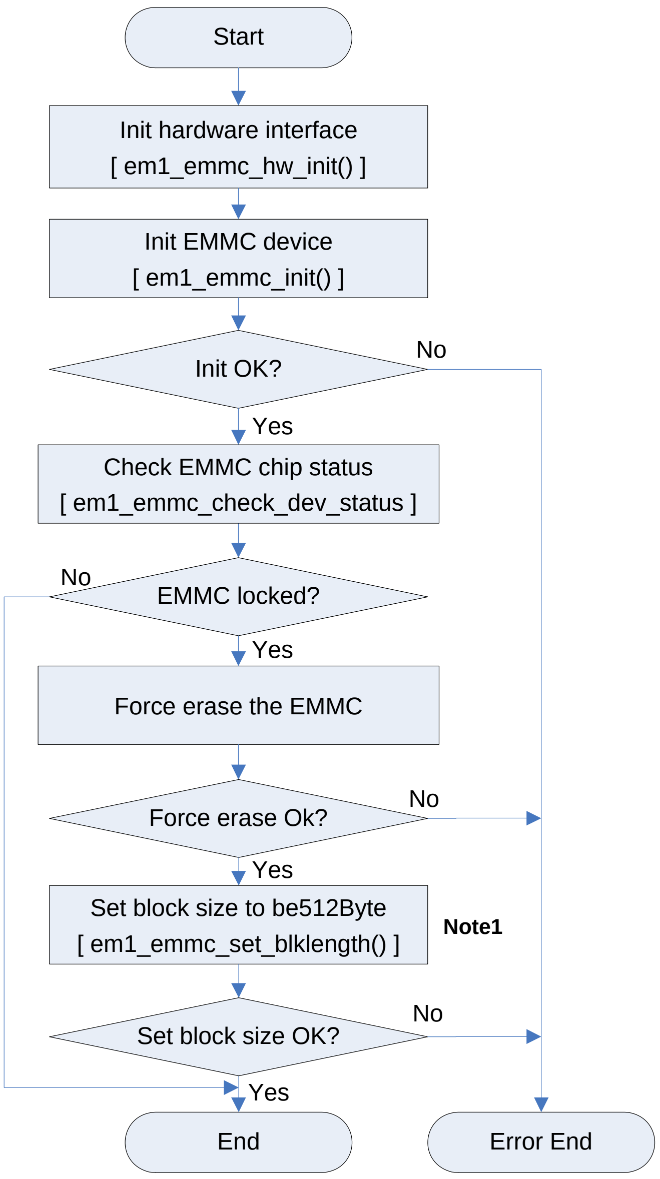

Figure 3-2 Initialization before Test

More details about the functions used in initialization please refer to “**APPENDIX A EMMC Driver Function**”

3.2.2 Operation Detail

(1) Init SDM module hardware

Init SDM module (function: em1_emmc_hw_init()).

Following steps shows the hardware initialization (em1_emmc_hw_init()) progress:

Step1: Reset setting

ASMU_RESETREQ3[4]	(0: reset; 1: cancel reset)
ASMU_RESETREQ3ENA [4]	(0: disable setting; 1: enable setting)

Step2: Clock setting

ASMU_GCLKCTRL4[7]	(0: close clock; 1: open clock)
ASMU_GCLKCTRL4ENA[7]	(0: disable setting; 1: enable setting)

Step3: Switch pins to SD function:

CHG_PINSEL_G80 = 0x04000000	(GIO_P93 -> SD2_CKI)
CHG_PINSEL_G112 = 0x00000555	(GIO_P112-117 -> SD2_CKO/CMD/DATA0-3)

Step4: Pull-up/down setting

CHG_PULL_G112 = 0x00666661	(SD2_CKO: Pull-up/down Disable, SD2_CMD and DATA0-3: Input Enable, Pull-up enable)

Step5: Drive capability setting

CHG_DRIVE1 = 0x05000000	(SD2_CK and SD2 pins : 4mA, default value)

Setp6: Read and wait control register setting

ASMU_AB1_SDICWAITCTRL = 0x00000300

ASMU_AB1_SDICREADCTRL = 0x00000000 (default value)

Setp7: Auto frequency control setting

ASMU_AUTO_FRQ_MASK3 = 0x07 (default value)

(2) Init EMMC device

EMMC device initialization (em1_emmc _init()) including following steps:

Step1: Power on EMMC device

Step2: Soft reset SD

SDIC_SOFT_RST = 0x0000	(module reset)
SDIC_SOFT_RST = 0x0007	(release reset)

Step3: Set bus width and time out value for response, card detect stable time

SDIC_OPTION = 0x80EE (bus width: 1bit; biggest time value for response time out and card detection stable time)

Step4: Clear and mask all information and interrupt

SDIC_INFO1 = 0x0000

SDIC_INFO2 = 0x0000

SDIC_INFO1_MASK = 0xFFFF

SDIC_INFO2_MASK = 0xFFFF

Step5: Set clock

SDIC_CLK_CTRL = 0x0140 (divide factor: 256, about 325KHz; Use the lower frequency when init EMMC)

Delay 1ms after setting the clock frequency

Step6: Send CMD0, make EMMC chip to idle state, including following steps:

Clear all error and information (SDIC_INFO1, SDIC_INFO2)

Enable response end interrupt occur (SDIC_INFO1_MASK)

Enable all error occur (SDIC_INFO2_MASK)

Send command (SDIC_CMD)

Wait for command send end or error occur (SDIC_INFO1, SDIC_INFO2)

Check whether error occur (SDIC_INFO2, SDIC_RSP0, SDIC_RSP1)

Note:

If command has argument, setting the argument at first (SDIC_ARG0, SDIC_ARG1)

Step7: Send CMD1 with argument 0x40FF8080, make EMMC chip to ready state.

SDIC_ARG0 = 0x8080

SDIC_ARG1 = 0x40FF

Check whether power up is ready? (SDIC_RSP1[15]). If not ready, send CMD1 again; if in ready state, run to next step. Details about send command please refer to step6.

Step8: All EMMC chip send CID to host, and get RCA (relative card address) information from host, including following steps:

Send CMD2(ALL_SEND_CID) with argument 0x00000000

Send CMD3 (SNED_RELATIVE_ADDR) with argument RCA (RCA init value is 1)

Increase RCA value and circle Step8 until all the EMMC chip on the SD bus has get a RCA from host, at this time, response time out error will occur, run to next step. Details about send command please refer to step6.

Step9: Get CSD information, including following steps:

Send CMD9 (SNED_CSD) with argument RCA (the one that need to send CSD information to host). Details about send command please refer to step6.

Get CSD information from the command response (SDIC_RSP0~SDIC_RSP7)

Step10: Select EMMC chip

Send CMD7 (SELECT) with argument RCA(which selected to communicate). Details about send command please refer to step6.

Step11: Re-setting clock

SDIC_CLK_CTRL = 0x0301 (divide factor: 4, about 20.8MHz; Use higher frequency after init EMMC)

Delay 1ms after setting the clock frequency

Step12: Setting extends CSD register, including following steps:

Send CMD6 (SWITCH) with argument 0x03B70100 (bus width: 4 bits) to configure extends CSD register.

Change bus width in SDIC_OPTION register

Send CMD6 (SWITCH) with argument 0x03B90100 (high speed) to configure extends CSD register. Details about send command please refer to step6.

Step13: Set block length, including following steps:

Send CMD16 (SET_BLOCKLEN) with argument “block length”. Details about send command please refer to step6.

Changing block size values in SDIC_SIZE register.

More details about the initialization progress please refer to “**APPENDIX A EMMC Driver Function**”.

(3) Check EMMC Device Status

According to the EMMC chip feature (if chip has password, the chip will be locked automatically after power on), in order to make sure read/write operation success after power on, check the device status at first (function: em1_emmc_check_dev_status()). If EMMC is locked, enforce erase the whole chip by lock/unlock command. If enforce erase OK, set block size to be 512 byte, then the read and right operation followed can works normally.

Check device status (em1_emmc_check_dev_status()) including following steps:

Step1: Send CMD13 (SEND_STATUS) with argument RCA. Details about send command please refer to step 6 of “**(2) Init EMMC device**” in “**Chapter 3.2.2 Operation Details**”

Step2: Read response value (SDIC_RSP0, SDIC_RSP1)

Step3: Check whether EMMC locked and error occurred according the EMMC status structure. More details about the EMMC status structure please refer to “**Chapter 4.9 moviNAND Status**” of EMMC chip user’s manual.

(4) Enforce Erase If EMMC Device Locked

If EMMC is locked, enforce erase the whole chip by lock/unlock command (function: em1_emmc_set_blklength() and em1_emmc_lock_unlock()).

a) Set block length to be 1.
Details about block length setting (em1_emmc_set_blklength()) please refer to step13 of "**(2) Init EMMC device**" in "**Chapter 3.2.2 Operation Details**".

b) Set lock/unlock data structure, make sure the bit3 "ERASE" of the lock/unlock data structure to be 1, and execute lock/unlock function. More details about lock/unlock operation and fixed data structure please refer to "**Chapter 4.2.11 moviNAND Lock/Unlock Operation**" of EMMC chip user's manual,
Lock/unlock function (em1_emmc_lock_unlock()) including following steps:

Step1: Ensure EMMC interface is not busy, check bit 14 of SDIC_INFO register

Step2: Enable sector setting and set sector number to 1 (SDIC_STOP, SDIC_SECCNT)

Step3: Prepare for data transfer, including following steps:
Clear interrupt information in SDIC_INFO1 register
Clear all error information in SDIC_INFO2 register
Set transfer none stop (SDIC_STOP[0] = 0)
Enable all error interrupt in SDIC_INFO2_MASK register
Enable read/write access interrupt occur in SDIC_INFO1_MASK register

Step4: Send CMD42 (LOCK_UNLOCK) without argument. Details about send command please refer to step 6 of "**(2) Init EMMC device**" in "**Chapter 3.2.2 Operation Details**"

Step5: Wait for write enable (SDIC_INFO2) and check whether error occur (SDIC_RSP0, SDIC_RSP1)

Step6: Write data
If no errors occur, write data (lock/unlock data structure) to EMMC chip (SDIC_BUF0 = data)

Step7: Wait for data transfer end or error occur (SDIC_INFO1, SDIC_INFO2)

Step8: Send CMD13 (SEND_STATUS) with argument RCA to check device status. Detail steps about check EMMC device status please refer to "**(3) Check EMMC Device Status**" in "**Chapter 3.2.2 Operation Details**".

Note:
Details about send command please refer to step 6 of "**(2) Init EMMC device**" in "**Chapter**

3.2.2 Operation Details”.

(5) Set Block Size after Enforce Erase

After enforce erase, in order to make sure the read and right operation followed can works normally, set block size to be 512 byte (function: em1_emmc_set_blklength()).

Details about block length setting (em1_emmc_set_blklength()) please refer to step13 of “**(2) Init EMMC device**” in “**Chapter 3.2.2 Operation Details**”.

Note:

The EMMA Mobile 1 SDM interface can support 512 byte as the maximum block size.

3.3 Example of EMMC Single Block Read/Write

In this example, we will write fixed data to EMMC chip, then read out and compare whether the data is right, both read and write operation will use single CPU operation mode.

3.3.1 Operation Flow

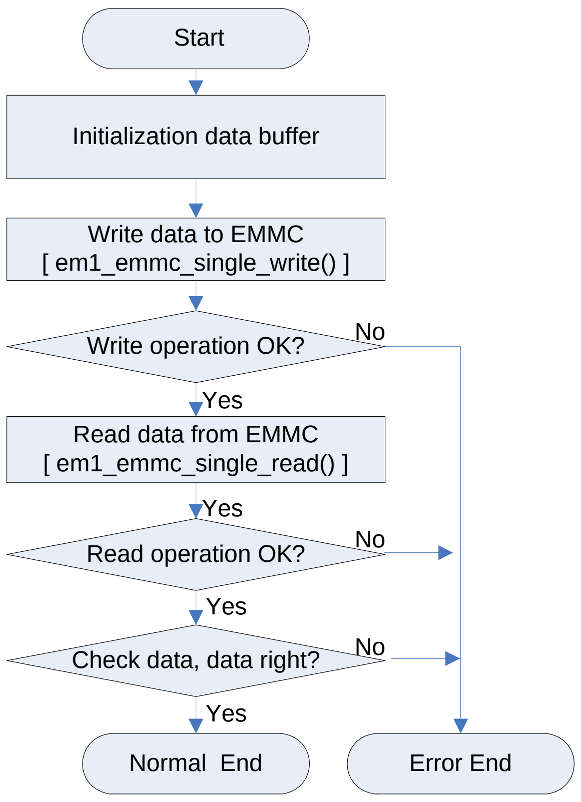

Figure 3-3 EMMC Single Read/Write Operation Flow

More details about the functions used in this example please refer to “**APPENDIX A EMMC Driver Function**”

3.3.2 Operation Detail

(1) Initialization Data Buffer

Initialize the write data buffer, set fixed value (as the following code segment shows) to write data buffer which will be wrote to EMMC chip. At the same time, initialize the read data buffer with 0, it will read out data from EMMC chip.

```
for (i =0; i<SDM_BLOCKLEN_VAL; i++ )
{
    g_write_buff[i] = (i&0xFF);
}
```

(2) Write Data to EMMC

Call the “em1_emmc_single_write()” function to write data into EMMC chip. If write operation failed (error occur during data transfer), end the operation and print error information; if write operation works OK, continue the test program.

Following steps shows the write operation progress:

Step1: Ensure EMMC interface is not busy, check bit 14 of SDIC_INFO register

Step2: Enable sector setting and set sector number (SDIC_STOP, SDIC_SECCNT)

Step3: Prepare for data transfer, including following steps:

Clear interrupt information in SDIC_INFO1 register
Clear all error information in SDIC_INFO2 register
Set transfer none stop (SDIC_STOP[0] = 0)
Enable all error interrupt in SDIC_INFO2_MASK register
Enable read/write access interrupt occur in SDIC_INFO1_MASK register

Step4: Send CMD24 (WRITE_SINGLE_BLOCK) with argument “write address”. Detail steps about send command please refer to step 6 of “**(2) Init EMMC device**” in “**Chapter 3.2.2 Operation Details**”.

Step5: Wait for write enable (SDIC_INFO2) and check whether error occur (SDIC_RSP0, SDIC_RSP1)

Step6: Write data

If no errors occur after send CMD24 (WRITE_SINGLE_BLOCK), write data to EMMC chip (SDIC_BUF0 = data)

Step7: Wait for data transfer end or error occur (SDIC_INFO1, SDIC_INFO2)

Step8: Send CMD13 (SEND_STATUS) with argument RCA to check device status. Detail steps about check EMMC device status please refer to "**(3) Check EMMC Device Status**" in "**Chapter 3.2.2 Operation Details**".

More details about single block write operation please refer to "**chapter A.4.12 Single Block Write**".

(3) Read Data

Call "em1_emmc_single_read()" function to read out the written data by step (2) from EMMC. If read failed, end the operation and print error; if read OK, continue the test program.
Following steps shows the read operation progress and registers configurations:

Step1: Ensure EMMC interface is not busy, check bit 14 of SDIC_INFO register

Step2: Enable sector setting and set sector number (SDIC_STOP, SDIC_SECCNT)

Step3: Prepare for data transfer, including following steps:
Clear interrupt information in SDIC_INFO1 register
Clear all error information in SDIC_INFO2 register
Set transfer none stop (SDIC_STOP[0] = 0)
Enable all error interrupt in SDIC_INFO2_MASK register
Enable read/write access interrupt occur in SDIC_INFO1_MASK register

Step4: Send CMD17 (READ_SINGLE_BLOCK) with argument "read address". Detail steps about send command please refer to step 6 of "**(2) Init EMMC device**" in "**Chapter 3.2.2 Operation Details**".

Step5: Wait for write enable (SDIC_INFO2) and check whether error occur (SDIC_RSP0, SDIC_RSP1)

Step6: Read data
If no errors occur after send CMD17 (READ_SINGLE_BLOCK), read data from EMMC chip (data = SDIC_BUF0)

Step7: Wait for data transfer end or error occur (SDIC_INFO1, SDIC_INFO2)

Step8: Send CMD13 (SEND_STATUS) with argument RCA to check device status. Detail steps about check EMMC device status please refer to "**(3) Check EMMC Device Status**" in "**Chapter 3.2.2 Operation Details**".

(4) Compare Data

Compare the read out data with the written data. If same, print OK; otherwise, print error end.

3.4 Example of EMMC Multi Block Operation

In this example, we will write fixed data (0x5A) to EMMC chip (first 16MB data area), then read out and compare whether the data is right. When read/write works OK, erase the first 2MB data area in EMMC chip, then read out and check whether data in first 2MB is zero or not. Both read and write operation will use multi block DMA operation mode.

3.4.1 Operation Flow

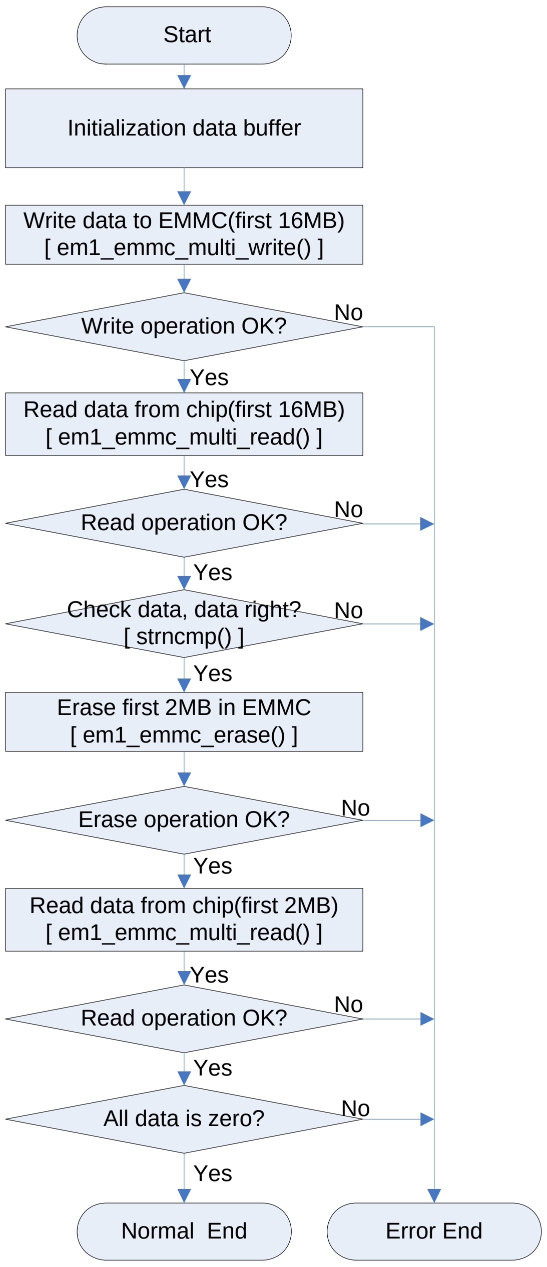

Figure 3-4 EMMC Multi block Operation Flow

More details about the functions used in this example please refer to "**APPENDIX A EMMC Driver Function**"

3.4.2 Operation Detail

(1) Initialization Data Buffer

Initialize the write data buffer, set fixed value (0x5A) to write data buffer which will be wrote to EMMC chip. At the same time, initialize the read data buffer with 0, it will read out data from EMMC chip.

(2) Write Data to EMMC

Setting parameters about the data transfer, call the "em1_emmc_multi_write()" function to write data into EMMC chip. Check the write operation, if failed, end the operation and print error information; if OK, continue the test program.

Details about multiple blocks write operation please refer to "**figure2-2 EMMA Mobile 1 EMMC Data Transfer in DMA Mode** ".

(3) Read Data from EMMC

Call "em1_emmc_multi_read()" function to read out the written data by step (2) from EMMC. Check the read operation, if read failed, end the operation and print error; if read OK, continue the test program.

Details about multiple blocks read operation please refer to "**figure2-2 EMMA Mobile 1 EMMC Data Transfer in DMA Mode** ".

(4) Compare Data

Compare the read out data with the written data. If same, print OK; otherwise, print error and end the test.

(5) Erase first 2MB in EMMC

Call "em1_emmc_erase()" function to erase the first 2MB data area in EMMC. If failed, end the operation and print error; if OK, continue the test program.

Details about EMMC erase operation please refer to "**figure2-3 EMMA Mobile 1 EMMC Erase Operation** ".

(6) Read Data from EMMC

Call "em1_emmc_multi_read()" function to read out the first 2MB data from EMMC. If read failed, end the operation and print error; if read OK, continue the test program.

(7) Compare Data

Compare the read out data. If 2MB data are all zero, test operation works OK; otherwise, error end.

3.5 Example of EMMC Write Protect Operation

In write protect operation example, set write protect(WP) to fixed address, then write data, if write operation works normally, it means the WP operation failed, otherwise, the WP operation works OK, then clear the write protect for fixed address, and write operation followed will works normally.

3.5.1 Operation Flow

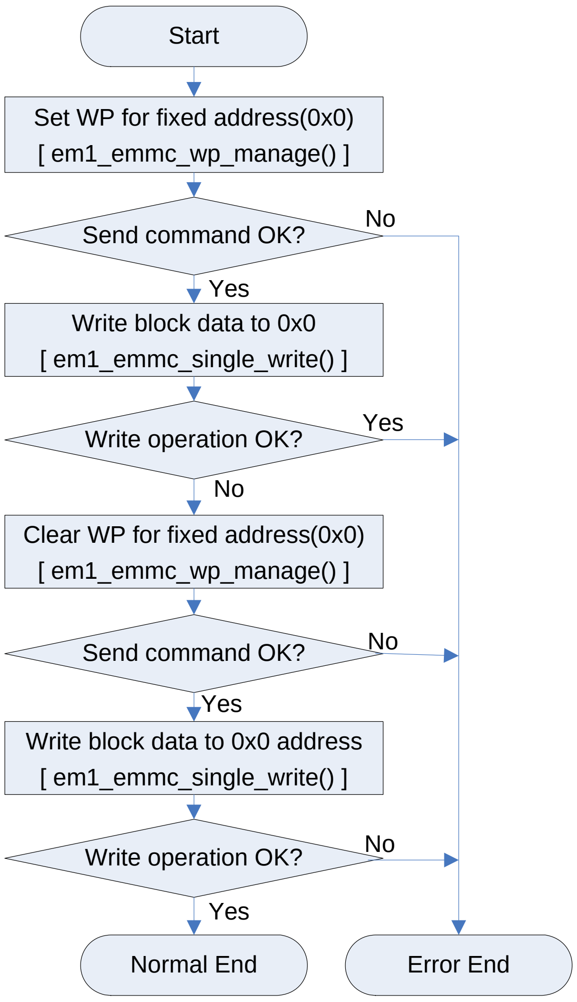

Figure 3-5 EMMC Write Protect Operation Flow

More details about the functions used in this example please refer to "**APPENDIX A EMMC Driver Function**"

3.5.2 Operation Detail

(1) Set WP to 0x0 Address

Send CMD28(SET_WRITE_PROT) to EMMC(function: em1_emmc_wp_manage()), with address 0x0, if command send failed, print error and end test, otherwise, continue the test.
Write protect management (m1_emmc_wp_manage()) including following steps:

Step1: Check whether the command is supported

Step2: Send command CMD28/CMD29 (SET_WRITE_PROT/ CLR_WRITE_PROT) with the argument “write protect unit” (based on unit of WP_GRP_SIZE). Detail steps about send command please refer to step 6 of “**(2) Init EMMC device**” in “**Chapter 3.2.2 Operation Details**”.

(2) Write Data to EMMC

Setting parameters about the data transfer, call the “em1_emmc_single_write()” function to write block data into 0x0 in EMMC chip. Check the write operation, if OK, means WP operation failed, end the operation and print error information; if OK, continue the test program.
Details about EMMC single write operation please refer to “**(2) Write Data to EMMC**” of “**Chapter 3.3.2 Operation Detail**”.

(3) Clear WP for 0x0 Address

Send CMD28 (CLR_WRITE_PROT) to EMMC(function: em1_emmc_wp_manage()), with address 0x0, if command send failed, print error and end test, otherwise, continue the test.
Details about Write protect management please refer to “**(1) Set WP to 0x0 Address”.**

(4) Write Data to EMMC

Setting parameters about the data transfer, call the “em1_emmc_single_write()” function to write block data into 0x0 in EMMC chip. Check the write operation, if OK, means WP test operation works OK, otherwise, the WP test failed.

3.6 Example of EMMC Lock/Unlock Operation

In lock/unlock example, set password for EMMC at first, then lock the device, check EMMC status, if EMMC is not locked, means lock operation failed, otherwise, lock works OK; then unlock the device, and read the EMMC status again to check the unlock operation.

Note:

In this test, use “123” as the password, and before end the lock/unlock test, in order not to affect other tests, should clear the password, and restore the block size (512 byte).

3.6.1 Operation Flow

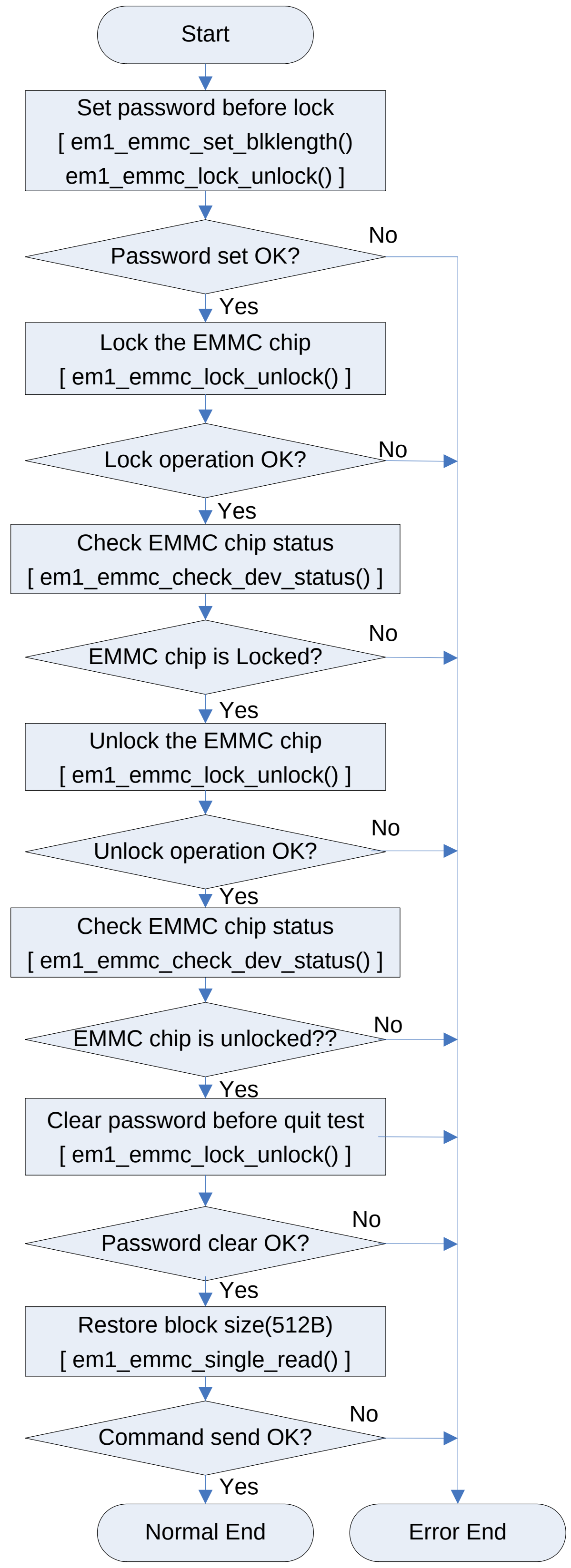

Figure 3-6 EMMC Lock/Unlock Operation Flow

3.6.2 Operation Detail

(1) Set Password before Lock

Call function “em1_emmc_set_blklength()” and “em1_emmc_lock_unlock()” to set password before lock the chip, before send command to EMMC, setting the lock/unlock data structure, and make sure the bit0(SET_PWD) in lock/unlock data structure set to 1. If failed, end the operation and print error information; if OK, continue the test program.

In this test, use “123” as the password contact.

Note:

Details about execution of lock/unlock function please refer to “**(4) Enforce Erase If EMMC Device Locked**” of “**Chapter 3.2.2 Operation Details**”.

(2) Lock EMMC

Call function “em1_emmc_lock_unlock()” to lock the EMMC chip, make sure the bit2(Lock/Unlock) in lock/unlock data structure set to 1. If failed, end the operation and print error information; if OK, continue the test program.

(3) Check EMMC Chip Status

Call “em1_emmc_check_dev_status ()” function to check EMMC status. if card is not locked, means lock function failed, end the operation and print error; otherwise, continue the test program.

Detail steps about check EMMC device status please refer to “**(3) Check EMMC Device Status**” in “**Chapter 3.2.2 Operation Details**”.

(4) Unlock EMMC

Call function “em1_emmc_lock_unlock()” to unlock the EMMC chip, make sure the bit2(Lock/Unlock) in lock/unlock data structure set to 0. If failed, end the operation and print error information; if OK, continue the test program.

(5) Check EMMC Chip Status

Call “em1_emmc_check_dev_status ()” function to check EMMC status. if card is still locked, means unlock function failed, end the operation and print error; otherwise, continue the test program.

(6) Clear Password

Call function “em1_emmc_lock_unlock()” to clear password, make sure the bit1(CLR_PWD) in lock/unlock data structure set to 1, “123” as the password contact. If failed, end the operation and print error information; if OK, continue the test program.

(7) Set block length to be 512 byte

Before end the lock/unlock test, call function “em1_emmc_set_blklength()” to set block size as 512 byte, if command works OK, end the test normally.

Details about block length setting (em1_emmc_set_blklength()) please refer to step13 of "**(2) Init EMMC device**" in "**Chapter 3.2.2 Operation Details**".

Note:

More details about lock/unlock data structure and its operation; please refer to "**Chapter 4.2.11 moviNAND Lock/Unlock Operation**" of EMMC chip user's manual.

More details about the functions used in this example please refer to "**APPENDIX A EMMC Driver Function**"

APPENDIX A EMMC Driver Function

A.1 Function List

The following table shows the EMMC driver interface functions:

Table A-1 EMMC Driver Function List

Class	Function Name	Function Detail
External function	em1_emmc_hw_init	Init the SDM module setting
	em1_emmc_init	EMMC device initialization
	em1_emmc_set_seccnt	Enable/disable sector and set sector number
	em1_emmc_send_cmd	Send comand
	em1_emmc_set_clk	Set SDM output clock value
	em1_emmc_set_blklength	Set block length
	em1_emmc_select_card	Select card
	em1_emmc_set_ext_csd	Set extend CSD register
	em1_emmc_check_dev_status	Check EMMC chip status
	em1_emmc_erase	Erase function
	em1_emmc_single_read	Single block read operation in CPU mode
	em1_emmc_signle_write	Single block write operation in CPU mode
	em1_emmc_multi_read	Multi block read operation in DMA mode
	em1_emmc_multi_write	Multi block write operation in DMA mode
	em1_emmc_lock_unlock	Lock/unlolck operation
	em1_emmc_wp_manage	Write protect management
Internal function	_em1_emmc_tranf_prepare	Prepare before data transfer
	_em1_emmc_decode_csd	Decode CSD structure
	_em1_emmc_check_rsp_status	Check response status

A.2 Global Variable Define

Table A-2 Global Variable Define

Name	Type	Detail
g_RCA_VAL	ushort	Globle flag for relative address
g_password_buff	uchar	Password data structure buffer
g_read_buff[]	uchar	Read buffer
g_write_buff[]	uchar	Write buffer

A.3 Structure Define

Table A-3 Structure Define

Structure Name	Detail
mmc_csd	CSD register sturcture

A.3.1 mmc_csd

Table A-4 Structure of mmc_csd

Member	Detail
uchar mmca_vsn	MMC structure version
ushort cmdclass	Command classes
ushort tacc_clks	Read access time in clocks
uint tacc_ns	Read access time in ns
uint max_dtr	Maximum data transfer speed
uint read_blkbits	Read block bits
uint read_blkbits	Write block bits
uint capacity	Device capacity
uint erase_grp_size	Erase group base size
uint erase_grp_mult	Erase group size multipile factor
uint wp_grp_size	Write protect froup size
uint read_partial	Whether enable read in partial block
uint read_misalign	Whether enable read block cross physical block boundaries
uint write_partial	Whether enable write in partial block
uint write_misalign	Whether enable write block cross physical block boundaries

A.4 Function Details

A.4.1 Hardware Initialization Function

[Function Name]

em1_emmc_hw_init

[Format]

void em1_emmc_hw_init (void);

[Argument]

None

[Function Return]

None

[Flow Chart]

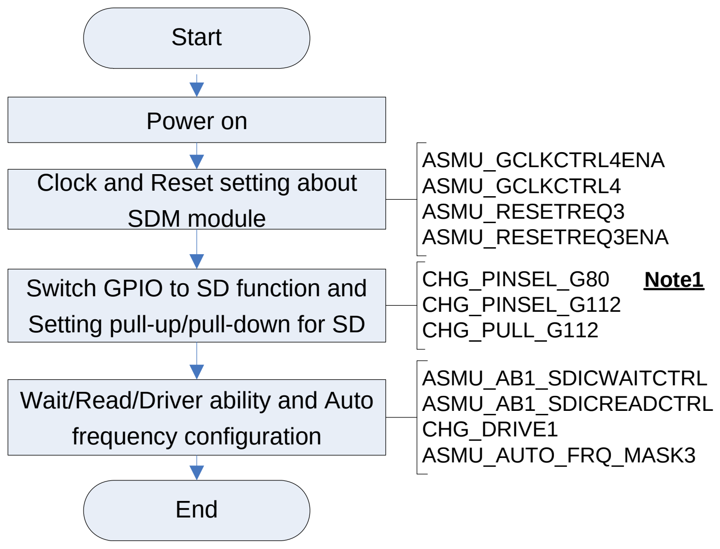

Figure A-1 SDM Hardware Initialization Flow

[Note]

1) Switch GPIO which used for EMMC to SD function, users should operate this step according the actual hardware connection.

A.4.2 EMMC Init Operation

[Function Name]

em1_emmc_init

[Format]

int em1_emmc_init(void) ;

[Argument]

None.

[Function Return]

DRV_OK

Others: error end

[Flow Chart]

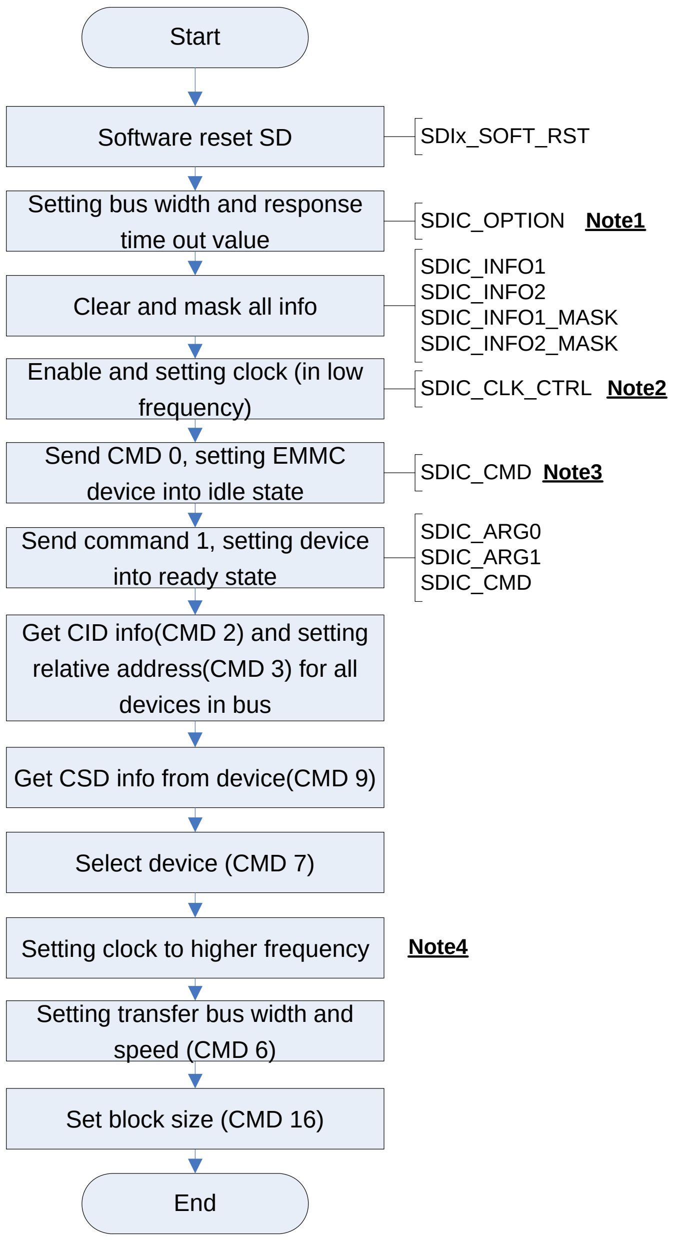
Start
Software reset SD
SDIx_SOFT_RST
Setting bus width and response time out value
SDIC_OPTION Note1
Clear and mask all info
SDIC_INFO1
SDIC_INFO2
SDIC_INFO1_MASK
SDIC_INFO2_MASK
Enable and setting clock (in low frequency)
SDIC_CLK_CTRL Note2
Send CMD 0, setting EMMC device into idle state
SDIC_CMD Note3
Send command 1, setting device into ready state
SDIC_ARG0
SDIC_ARG1
SDIC_CMD
Get CID info(CMD 2) and setting relative address(CMD 3) for all devices in bus
Get CSD info from device(CMD 9)
Select device (CMD 7)
Setting clock to higher frequency
Note4
Setting transfer bus width and speed (CMD 6)
Set block size (CMD 16)
End

Figure A-2 EMMC Chip Initialization Flow

[Note]

1) Before the initialization of EMMC device, set the bus width to be 1bit and the response time out value to be maximum, reason is: in EMMC identification progress, bus width will use 1bit and the clock frequency should in 10-400 KHz.

2) After setting the clock for SDM transfer, please wait 1ms for stability.

3) After send command, user should check the command response, if error occurred (except when response time out for CMD 2), the initialization will be ended abnormally.

4) After initialization, users can configure the bus width and clock frequency for transfer speed according the CSD parameter of EMMC chip, the transfer speed defines in CSD register just clock frequency not in high speed mode, so the actual transfer speed is related to EMMC chip specification version.

A.4.3 Sector Setting

[Function Name]

em1_emmc_set_seccnt

[Format]

void em1_emmc_set_seccnt(BOOL bEnable, uint sec_num);

[Argument]

Parameter	Type	I/O	Detail
bEnable	BOOL	I	Enable/disable sector
sec_num	uint	I	Sector number

[Function Return]

None

[Flow Chart]

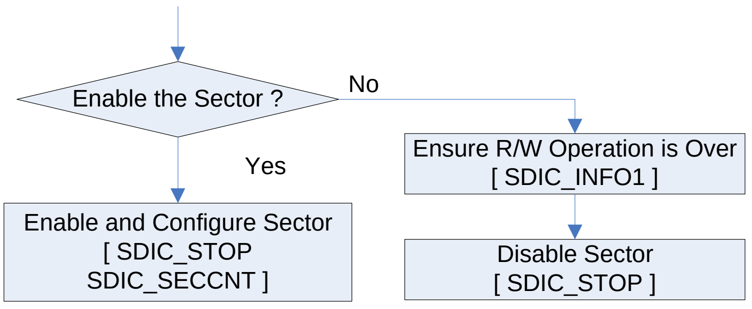

Figure A-3 Sector Setting Flow

[Note]

None

A.4.4 Send Command

[Function Name]

em1_emmc_send_cmd

[Format]

int em1_emmc_send_cmd (int cmd) ;

[Argument]

Parameter	Type	I/O	Detail
cmd	int	I	Command index that need to be send

[Function Return]

DRV_OK

Others: error end

[Flow Chart]

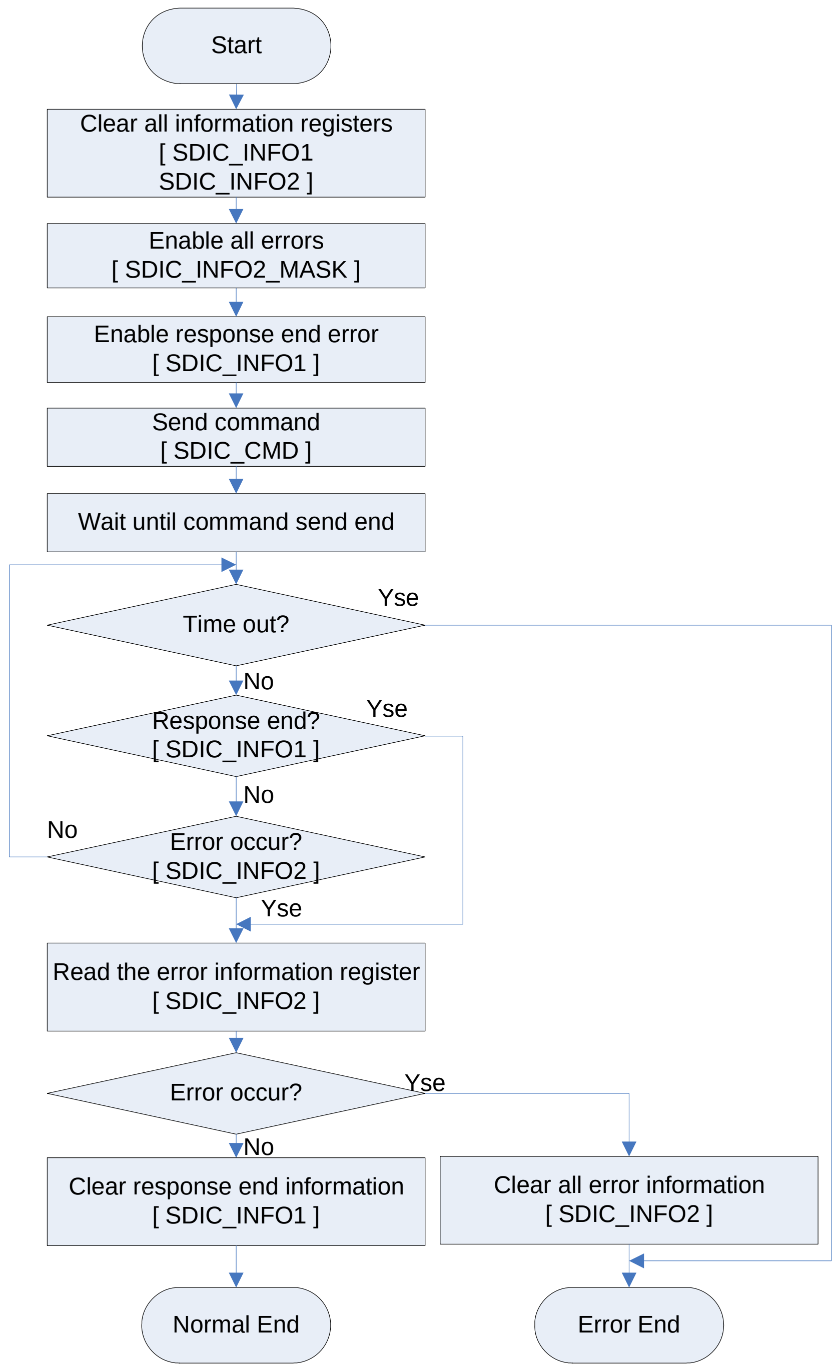
Start
Clear all information registers [SDIC_INFO1 SDIC_INFO2]
Enable all errors [SDIC_INFO2_MASK]
Enable response end error [SDIC_INFO1]
Send command [SDIC_CMD]
Wait until command send end
Time out?
Yse
No
Response end? [SDIC_INFO1]
Yse
No
Error occur? [SDIC_INFO2]
No
Yse
Read the error information register [SDIC_INFO2]
Error occur?
Yse
No
Clear response end information [SDIC_INFO1]
Clear all error information [SDIC_INFO2]
Normal End
Error End

Figure A-4 Send Command

[Note]

Send command to EMMC chip

A.4.5 Set Clock

[Function Name]

em1_emmc_set_clk

[Format]

void em1_emmc_set_clk(ushort value);

[Argument]

Parameter	Type	I/O	Detail
value	ushort	I	Clock setting value

[Function Return]

None

[Flow Chart]

None

[Note]

Set SDIx_CLK_CTRL register.

A.4.6 Set Block Length

[Function Name]

em1_emmc_set_blklength

[Format]

int em1_emmc_set_blklength(ushort length);

[Argument]

Parameter	Type	I/O	Detail
length	ushort	I	Block length setting value

[Function Return]

DRV_OK

Others: error end

[Flow Chart]

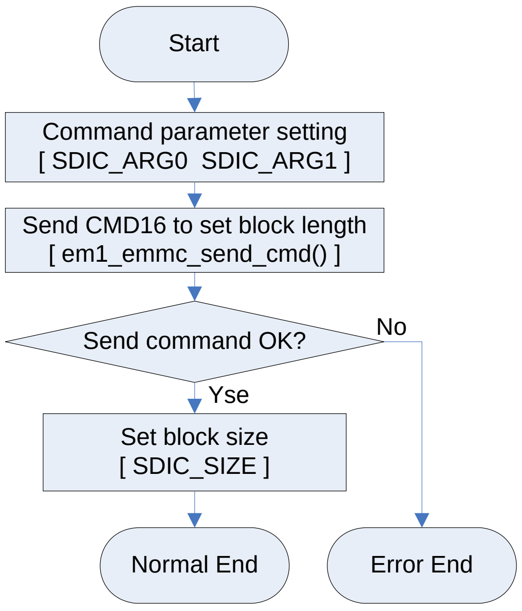

Figure A-5 Set Block Length

[Note]

Data transfer block length setting.

A.4.7 Select Card

[Function Name]

em1_emmc_select_card

[Format]

int em1_emmc_select_card(uint RCA);

[Argument]

Parameter	Type	I/O	Detail
RCA	uint	I	Relative card address that need to be select

[Function Return]

DRV_OK

Others: error end

[Flow Chart]

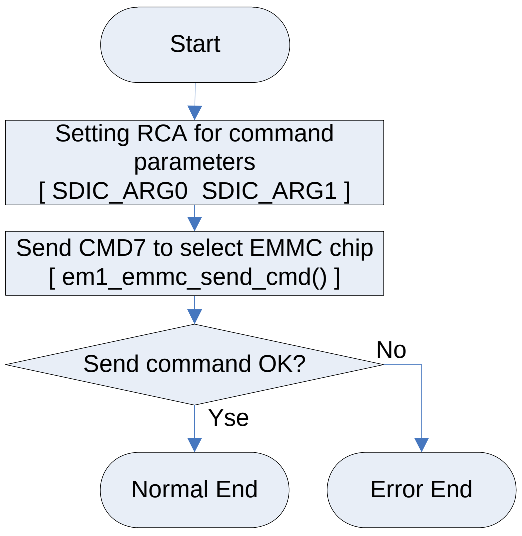
Start
Setting RCA for command parameters [SDIC_ARG0 SDIC_ARG1]
Send CMD7 to select EMMC chip [em1_emmc_send_cmd()]
Send command OK?
No
Yse
Normal End
Error End

Figure A-6 Select Card

[Note]

None.

A.4.8 Configuration Extend CSD

[Function Name]

em1_emmc_set_ext_csd

[Format]

int em1_emmc_set_ext_csd(void) ;

[Argument]

None

[Function Return]

DRV_OK

Others: error end

[Flow Chart]

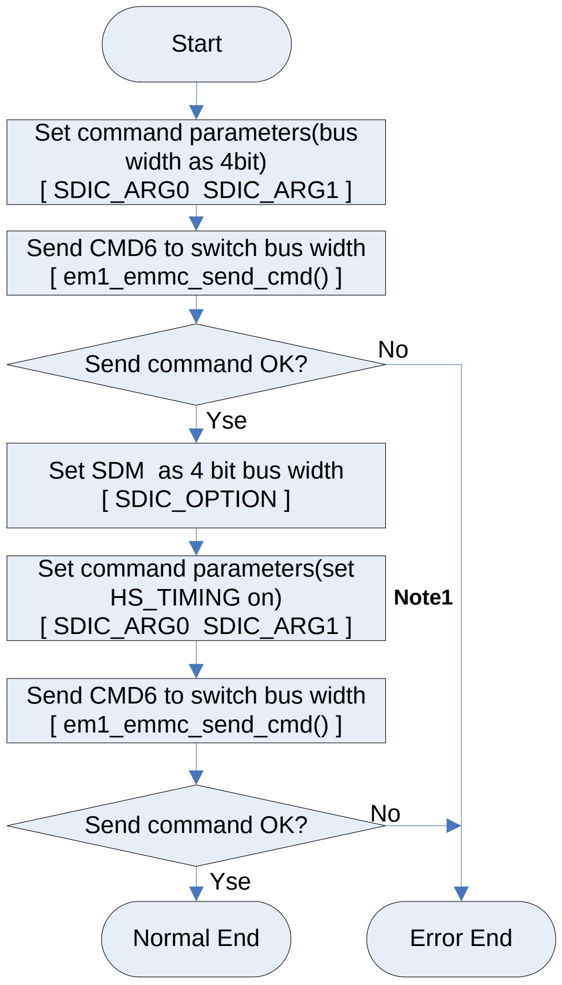
Start
Set command parameters(bus width as 4bit) [SDIC_ARG0 SDIC_ARG1]
Send CMD6 to switch bus width [em1_emmc_send_cmd()]
Send command OK?
No
Yse
Set SDM as 4 bit bus width [SDIC_OPTION]
Set command parameters(set HS_TIMING on) [SDIC_ARG0 SDIC_ARG1]
Note1
Send CMD6 to switch bus width [em1_emmc_send_cmd()]
Send command OK?
No
Yse
Normal End
Error End

Figure A-7 Configure Extend CSD

[Note]

Change bus width and other operation mode.

1) Details about extend CSD information, please refer to “**Chapter 5.5 Extended CSD Register**” of EMMC chip user’s manual

A.4.9 Check Device Status

[Function Name]

em1_emmc_check_dev_status

[Format]

int em1_emmc_check_dev_status (void) ;

[Argument]

None

[Function Return]

DRV_EMMC_LOCKED

DRV_EMMC_UNLOCKED

DRV_ERR_STATE

[Flow Chart]

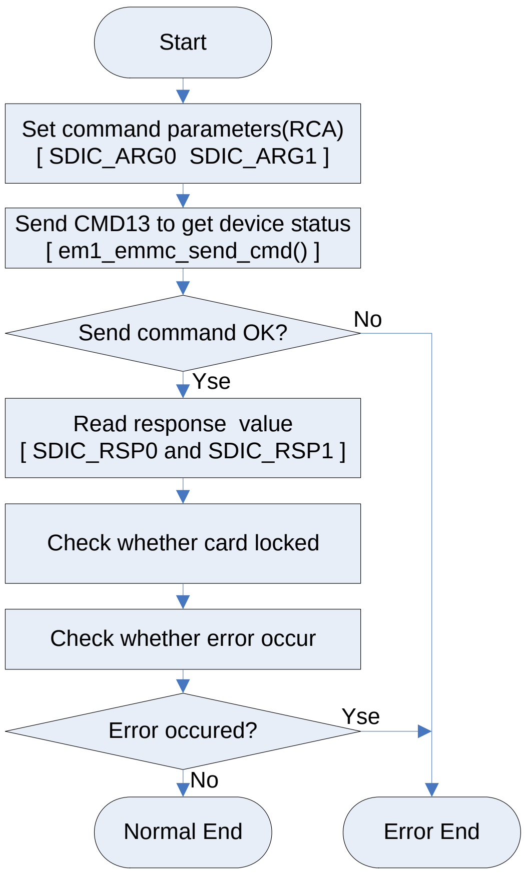
Start
Set command parameters(RCA)
[SDIC_ARG0 SDIC_ARG1]
Send CMD13 to get device status
[em1_emmc_send_cmd()]
Send command OK?
No
Yse
Read response value
[SDIC_RSP0 and SDIC_RSP1]
Check whether card locked
Check whether error occur
Error occured?
Yse
No
Normal End
Error End

Figure A-8 Check EMMC Chip Stauts

[Note]

Check EMMC chip status register.

A.4.10 Erase Function

[Function Name]

em1_emmc_erase

[Format]

int em1_emmc_erase(uint str_addr, uint end_addr) ;

[Argument]

Parameter	Type	I/O	Detail
str_addr	uint	I	Start erase group unit address
end_addr	uint	I	End of erase group unit address

[Function Return]

DRV_OK

Others: error end

[Flow Chart]

Please refer to “figure2-3 EMMA Mobile 1 Erase Operation” in “chapter 2.3 Erase”.

[Note]

None

A.4.11 Single Block Read

[Function Name]

em1_emmc_single_read

[Format]

int em1_emmc_single_read(uint address, uchar *data);

[Argument]

Parameter	Type	I/O	Detail
address	uint	I	block address to be read
data	uchar *	I/O	read out data buffer

[Function Return]

DRV_OK

Others: error end

[Flow Chart]

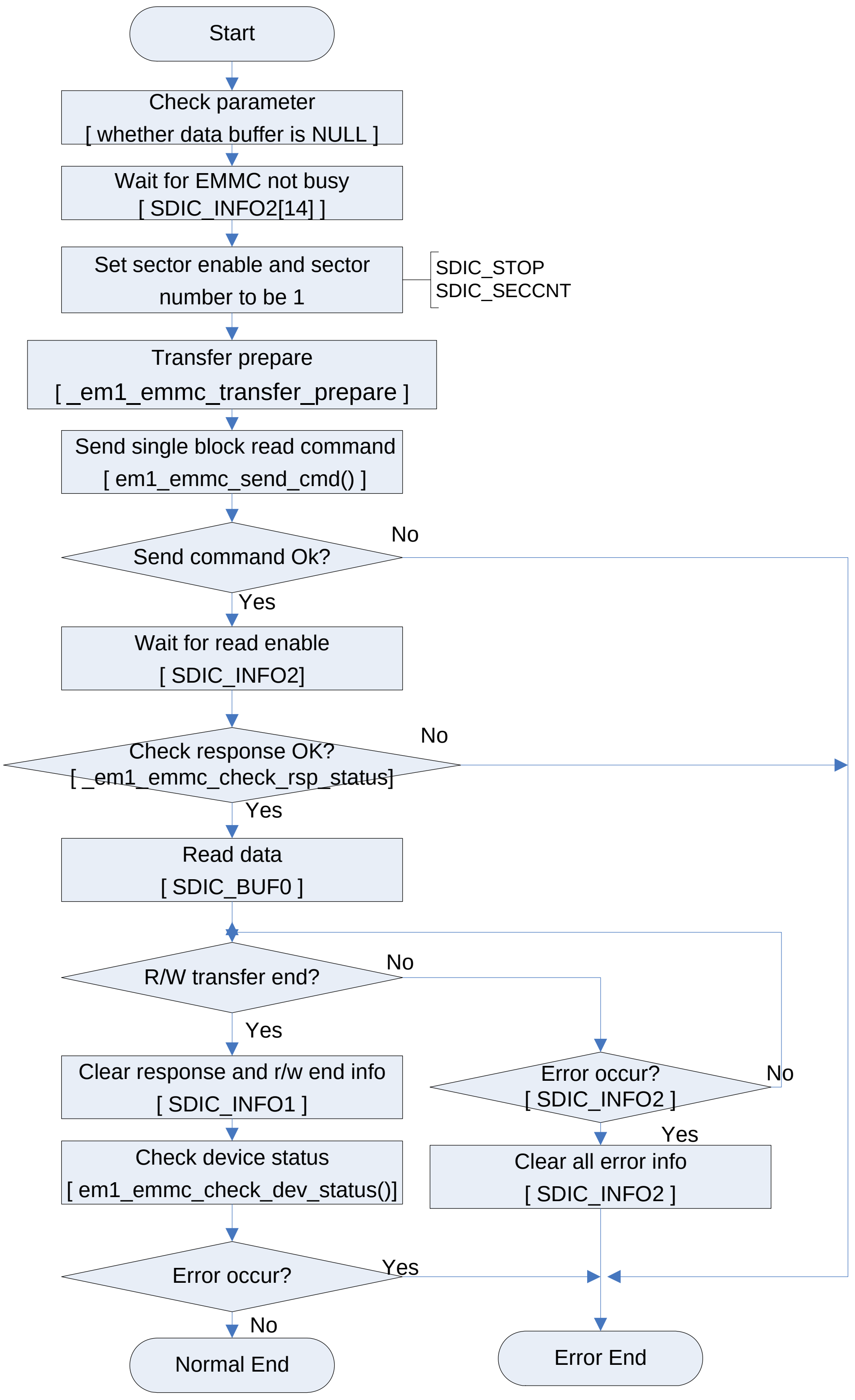

Figure A-9 Single Block Read in CPU mode

[Note]

None

A.4.12 Single Block Write

[Function Name]

em1_emmc_single_write

[Format]

int em1_emmc_single_write(uint address, uchar *data);

[Argument]

Parameter	Type	I/O	Detail
address	uint	I	block address to be write
data	uchar *	I/O	Write source data buffer

[Function Return]

DRV_OK

Others: error end

[Flow Chart]

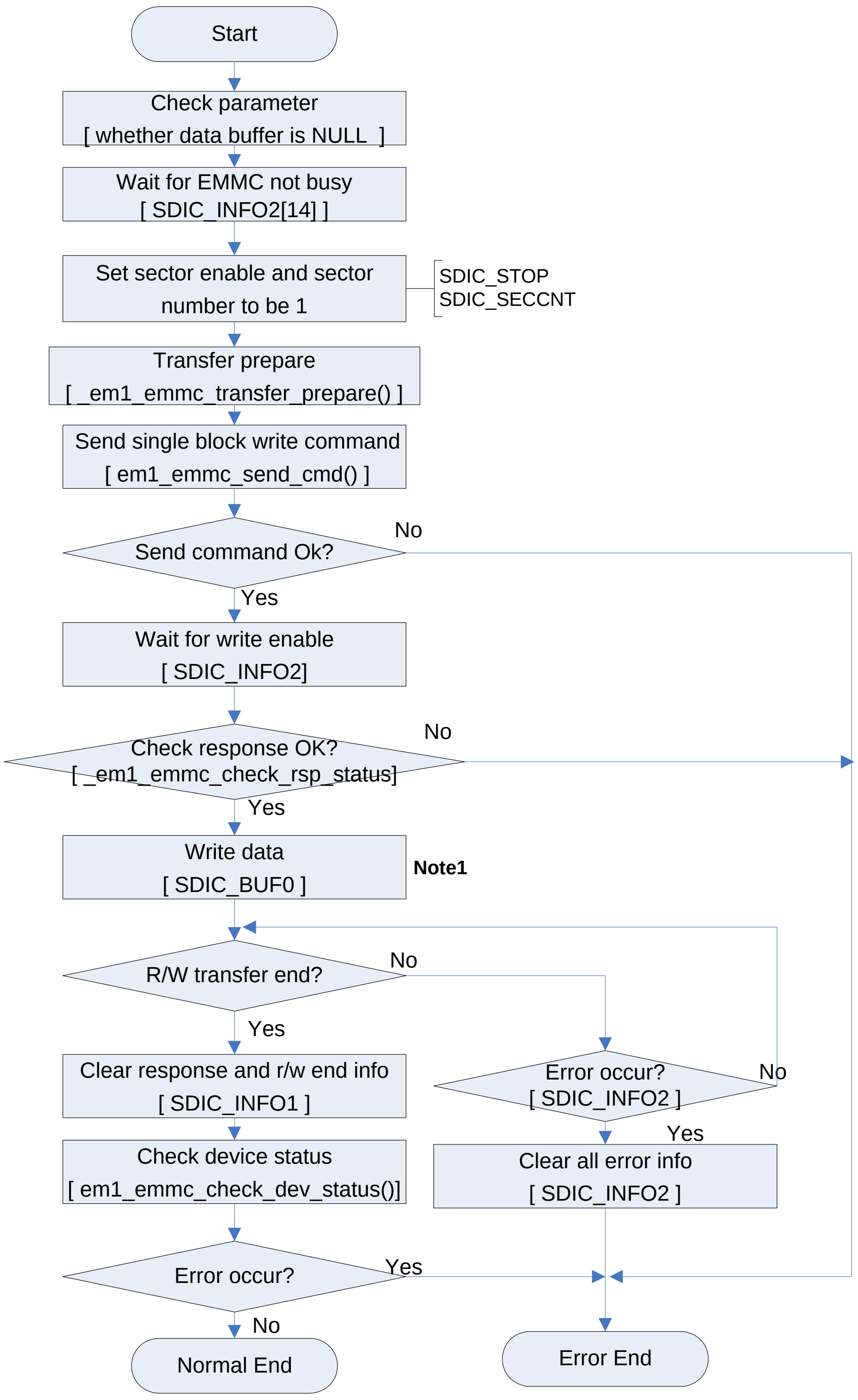

Figure A-10 Single Block Write in CPU mode

[Note]

1) During write operation, check SDIC_INFO2, if buffer write access error occur, end the write operation and return with error information.

A.4.13 Multiple Block Read

[Function Name]

em1_emmc_multi_read

[Format]

int em1_emmc_multi_read (uint address, uchar *read_buf, uint blk_num);

[Argument]

Parameter	Type	I/O	Detail
address	uint	I	block address to be read
read_buf	uchar *	I/O	read out data buffer
blk_num	uint	I	block number to be read

[Function Return]

DRV_OK

Others: error end

[Flow Chart]

Please refer to “figure2-2 EMMA Mobile 1 EMMC Data Transfer in DMA Mode” in “chapter 2.2 Data Transfer”.

[Note]

None.

A.4.14 Multiple Block Write

[Function Name]

em1_emmc_multi_write

[Format]

int em1_emmc_multi_write (uint address, uchar * write_buf, uint blk_num);

[Argument]

Parameter	Type	I/O	Detail
address	uint	I	block address to be write
write_buf	uchar *	I/O	write source data buffer
blk_num	uint	I	block number to be write

[Function Return]

DRV_OK

Others: error end

[Flow Chart]

Please refer to “figure2-2 EMMA Mobile 1 EMMC Data Transfer in DMA Mode” in “chapter 2.2 Data Transfer”.

[Note]

None.

A.4.15 Lock/Unlock Function

[Function Name]

em1_emmc_lock_unlock

[Format]

int em1_emmc_lock_unlock(uchar *data);

[Argument]

Parameter	Type	I/O	Detail
data	uchar*	I	Lock data structure buffer

[Function Return]

DRV_OK

Others: error end

[Flow Chart]

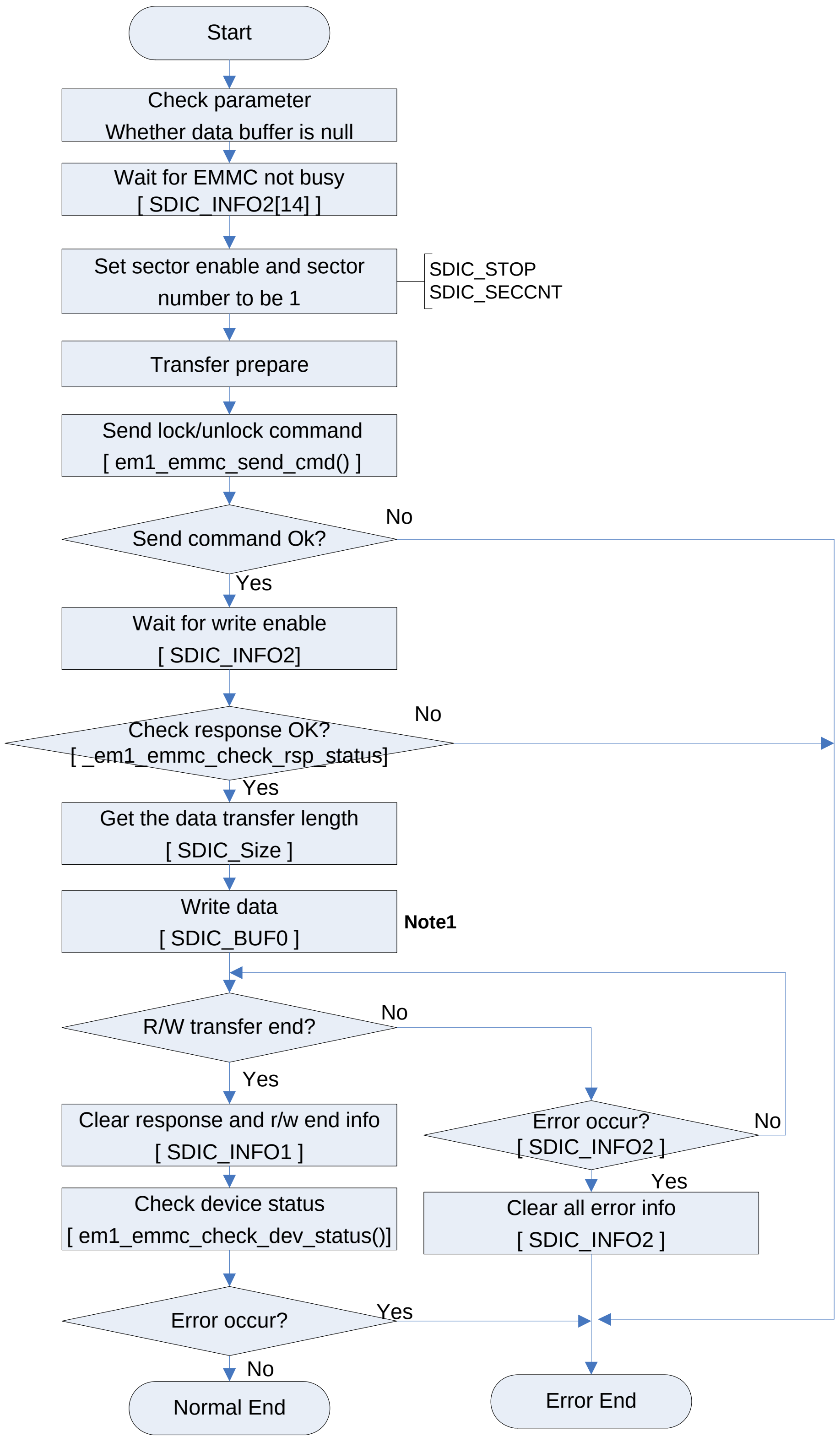

Figure A-11 Lock/Unlock Progress

[Note]

1) During write operation, check SDIC_INFO2, if buffer write access error occur, end the write operation and return with error information.

A.4.16 Write Protect Manage

[Function Name]

em1_emmc_wp_manage

[Format]

int em1_emmc_wp_manage(int cmd, uint addr) ;

[Argument]

Parameter	Type	I/O	Detail
cmd	int	I	Write protect command
addr	uint	I	Address to be set write protect

[Function Return]

DRV_OK

Others: error end

[Flow Chart]

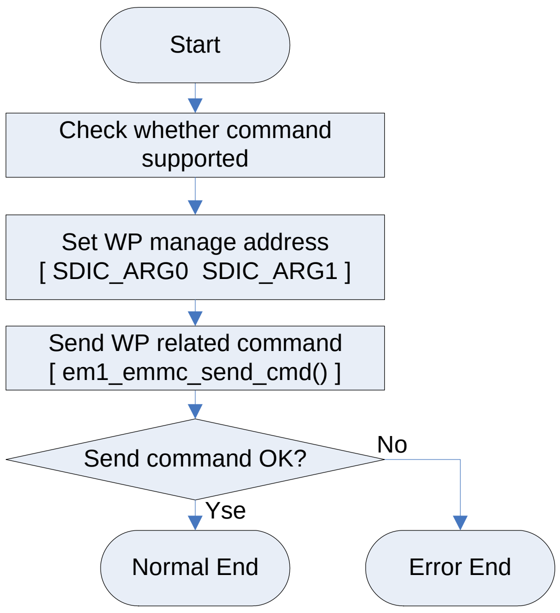
Start
Check whether command supported
Set WP manage address [SDIC_ARG0 SDIC_ARG1]
Send WP related command [em1_emmc_send_cmd()]
Send command OK?
No
Yse
Normal End
Error End

Figure A-12 Write Protect Management

[Note]

None.

A.4.17 Transfer Prepare

[Function Name]

_em1_emmc_transfer_prepare

[Format]

void _em1_emmc_transfer_prepare(void);

[Argument]

None

[Function Return]

None

[Flow Chart]

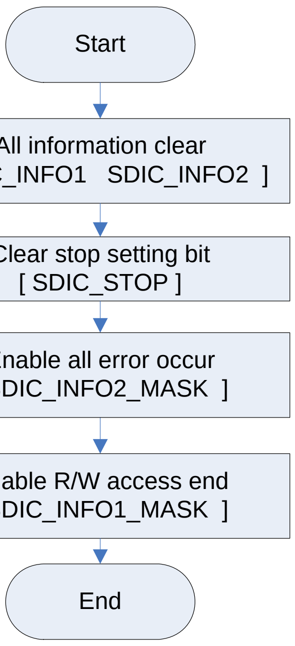

Figure A-13 Register Prepare before Data transfer

[Note]

None.

A.4.18 Decode CSD

[Function Name]

_em1_emmc_decode_csd

[Format]

void _em1_emmc_decode_csd(uint *raw_csd);

[Argument]

Parameter	Type	I/O	Detail
raw_csd	uint	I	Raw CSD value buffer

[Function Return]

None

[Flow Chart]

None

[Note]

This function will decode the response of SEND_CSD command; get CSD members that useful for data transfer.

A.4.19 Check Response Status

[Function Name]

_em1_emmc_check_rsp_status

[Format]

int _em1_emmc_check_rsp_status(void);

[Argument]

None

[Function Return]

DRV_OK

DRV_ERR_STATE

[Flow Chart]

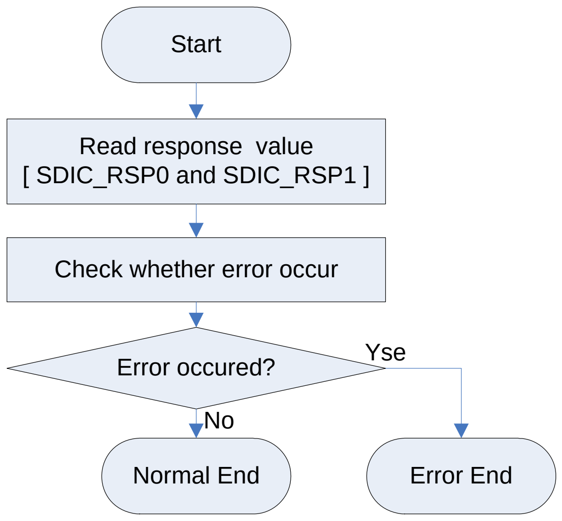

Figure A-14 Check Response Status

[Note]

None

APPENDIX B COMMANDS

Following table shows the simple function of the EMMC chip command which used in this document.
More details about the commands format and function please refer to "**Chapter4.6 Commands**" of the EMMC chip user's manual.

Table B-1 Command Description List

CMD INDEX	Argument	Abbreviation	Command Description
CMD0	[31:0] stuff bits	GO_IDLE_STATE	Resets the EMMC chip to idle state
CMD1	[31:0] OCR without busy	SEND_OP_COND	Ask chip send its Operating Conditions Register contents.
CMD2	[31:0] stuff bits	ALL_SEND_CID	Asks the EMMC chip to send its CID number on the CMD line
CMD3	[31:16] RCA [15:0] stuff bits	SET_RELATIVE_ADDR	Assigns relative address to the EMMC chip
CMD4	Not Supported		
CMD5	Reserved		
CMD6	[31:26] Set to 0 [25:24] Access [23:16] Index [15:8] Value [7:3] Set to 0 [2:0] Cmd Set	SWITCH	Switches operation mode of the selected EMMC chip or modifies the EXT_CSD registers.
CMD7	[31:16] RCA [15:0] stuff bits	SELECT/DESELECT_CARD	Select device by its own relative address and gets deselected by any other address; address 0 deselects the EMMC chip.
CMD8	[31:0] stuff bits	SEND_EXT_CSD	The EMMC chip sends its EXT_CSD register as a block of data.
CMD9	[31:16] RCA [15:0] stuff bits	SEND_CSD	Addressed EMMC chip sends its Card-Specific Data (CSD) on the CMD line.
CMD10	[31:16] RCA [15:0] stuff bits	SEND_CID	Addressed EMMC chip sends its Card Identification (CID) on CMD the line.
CMD11	Not Supported		
CMD12	[31:0] stuff bits	STOP_TRANSMISSION	Forces the EMMC chip to stop transmission
CMD13	[31:16] RCA [15:0] stuff bits	SEND_STATUS	Addressed EMMC chip sends its status register.
CMD14	[31:0] stuff bits	BUSTEST_R	A host reads the reversed bus testing data pattern from a EMMC chip.
CMD15	[31:16] RCA [15:0] stuff bits	GO_INACTIVE_STATE	Sets the EMMC chip to inactive state
CMD16	[31:0] block	SET_BLOCKLEN	Sets the block length (in bytes) for all

	length		following block commands (read and write).
CMD17	[31:0] data address	READ_SINGLE_BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command
CMD18	[31:0] data address	READ_MULTIPLE_BLOCK	Multipile block read command
CMD19	[31:0] stuff bits	BUSTEST_W	A host sends the bus test data pattern to a EMMC chip
CMD20	Not supported		
CMD21 CMD22	Reserved		
CMD23	[31:16] set to 0 [15:0] number of blocks	SET_BLOCK_COUNT	Defines the number of blocks which are going to be transferred in the immediately succeeding multiple block read or write command.
CMD24	[31:0] data address	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.
CMD25	[31:0] data address	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows or the requested number of block received
CMD26	Not applicable		
CMD27	[31:0] stuff bits	PROGRAM_CSD	Programming of the programmable bits of the CSD
CMD28	[31:0] data address	SET_WRITE_PROT	Sets the write protection bit of the addressed group.
CMD29	[31:0] data address	CLR_WRITE_PROT	Clears the write protection bit of the addressed group
CMD30	[31:0] write protect data address	SEND_WRITE_PROT	Asks the EMMC chip to send the status of the write protection bits.
CMD31 … CMD34	Reserved		
CMD35	[31:0] data address	ERASE_GROUP_START	Sets the address of the first erase group within a range to be selected for erase
CMD36	[31:0] data address	ERASE_GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase
CMD37	Reserved		
CMD38	[31:0] stuff bits	ERASE	Erases all previously selected write blocks
CMD39	[31:16] RCA [15:15]register write Flag [14:8] register address [7:0] register data	FAST_IO	Used to write and read 8 bit (register) data fields.
CMD40	[31:0] stuff bits	GO_IRQ_STATE	Sets the system into interrupt mode
CMD41	Reserved		

CMD42	[31:0] stuff bits.	LOCK_UNLOCK	Used to set/reset the password or lock/unlock the EMMC chip. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43 … CMD54	Reserved		
CMD55	MMCA Optional Command, currently not supported.		
CMD56 … CMD59	Reserved		

ANNEX Modification History

Number	Modification Contents	Author	Date
Ver 1.00	New version		Aug,4.2009