

EC2x&AG35-QuecOpen

EMMC&SD Card

Adaptation Guide

LTE Standard/Automotive Module Series

Rev. EC2x&AG35-QuecOpen_EMMC&SD_Card_Adaptation_Guide_V1.2

Date: 2019-03-04

Status: Preliminary

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.

7th Floor, Hongye Building, No.1801 Hongmei Road, Xuhui District, Shanghai 200233, China

Tel: +86 21 5108 6236

Email: info@quectel.com

Or our local office. For more information, please visit:

<http://www.quectel.com/support/sales.htm>

For technical support, or to report documentation errors, please visit:

<http://www.quectel.com/support/technical.htm>

Or email to: support@quectel.com

GENERAL NOTES

QUECTEL OFFERS THE INFORMATION AS A SERVICE TO ITS CUSTOMERS. THE INFORMATION PROVIDED IS BASED UPON CUSTOMERS' REQUIREMENTS. QUECTEL MAKES EVERY EFFORT TO ENSURE THE QUALITY OF THE INFORMATION IT MAKES AVAILABLE. QUECTEL DOES NOT MAKE ANY WARRANTY AS TO THE INFORMATION CONTAINED HEREIN, AND DOES NOT ACCEPT ANY LIABILITY FOR ANY INJURY, LOSS OR DAMAGE OF ANY KIND INCURRED BY USE OF OR RELIANCE UPON THE INFORMATION. ALL INFORMATION SUPPLIED HEREIN IS SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.

COPYRIGHT

THE INFORMATION CONTAINED HERE IS PROPRIETARY TECHNICAL INFORMATION OF QUECTEL WIRELESS SOLUTIONS CO., LTD. TRANSMITTING, REPRODUCTION, DISSEMINATION AND EDITING OF THIS DOCUMENT AS WELL AS UTILIZATION OF THE CONTENT ARE FORBIDDEN WITHOUT PERMISSION. OFFENDERS WILL BE HELD LIABLE FOR PAYMENT OF DAMAGES. ALL RIGHTS ARE RESERVED IN THE EVENT OF A PATENT GRANT OR REGISTRATION OF A UTILITY MODEL OR DESIGN.

Copyright © Quectel Wireless Solutions Co., Ltd. 2019. All rights reserved.

About the Document

History

Revision	Date	Author	Description
1.0	2018-03-21	Quinn LV	Initial
1.1	2018-09-15	Larry ZHANG	Updated tool parameters for making ext4 file system
1.2	2019-03-04	Larry ZHANG	Added EMMC power supply notes

Contents

About the Document.....	2
Contents.....	3
Table Index.....	5
Figure Index.....	6
1 Introduction	7
2 Hardware Interface and Pin Definition of EMMC&SD Card	8
2.1. Usage Definition of EC2x Connecting Externally SD Card/EMMC Pin.....	8
2.2. Usage Definition of AG35 Connecting Externally SD Card/EMMC Pin.....	8
3 Recommendation of Hardware Circuit Design	10
3.1. EC2x&AG35 Modules Connection Externally SD Card Hardware Circuit Reference Design .	10
3.2. EC2x&AG35 Modules Connection Externally EMMC Hardware Circuit Reference Design	10
4 EMMC Device Tree Configuration and Kernel Driver Adaptation.....	12
4.1. Modifying Device Tree File to Add Device Information	12
4.2. Modifying Kernel Configuration to Support SD Controller and EMMC Driver	14
4.3. Modifying Kernel Configuration to Support Ext4 File System	14
4.4. Modifying Kernel Driver to Avoid Switching Signal Voltage to 2.85v	14
4.5. Compilation Test.....	14
5 SD Card Device Tree Configuration and Kernel Driver Adaptation	16
5.1. Modifying Device Tree File to Add Device Information	16
5.2. Modifying the Kernel Configuration to Support SD Controller and SD Card Driver	17
5.3. Modifying the Kernel Configuration to Support FAT File System	17
5.4. Compilation Test.....	17
6 EMMC File System Production and Functional Test Verification.....	18
6.1. Building of LTE OPEN EVB Hardware Test Environment.....	18
6.2. Checking If the EMMC Device is Correctly Identified.....	18
6.3. Partition Management.....	18
6.3.1. Checking Existing EMMC Partition Information.....	19
6.3.2. Existing Partition Deletion	19
6.3.3. Creation of New Partition	19
6.3.4. Writing into Newly Created Partition Information	20
6.4. Importing Ext4 File System Making Tool.....	20
6.5. Formatting Partition to the Ext4 File System	20
6.6. Read and Write Test.....	21
7 SD Card File System Making and Functional Test Verification	23
7.1. Building of LTE OPEN EVB Hardware Test Environment.....	23
7.2. Checking Whether SD Card is Correctly Identified.....	23
7.3. Partition Management.....	23
7.3.1. Checking the Existing SD Card Partition Information	24
7.3.2. Existing Partition Deletion	24
7.3.3. Creation of New Partition	24
7.3.4. Partition Information Updating	25

7.4.	Formatting Partition to FAT File System	25
7.5.	Test on Read and Write.....	26
8	Common Debugging Method for SD Driver	27
8.1.	Simple Read and Write Test	27
8.1.1.	Data Writing.....	27
8.1.2.	Data Reading	27
8.2.	Viewing the File Systems Supported by the Current System	27
8.3.	Getting Basic Information of SD Card/EMMC	28
8.4.	Brief Introduction of Linux MMC Driver Subsystem.....	28
8.4.1.	Introduction of Linux MMC Subsystem	28
8.4.2.	Linux MMC Subsystem Driver Initialization Process	29
8.5.	Open Debug Information.....	30
8.5.1.	Open Debug Macro of the MMC Subsystem	30
8.5.2.	Dmesg Output Debug Log	31
9	Appendix A Reference Documents and Terminology Abbreviations.....	32

Table Index

TABLE 1: REFERENCE DOCUMENTS.....	32
TABLE 2: TERMINOLOGY ABBREVIATIONS	32

Quectel
Confidential

Figure Index

FIGURE 1: SD CARD HARDWARE CIRCUIT REFERENCE DESIGN	10
FIGURE 2: EMMC HARDWARE CIRCUIT REFERENCE DESIGN.....	11
FIGURE 3: LINUX MMC SUBSYSTEM ARCHITECTURE DIAGRAM	29
FIGURE 4: LINUX MMC DRIVER INITIALIZATION FLOW CHART.....	30

Quectel
Confidential

1 Introduction

Secure Digital is abbreviated as SD, its full name is Secure Digital Memory Card. A memory card that is widely used in portable devices such as digital cameras, personal digital assistants, and multimedia players.

EMMC is short for Embedded MultiMedia Card. MMC is short for MultiMedia Card which is a Flash Memory Card standard that defined the architecture of the MMC and the interfaces and protocols for accessing Flash Memory. EMMC is an extension of MMC to meet the higher standard requirements such as performance, cost, size, stability, and use convenience and so on.

Linux Kernel uniformly manages EMMC, SD card, SDIO WIFI and other devices by MMC subsystem. EMMC emphasizes multimedia storage, SD emphasizes security and data protection, SDIO was evolved from SD which emphasizes interface (Input/Output) and it no longer pays attention to the specific form of the other end (it can be SDIO WIFI card, etc.)

In order to meet the diversified needs of customers, both EC2x and AG35 reserve one SDIO interface SDHC2 which is used to connect SD card or EMMC. The SDIO interface is compliant with the SD 3.0 protocol.

This document mainly describes circuit design, software drivers, test verification, etc. it can help customers to conduct development work easily and quickly.

2 Hardware Interface and Pin Definition of EMMC&SD Card

Both AG35-QuecOpen and EC2x-QuecOpen provide two SDIO interfaces that support SD 3.0 protocol. SDIO1 is used to connect externally WiFi devices. SDIO2 is used to connect externally SD card/EMMC devices. The pin of SDIO2 is defined as follows:

2.1. Usage Definition of EC2x Connecting Externally SD Card/EMMC Pin

Pin Name	Pin No.	I/O	Function		Comment
			Alternate Function 1 (Default)	Alternate Function 2	
SDC2_DATA3	28	IO	SDC2_DATA3		
SDC2_DATA2	29	IO	SDC2_DATA2		
SDC2_DATA1	30	IO	SDC2_DATA1		
SDC2_DATA0	31	IO	SDC2_DATA0		
SDC2_CLK	32	DO	SDC2_CLK		
SDC2_CMD	33	IO	SDC2_CMD		
VDD_SDIO	34	PO	VDD_SDIO		Input 2.85V/1.8V Configurable it can not be used to supply power for SD card
SD_INS_DET	23	DI	SD_INS_DET	GPIO_26	

For more information, please refer to the document *Quectel_AG35-QuecOpen_Hardware_Design.pdf*.

2.2. Usage Definition of AG35 Connecting Externally SD Card/EMMC Pin

Pin Name	Pin No.	I/O	Function		Comment
			Alternate Function 1 (Default)	Alternate Function 2	
VDD_SDIO	46	PO	VDD_SDIO		1.8V/2.85V configurable output. SDIO pull up power source for SD card. Keep it open for eMMC.
SDC2_DATA2	47	IO	SDC2_DATA2		SDIO signal level can be selected according to the one supported by SD card. 1.8V power domain for eMMC. Please refer to SD 3.0 protocol for more details.
SDC2_DATA3	48	IO	SDC2_DATA3		
SDC2_DATA0	49	IO	SDC2_DATA0		
SDC2_DATA1	50	IO	SDC2_DATA1		
SDC2_CMD	51	IO	SDC2_CMD		DI: Insertion detection for SD card. DO: Reset eMMC ¹⁾ .
SDC2_CLK	53	DO	SDC2_CLK		
SD_INS_DET	52	IO	SD_INS_DET	GPIO_26	

For more information, please refer to the document *Quectel_AG35-QuecOpen_Hardware_Design.pdf*.

3 Recommendation of Hardware Circuit Design

3.1. EC2x&AG35 Modules Connection Externally SD Card Hardware

Circuit Reference Design

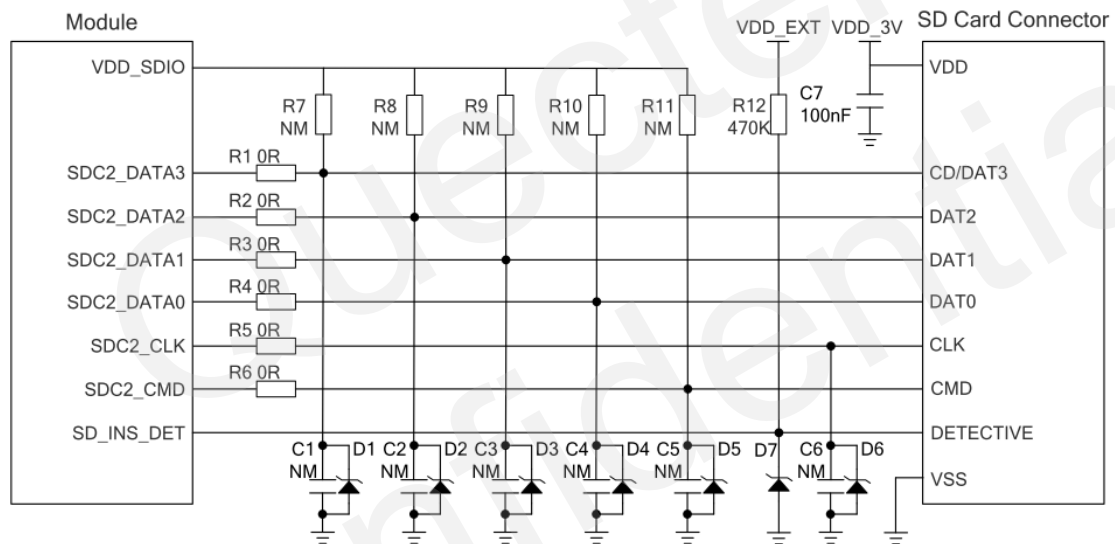


Figure 1: SD Card Hardware Circuit Reference Design

3.2. EC2x&AG35 Modules Connection Externally EMMC Hardware Circuit

Reference Design

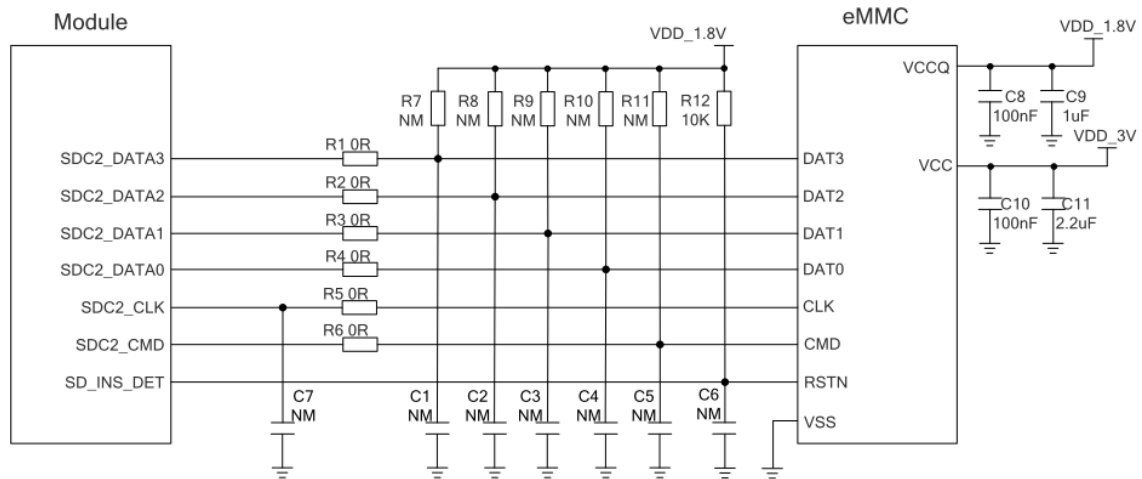


Figure 2: EMMC Hardware Circuit Reference Design

For more information, please refer to the document *Quectel_EC20_R2.0 QuecOpen_Hardware_Design.pdf* and *Quectel_AG35-QuecOpen_Hardware_Design.pdf*.

NOTE

VCC and VCCQ should keep constant power supply and even if the module is in a sleep status, it can not also be powered off.

4 EMMC Device Tree Configuration and Kernel Driver Adaptation

4.1. Modifying Device Tree File to Add Device Information

Checking whether configuration information of *sdhc_2* of *mdm9607-mtp.dtsi* under the path of *ql-ol-sdk/ql-ol-kernel/arch/arm/boot/dts/qcom/* is shown as follow. If there are any differences, please modify it as shown below:

```
&sdhc_2 {
    /*vdd-supply = <&sdcard_ext_vreg>;*/
    qcom,vdd-voltage-level = <2850000 2850000>;
    qcom,vdd-current-level = <15000 400000>;

    vdd-io-supply = <&mdm9607_l13>;
    qcom,vdd-io-always-on;
    qcom,vdd-io-voltage-level = <1800000 1800000>;
    qcom,vdd-io-current-level = <200 50000>;

    #address-cells = <0>;
    interrupt-parent = <&sdhc_2>;
    interrupts = <0 1 2>;
    #interrupt-cells = <1>;
    interrupt-map-mask = <0xffffffff>;
    interrupt-map = <0 &intc 0 125 0
                    1 &intc 0 221 0
                    2 &tlmm_pinmux 26 0>;
    interrupt-names = "hc_irq", "pwr_irq", "status_irq";
    qcom,nonhotplug;
    pinctrl-names = "active", "sleep";
    pinctrl-0 = <&sdhc2_clk_on &sdhc2_cmd_on &sdhc2_data_on &sdhc2_cd_on>;
    pinctrl-1 = <&sdhc2_clk_off &sdhc2_cmd_off &sdhc2_data_off>;

    status = "ok";
};
```

Checking whether sdhc_2 configuration information *mdm9607.dtsi* under the path of *ql-ol-sdk/ql-ol-kernel/arch/arm/boot/dts/qcom/* is shown as follows. If there are any differences, please modify the following information:

```
sdhc_2: sdhci@07864900 {
    compatible = "qcom,sdhci-msm";
    reg = <0x07864900 0x200>, <0x07864000 0x800>;
    reg-names = "hc_mem", "core_mem";

    interrupts = <0 125 0>, <0 221 0>;
    interrupt-names = "hc_irq", "pwr_irq";

    qcom,bus-width = <4>;

    qcom,devfreq,freq-table = <25000000 50000000>;

    qcom,msm-bus,name = "sdhc2";
    qcom,msm-bus,num-cases = <8>;
    qcom,msm-bus,num-paths = <1>;
    qcom,msm-bus,vectors-KBps = <81 512 0 0>, /* No vote */
        <81 512 1600 3200>, /* 400 KB/s */
        <81 512 80000 160000>, /* 20 MB/s */
        <81 512 100000 200000>, /* 25 MB/s */
        <81 512 200000 400000>, /* 50 MB/s */
        <81 512 400000 800000>, /* 100 MB/s */
        <81 512 800000 800000>, /* 200 MB/s */
        <81 512 2048000 4096000>; /* Max. bandwidth */
    qcom,bus-bw-vectors-bps = <0 400000 20000000 25000000 50000000
        100000000 200000000 4294967295>;

    clocks = <&clock_gcc clk_gcc_sdcc2_ahb_clk>,
        <&clock_gcc clk_gcc_sdcc2_apps_clk>;
    clock-names = "iface_clk", "core_clk";

    qcom,clk-rates = <400000 25000000 50000000>;

    qcom,pm-qos-irq-type = "affine_irq";
    qcom,pm-qos-irq-latency = <2 250>;

    status = "disabled";
};
```

For more information on device tree parameters, please refer to *sdhci-msm.txt* of device tree description text

under the path of *ql-ol-sdk/ql-ol-kernel/Documentation/devicetree/bindings/mmc*.

4.2. Modifying Kernel Configuration to Support SD Controller and EMMC

Driver

Checking whether the following configuration items have been added to *mdm9607_defconfig* and *mdm9607-perf_defconfig* under the path of *ql-ol-sdk/ql-ol-kernel/arch/arm/configs*:

```
CONFIG_MMC=y
CONFIG_MMC_PERF_PROFILING=y
CONFIG_MMC_CLKGATE=y
CONFIG_MMC_PARANOID_SD_INIT=y
CONFIG_MMC_BLOCK_MINORS=32
CONFIG_MMC_SDHCI=y
CONFIG_MMC_SDHCI_PLTFM=y
CONFIG_MMC_SDHCI_MSM=y
```

4.3. Modifying Kernel Configuration to Support Ext4 File System

Checking whether the following configuration items have been added to *mdm9607_defconfig* and *mdm9607-perf_defconfig* under the path of *ql-ol-sdk/ql-ol-kernel/arch/arm/configs*:

```
CONFIG_EXT4_FS=y
```

4.4. Modifying Kernel Driver to Avoid Switching Signal Voltage to 2.85v

Modifying *QUECTEL_DISABLE_SWITCH_SIGNAL_VOLTAGE_330* to 1 of macro definition in the source file of *core.c* under the path of *ql-ol-sdk/ql-ol-kernel/drivers/mmc/core/* and modify it as follows:

```
#define QUECTEL_DISABLE_SWITCH_SIGNAL_VOLTAGE_330 1
```

4.5. Compilation Test

If user modified *mdm9607_defconfig* and *mdm9607-perf_defconfig* under the path of *ql-ol-sdk/ql-ol-ke*

ernel/arch/arm/configs/ and these files have already been compiled, please make sure that the build folder was deleted under the path of *ql-ol-sdk/ql-ol-kernel/* and recompile the kernel.

Please refer to the document *Quectel_EC2x&AG35-QuecOpen_Quick Start* to compile and download the Kernel. Please refer to **Chapter 6** of this document to conduct read and write test on EMMC partition and formatting.

Quectel
Confidential

5 SD Card Device Tree Configuration and Kernel Driver Adaptation

5.1. Modifying Device Tree File to Add Device Information

Checking mdm9607-mtp.dtsi under the path of *ql-ol-sdk/ql-ol-kernel/arch/arm/boot/dts/qcom* to confirm whether the following information were added:

```
&sdhc_2 {
    /*vdd-supply = <&sdcard_ext_vreg>;*/
    qcom,vdd-voltage-level = <2850000 2850000>;
    qcom,vdd-current-level = <15000 400000>;

    vdd-io-supply = <&mdm9607_l13>;
    qcom,vdd-io-always-on;
    qcom,vdd-io-voltage-level = <1800000 2850000>;
    qcom,vdd-io-current-level = <200 50000>;

    #address-cells = <0>;
    interrupt-parent = <&sdhc_2>;
    interrupts = <0 1 2>;
    #interrupt-cells = <1>;
    interrupt-map-mask = <0xffffffff>;
    interrupt-map = <0 &intc 0 125 0
                    1 &intc 0 221 0
                    2 &tlmm_pinmux 26 0>;
    interrupt-names = "hc_irq", "pwr_irq", "status_irq";
    cd-gpios = <&tlmm_pinmux 26 0x1>;
    qcom,nonhotplug;
    pinctrl-names = "active", "sleep";
    pinctrl-0 = <&sdhc2_clk_on &sdhc2_cmd_on &sdhc2_data_on &sdhc2_cd_on>;
    pinctrl-1 = <&sdhc2_clk_off &sdhc2_cmd_off &sdhc2_data_off>;

    status = "ok";
};
```

For more information on device tree parameters, please refer to `sdhci-msm.txt` under the path `ql-ol-sdk/ql-ol-kernel/Documentation/devicetree/bindings/mmc`.

5.2. Modifying the Kernel Configuration to Support SD Controller and SD Card Driver

Checking whether the following configuration items were added to `mdm9607_defconfig` and `mdm9607-perf_defconfig` under the path of `ql-ol-sdk/ql-ol-kernel/arch/arm/configs/`:

```
CONFIG_MMC=y
CONFIG_MMC_PERF_PROFILING=y
CONFIG_MMC_CLKGATE=y
CONFIG_MMC_PARANOID_SD_INIT=y
CONFIG_MMC_BLOCK_MINORS=32
CONFIG_MMC_SDHCI=y
CONFIG_MMC_SDHCI_PLTFM=y
CONFIG_MMC_SDHCI_MSM=y
```

5.3. Modifying the Kernel Configuration to Support FAT File System

Checking whether the following configuration items were added to `mdm9607_defconfig` and `mdm9607-perf_defconfig` under the path of `ql-ol-sdk/ql-ol-kernel/arch/arm/configs/`:

```
CONFIG_FAT_FS=y
CONFIG_VFAT_FS
```

5.4. Compilation Test

If user modified `mdm9607_defconfig` and `mdm9607-perf_defconfig` under the path of `ql-ol-sdk/ql-ol-kernel/arch/arm/configs/` and these files have already been compiled, please make sure that the `buid` folder was deleted under the path of `ql-ol-sdk/ql-ol-kernel/` and recompile the kernel.

Please refer to the document *Quectel_EC2x&AG35-QuecOpen_Quick Start* to compile and download the kernel. Please refer to **Chapter 7** of this document to conduct read and write test on EMMC partition and formatting.

6 EMMC File System Production and Functional Test Verification

6.1. Building of LTE OPEN EVB Hardware Test Environment

Please switch to S0701 of LTE OpenEVB to the right of EMMC and connect GPIO_26 and EMMC_RSTN of J0201 with a wire.

6.2. Checking If the EMMC Device is Correctly Identified

Please input `ls/dev/mmc*-l` command at the command line terminal to check whether the EMMC is successfully identified. A sample screenshot is as follows:

```
root@mdm9607-perf:~# ls /dev/mmc*  
/dev/mmcblk0      /dev/mmcblk0rpb  
root@mdm9607-perf:~#
```

According to the screenshot, EMMC storage device `mmcblk0` is successfully identified under `dev` directory. Please input `cat /proc/partitions` at the command line terminal to check BlockSize of EMMC.

6.3. Partition Management

After entering the Linux system, please input the following command in the shell terminal to perform EMMC partition:

```
root@mdm9607:~# fdisk /dev/mmcblk0  
  
The number of cylinders for this disk is set to 117504.  
There is nothing wrong with that, but this is larger than 1024,  
and could in certain setups cause problems with:  
1) software that runs at boot time (e.g., old versions of LILO)  
2) booting and partitioning software from other OSs  
   (e.g., DOS FDISK, OS/2 FDISK)
```

6.3.1. Checking Existing EMMC Partition Information

Please input command p to check the existing partition information of the EMMC.

```
Command (m for help): p

Disk /dev/mmcblk0: 3850 MB, 3850371072 bytes
4 heads, 16 sectors/track, 117504 cylinders
Units = cylinders of 64 * 512 = 32768 bytes

   Device Boot      Start         End      Blocks   Id System
/dev/mmcblk0p1          1        117504       3760120    c Win95 FAT32

Command (m for help):
```

There is one partition existed in the current EMMC according to above figure.

6.3.2. Existing Partition Deletion

If there is no partition for current EMMC, please skip this step. If EMMC has partitions, please input command d and then input the partition number to be deleted in turn. There is only one partition existed in the current EMMC according to the above figure, and the only one partition will be deleted by default when inputting command d. The example is shown in the following figure:

```
   Device Boot      Start         End      Blocks   Id System
/dev/mmcblk0p1          1        117504       3760120    c Win95 FAT32 (LBA)

Command (m for help): d
Selected partition 1
```

6.3.3. Creation of New Partition

Suppose that two partitions need to be created. The first partition is 50M in size, and the remaining storage space is allocated to the second partition. Please input command n and then input command p to create the first partition (size: 50M), among them, blank area command means ENTER key.

```
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-117504, default 1): Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-117504, default 117504): +50M
```

Please input command n to create the second partition (its size is equal to the remaining space) where the blank area command means the ENTER key. Please input p to check partition information.

```
Command (m for help): p

Disk /dev/mmcblk0: 3850 MB, 3850371072 bytes
4 heads, 16 sectors/track, 117504 cylinders
Units = cylinders of 64 * 512 = 32768 bytes

   Device Boot      Start         End      Blocks   Id System
/dev/mmcblk0p1          1         1527       48856    c Win95 FAT32 (LBA)
/dev/mmcblk0p2       1528       117504      3711264    c Win95 FAT32 (LBA)
```

6.3.4. Writing into Newly Created Partition Information

Please input command w to save and write the newly created EMMC partition information.

```
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table
[ 225.455467] mmcblk0: p1 p2
root@mdm9607:/# [ 225.461386] fdisk (1373) used greatest stack depth: 40
[ 225.587798] FAT-fs (mmcblk0p1): Volume was not properly unmounted. So
root@mdm9607:/# ls /dev/mmcblk0
mmcblk0      mmcblk0p1   mmcblk0p2   mmcblk0rpm
```

6.4. Importing Ext4 File System Making Tool

Please import *mke2fs* under the path of */release/sbinql-ol-sdk/ql-ol-extsdk/tools/filesystem_make* to */sbin/* directory of EC2x or AG35 via adb or other methods, and modify its permission to executable permission by *chmod* command.

6.5. Formatting Partition to the Ext4 File System

Here is an example of partition 2 to demonstrate how to format partition 2 to Ext4 file system by *mke2fs* tool.

```
# mke2fs -t ext4 /dev/mmcblk0p2
```

```
/ # mke2fs -t ext4 /dev/mmcblk0p2
mke2fs 1.42.9 (28-Dec-2013)
Discarding device blocks: done
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
233392 inodes, 931840 blocks
46592 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=956301312
29 block groups
32768 blocks per group, 32768 fragments per group
8048 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

6.6. Read and Write Test

Creating mmcblk0p2 folder under directory of /mnt/ and then mounting EMMC partition 2 to under directory of /mnt/mmcblk0p2.

```
root@mdm9607:/# ls /mnt/
mmcblk0p1
root@mdm9607:/# mkdir /mnt/mmcblk0p2
root@mdm9607:/# mount -t ext4 /dev/mmcblk0p2 /mnt/mmcblk0p2/
[ 542.934444] EXT4-fs (mmcblk0p2): mounted filesystem without journal.
```

Checking if the mounting is successful and the partition format is correct by command df -hT. An example is shown below:

```
root@mdm9607:/# df -hT
Filesystem      Type      Size      Used Available Use% Mounted on
ubi0:rootfs     ubifs     55.8M     43.8M      12.0M   78% /
ubi1:modem      ubifs     41.0M     33.1M       7.9M   81% /firmware
tmpfs           tmpfs      64.0K       4.0K     60.0K    6% /dev
tmpfs           tmpfs      75.0M     20.0K     75.0M    0% /run
tmpfs           tmpfs      75.0M     60.0K     74.9M    0% /var/volatile
tmpfs           tmpfs      75.0M       0        75.0M    0% /media/ram
/dev/ubi2_0     ubifs     99.5M     24.0K     99.5M    0% /usrdata
/dev/mmcblk0p1  vfat      47.0M     46.0M     996.5K   98% /mnt/mmcblk0p1
/dev/mmcblk0p2  ext4      3.5G       7.1M      3.3G    0% /mnt/mmcblk0p2
```

Using command of dd if=/dev/zero of=/mnt/mmcblk0p2/test1.img bs=1M count=3072 to generate a 3G large file, which is stored in /mnt/mmcblk0p2, namely the EMMC partition 2. Checking the newly generated file size to determine whether read and write operation of the EMMC are performed successfully.

```
root@mdm9607:/# dd if=/dev/zero of=/mnt/mmcblk0p2/test1.img bs=1M count=3072
3072+0 records in
3072+0 records out
3221225472 bytes (3.0GB) copied, 337.457677 seconds, 9.1MB/s
root@mdm9607:/# ls /mnt/mmcblk0p2/ -l -h
total 3145728
-rwxrwxrwx  1 root    root      3.0G Jan  6 04:11 test1.img
root@mdm9607:/#
```

Quectel
Confidential

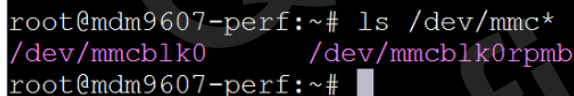
7 SD Card File System Making and Functional Test Verification

7.1. Building of LTE OPEN EVB Hardware Test Environment

Please switch to S0701 of LTE OpenEVB to the SD in the left. Then the GPIO_26 and SD_INS_DET of J0201 will be connected by a wire short connection.

7.2. Checking Whether SD Card is Correctly Identified

Please input the command `ls /dev/mmc*-l` at the command line terminal to check whether the SD card is successfully identified. The following is sample screenshot:



```
root@mdm9607-perf:~# ls /dev/mmc*  
/dev/mmcblk0          /dev/mmcblk0rpmb  
root@mdm9607-perf:~#
```

According to the screenshot, `mmcblk0` was successfully mounted under the directory of `dev`. If users want to check the block size of the SD card, please input the command `cat /proc/partitions` at the command line terminal.

7.3. Partition Management

After entering the Linux system, please input the following command in the shell terminal to perform EMMC partition:

```
# fdisk /dev/mmcblk0
```



```
root@mdm9607:~# fdisk /dev/mmcblk0

The number of cylinders for this disk is set to 117504.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)
```

7.3.1. Checking the Existing SD Card Partition Information

Please input command p to check detailed information of SD card

```
Command (m for help): p

Disk /dev/mmcblk0: 3850 MB, 3850371072 bytes
4 heads, 16 sectors/track, 117504 cylinders
Units = cylinders of 64 * 512 = 32768 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/mmcblk0p1        1         117504      3760120    c  Win95 FAT32

Command (m for help): █
```

There is one partition existed in the currently used SD card according to above figure

7.3.2. Existing Partition Deletion

If there is no partition existed in current EMMC, please skip this step. If EMMC has partition, please input command d, and then input the partition number to be deleted in turn. There is only one partition existed in the current EMMC according to the above figure, and the only existed one partition will be deleted by default when inputting command d. The example is shown in the following figure:

```
   Device Boot      Start         End      Blocks   Id  System
/dev/mmcblk0p1        1         117504      3760120    c  Win95 FAT32 (LBA)

Command (m for help): d
Selected partition 1
```

7.3.3. Creation of New Partition

Suppose that two partitions need to be created. The first partition is 50M in size and the remaining storage space is allocated to the second partition. Please input command n and then input command p to create the first partition (size: 50M), among them, blank area command means ENTER key.

```
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-117504, default 1): Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-117504, default 117504): +50M
```

Please input command n to create the second partition (its size is equal to the remaining space of SD card) where the blank area command represents the ENTER key.

```
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (1528-117504, default 1528): Using default value 1528
Last cylinder or +size or +sizeM or +sizeK (1528-117504, default 117504): Us
```

Please input p to check partition information.

```
Command (m for help): p

Disk /dev/mmcblk0: 3850 MB, 3850371072 bytes
4 heads, 16 sectors/track, 117504 cylinders
Units = cylinders of 64 * 512 = 32768 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/mmcblk0p1        1          1527       48856    c   Win95 FAT32 (LBA)
/dev/mmcblk0p2      1528      117504      3711264    c   Win95 FAT32 (LBA)
```

7.3.4. Partition Information Updating

Please input the command w to update the SD card partition information

```
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table
[ 225.455467] mmcblk0: p1 p2
root@mdm9607:/# [ 225.461386] fdisk (1373) used greatest stack depth: 41
[ 225.587798] FAT-fs (mmcblk0p1): Volume was not properly unmounted. So
root@mdm9607:/# ls /dev/mmcblk0
mmcblk0      mmcblk0p1   mmcblk0p2   mmcblk0rpmb
```

7.4. Formatting Partition to FAT File System

Here is an example of formatting partition 1 to fat file system by the mkfs.vfat tool that comes with the

module itself.

```
# mkfs.vfat /dev/mmcb1k0p1
```

```
root@mdm9607:~# mkfs.vfat /dev/mmcb1k0p1
```

7.5. Test on Read and Write

Creating the folder mmcb1k0p1 under the directory of /mnt/ and mounting partition 1 of SD card to the directory of /mnt/mmcb1k0p1.

```
root@mdm9607:/# ls /mnt/
root@mdm9607:/# mkdir /mnt/mmcb1k0p1
root@mdm9607:/# ls /mnt/
mmcb1k0p1
root@mdm9607:/# mount -t vfat /dev/mmcb1k0p1 /mnt/mmcb1k0p1/
```

Checking if the mounting was successful and if the partition format was correct by df-hT. The example is shown in the following figure:

```
root@mdm9607:/# df -hT
```

Filesystem	Type	Size	Used	Available	Use%	Mounted on
ubi0:rootfs	ubifs	55.8M	43.8M	12.0M	78%	/
ubi1:modem	ubifs	41.0M	33.1M	7.9M	81%	/firmware
tmpfs	tmpfs	64.0K	4.0K	60.0K	6%	/dev
tmpfs	tmpfs	75.0M	20.0K	75.0M	0%	/run
tmpfs	tmpfs	75.0M	60.0K	74.9M	0%	/var/volatile
tmpfs	tmpfs	75.0M	0	75.0M	0%	/media/ram
/dev/ubi2_0	ubifs	99.5M	24.0K	99.5M	0%	/usrdata
/dev/mmcb1k0p1	vfat	47.0M	512	47.0M	0%	/mnt/mmcb1k0p1

Using command of `dd if=/dev/zero of=/mnt/mmcb0p2/test1.img bs=1M count=46` to generate a 46M file, which is stored in /mnt/mmcb1k0p1, namely the SD card partition 1. Check the size of newly generated file to determine whether the read and write operation of SD Card are performed successfully.

8 Common Debugging Method for SD Driver

8.1. Simple Read and Write Test

After the SD device was recognized, a device named mmcblk0 will be generated under the directory of dev. There are many steps for the EMMC/SD card partition and formatting, so the errors are very difficult to be avoided. Users can test basic read and write by simple commands such as echo, hexdump and others. Methods are as below:

8.1.1. Data Writing

Using the echo command to write some data to the identified SD device (/dev/mmcblk0). The command is echo HelloWorld! > /dev/mmcblk0. The example is as follows:

```
root@mdm9607-perf:~# ls /dev/mmc*  
/dev/mmcblk0      /dev/mmcblk0rpmb  
root@mdm9607-perf:~# echo HelloWorld! > /dev/mmcblk0  
root@mdm9607-perf:~#
```

8.1.2. Data Reading

Using the hexdump command to read the written data from the identified SD device (/dev/mmcblk0). The command is hexdump -C -n 15 /dev/mmcblk0.

```
root@mdm9607-perf:~# hexdump -C -n 15 /dev/mmcblk0  
00000000  48 65 6c 6c 6f 57 6f 72  6c 64 21 0a 02 08 06  |HelloWorld!....|  
0000000f  
root@mdm9607-perf:~#
```

8.2. Viewing the File Systems Supported by the Current System

User can view the type of file system supported by the current Linux system by cat/proc/file systems

```
root@mdm9607-perf:~# cat /proc/filesystems
nodev    sysfs
nodev    rootfs
nodev    ramfs
nodev    bdev
nodev    proc
nodev    cgroup
nodev    tmpfs
nodev    debugfs
nodev    sockfs
nodev    pipefs
nodev    configfs
nodev    devpts
          ext3
          ext2
          ext4
          vfat
nodev    mtd_inodefs
nodev    ubifs
nodev    functionfs
root@mdm9607-perf:~#
```

The current system supports file systems such as vfat, ext2, ext3, ext4, etc according to the above figure

8.3. Getting Basic Information of SD Card/EMMC

When identifying SD card or EMMC, the driver has created some commonly used information under the directory of sys in the form of files. Users can view device information through commands such as cat. The following are commonly used commands:

```
ls/sys/block/mmcblk0-l
```

```
cat/sys/block/mmcblk0/size
```

```
cat/proc/self/mounts
```

```
cat/sys/block/mmcblk0/device/cid, csd, date, fwrev, hwrev, manfid, name, oemid, serial, type, uevent
```

8.4. Brief Introduction of Linux MMC Driver Subsystem

8.4.1. Introduction of Linux MMC Subsystem

MMC, SD and SDIO technologies are all comes from MMC technology. There are many commonalities for them. Therefore, Linux Kernel uses MMC framework to manage all devices related to these three technologies.

MMC subsystem of Linux is implemented in the Kernel *drivers/mmc* directory. There are three subdirectories in the directory, namely card, core and host. They stored codes which is relevant to block

device, core layer and main controller of SD/MMC card respectively, and reflect the hierarchical structure of the subsystem.

SD/MMC subsystem can be divided into three layers from top to bottom: namely Host layer, Core layer and Card layer. Their roles are defined as follows:

Host layer: It also known as the main controller driver layer. The main controller is usually a peripheral on the SoC, through which the SoC can transmit data with the card according to the SD/MMC protocol. There are many main controller drivers in *drivers/mmc/host subdirectory*, including SDIO driver *sdhci-msm.c* of MDM9607.

Core layer: namely the core layer of MMC subsystem, whose implementation code is located in subdirectory of *driver/MMC/core*. The core layer (main controller driver) not only provides interface for the Host layer driver such as *mmc_alloc_host()/mmc_add_host()/mmc_remove_host()* to allocate, add and delete MMC main controller objects, but it also implements corresponding protocol codes for SD, MMC and SDIO specifications.

Card layer: as SD/MMC cards belong to block device, so it is necessary to provide block device drivers for these devices. *block.c* under the directory of *drivers/mmc/car*, mainly implements the general block device driver of SD/MMC card , and the request queue and processing of the block device were implemented in *queue.c* . The main device number of SD/MMC card device is 179.

mmc Device Driver block diagram

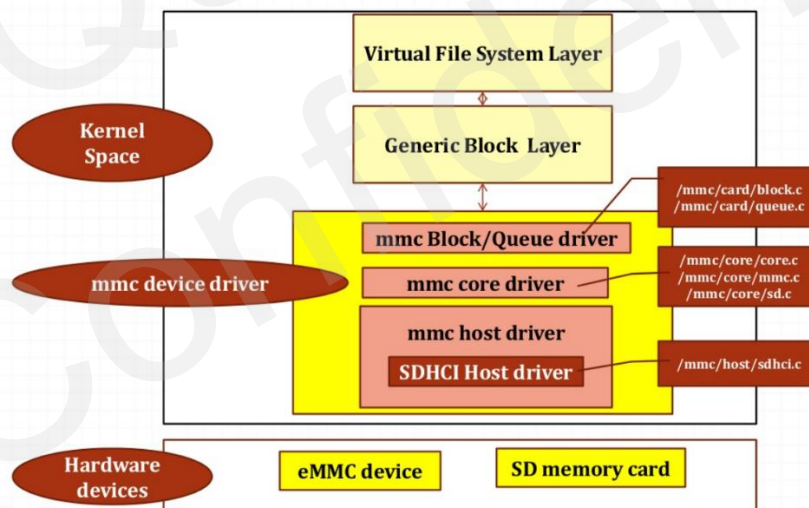


Figure 3: Linux MMC Subsystem Architecture Diagram

8.4.2. Linux MMC Subsystem Driver Initialization Process

Generally, most of the problems will happen in the process of driver initialization. The initialization process of the MMC driver framework is as follows:

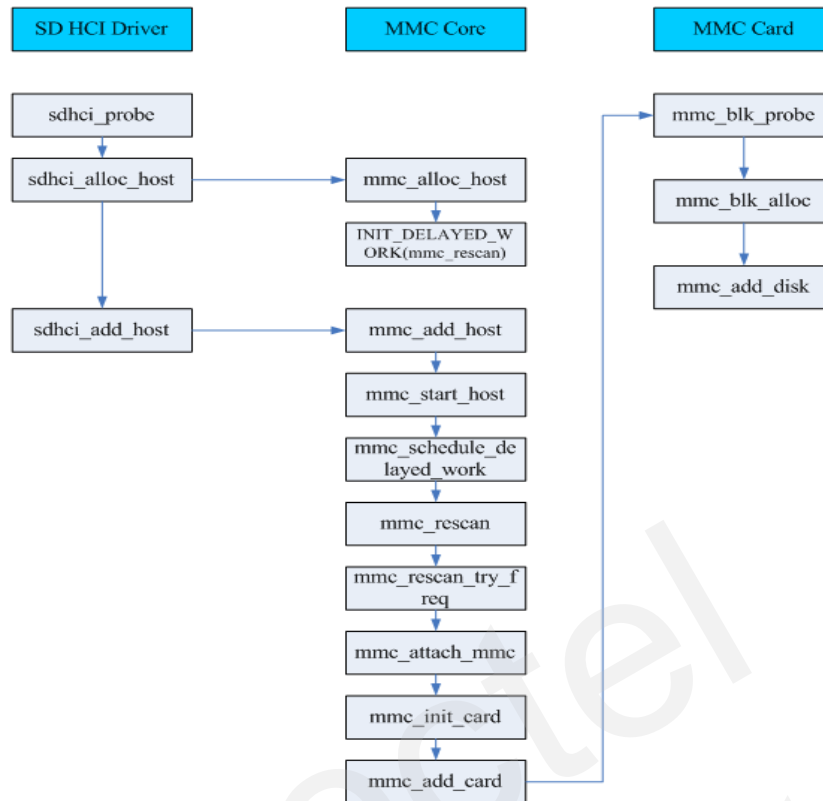


Figure 4: Linux MMC Driver Initialization Flow Chart

8.5. Open Debug Information

8.5.1. Open Debug Macro of the MMC Subsystem

```

1 #
2 # Makefile for the kernel mmc device drivers.
3 #
4 # File and path: Kernel/drivers/mmc/Makefile
5 #subdir-ccflags-$(CONFIG_MMC_DEBUG) := -DDEBUG
6 #subdir-ccflags-y := -DDEBUG
7
8 obj-$(CONFIG_MMC) += core/
9 obj-$(CONFIG_MMC) += card/
10 obj-$(subst m,y,$(CONFIG_MMC)) += host/
  
```

The image shows a snippet of the `Makefile` for the kernel MMC device drivers. A red box highlights the line `#subdir-ccflags-$(CONFIG_MMC_DEBUG) := -DDEBUG`, which is labeled "Before Modifying". Another red box highlights the line `#subdir-ccflags-y := -DDEBUG`, which is labeled "After Modifying". A red arrow points from the "Before Modifying" box to the "After Modifying" box, indicating the change made to the Makefile.

8.5.2. Dmesg Output Debug Log

In order to reduce the amount of information printed by the terminal, the kernel log will be stored in the log_buffer kernel. When users use dmesg, the Kernel log of log_buffer will be outputted to the terminal. The example is as follow:

```
root@mdm9607-perf:~# dmesg -c
[ 567.072599] mmc1: sdhci_msm_set_uhs_signaling-clock:500
[ 567.072626] mmc1: ungated MCI clock
[ 567.072670] sdhci [sdhci_irq()]: *** mmc1 got interrupt
[ 567.072701] mmc1: req done (CMD13): 0: 00000900 00000000
[ 567.270157] mmc1: clock 0Hz busmode 1 powermode 2 cs 0
```


9 Appendix A Reference Documents and Terminology Abbreviations

Table 1: Reference Documents

SN	Document Name	Notes
[1]	80-NL199-1 sdhci_architecture_and_debugging	Qualcomm SD Controller Spec
[2]	Quectel_AG35-QuecOpen_Hardware_Design	AG35 Hardware Design Guide
[3]	Quectel_EC20-QuecOpen_Hardware_Design	EC20 Hardware Design Guide

Table 2: Terminology Abbreviations

Terminology Abbreviations	Description
EMMC	Embedded Multi Media Card
Ext4	Fourth extended filesystem
FAT	File Allocation Table
SD Card	Secure Digital Memory Card