EC2x&AG35-QuecOpen EMMC&SD 卡适配指导说明

LTE Standard/Automotive Module 系列

版本：EC2x&AG35-QuecOpen_EMMC&SD 卡适配指导说明_V1.2

日期：2019-03-04

状态：临时文件

上海移远通信技术股份有限公司始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司
上海市徐汇区虹梅路 1801 号宏业大厦 7 楼　　邮编：200233
电话：+86 21 51086236　　邮箱：info@quectel.com

或联系我司当地办事处，详情请登录：
http://www.quectel.com/cn/support/sales.htm

如需技术支持或反馈我司技术文档中的问题，可随时登陆如下网址：
http://www.quectel.com/cn/support/technical.htm
或发送邮件至：support@quectel.com

前言

上海移远通信技术股份有限公司提供该文档内容用以支持其客户的产品设计。客户须按照文档中提供的规范、参数来设计其产品。由于客户操作不当而造成的人身伤害或财产损失，本公司不承担任何责任。在未声明前，上海移远通信技术股份有限公司有权对该文档进行更新。

版权申明

文档历史

修订记录

版本	日期	作者	变更表述
1.0	2018-03-21	吕恒利	初始版本
1.1	2018-09-15	张文郑	更新制作 ext4 文件系统的工具参数
1.2	2019-03-04	张文郑	添加 EMMC 电源注意说明

目录

表格索引

图表索引

1 引言

Secure Digital 缩写为 SD，全名为 Secure Digital Memory Card，为一种存储卡，被广泛地用于便携式设备上，例如数码相机、个人数码助理和多媒体播放器等。

EMMC 是 Embedded MultiMedia Card 的简称。MMC 是 MultiMedia Card 的简称， 是一种闪存卡（Flash Memory Card）标准，它定义了 MMC 的架构以及访问 Flash Memory 的接口和协议。而 EMMC 则是对 MMC 的一个拓展，以满足更高标准的性能、成本、体积、稳定、易用等的需求。

Linux Kernel 使用 MMC 子系统统一管理 EMMC、SD 卡、SDIO WiFi 等设备。EMMC 强调的是多媒体存储；SD 强调的是安全和数据保护；SDIO 是从 SD 演化出来的，强调的是接口（Input/Output），不再关注另一端的具体形态（可以是 SDIO WiFi 卡等）。

为了满足客户多元化的需求，EC2x 和 AG35 均预留一路 SDIO 接口 SDHC2，用于连接 SD 卡或者 EMMC。该 SDIO 接口符合 SD 3.0 协议。

该文档介绍了电路设计、软件驱动、测试验证等；可以帮助客户简易而快速的进行开发。

2 EMMC&SD 卡硬件接口及引脚定义

AG35-QuecOpen 和 EC2X-QuecOpen 均提供了两路支持 SD 3.0 协议的 SDIO 接口。SDIO1 用来外接 WiFi 设备。SDIO2 用于外接 SD 卡/EMMC 设备。SDIO2 的引脚定义如下：

2.1. EC2x 外接 SD 卡/EMMC 管脚使用定义

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		输出 2.85V/1.8V 可配置。不能用于 SD 卡供电。
SD_INS_DET	23	DI	SD_INS_DET	GPIO_26	

详细信息请查阅文档 Quectel_EC20_R2.0-QuecOpen_Hardware_Design.pdf。

2.2. AG35 外接 SD 卡/EMMC 管脚使用定义

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		
SDC2_CLK	53	DO	SDC2_CLK		
SD_INS_DET	52	IO	SD_INS_DET	GPIO_26	**DI:** Insertion detection for SD card. **DO**: Reset eMMC [1].

详细信息请查阅文档 Quectel_AG35-QuecOpen_Hardware_Design.pdf。

3 硬件电路设计推荐

3.1. EC2x&AG35 外接 SD 卡硬件电路参考设计

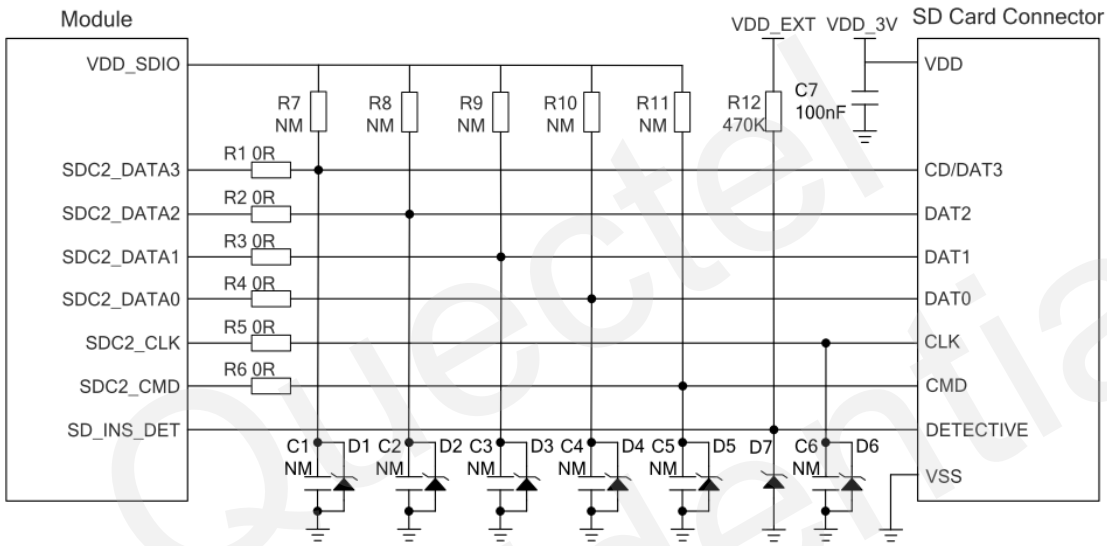

图 1：SD 卡硬件电路参考设计

3.2. EC2x&AG35 外接 EMMC 硬件电路参考设计

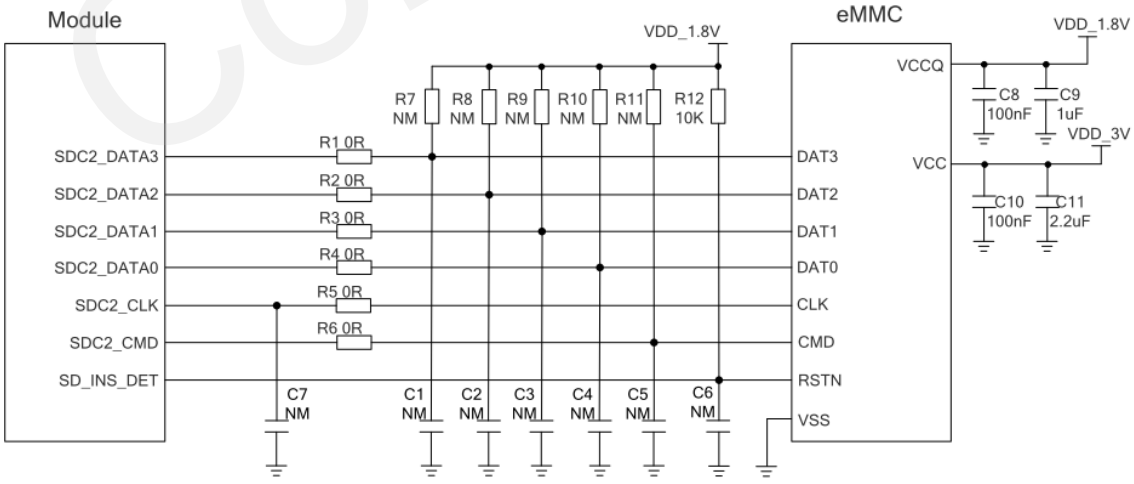

图 2：EMMC 硬件电路参考设计

详细信息请查阅文档 Quectel_EC20_R2.0-QuecOpen_Hardware_Design.pdf 和 Quectel_AG35-QuecOpen_Hardware_Design.pdf。

备注

VCC 和 VCCQ 要常供电，在模块休眠情况下，也不可断电。

4 EMMC 设备树配置及内核驱动适配

4.1. 修改设备树文件，增加设备信息

检查路径为 *ql-ol-sdk/ql-ol-kernel/arch/arm/boot/dts/qcom/*设备树文件 mdm9607-mtp.dtsi 的设备树节点 sdhc_2 的配置信息是否如下所示，如有差异，请修改如下：

```
&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 = <&sdc2_clk_on &sdc2_cmd_on &sdc2_data_on &sdc2_cd_on>;
	pinctrl-1 = <&sdc2_clk_off &sdc2_cmd_off &sdc2_data_off>;

	status = "ok";
};
```

检查路径为 *ql-ol-sdk/ql-ol-kernel/arch/arm/boot/dts/qcom/*设备树文件 mdm9607.dtsi 的设备树节点 sdhc_2 的配置信息是否如下所示，如有差异，请修改为和下图一样：

```
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";
};
```

有关设备树参数的详细信息，请参考路径为 *ql-ol-sdk/ql-ol-kernel/Documentation/devicetree/bindings/mmc* 设备树说明文本 sdhci-msm.txt。

4.2. 修改内核配置，以支持 SD 控制器和 EMMC 驱动

检查路径为 ql-ol-sdk/ql-ol-kernel/arch/arm/configs/ 的配置文件 mdm9607_defconfig 和 mdm9607-perf_defconfig 是否已添加如下配置项：

```
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. 修改内核配置，以支持 Ext4 文件系统

检查路径为 *ql-ol-sdk/ql-ol-kernel/arch/arm/configs/* 的配置文件 mdm9607_defconfig 和 mdm9607-perf_defconfig 是否已添加如下配置项：

```
CONFIG_EXT4_FS=y
```

4.4. 修改内核驱动，避免切换信号电压至 2.85v

修改路径为 *ql-ol-sdk/ql-ol-kernel/drivers/mmc/core/*的源文件 core.c，宏定义 QUECTEL_DISABLE_SWITCH_SIGNAL_VOLTAGE_330 为 1，修改为如下：

```
#define  QUECTEL_DISABLE_SWITCH_SIGNAL_VOLTAGE_3301
```

4.5. 编译测试

若修改了路径为 *ql-ol-sdk/ql-ol-kernel/arch/arm/configs/*的配置文件 mdm9607_defconfig 和 mdm9607-perf_defconfig，且已经编译过，务必删除路径为 *ql-ol-sdk/ql-ol-kernel/*下的 buid 文件夹，重新编译内核。

请参考文档 *Quectel_EC2x&AG35-QuecOpen_快速入门*编译下载内核。参考本文档的***第六章***对 EMMC 分区、格式化，进行读写测试。

5 SD 卡设备树配置及内核驱动适配

5.1. 修改设备树文件，增加设备信息

检查路径为 *ql-ol-sdk/ql-ol-kernel/arch/arm/boot/dts/qcom* 设备树文件 mdm9607-mtp.dtsi 是否已添加以下信息：

```
&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 = <&sdc2_clk_on &sdc2_cmd_on &sdc2_data_on &sdc2_cd_on>;
	pinctrl-1 = <&sdc2_clk_off &sdc2_cmd_off &sdc2_data_off>;

	status = "ok";
};
```

有关设备树参数的详细信息，请参考路径为 *ql-ol-sdk/ql-ol-kernel/Documentation/devicetree/bindings/mmc* 设备树说明文本 sdhci-msm.txt。

5.2. 修改内核配置，以支持 SD 控制器和 SD 卡驱动

检查路径为 *ql-ol-sdk/ql-ol-kernel/arch/arm/configs/*的配置文件 mdm9607_defconfig 和 mdm9607-perf_defconfig 是否已添加如下配置项：

```
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. 修改内核配置，以支持 FAT 文件系统

检查路径为 *ql-ol-sdk/ql-ol-kernel/arch/arm/configs/*的配置文件 mdm9607_defconfig 和 mdm9607-perf_defconfig 是否已添加如下配置项：

```
CONFIG_FAT_FS=y
    CONFIG_VFAT_FS
```

5.4. 编译测试

若修改了路径为 *ql-ol-sdk/ql-ol-kernel/arch/arm/configs/*的配置文件 mdm9607_defconfig 和 mdm9607-perf_defconfig，且已经编译过，务必删除路径为 *ql-ol-sdk/ql-ol-kernel/*下的 buid 文件夹，重新编译内核。

请参考文档 *Quectel_EC2x&AG35-QuecOpen_快速入门*编译下载内核。参考本文档的***第七章***对 SD 卡分区、格式化，进行读写测试。

6 EMMC 文件系统制作和功能测试验证

6.1. 搭建 LTE OPEN EVB 硬件测试环境

请把 LTE OpenEVB 的 S0701 拨码拨到右侧 EMMC 处，使用短接帽短接 J0201 的 GPIO_26 和 EMMC_RSTN。

6.2. 查看 EMMC 设备是否正确识别

在命令行终端，输入命令 ls/dev/mmc*-l，查看 EMMC 是否成功识别。示例截图如下：

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

根据截图可知，dev 目录下成功识别到了 EMMC 存储器件 mmcblk0。在命令行终端输入命令 cat/proc/partions 查看 EMMC 的 BlockSize。

6.3. 分区管理

进入 Linux 系统后，在 shell 终端输入以下命令，对 EMMC 进行分区：
 # 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)
```

6.3.1. 查看 EMMC 已有分区信息

输入命令 p，查看 EMMC 已有分区信息。

```
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):
```

从上图可知，当前使用的 EMMC 存在一个分区。

6.3.2. 删除已有分区

若 EMMC 当前无分区，跳过该步骤。若 EMMC 已有分区，输入命令 d，并依次输入要删除的分区号。由上图可知，当前 EMMC 只有一个分区，输入 d 就默认删除仅有的分区了，示例如下图所示：

```
        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. 创建新分区

假设要创建两个分区。第一个分区大小是 50M，剩余的存储空间分配给第二个分区。
输入命令 n，接着输入命令 p，建立第一个分区（大小 50M），其中空白区命令表示 ENTER 键。

```
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
```

输入命令 n，建立第二个分区（大小为 EMMC 剩余空间大小），其中空白区命令表示 ENTER 键。
输入 p，查看分区信息。

```
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. 写入新创建分区信息

输入命令 w，保存并写入新创建的 EMMC 分区信息。

```
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: 4
[  225.587798] FAT-fs (mmcblk0p1): Volume was not properly unmounted. So

root@mdm9607:/# ls /dev/mmcblk0
mmcblk0      mmcblk0p1    mmcblk0p2    mmcblk0rpmb
```

6.4. 导入 Ext4 文件系统制作工具

将路径为*/release/sbinql-ol-sdk/ql-ol-extsdk/tools/filesystem_make* 下的可执行文件 mke2fs 通过 adb 或其他方法导入 EC2X 或 AG35 的*/sbin/*目录下。并使用 chmod 命令修改其权限为可执行权限。

6.5. 格式化分区为 Ext4 文件系统

此处以分区 2 为例，演示使用制作的 mke2fs 工具格式化分区 2 为 Ext4 文件系统。

```
# 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. 读写测试

在*/mnt/*下创建文件夹 mmcblk0p2。将 EMMC 分区 2 挂载到*/mnt/ mmcblk0p2* 目录下。

```
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.
```

使用 df –hT 查看是否挂载成功，以及分区格式是否正确。示例如下图：

```
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
```

使用 dd if=/dev/zero of=/mnt/mmcblk0p2/test1.img bs=1M count=3072 命令生成大文件，生成 1 个 3G 的文件，存储于*/mnt/mmcblk0p2*，即 EMMC 分区 2。查看新生成的文件大小，判定 EMMC 读写是否成功。

```
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:/#
```

7 SD 卡文件系统制作和功能测试验证

7.1. 搭建 LTE OPEN EVB 硬件测试环境

请把 LTE OpenEVB 的 S0701 拨码拨到左侧 SD 处。再使用短接帽短接 J0201 的 GPIO_26 和 SD_INS_DET。

7.2. 查看 SD 卡是否正确识别

在命令行终端，输入命令 ls /dev/mmc*-l，查看 SD 卡是否成功识别。示例截图如下：

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

根据截图可知，dev 目录下成功挂载了 mmcblk0。如果想查看 SD 卡的 block size，在命令行终端输入命令 cat /proc/partions 查看。

7.3. 分区管理

进入 Linux 系统后，在 shell 终端输入以下命令，对 SD 卡进行分区：
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. 查看 SD 卡已有分区信息

输入命令 p，查看 SD 卡详细信息

```
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):
```

从上图可知，当前使用的 SD 卡存在一个分区。

7.3.2. 删除已有分区

若 SD 卡当前无分区，跳过该步骤。若 SD 卡已有分区，输入命令 d，并依次输入要删除的分区号。由上图可知，当前 SD 卡只有一个分区，输入 d 就默认删除仅有的分区了，示例如下图所示：

```
        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. 创建新分区

创建新分区假设要创建两个分区，第一个分区大小是 50M，剩余的存储空间分配给第二个分区。输入命令 n，接着输入命令 p，建立第一个分区（大小 50M），其中空白区命令表示 ENTER 键。

```
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
```

输入命令 n，建立第二个分区（大小为 SD 卡剩余空间大小），其中空白区命令表示 ENTER 键。

```
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
```

输入 p，查看分区信息。

```
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. 更新分区信息

输入命令 w，更新 SD 卡分区信息。

```
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: 4
[  225.587798] FAT-fs (mmcblk0p1): Volume was not properly unmounted. So

root@mdm9607:/# ls /dev/mmcblk0
mmcblk0      mmcblk0p1    mmcblk0p2    mmcblk0rpmb
```

7.4. 格式化分区为 FAT 文件系统

此处以分区 1 为例，使用模块自带的 mkfs.vfat 工具格式化分区 1 为 fat 文件系统。

 # mkfs.vfat /dev/mmcblk0p1

```
root@mdm9607:~# mkfs.vfat /dev/mmcblk0p1
```

7.5. 读写测试

在*/mnt/*下创建文件夹 mmcblk0p1。将 SD 卡分区 1 挂载到*/mnt/ mmcblk0p1* 目录下。

```
root@mdm9607:/# ls /mnt/
root@mdm9607:/# mkdir /mnt/mmcblk0p1
root@mdm9607:/# ls /mnt/
mmcblk0p1
root@mdm9607:/# mount -t vfat /dev/mmcblk0p1 /mnt/mmcblk0p1/
```

使用 df –hT 查看是否挂载成功，以及分区格式是否正确。示例如下图：

```
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       512     47.0M   0% /mnt/mmcblk0p1
```

使用 *dd if=/dev/zero of=/mnt/mmcblk0p1/test0.img bs=1M count=46* 命令生成 1 个 46M 的文件，存储于*/mnt/ mmcblk0p1*，即 SD 卡分区 1。查看新生成的文件大小，判定 SD 卡读写是否成功。

8 SD 驱动常用调试方法

8.1. 简单读写测试

SD 设备被识别后，会在 dev 目录下生成名为 mmcblk0 的设备。EMMC/SD 卡分区、格式化的步骤较多，难免出错。用户可以先使用简单的 echo、hexdump 等命令测试基本读写。方法如下：

8.1.1. 写入数据

使用 echo 命令向识别到的 SD 设备（/dev/mmcblk0）写入一些数据，命令为 echo HelloWorld! > /dev/mmcblk0，示例如下图：

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

8.1.2. 读出数据

使用 hexdump 命令从识别到的 SD 设备（/dev/mmcblk0）读出写入的数据，命令为 hexdump-C-n 15 /dev/mmcblk0，示例如下图：

```
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. 查看当前系统支持文件系统

通过命令 cat/proc/filesystems，可以获知当前的 Linux 系统所支持的文件系统类型。

```
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:~#
```

如上图可知，当前系统支持 vfat、ext2、ext3、ext4 等文件系统。

8.3. 获取 SD 卡/EMMC 基本信息

识别 SD 卡或者 EMMC 时，驱动已经将一些常用信息通过文件的形式创建在 sys 目录下。用户可以方便的通过 cat 等命令查看设备信息。如下为常用命令：

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. Linux MMC 驱动子系统简述

8.4.1. Linux MMC 子系统介绍

MMC、SD、SDIO 这三种技术都是起源于 MMC 技术，有很多共性，因此 Linux Kernel 统一使用 MMC framework 管理所有和这三种技术有关的设备。

Linux 的 MMC 子系统在内核 drivers/mmc 目录中实现，该目录中有 3 个子目录，分别是 card、core 和 host，它们分别存放 SD/MMC 卡的块设备、核心层和主控制器的相关代码，并反映该子系统的层次架构。

SD/MMC 子系统从上到下可分为 3 个层次：Host 层、Core 层和 Card 层。这 3 个层次的作用定义如下：

Host 层：也叫主控制器驱动层。主控制器通常是 SoC 上的一个外设，通过它，SoC 能够按照 SD/MMC 协议规定的方式与卡进行数据传输。在 *drivers/mmc/host* 子目录下有很多主控制器驱动，其中就包括 MDM9607 的 SDIO 驱动 sdhci-msm.c。

Core 层：即 MMC 子系统核心层，它的实现代码位于 *drivers/mmc/core* 子目录中。核心层（主控制器驱动程序）除为 Host 层驱动提供诸如 mmc_alloc_host()/mmc_add_host()/mmc_remove_host()等分配、添加、删除 MMC 主控制器对象的接口外，还分别就 SD、MMC 和 SDIO 三种规范实现了相应的协议代码。

Card 层：因为 SD/MMC 卡属于块设备， 所以需要为这些设备提供块设备驱动。在 drivers/mmc/card 中，block.c 主要实现了 SD/MMC 卡的通用块设备驱动，在 queue.c 中则实现了该块设备的请求队列及处理。SD/MMC 卡块设备的主设备号是 179。

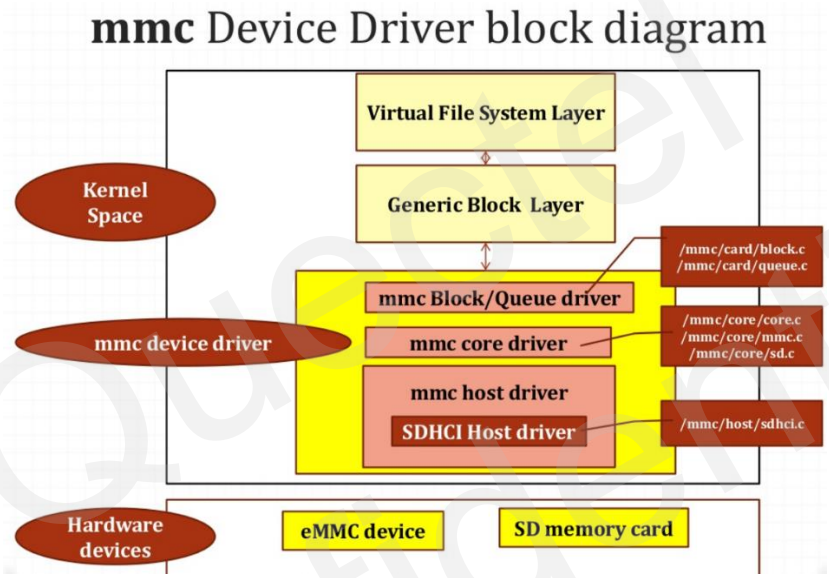

图 3：Linux MMC 子系统架构图

8.4.2. Linux MMC 子系统驱动初始化流程

通常，大多数问题出现在驱动初始化过程中，整理 MMC 驱动框架的初始化过程如下：

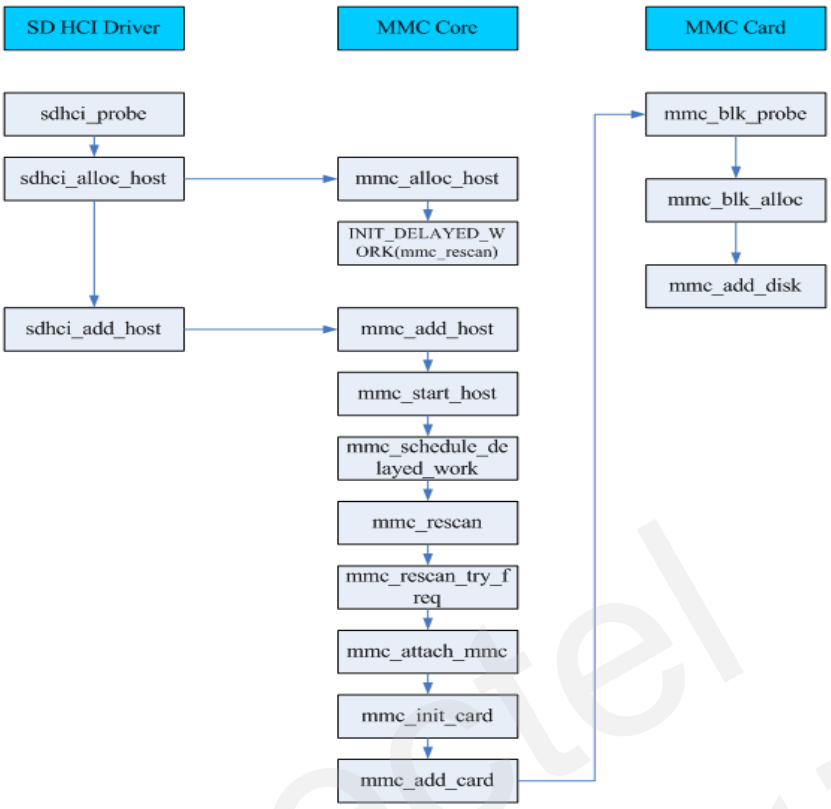

图 4：Linux MMC 驱动初始化流程图

8.5. 打开 Debug 信息

8.5.1. 打开 mmc 子系统调试宏

```
1  #
2  # Makefile for the kernel mmc device drivers.
3  #
4
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/
```

文件及路径kernel/drivers/mmc/Makefile

修改前

修改后

8.5.2. Dmesg 输出 Debug Log

为了减少终端打印信息量，kernel log 会被存放在 log_buffer 内核中。当我们使用 dmesg 时才会把 log_buffer 的 KerneL log 输出到终端，示例如下：

```
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 0000000
[  567.270157] mmc1: clock 0Hz busmode 1 powermode 2 cs 0
```

9 附录 A 参考文档及术语缩写

表 1：参考文档

序号	文档名称	备注
[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

表 2：术语缩写

术语	描述
EMMC	Embedded Multi Media Card
Ext4	Fourth extended filesystem
FAT	File Allocation Table
SD Card	Secure Digital Memory Card