



Linux MMC 开发指南

版本号: 2.7
发布日期: 2022.7.11

版本历史

版本号	日期	制/修订人	内容描述
2.0	2021.11.23	AWA0332	1.init from Linux MMC 开发指南 2. 修改部分格式
2.1	2021.11.25	AWA0332	1. 添加版本历史, 适用范围改用表格, 修复调试节点信息, 添加 dts 或者 sysconfig 一些需要客户自行确认修改的项目
2.2	2021.11.26	AWA0332	1. 添加封面, 修改配置项目格式混乱问题, 修改常见问题的格式问题, 删除多余的版本历史
2.3	2021.12.21	AWA0332	1. 去掉名字版本号和空格问题
2.4	2022.2.18	AWA1579	1. 适配 linux4.9 配置, 并且进行节点修改
2.5	2021.12.11	AWA1767	1. 增加一些常用的 dts 配置项说明
2.6	2022.4.18	AWA1579	1. 增加性能测试节点说明
2.7	2022.7.11	AWA1767	1. 增加新的适用平台

目 录

1 前言	1
1.1 文档简介	1
1.2 目标读者	1
1.3 适用范围	1
2 模块介绍	2
2.1 模块功能介绍	2
2.2 相关术语介绍	2
2.2.1 硬件术语	2
2.2.2 软件术语	2
2.3 模块配置介绍	3
2.3.1 sys_config.fex 配置说明	3
2.3.2 Device Tree 配置说明	4
2.3.2.1 1. uboot 阶段	4
2.3.2.2 2. 内核阶段	6
2.3.3 kernel menuconfig 配置说明	10
2.4 源码结构介绍	11
2.5 驱动框架介绍	12
3 模块接口说明	13
3.1 sunxi_mmc_rescan_card()	13
4 模块使用范例	14
5 FAQ	15
5.1 调试方法	15
5.1.1 调试工具	15
5.1.2 调试节点	15
5.1.2.1 1. 寄存器信息	15
5.1.2.2 2. emmc 信息	17
5.1.2.3 3. 性能验证节点	18
5.2 常见问题	20

插图

2-1 menuconfig 主界面	10
2-2 Device drivers 界面	11
2-3 sdmmc 支持界面	11
5-1 sunxi_host_perf	18
5-2 sysconfig 性能测试配置	20



1 前言

1.1 文档简介

介绍 Linux 内核中 SD/MMC 子系统的接口及使用方法，为 SD/MMC 设备驱动的开发提供参考。

1.2 目标读者

SD/MMC 驱动的开发/维护人员。

1.3 适用范围

产品名称	内核版本	驱动文件
A133	Linux-5.4	sunxi_mmc*
h616	Linux-5.4	sunxi_mmc*
v853	Linux-4.9	sunxi_mmc*
R853	Linux-4.9	sunxi_mmc*
F133	Linux-5.4	sunxi_mmc*

2 模块介绍

2.1 模块功能介绍

Linux 提供了 MMC 子系统来实现对各种 SD/MMC/EMMC/SDIO 设备访问，MMC 子系统由上到下可以分为三层，MMC/SD card 层，MMC/SD core 层以及 MMC/SD host 层，它们之间的层次关系如下所示。

MMC/SD card 层主要负责按照 LINUX 块设备驱动程序的框架实现一个卡的块设备驱动。负责块设备请求的处理，以及请求队列的管理。

MMC/SD core 层负责通信协议的处理，包括 SD/MMC/eMMC/SDIO，为上一层提供具体读写接口，同时为下一层提供 host 端接口。

MMC/SD host 层是实现对 SD/MMC 控制器相关的操作，直接操作硬件，也是主要实现部分。

2.2 相关术语介绍

2.2.1 硬件术语

术语	解释说明
Sunxi	指 Allwinner 的一系列 SOC 硬件平台。
SD	Secure Digital Memory Card
MMC	Multimedia Card
eMMC	Embedded MultiMediaCard
host	指具体的 SD/MMC 控制器

2.2.2 软件术语

无

2.3 模块配置介绍

2.3.1 sys_config.fex 配置说明

[card0_boot_para]

```
card_ctrl      = 0
card_high_speed = 1
card_line      = 4
sd_c_d1        = port:PF0<2><1><3><default>
sd_c_d0        = port:PF1<2><1><3><default>
sd_c_clk       = port:PF2<2><1><3><default>
sd_c_cmd       = port:PF3<2><1><3><default>
sd_c_d3        = port:PF4<2><1><3><default>
sd_c_d2        = port:PF5<2><1><3><default>
```

各个配置项的意义如下：

配置项	配置项含义
card_ctrl	0: 选择卡量产相关的控制器
card_high_speed	速度模式 0 为低速, 1 为高速
card_line	代表卡总线宽度, 分别有 1,4,8
sd_c_d1	sd_c data1 的 GPIO 配置
sd_c_d0	sd_c data0 的 GPIO 配置
sd_c_clk	sd_c clk 的 GPIO 配置
sd_c_cmd	sd_c cmd 的 GPIO 配置
sd_c_d3	sd_c data3 的 GPIO 配置
sd_c_d2	sd_c data2 的 GPIO 配置

[card2_boot_para]

```
card_ctrl      = 2
card_high_speed = 1
card_line      = 8
sd_c_clk       = port:PC5<3><1><3><default>
sd_c_cmd       = port:PC6<3><1><3><default>
sd_c_d0        = port:PC10<3><1><3><default>
sd_c_d1        = port:PC13<3><1><3><default>
sd_c_d2        = port:PC15<3><1><3><default>
sd_c_d3        = port:PC8<3><1><3><default>
sd_c_d4        = port:PC9<3><1><3><default>
sd_c_d5        = port:PC11<3><1><3><default>
sd_c_d6        = port:PC14<3><1><3><default>
sd_c_d7        = port:PC16<3><1><3><default>
sd_c_emmc_rst  = port:PC1<3><1><3><default>
sd_c_ds        = port:PC0<3><2><3><default>
sd_c_ex_dly_used = 2
sd_c_io_1v8    = 1
```

各个配置项的意义如下：

配置项	配置项含义
card_ctrl	0: 选择卡量产相关的控制器
card_high_speed	速度模式 0 为低速, 1 为高速
card_line	代表卡总线宽度, 分别有 1,4,8
sdclk	sdclk 的 GPIO 配置
sdcmd	sdcmd 的 GPIO 配置
sdcd0	sdcd0 的 GPIO 配置
sdcd1	sdcd1 的 GPIO 配置
sdcd2	sdcd2 的 GPIO 配置
sdcd3	sdcd3 的 GPIO 配置
sdcd4	sdcd4 的 GPIO 配置
sdcd5	sdcd5 的 GPIO 配置
sdcd6	sdcd6 的 GPIO 配置
sdcd7	sdcd7 的 GPIO 配置
sdceMMC_rst	emmc 复位信号的 GPIO 配置
sdcds	sdcds 线的 GPIO 配置
sdcd_ex_dly_used	采样模式控制, 2: tune 采样点; 1: 固定采样点方式, 烧写阶段和启动阶段, 通过 sys_config 配置采样点; 其它值: 烧写阶段和启动阶段使用预设的采样点, 通常用 2, 不建议修改
sdcd_io_1v8	1: 表示 eMMC IO 电平是 1.8V, 需要根据实际 emmc io 供电配置
sdcd_force_boot_tuning	1: 强制启动 tuning
sdcd_tm4_win_th	Tune 采样最小可选窗口
sdcd_dis_host_caps	禁止某一种速度模式, bit1: Host 不支持 HS-SDR; Bit3: Host 不支持 8 线模式; bit6:Host 不支持 HS-DDR;bit7:Host 不支持 HS200;bit8:Host 不支持 HS400; 其他值, 不建议使用

2.3.2 Device Tree 配置说明

2.3.2.1 1.uboot 阶段

放到板级目录下面: uboot-board.dts

2.3.2.1.1 (1) sdc0

```
&sdc0_pins_a {
    allwinner,pins = "PF0", "PF1", "PF2",
                    "PF3", "PF4", "PF5";
    allwinner,function = "sdc0";
    allwinner,muxsel = <2>;
}
```

```

allwinner,drive = <3>;
allwinner,pull = <1>;
};

&card0_boot_para {
    /* reg = <0x0 0x2 0x0 0x0>; */
    device_type = "card0_boot_para";
    card_ctrl = <0x0>;
    card_high_speed = <0x1>;
    card_line = <0x4>;
    pinctrl-0 = <&sdc0_pins_a>;
};

```

配置项	配置项含义
card_ctrl	0: 选择卡量产相关的控制器
card_high_speed	速度模式 0 为低速, 1 为高速
card_line	代表卡总线宽度, 分别有 1,4,8
pinctrl-0	代表卡的 pin 设置
sdc0_pins_a	具体卡的 pin 设置, allwinner,pins 代表具体的 pin 名字, allwinner,function 表示 pin 选择的功能, 这里选择 sdc0, allwinner,muxsel 代表 sdc0 对应 spec 里面的功能值, allwinner,drive 代表去掉能力, allwinner,pull 代表上下拉

2.3.2.1.2 (2) sdc2

```

&sdc2_pins_a {
    allwinner,pins = "PC1", "PC5", "PC6",
                    "PC8", "PC9", "PC10", "PC11",
                    "PC13", "PC14", "PC15", "PC16";
    allwinner,function = "sdc2";
    allwinner,muxsel = <3>;
    allwinner,drive = <3>;
    allwinner,pull = <1>;
};

&sdc2_pins_b {
    allwinner,pins = "PC0", "PC1", "PC5", "PC6",
                    "PC8", "PC9", "PC10", "PC11",
                    "PC13", "PC14", "PC15", "PC16";
    allwinner,function = "io_disabled";
    allwinner,muxsel = <7>;
    allwinner,drive = <1>;
    allwinner,pull = <1>;
};

&sdc2_pins_c {
    allwinner,pins = "PC0";
    allwinner,function = "sdc2";
    allwinner,muxsel = <3>;
    allwinner,drive = <3>;
    allwinner,pull = <2>;
};

```

```

&card2_boot_para {
    /*reg = <0x0 0x3 0x0 0x0>; */
    device_type = "card2_boot_para";
    card_ctrl = <0x2>;
    card_high_speed = <0x1>;
    card_line = <0x8>;
    pinctrl-0 = <&sd2_pins_a &sd2_pins_c>;
    sdc_ex_dly_used = <0x2>;
    sdc_io_lv8 = <0x1>;
    sdc_tm4_win_th = <0x08>;
    sdc_tm4_hs200_max_freq = <150>;
    sdc_tm4_hs400_max_freq = <100>;
    sdc_type = "tm4";
};

```

配置项	配置项含义
card_ctrl	0: 选择卡量产相关的控制器
card_high_speed	速度模式 0 为低速, 1 为高速
card_line	代表卡总线宽度, 分别有 1,4,8
sd2_ex_dly_used	采样模式控制, 2: tune 采样点; 1: 固定采样点方式, 烧写阶段和启动阶段, 通过 sys_config 配置采样点; 其它值: 烧写阶段和启动阶段使用预设的采样点, 通常用 2
sd2_io_lv8	1: 表示 eMMC IO 电平是 1.8V, 需要根据实际 emmc io 供电配置
sd2_tm4_win_th	Tune 采样最小可选窗口
sd2_tm4_hs200_max_freq/ sd2_tm4_hs400_max_freq	代表 emmc 的 hs200/hs400 最大频率设置, 不建议修改
sd2_pins_a, sd2_pins_c	具体卡的 pin 设置, allwinner,pins 代表具体的 pin 名字, allwinner,function 表示 pin 选择的功能, 这里选择 sdc0, allwinner,muxsel 代表 sdc0 对应 spec 里面的功能值, allwinner,drive 代表去掉能力, allwinner,pull 代表上下拉

2.3.2.2 2. 内核阶段

存放在 board.dts 或者内核目录下面 arch/armXX/boot/dts/sunxi/sunxiXiwXpX 中在不同的 Sunxi 硬件平台中, SD/MMC 控制器的数目也不一定相同, 但对于每一个 SD/MMC 控制器来说, 在 board.dts 中配置的参数相似。

各个项目的意义如下

2.3.2.2.1 [sd20] 通常用作 SD 卡

```

sdcc0: sdmmc@04020000 {
device_type = "sdcc0";
cd-used-24M;
disable-wp;
pinctrl-0 = <&sdcc0_pins_a>;
bus-width = <4>;
cd-gpios = <&pio PF 6 6 1 3 0xffffffff>;
/*non-removable;*/
/*broken-cd;*/
/*cd-inverted*/
/*data3-detect;*/
cap-sd-highspeed;
sd-uhs-sdr50;
sd-uhs-ddr50;
sd-uhs-sdr104;
no-sdio;
no-mmc;
sunxi-power-save-mode;
/*sunxi-dis-signal-vol-sw;*/
max-frequency = <150000000>;
ctl-spec-caps = <0x8>;
vmmc-supply = <&reg_dldo1>;
vqmmc33sw-supply = <&reg_dldo1>;
vqmmc33sw-supply = <&reg_dldo1>;
vqmmc18sw-supply = <&reg_aldo1>;
vqmmc18sw-supply = <&reg_aldo1>;
status = "okay";
};

```

各个配置项的意义如下：

配置项	配置项含义
device_type	phy 索引值的选择
cd-used-24M	使用 24M 时钟检测插拔卡中断
disable-wp	卡设置写保护，ro
pinctrl-0	第一组 pin 脚的 GPIO 配置
bus-width	线宽
cd-gpios	卡检测的 GPIO 配置
non-removable	不可移除
broken-cd	sd 卡检测方式：轮训
cd-inverted	卡检测的高电平有效还是低电平有效
data3-detect	data3 线检测卡
cap-sd-highspeed	SD 卡的 High speed
sd-uhs-sdr50	SD 卡的 uhs-sdr50
sd-uhs-ddr50	SD 卡的 uhs-ddr50
sd-uhs-sdr104	SD 卡的 uhs-sdr104
no-sdio	无 sdio
no-mmc	无 mmc
sunxi-power-save-mode	发送数据或者命令才有时钟输出
sunxi-dis-signal-vol-sw	关闭电压切换
max-frequency	最大频率

配置项	配置项含义
ctl-spec-caps	控制 spec 能力
vmmc-supply	供电电压、工作电压，需要根据实际 pmu 供电方案修改
vqmmc33sw-supply	3.3V 的 IO 电压，需要根据实际 pmu 供电方案修改
vdmmc33sw-supply	3.3V 的卡检测电压，需要根据实际 pmu 供电方案修改
vqmmc18sw-supply	1.8V 的 IO 电压，需要根据实际 pmu 供电方案修改
vdmmc18sw-supply	1.8V 的卡检测电压，需要根据实际 pmu 供电方案修改
status	设备树的状态

2.3.2.2.2 [sdci1] 通常用作 SDIO WIFI

```
sdci1: sdmmc@04021000 {
    pinctrl-0 = <&sdci0_pins_a>;
    bus-width = <4>;
    cap-sd-highspeed;
    sd-uhs-sdr50;
    sd-uhs-ddr50;
    sd-uhs-sdr104;
    no-sd;
    no-mmc;
    /*sunxi-power-save-mode;*/
    /*sunxi-dis-signal-vol-sw;*/
    cap-sdio-irq;
    keep-power-in-suspend;
    ignore-pm-notify;
    max-frequency = <150000000>;
    ctl-spec-caps = <0x8>;
    sunxi-dly-208M = <1 1 0 0 0 1>;
    vmmc-supply = <&reg_dldo1>;
    vqmmc33sw-supply = <&reg_dldo1>;
    vdmmc33sw-supply = <&reg_dldo1>;
    vqmmc18sw-supply = <&reg_aldo1>;
    vdmmc18sw-supply = <&reg_aldo1>;
    status = "okay";
};
```

各个配置项的意义如下：

配置项	配置项含义
pinctrl-0	第一组 pin 脚的 GPIO 配置
bus-width	线宽
cap-sd-highspeed	SDIO 卡的 High speed
sd-uhs-sdr50	SDIO 卡的 uhs-sdr50
sd-uhs-ddr50	SDIO 卡的 uhs-ddr50
sd-uhs-sdr104	SDIO 卡的 uhs-sdr104
no-sd	无 sd
no-mmc	无 mmc
sunxi-power-save-mode	发送数据或者命令才有时钟输出

配置项	配置项含义
sunxi-dis-signal-vol-sw	关闭电压切换
cap-sdio-irq	开启 SDIO 中断
keep-power-in-suspend	休眠时保持电源
ignore-pm-notify	忽略电源管理的通知
max-frequency	最大频率
ctl-spec-caps	控制 spec 能力
sunxi-dly-208M	<1(cmd driver phase) 1(data driver phase) 0 0 0(data sample phase) 0(cmd sample phase)>
vmmc-supply	供电电压、工作电压，需要根据实际 pmu 供电方案修改
vqmmc33sw-supply	3.3V 的 IO 电压，需要根据实际 pmu 供电方案修改
vdmmc33sw-supply	3.3V 的卡检测电压，需要根据实际 pmu 供电方案修改
vqmmc18sw-supply	1.8V 的 IO 电压，需要根据实际 pmu 供电方案修改
vdmmc18sw-supply	1.8V 的卡检测电压，需要根据实际 pmu 供电方案修改
status	设备树的状态

2.3.2.2.3 [sdcard] 通常用作 eMMC

```
sdcard: sdmmc@04022000 {
    pinctrl-0 = <&sdcard_pins_a &sdcard_pins_c>;
    bus-width = <8>;
    non-removable;
    cap-mmc-highspeed;
    mmc-ddr-1_8v;
    mmc-hs200-1_8v;
    mmc-hs400-1_8v;
    no-sdio;
    no-sd;
    sunxi-power-save-mode;
    sunxi-dis-signal-vol-sw;
    max-frequency = <100000000>;
    ctl-spec-caps = <0x308>;
    vmmc-supply = <&reg_dldo1>;
    vqmmc-supply = <&reg_aldo1>;
    fixed-emmc-driver-type = <0x1>;
    sdcard_tm4_sm0_freq0 = <0>;
    status = "disabled";
};
```

各个配置项的意义如下：

配置项	配置项含义
pinctrl-0	第一组 pin 脚的 GPIO 配置
bus-width	线宽
non-removable	不可移除
cap-mmc-highspeed	MMC 卡的 High speed
mmc-ddr-1_8v	MMC 卡的 ddr50

配置项	配置项含义
mmc-hs200-1_8v	MMC 卡的 hs200
mmc-hs400-1_8v	MMC 卡的 hs400
no-sdio	无 sdio
no-sd	无 sd
sunxi-power-save-mode	发送数据或者命令才有时钟输出
sunxi-dis-signal-vol-sw	关闭电压切换
max-frequency	最大频率
vmmc-supply	供电电压、工作电压，需要根据实际 pmu 供电方案修改
vqmmc-supply	IO 电压，需要根据实际 pmu 供电方案修改
fixed-emmc-driver-type	驱动能力等级调整，对应设置 Extended CSD register 中 HS_TIMING
sdc_tm4_sm0_freqn	host timing setting
status	设备树的状态

2.3.3 kernel menuconfig 配置说明

在命令行中进入内核 longan 根目录，执行 ./build.sh menuconfig 进入配置界面，并按以下步骤操作：

1. menuconfig 主界面

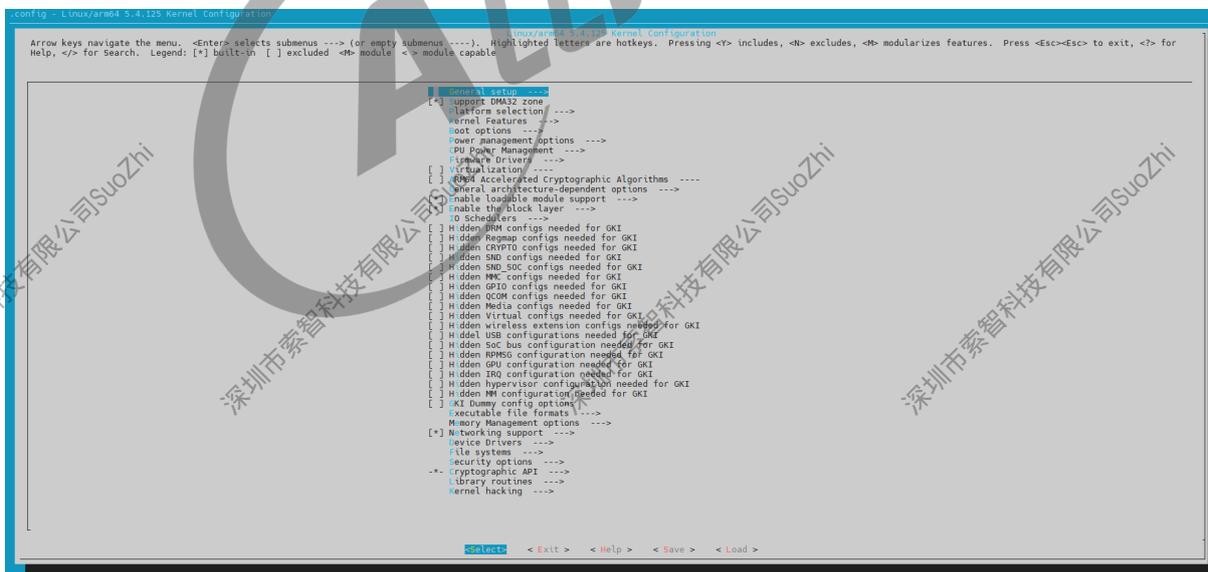


图 2-1: menuconfig 主界面

2. 进入 Device Drivers，并选中 MMC/SD/SDIO card support 为 “* ”

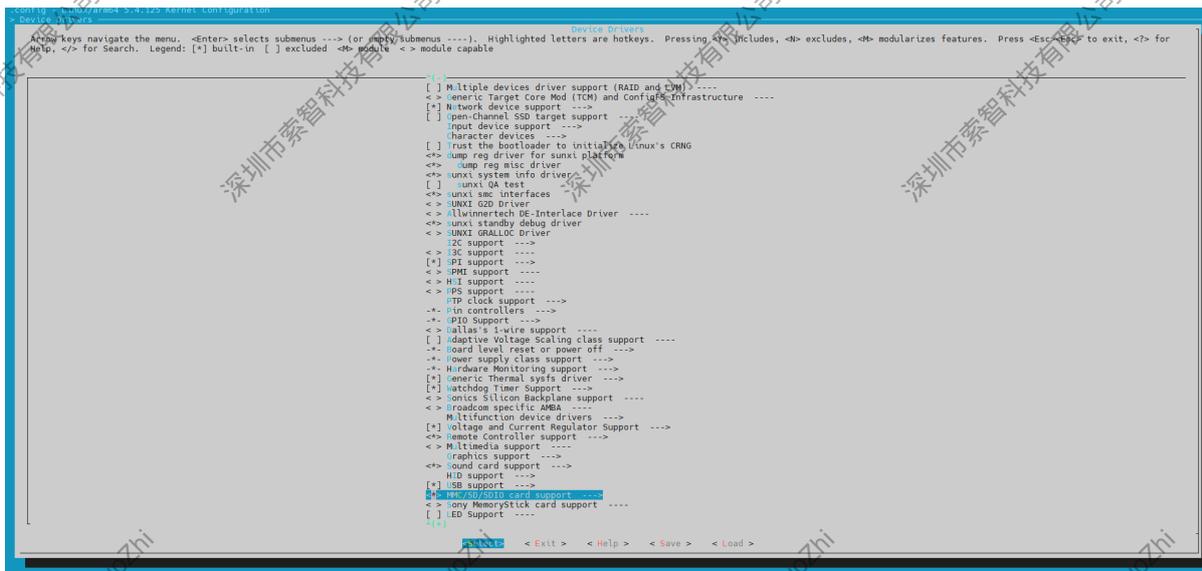


图 2-2: Device drivers 界面

3. 选择 Allwinner sunxi SD/MMC Host Controller support 为 “*”，编译进内核（M 为编译进模块）

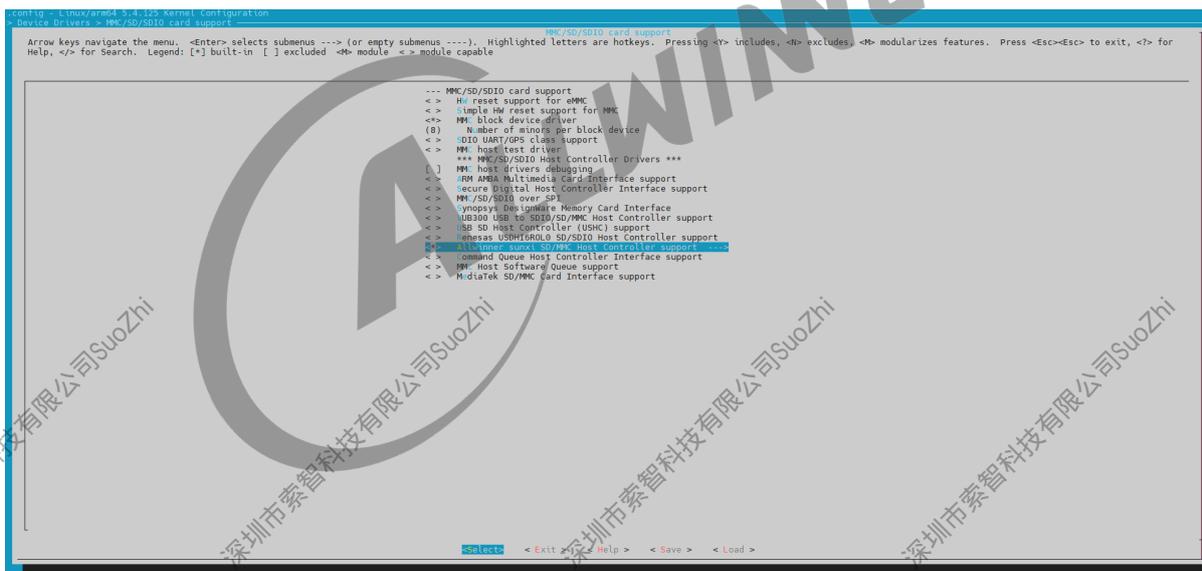


图 2-3: sdmmc 支持界面

2.4 源码结构介绍

SD/MMC 总线驱动的源代码位于内核在 drivers/mmc/host 目录下：drivers/mmc/host

└─ sunxi-mmc.c // 集中了所有的控制器的公共部分以及主要的控制逻辑代码，以及大部分的资源申请和使用，包括中断，pin，ccmu 等

- |— sunxi-mmc.h // 为 Sunxi 平台的 SD/MMC 控制器驱动定义了一些公共的宏、数据结构
- |— sunxi-mmc-v4p1x.c // 部分平台的 sdc0、sdc1 的控制器差异部分驱动代码
- |— sunxi-mmc-v4p1x.h // 部分平台的 sdc0、sdc1 的控制器驱动定义了一些的宏、数据结构
- |— sunxi-mmc-v4p5x.c // sdc2 的控制器差异部分驱动代码
- |— sunxi-mmc-v4p5x.h // sdc2 的控制器驱动定义了一些的宏、数据结构
- |— sunxi-mmc-v5p3x.c /部分平台的 sdc0、sdc1 的控制器差异部分驱动代码
- |— sunxi-mmc-v5p3x.h//部分平台的 sdc0、sdc1 的控制器驱动定义了一些的宏、数据结构
- |— sunxi-mmc-debug.c 用于 debug 的代码
- |— sunxi-mmc-export.c 提供给其他模块的独立接口

2.5 驱动框架介绍

如源码结构介绍

3 模块接口说明

对外函数接口

3.1 sunxi_mmc_rescan_card()

- 作用：用于其他模块手动扫描 sd/sdio，主要提供给 wifi 驱动使用
- 参数：
- 参数 1:id 对于的物理控制器 id
- 返回：
- 无

4 模块使用范例

无



5 FAQ

```
55529 20220303:10:27:16[83134.953729] sunxi-mmc sdc0: smc 0 p0 err, cmd 25, WR EBE !!
55530 20220303:10:27:16[83134.964738] sunxi-mmc sdc0: retry:start
55531 20220303:10:27:16[83134.969109] sunxi-mmc sdc0: retry:stop
55532 20220303:10:27:16[83134.973355] sunxi-mmc sdc0: retry:stop recover
55533 20220303:10:27:16[83134.982714] sunxi-mmc sdc0: REG_DRV_DL: 0x00030000
55534 20220303:10:27:16[83134.988084] sunxi-mmc sdc0: REG_SD_NTSR: 0x81710000
55535 20220303:10:27:16[83134.993551] sunxi-mmc sdc0: REG_NTDL_HS400: 0x20000010
1、 55536 20220303:10:27:16[83134.999559] sunxi-mmc sdc0: *****retry:re-send cmd*****
```

问：sd 卡和 wifi 经常出现 retry: start 等打印的原因和原理；

答：出现这种情况是因为 sdio 通信失败了，一般是数据 crc 校验错误；对于这种错误主要怀疑，硬件信号受到干扰；而这个 retry 机制是通过重发改变相位等方法来规避单次通信失败；在通信无法回复的情况下，会出现大量的 retrylog，此时请检查硬件信号，供电等关键信息。

5.1 调试方法

5.1.1 调试工具

5.1.2 调试节点

5.1.2.1 1. 寄存器信息

linux5.4 内核

a.sdc2

(1).sdc2 gpio 寄存器信息

```
cat /sys/devices/platform/soc@2900000/4022000.sdmmc/sunxi_dump_gpio_register
```

(2).sdc2 ccmu 寄存器信息

```
cat /sys/devices/platform/soc@2900000/4022000.sdmmc/sunxi_dump_ccmu_register
```

(3).sdc2 host 寄存器信息

```
cat /sys/devices/platform/soc@2900000/4022000.sdmmc/sunxi_dump_host_register
```

b.sdc0

(1).sdc0 gpio 寄存器信息

```
cat /sys/devices/platform/soc@2900000/4020000.sdmmc/sunxi_dump_gpio_register
```

(2).sdc0 ccmu 寄存器信息

```
cat /sys/devices/platform/soc@2900000/4020000.sdmmc/sunxi_dump_ccmu_register
```

(3).sdc0 host 寄存器信息

```
cat /sys/devices/platform/soc@2900000/4020000.sdmmc/sunxi_dump_host_register
```

(4) 手动扫描卡接口

```
echo 1 > /sys/devices/platform/soc@2900000/4020000.sdmmc/sunxi_insert
```

c.sdc1

(1).sdc1 gpio 寄存器信息

```
cat /sys/devices/platform/soc@2900000/4021000.sdmmc/sunxi_dump_gpio_register
```

(2).sdc1 ccmu 寄存器信息

```
cat /sys/devices/platform/soc@2900000/4021000.sdmmc/sunxi_dump_ccmu_register
```

(3).sdc1 host 寄存器信息

```
cat /sys/devices/platform/soc@2900000/4021000.sdmmc/sunxi_dump_host_register
```

linux4.9 内核

a.sdc2

(1).sdc2 gpio 寄存器信息

```
cat /sys/devices/platform/soc/sdc2/sunxi_dump_gpio_register
```

(2).sdc2 ccmu 寄存器信息

```
cat /sys/devices/platform/soc/sdc2/sunxi_dump_ccmu_register
```

(3).sdc2 host 寄存器信息

```
cat /sys/devices/platform/soc/sdc2/sunxi_dump_host_register
```

b.sdc0

(1).sdc0 gpio 寄存器信息

```
cat /sys/devices/platform/soc/sdc0/sunxi_dump_gpio_register
```

(2).sdc0 ccmu 寄存器信息

```
cat /sys/devices/platform/soc/sdc0/sunxi_dump_ccmu_register
```

(3).sdc0 host 寄存器信息

```
cat /sys/devices/platform/soc/sdc0/sunxi_dump_host_register
```

(4) 手动扫描卡接口

```
echo 1 > /sys/devices/platform/soc/sdc0/sunxi_insert
```

```
c.sdc1
```

(1).sdc1 gpio 寄存器信息

```
cat /sys/devices/platform/soc/sdc1/sunxi_dump_gpio_register
```

(2).sdc1 ccmu 寄存器信息

```
cat /sys/devices/platform/soc/sdc1/sunxi_dump_ccmu_register
```

(3).sdc1 host 寄存器信息

```
cat /sys/devices/platform/soc/sdc1/sunxi_dump_host_register
```

5.1.2.2 2.emmc 信息

(1) 获取路径：`cd /sys/block/mmcblk0/device`. 这里包含了大部分的 emmc 信息

```
block ffu_capable preferred_erase_size cid fwrev prv cmdq_en hwrev raw_rpmb_size_mult  
csd life_time rca date manfid rel_sectors driver mmcblk0rpmb rev dsr name  
serial enhanced_area_offset ocr subsystem enhanced_area_size oemid type en-  
hanced_rpmb_supported power uevent erase_size pre_eol_info
```

(2) cat 需要的信息，

例如获取寿命信息

```
cat life_time
```

```
cat pre_eol_info
```

获取 emmc 名字

```
cat namd
```

获取生产日期

```
cat data
```

获取唯一识别码

```
cat cid
```

获取制造商 id

```
cat manfid
```

5.1.2.3 3、性能验证节点

该节点主要是用来记录底层存储的读写性能，该节点记录的数据不参含调度以及文件系统的影响。

为了描述方便，这里设定 base 目录这一概念，其中 X 代表控制器号；

内核 linux4.9 base=/sys/devices/platform/soc/sdcX

内核 linux5.4 base=/sys/devices/platform/soc@2900000/402X000.sdmmc

(1) 开始测量：echo 1 > /\$base/sunxi_host_perf

(2) 进行读写操作

(3) 获取测试结果：cat /\$base/sunxi_host_perf

```
Write performance at host driver Level:1073741824 bytes in 105978400 microseconds
```

```
Read performance at host driver Level:0 bytes in 0 microseconds
```

图 5-1: sunxi_host_perf

(4) 速度计算：1073741824byte/105978400us = 9.66MB/s

(5) 清除测量数据：echo 0 > /\$base/sunxi_host_perf

动态设置

以下动态设置的节点均于 base 目录下：

sunxi_host_perf 总开关，打开后下面设置才有效

sunxi_host_filter_w_sector：单笔数据传输的数据大于等于这个数据量，

sunxi_host_filter_w_speed 才生效，单位是扇区

sunxi_host_filter_w_speed：速度低于这个值就打印出来，单位是 B/S

参考

```
echo 20971520 > /$base/sunxi_host_filter_w_speed
```

```
echo 8 > /$base/sunxi_host_filter_w_sector
```

```
echo 1 > /$base/sunxi_host_perf
```

效果

```
20190322_17:24:37.207 [ 64.922940] c=25,a=0x 3fc00,bs= 2560,t= 105463us,sp=
12136KB/s
```

```
20190322_17:24:37.586 [ 65.301113] c=25,a=0x 43800,bs= 2560,t= 92740us,sp=
13802KB/s
```

```
20190322_17:24:37.829 [ 65.544155] c=25,a=0x 46000,bs= 2560,t= 94162us,sp=
13593KB/s
```

```
20190322_17:24:37.967 [ 65.682744] c=25,a=0x 47400,bs= 2560,t= 77371us,sp=
16543KB/s
```

```
20190322_17:24:38.041 [ 65.755126] c=25,a=0x 47e00,bs= 2560,t= 64860us,sp=
19734KB/s
```

开机默认启动

在 dts 或者 sysconfig.fex 里面加入下面配置

per_enable: 总开关，打开后下面设置才有效

filter_sector: 传输的数据大于等于这个数据量，

filter_speed 才生效，单位是扇区

filter_speed: 速度低于这个值就打印出来，单位是 B/S

参考

```
68 ;
69 [sdcard]
70 sdc0_used = 1
71 bus-width = 4
72 sdc0_d1 = port:PF00<2><1><3><default>
73 sdc0_d0 = port:PF01<2><1><3><default>
74 sdc0_clk = port:PF02<2><1><3><default>
75 sdc0_cmd = port:PF03<2><1><3><default>
76 sdc0_d3 = port:PF04<2><1><3><default>
77 sdc0_d2 = port:PF05<2><1><3><default>
78 cd-gpios = port:PF06<6><1><3><default>
79 sunxi-power-save-mode =
80 ;sunxi-dis-signal-vol-sw =
81 sd-uhs-sdr50 =
82 sd-uhs-ddr50 =
83 sd-uhs-sdr104 =
84 max-frequency = 150000000
85 ;broken-cd =
86 vmmc="vcc-sdc"
87 vqmmc33sw="vcc33-pf"
88 vqmmc18sw="vcc18-pf"
89 vdmcc33sw="vcc33-pf"
90 vdmcc18sw="vcc18-pf"
91 filter_speed = 209/1520
92 filter_sector = 8
93 per_enable = 1
94
```

图 5-2: sysconfig 性能测试配置

效果

```
20190322_17:24:37.207 [ 64.922940] c=25,a=0x 3fc00,bs= 2560,t= 105463us,sp=
12136KB/s
```

```
20190322_17:24:37.586 [ 65.301113] c=25,a=0x 43800,bs= 2560,t= 92740us,sp=
13802KB/s
```

```
20190322_17:24:37.829 [ 65.544155] c=25,a=0x 46000,bs= 2560,t= 94162us,sp=
13593KB/s
```

```
20190322_17:24:37.967 [ 65.682744] c=25,a=0x 47400,bs= 2560,t= 77371us,sp=
16543KB/s
```

```
20190322_17:24:38.041 [ 65.755126] c=25,a=0x 47e00,bs= 2560,t= 64860us,sp=
19734KB/s
```

5.2 常见问题

参考《MMC 量产问题快速排查指南》《eMMC 硬件排查指南》

著作权声明

版权所有 © 2022 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明

、 全志科技、**全志科技** （不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。