# MMC 量产问题

# 快速排查指南

**版本号: 1.5**
**发布日期: 2022.11.24**

# 版本历史

| 版本号 | 日期 | 制/修订人 | 内容描述 |
|---|---|---|---|
| V0.1 | 2020.1.2 | AWA1579 | 文档初始版本 |
| V0.2 | 2020.1.06 | AWA1579 | 根据小组成员建议修改文档格式 |
| 0.3 | 2020.1.09 | AWA1579 | 根据小组成员建议修改具体配置 |
| 0.4 | 2020.03.23 | AWA1579 | 添加 dto 新案例 |
| 0.5 | 2020.8.21 | AWA0332 | 增加 eMMC 支持情况检查，eMMC 里数据异常分析，增加 DTO 案例 |
| 0.6 | 2020.8.31 | AWA0332 | 增加 sample phase 具体意义，dump 工具使用连接 |
| 0.7 | 2020.8.31 | AWA0332 | 增加增加 brom 启动 boot0 失败 |
| 0.8 | 2020.1.19 | AWA1579 | 增加 uboot2018 的 bias 配置 |
| 0.9 | 2020.3.18 | AWA1579 | 增加 sdio 类问题以及卡死问题分析 |
| 1.0 | 2021.7.29 | AWA1767 | 1、内核查询 mmc 设备属性的方法 2、增加 SD 卡电压切换导致出现 RTO 问题 3、最佳输出相位的配置方法 4、数据不导通问题，增加案例 5、增加 DCE/RCE 类 CRC 问题排查方法 6、SDC 的 CLK 不连续 7、同一固件在不同板子 eMMC 跑的速度不同 8、mmc 设备擦除慢 9、uboot 加载内核失败情况，增加 dram 异常造成的案例 |
| 1.1 | 2021.12.20 | AWA1579 | 增加电源配置异常问题分析 |
| 1.2 | 2022.2.27 | AWA1767 | 增加 R 线存储功能问题排查 |
| 1.3 | 2022.3.31 | AWA1767 | 增加近一年存储问题排查整理 |
| 1.4 | 2022.8.29 | AWA1767 | 增加 brom 耗时久问题 |
| 1.5 | 2022.11.24 | AWA1767 | 修改文件大小写格式和增加耐压配置说明 |

# 目　录

# 插　图

# 1 前言

## 1.1 概述

- 本文档提供无线网 sd/eMMC/sdio 等常规问题快速排查方案
- 本文档提供的快速排查方案模型如下：

常规问题场景 + 基础硬件排查 + 基础软件配置排查 + 软件快速调试排查

- 本文档仅用于帮助开发人员快速排查和解决初级问题问题，如果按照文档中的指导仍无法解决问题，请联系全志原厂 FAE 进一步协助解决，并提供说明是否按照文档中的排查方案进行过必要的排查，提供排查过程的必要记录和数据

## 1.2 读者对象

本文档主要适用于以下工程师:

- sd/sdio/mmc 技术基础的 RD 研发人员
- sd/sdio/mmc 技术基础的 FAE 工程人员

# 2 eMMC/sd/sdio 问题背景知识

因为驱动的原因，打印格式不一定完全一样，但是大部分关键字是能对上的。

## 2.1 Boot 阶段打印参考说明

Boot 阶段打印说明 Boot 阶段

```
（包括boot0、boot1、uboot）
常见log打印boot阶段打印 [mmc]: mmc 2 cmd 8 timeout, err 0x00000100
[mmc]: mmc 2 cmd 8 err 0x00000100
[mmc]: mmc 2 send if cond failed
[mmc]: mmc 2 cmd 55 timeout, err 0x00000100
[mmc]: mmc 2 cmd 55 err 0x00000100
[mmc]: mmc 2 send app cmd failed
err 说明  err 0x00000100 表示 Response timeout
err 0x00000042 表示 Response CRC error
err 0x00000080 表示 Data CRC error
err 0x00000200 表示 Data timeout
err 0x00002000 表示 Data Start Error
err 0x00008000 表示 Data End-bit error
```

## 2.2 内核阶段打印参考说明

```
Linux内核驱动打印 [mmc-err] smc 2 err, cmd 55,  RTO
[mmc-err] smc 2 err, cmd 8,  EBE
Updataclock fail
...
err 说明  RTO：  Response timeout
RCE：  Response CRC error
EBE：  Data End-bit error
SBE：  Data Start Error
DCE：  Data CRC error
DTO：  Data timeout
```

表示的错误跟 boot 的阶段一样

## 2.3 Sd/eMMC/sdio 日志信息调试

**打印添加原则，在错误第一现场添加打印：**

### 2.3.1 Uboot 阶段

可以在需要的地方添加如下打印，dump 出 pio，ccmu 以及控制器的寄存器：

```
//>-----dumphex32("ccmu", (char *)SUNXI_CCM_BASE, 0x100);
//>-----dumphex32("gpio", (char *)SUNXI_PIO_BASE, 0x100);
//>-----dumphex32("mmc", (char *)priv->reg, 0x100);
```

图 2-1: uboot 调试接口

### 2.3.2 Kernel 阶段

Linux 平台通过如下步骤抓取系统日志信息：

1. echo 8 > /proc/sys/kernel/printk。
2. 通过/sys/class/sunxi_dump/dump 打印寄存器确定。
3. 代码添加 sunxi_mmc_dumphex32/sunxi_dump_reg 打印出寄存器确定状态。

```
>-------dev_err(mmc_dev(host->mmc),
>-------->-------"smc %d p%d err, cmd %d,%s%s%s%s%s%s%s%s%s%s%s !!\n",
>-------->-------host->mmc->index, host->phy_index, cmd->opcode,
>-------->-------data ? (data->flags & MMC_DATA_WRITE ? " WR" : " RD") : "",
>-------->-------host->int_sum & SDXC_RESP_ERROR ? " RE" : "",
>-------->-------host->int_sum & SDXC_RESP_CRC_ERROR ? " RCE" : "",
>-------->-------host->int_sum & SDXC_DATA_CRC_ERROR ? " DCE" : "",
>-------->-------host->int_sum & SDXC_RESP_TIMEOUT ? " RTO" : "",
>-------->-------host->int_sum & SDXC_DATA_TIMEOUT ? " DTO" : "",
>-------->-------host->int_sum & SDXC_FIFO_RUN_ERROR ? " FE" : "",
>-------->-------host->int_sum & SDXC_HARD_WARE_LOCKED ? " HL" : "",
>-------->-------host->int_sum & SDXC_START_BIT_ERROR ? " SBE" : "",
>-------->-------host->int_sum & SDXC_END_BIT_ERROR ? " EBE" : "");
>-------/*sunxi_mmc_dumphex32(host,"sunxi_mmc",host->reg_base,0x180); */
>-------/*sunxi_mmc_dump_des(host,host->sg_cpu,PAGE_SIZE); */
}
```

图 2-2: kernel 调试接口

4. 内核查询 mmc 设备属性。命令行输入：

   mount -t debugfs none /sys/kernel/debug（确认 debugfs 已经挂载）

   cat /sys/kernel/debug/mmc0/ios（cat 对应存储设备的节点的 ios）

   打印结果如下：clock: 150000000 Hz
   vdd: 23 (3.5 ~ 3.6 V)
   bus mode: 2 (push-pull)
   chip select: 0 (don’ t care)

power mode: 2 (on)
bus width: 3 (8 bits)
timing spec: 10 (mmc HS400)
signal voltage: 1 (1.80 V)
driver type: 0 (driver type B)

上述 ios 的节点对应的 root 目录 mmc0 和我们控制器的定义顺序不同，这个是根据识别的顺序定义的 mmcx 的序号。该结果查询出来的值为当前 eMMC 的状态值，比如上面的 signal voltage 的值，是直接打印 host->ios->signal_voltage 的值，每次电压值更新都会对该值进行修改，即为当前值。

# 3 sd/eMMC/sdio 常见问题场景排查

在进行软件分析前注意点，软件调试和排查都应该在 layout 和硬件正常情况下进行：

1）**Layout 排查：**

eMMC PCB 布局规定：（具体参考《芯片应用指南-硬件》）

eMMC NC/RFU 等保留引脚都悬空，不可为了走线方便将这些信号与电源、地、或其他 eMMC 信号连接在一起。如果确实走线有困难，可适当修改 eMMC PCB 封装，去掉一些 NC/RFU 的 ball。

出现问题，需先排查是否 NC/RFU 的 ball 参与走线，如果存在这些问题，请按照《芯片应用指南-硬件》layout，避免 NC/RFU 的 ball 走线引起的各种问题。

1）**硬件信息排查：**

具体方法参考《AW_eMMC_SD 量产问题信息收集模板》上的 “SD、eMMC 硬件排查方向指导”一节。

日常量产过程中，sd/eMMC/sdio 模块会遇到” 高低温实验失败”、” 初始化失败” 、“读写失败”，几类情况，请客户端分别按照以下说明进行调试。

## 3.1 eMMC 支持情况检查

遇到问题或者客户量产前，先检查一下 eMMC 是否在支持列表里面，不在支持列表里面，先按照要求做 eMMC 验证。

## 3.2 eMMC 读写数据失败

先检查 bias 设置是否正确：参考下面《bias 设置》章节

## 3.3 eMMC 高低温实验失败

**问题现象**

eMMC 常温下，使用以及相关测试正常，一到高低温实验，就出现 eMMC 读写相关的错误。

比如高温实验中出现读 cmd18 错误：

```
[mmc]: smc 2 err, cmd 18,  DCE EBE
[mmc]: mmc 2 close bus gating and reset
[mmc]: mmc 2 mmc cmd 18 err 0x00008080
[mmc]: read block failed, mmc_read_blocks 280
[mmc]: block read failed, mmc_bread 333
read mbr copy[0] failed
[mmc]: mmc 2 cmd 18 timeout, err 8180
[mmc]: smc 2 err, cmd 18,  DCE RTO EBE
[mmc]: mmc 2 close bus gating and reset
[mmc]: mmc 2 mmc cmd 18 err 0x00008180
[mmc]: read block failed, mmc_read_blocks 280
[mmc]: block read failed, mmc_bread 333
read mbr copy[1] failed
[mmc]: mmc 2 cmd 18 timeout, err 100
[mmc]: smc 2 err, cmd 18,  RTO
[mmc]: mmc 2 close bus gating and reset
[mmc]: mmc 2 mmc cmd 18 err 0x00000100
[mmc]: read block failed, mmc_read_blocks 280
[mmc]: block read failed, mmc_bread 333
read mbr copy[2] failed
[mmc]: mmc 2 cmd 18 timeout, err 100
[mmc]: smc 2 err, cmd 18,  RTO
[mmc]: mmc 2 close bus gating and reset
[mmc]: mmc 2 mmc cmd 18 err 0x00000100
[mmc]: read block failed, mmc_read_blocks 280
[mmc]: block read failed, mmc_bread 333
read mbr copy[3] failed
[0.731]hdmi hdcp not enable!
```

图 3-1: 高温校验错误

**问题分析**

1、确认 eMMC，dram 和 IC 工作温度范围是否在高低温实验范围。工作温度范围合适的情况下，进行步骤 2 操作；否则，所有模组工作温度范围取最小值。

2、确认 sys_config.fex [card2_boot_para] 下配置了 sdc_ex_dly_used = 配置该属性后，进行步骤 3。

3、更新相应平台的 mmc 驱动代码, 如果问题还在，找 mmc 驱动相关同事确定代码更新是否对了，再做实验。如问题还在，走步骤 4

4、在相同的高低温环境下进行固件烧写，并且进行同样的实验。将实验结果、常温烧写 log、高低温烧写 log 反馈全志 FAE。

5、反馈的 log 得包含烧写过程中，如下格式的采样点窗口 +tuning 信息：

```
[2019/10/31 14:49:35] [9.400][mmc]: write_tuning_try_freq: write ok
[2019/10/31 14:49:35] [9.402][mmc]: Pattern compare ok
[2019/10/31 14:49:35] [9.404][mmc]: Write tuning pattern ok
[2019/10/31 14:49:35] [9.408][mmc]: ================== HSSDR52_SDR25...
[2019/10/31 14:49:35] [9.412][mmc]: skip freq 400000
[2019/10/31 14:49:35] [9.415][mmc]: skip freq 25000000
[2019/10/31 14:49:35] [9.418][mmc]: freq: 2-50000000
[2019/10/31 14:49:38] [mmc]: [0-63|64]
[2019/10/31 14:49:38] [12.037][mmc]: ================== HS200_SDR104...
[2019/10/31 14:49:38] [12.039][mmc]: skip freq 400000
[2019/10/31 14:49:38] [12.039][mmc]: skip freq 25000000
[2019/10/31 14:49:38] [12.039][mmc]: freq: 2-50000000
[2019/10/31 14:49:40] [14.537][mmc]: freq: 3-100000000
[2019/10/31 14:49:42] [16.544][mmc]: freq: 4-150000000
[2019/10/31 14:49:44] [18.511][mmc]: freq: 5-200000000
[2019/10/31 14:49:46] [mmc]: [0-62|63]
[2019/10/31 14:49:46] [mmc]: [0-10|11] [15-41|27] [52-62|11]
[2019/10/31 14:49:46] [mmc]: [0-24|25] [33-59|27]
[2019/10/31 14:49:46] [mmc]: [0-9|10] [15-15|1] [24-36|13] [52-62|11]
[2019/10/31 14:49:46] [19.951][mmc]: ================== HSDDR52_DDR50...
[2019/10/31 14:49:46] [19.953][mmc]: skip freq 400000
[2019/10/31 14:49:46] [19.953][mmc]: freq: 1-25000000
[2019/10/31 14:49:48] [22.045][mmc]: freq: 2-50000000
[2019/10/31 14:49:49] [mmc]: [0-51|52]
[2019/10/31 14:49:49] [mmc]: [0-14|15] [19-24|6] [52-63|12]
[2019/10/31 14:49:49] [23.436][mmc]: ================== HS400...
[2019/10/31 14:49:49] [23.439][mmc]: skip freq 400000
[2019/10/31 14:49:49] [23.439][mmc]: skip freq 25000000
[2019/10/31 14:49:49] [23.439][mmc]: freq: 2-50000000
[2019/10/31 14:49:49] [23.455][mmc]: freq: 3-100000000
[2019/10/31 14:49:49] [23.469][mmc]: freq: 4-150000000
[2019/10/31 14:49:49] [23.486][mmc]: skip freq 200000000
[2019/10/31 14:49:49] [mmc]: [0-63|64]
[2019/10/31 14:49:49] [mmc]: [0-46|47] [51-63|13]
[2019/10/31 14:49:49] [mmc]: [0-29|30] [33-63|31]
[2019/10/31 14:49:49] [23.486][mmc]: skip freq 400000
[2019/10/31 14:49:49] [23.486][mmc]: skip freq 25000000
[2019/10/31 14:49:49] [23.486][mmc]: freq: 2-50000000
[2019/10/31 14:49:51] [25.309][mmc]: freq: 3-100000000
[2019/10/31 14:49:52] [25.999][mmc]: freq: 4-150000000
[2019/10/31 14:49:52] [26.463][mmc]: skip freq 200000000
[2019/10/31 14:49:52] [mmc]: [0-47|48]
[2019/10/31 14:49:52] [mmc]: [0-10|11] [15-21|7]
[2019/10/31 14:49:52] [mmc]: [0-11|12]
[2019/10/31 14:49:52] [26.466][mmc]: DS26/SDR12: 0xffffffff 0xffffffff
[2019/10/31 14:49:52] [26.466][mmc]: HSSDR52/SDR25: 0xff20ffff 0xffffffff
[2019/10/31 14:49:52] [26.466][mmc]: HSDDR52/DDR50: 0xff071aff 0xffffffff
[2019/10/31 14:49:52] [26.466][mmc]: HS200/SDR104: 0x1c1fffff 0xffff1e2e
[2019/10/31 14:49:52] [26.467][mmc]: HS400: 0xff18ffff 0xffffff06
[2019/10/31 14:49:52] [26.471][mmc]: HS400: 0x1720ffff 0xffffff30
[2019/10/31 14:49:52] [26.475][mmc]: Best spd md: 4-HS400, freq: 2-50000000
[2019/10/31 14:49:52] [26.482][mmc]: Bus width 8
```

采样点窗口

tuning结果

图 3-2: 采样点参数

# 3.4 mmc 卡初始化失败

**正常的初始化 log:**

**Uboot 阶段**

```
[2019/10/31 14:50:29] SUNXI SD/MMC: 2
[2019/10/31 14:50:29] [0.559][cpu1]screen 0 don't support TV!
[2019/10/31 14:50:29] [0.560][cpu1]screen 1 don't support TV!
[2019/10/31 14:50:29] [0.562][mmc]: media type 0x8000000
[2019/10/31 14:50:29] [cpu1]drv_disp_init finish
[2019/10/31 14:50:29] [0.566][mmc]: Try MMC card 2
[2019/10/31 14:50:29] [0.569][cpu1]hdmi hdcp not enable!
[2019/10/31 14:50:29] [0.577][cpu1]boot_disp.output_disp=0
[2019/10/31 14:50:29] [0.578][cpu1]boot_disp.output_mode=14
[2019/10/31 14:50:29] [0.580][mmc]: *****grp info 4000 4000 400 400
[2019/10/31 14:50:29] [0.581][cpu1]fetch script data boot_disp.auto_hpd fail
[2019/10/31 14:50:29] [0.585][mmc]: def wp_grp_size 4000
[2019/10/31 14:50:29] [0.590][cpu1]disp0 device type(2) enable
[2019/10/31 14:50:29] [0.593][mmc]: wp_grp_size 0x4000
[2019/10/31 14:50:29] [0.600][mmc]: *****grp info 4000 4000 400 400
[2019/10/31 14:50:29] [0.607][mmc]: host caps: 0x1ef
[2019/10/31 14:50:29] [0.608][mmc]: MID 000045 PSN ace26a0d
[2019/10/31 14:50:29] [0.610][mmc]: PNM DG4008 -- 0x44-47-34-30-30
[2019/10/31 14:50:29] [0.614][mmc]: PRV 0.1
[2019/10/31 14:50:29] [0.616][mmc]: MDT m-5 y-2019
[2019/10/31 14:50:29] [0.619][mmc]: MMC v5.1
[2019/10/31 14:50:29] [0.621][mmc]: user capacity  : 7457 MB
[2019/10/31 14:50:29] [0.624][mmc]: wp_grp_size: 0x4000 sector
[2019/10/31 14:50:29] [0.628][mmc]: don't support write protect operation!!
[2019/10/31 14:50:29] [0.633][mmc]: SD/MMC 2 init OK!!!
[2019/10/31 14:50:29] [0.638][mmc]: Best spd md: 4-HS400, freq: 2-50000000
[2019/10/31 14:50:29] [0.643][mmc]: Bus width 8
[2019/10/31 14:50:29] [0.643][mmc]: EOL Info(Rev blks): Normal
[2019/10/31 14:50:29] [0.646][mmc]: Wear out(type A): [0.649][mmc]: 10%-20% life time used
[2019/10/31 14:50:29] [0.652][mmc]: Wear out(type B): [0.655][mmc]: 10%-20% life time used
[2019/10/31 14:50:29] [0.658][mmc]: End mmc_init_boot
[2019/10/31 14:50:29] [0.661]sunxi flash init ok
[2019/10/31 14:50:29] used mbr [0], count = 7
[2019/10/31 14:50:29] [0.666]hdmi hdcp not enable!
```

图 3-3: uboot-eMMC 初始化成功

eMMC 等设备初始化成功的 log。

**Kernel 阶段**

```
[    3.286010] sunxi-mmc sdc2: sdc set ios:clk 400000Hz bm PP pm ON vdd 22 width 8 timing MMC-HS200 dt B
[    3.286407] sunxi-mmc sdc2: sdc set ios:clk 100000000Hz bm PP pm ON vdd 22 width 8 timing MMC-HS200 dt B
[    3.286645] sunxi-mmc sdc2: sdc set ios:clk 100000000Hz bm PP pm ON vdd 22 width 8 timing MMC-HS(SDR20) dt B
[    3.286750] sunxi-mmc sdc2: sdc set ios:clk 52000000Hz bm PP pm ON vdd 22 width 8 timing MMC-HS(SDR20) dt B
[    3.286935] sunxi-mmc sdc2: sdc set ios:clk 50000000Hz bm PP pm ON vdd 22 width 8 timing MMC-HS400 dt B
[    3.287040] sunxi-mmc sdc2: sdc set ios:clk 100000000Hz bm PP pm ON vdd 22 width 8 timing MMC-HS400 dt B
[    3.287244] mmc0: new HS400 MMC card at address 0001
[    3.291428] mmcblk0: mmc0:0001 ISOCOM 28.9 GiB
[    3.295032] mmcblk0boot0: mmc0:0001 ISOCOM partition 1 4.00 MiB
[    3.298597] mmcblk0boot1: mmc0:0001 ISOCOM partition 2 4.00 MiB
[    3.302159] mmcblk0rpmb: mmc0:0001 ISOCOM partition 3 4.00 MiB
[    3.305742] ALSA device list:
[    3.305745]   #0: sun50iw10-codec
[    3.329256]  mmcblk0: p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12 p13 p14 p15 p16 p17
```

图 3-4: kernel-eMMC 初始化成功

sd/eMMC/sdio 等设备初始化成功的 log，能正常出现检测到新卡和相应地址以及容量的 log，以上是 eMMC 的检测成功的 log，具体情况参见上图。具体 sd/sdio log 类似。

当设备正常上电启动后，如果发现 sd/eMMC/sdio 初始化失败，可能存在以下问题：

## 3.4.1 ACMD41、CMD1 Card ready 超时

**问题现象**

常见打印（boot）

wait for mmc init failed

wait for card init failed

问题原因：

1、样片问题：

解决步骤：

a. 直接找原厂硬件工程师协助解决。

b. 或者，手动修改驱动代码，把等待超时设置长，例如设置成 30s，如果任然有问题找厂家解决

如何把超时设置时间改成改长，步骤如下：进入 boo0/uboot 代码里面的 mmc_send_op_cond 函数，更改 timeout 值

c. 直接使用逻辑分析仪捉取相应 cmd 的响应是否一直未 ready。确认设备回复未 ready，找 eMMC 原厂厂家解决。

问题原因：

2、串口倒灌电

解决步骤：

a. 拔掉 RX 信号线和电源线（TX 可以正常连接）

## 3.4.2 Pin 相关问题

**问题现象**

常见错误是在传输的时候出现报错 RCE 或者 RE，即命令对方回应命令出错。

**问题分析**

1、Pin 没有设置

检测 sys_config.fex，看看设置时候是否和 datasheet 对应。

2、电源问题

检测 pin 的 io 电压是否有电压，电压是否正常，电路图是否连接对了。

sys_config.fex 对应的电源选项是否设置对了。

3、频率过高

检测寄存器，或者看波形，确定频率是否设置过高。

## 3.4.3 RE

**问题现象**

eMMC RE 问题目前有规律可循，能有共同规律的案例基本都处于初始化阶段，cmd1 命令时，这里如果出现了 cmd1 RE 就可以按照以下信息进行分析。

**问题分析**

（1）量取初始化阶段，vcc-emmc 和 vcc-emmc-io 这两路点。确定是否存在波动干扰，或者电压异常。

（2）如果存在电压异常, 并且电压切换（如 1.8V->3.3V），主要存在以下两个可能：

1）软件配置的 vcc-emmc 和 vcc-emmc-io 存在和硬件设计不匹配。当使能电资源时，电压从默认的硬件电压错误配置为软件配置电压。sdcx 中的 vmmc 和 vqmm 是否和实际硬件设计匹配，vmmc：特指 eMMC 的供电电压，一般为 3.3V。vqmm 特指 pin 的 io 电压，在全志一般平台特指 vcc-pc。

2）存在模块在 mmc 模块初始化获取电源资源之前，对 mmc 使用到的电源 vcc-emmc/vcc-emmc-io 进行开关，导致 eMMC 存在灌电导致 eMMC 异常。这种得根据 dts 等确认哪些模块进行了该电源配置，进行实验确认影响 eMMC 供电的模块，根据实际问题进行分析。

## 3.4.4 STO

参考 STO。

## 3.4.5 RTO（response timeout）

**问题现象一：**

出现 RTO 错误, 并且在 kernel 阶段。

```
[    2.832151] sunxi-mmc sdc2: drivers/mmc/host/sunxi-mmc.c 1263 sunxi_mmc_set_ios ios->power_mode:2
[    2.832164] sunxi-mmc sdc2: drivers/mmc/host/sunxi-mmc.c 1491 sunxi_mmc_set_ios
[    2.832169] sunxi-mmc sdc2: drivers/mmc/host/sunxi-mmc-v4p10x.c 323 sunxi_mmc_clk_set_rate_for_sdmmc_v
[    2.839206] sunxi-mmc sdc2: smc 0 p2 err, cmd 8, RTO !!
```

图 3-5: RTO 错误

**问题分析**

1. 出现 RTO 错误：并且在 sun8iw5、sun8iw6、sun8iw7 或者 sun8iw8 等平台，使用的是 sdc2。如果是进行步骤 2，否则进行步骤 5。
2. 打上如下补丁，确定能否规避此问题，如不能规避进行步骤 5。

```
--- a/drivers/mmc/host/sunxi-mmc.c
+++ b/drivers/mmc/host/sunxi-mmc.c
@@ -2447,7 +2447,11 @@ static int sunxi_mmc_probe(struct platform_device *pdev)
                mmc->max_busy_timeout = SUNXI_DEF_MAX_R1B_TIMEOUT_MS;
                mmc->caps |= MMC_CAP_WAIT_WHILE_BUSY;
                dev_info(&pdev->dev, "set host busy\n");
-       }
+       } else {
+       /*h3-sdmmc-host can't get the busy signal from card*/
+               sunxi_mmc_ops.card_busy = NULL;
+       }
+
```

图 3-6: 规避 RTO 的补丁

3. 出现 RTO 错误：并且在 sun8iw16、sun8iw19、sun50iw9、sun50iw10 等平台，使用的是 Sdc0，如果是进行步骤 4，否则进行步骤 5，
4. 识别过程的 CMD 本来可以正常接收发送，进行过 CMD11 命令发送后开始出现 RTO 错误，则有可能是电压切换时对应的 PIN 没有切换电压正确造成的。
5. 使用逻辑分析仪，捉取出现问题时的波形。

**问题现象二**：在其他阶段出现的 RTO 报错，且经过软件和硬件排查未查找到原因

**问题分析：**

将 eMMC 的 rst 引脚接上拉电阻看能否解决问题

## 3.4.6 Sd 概率性识别不到

**问题现象**

插入/拔出 sd 卡后，没有出现 sd 检测和 sd 移除的 log 出现。

**问题分析**

1. 测量插入和拔出 sd 卡时的 cd 脚上的电压是否有变化，无变化请确定 sd 卡座是否正确连接，是否存在虚焊短接等问题。否则进行步骤 2。
2. （echo 1 > /sys/devices/platform/soc/sdc0/sunxi_insert），看是否能正确检测到 sd 卡。
3. 修改成轮询，查看是否会存在丢卡。
4. 轮询/手动扫卡正常，则证明 sd 卡 cd 脚中断没采集到，使用下面的解决方法。

**解决方法：**

1. 改成轮询扫卡，sd 卡节点添加如下属性:

   sys_config.fex 添加 broken-cd= ，如下图：

```
925 [sdc0]
926 sdc0_used               = 1
927 bus-width               = 4
928 sdc0_d1                 = port:PF00<2><1><2><default>
929 sdc0_d0                 = port:PF01<2><1><2><default>
930 sdc0_clk                = port:PF02<2><1><2><default>
931 sdc0_cmd                = port:PF03<2><1><2><default>
932 sdc0_d3                 = port:PF04<2><1><2><default>
933 sdc0_d2                 = port:PF05<2><1><2><default>
934 sd-uhs-sdr50            =
935 sd-uhs-ddr50            =
936 sd-uhs-sdr104           =
937 broken-cd               =
```

图 3-7: 轮询扫卡 sys_config

   board.dts 添加 broke-cd，如下图：

```
sdc0: sdmmc@04020000 {
        broken-cd;
        bus-width = <4>;
        cd-gpios = <&pio PF 6 6 1 3 0xffffffff>;
        /*non-removable;*/
        /*cd-inverted*/
        /*data3-detect;*/
        /*card_pwr_gpios = <&pio PH 14 1 1 2 0xffffffff>;*/
        cd-used-24M;
        cap-sd-highspeed;
        sd-uhs-sdr50;
        sd-uhs-ddr50;
```

图 3-8: 轮询扫卡 dts

2. 确定环境是否打上了如下补丁，如果没有，请打上如下补丁再在 sys_config.fex 或者 dts 上添加 cd-used-24M，提高中断脚的检测时钟为 24M，否则只能使用步骤 1 进行解决。

   补丁：

```
commit ef587a456d472bcf8c5c61cdf88cb635492dd23b
Author: [redacted]
Date:   Fri Nov 15 10:39:22 2019 +0800

    sdmmc:all platform:add card detect freq switch to 24MHz config
    cd-used-24M

    Change-Id: I75947f2568693fbeff628282e6f36eafdabf09b5
```

图 3-9: 24M-cd-gpio

## 3.4.7 SD 卡导致系统卡死

如果是用的 linux4.9 内核，确定环境是否打上了如下两个补丁：


图 3-10: broken handle

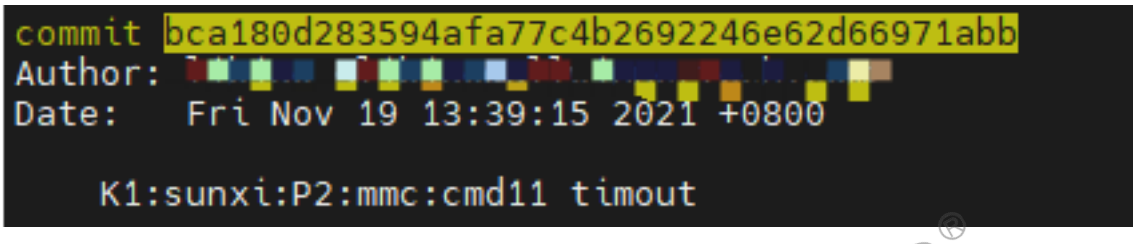


图 3-11: cmd11 timeout

如果已有补丁，则在对应项目的 dts 将 ctl-spec-caps 改为 0x608，max-busy-timeout 改为 5000。

```
sdc0: sdmmc@04020000 {
        pinctrl-0 = <&sdc0_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 = <0x608>;
        max-busy-timeout = <5000>;
        vmmc-supply = <&reg_dldo1>;
        vqmmc33sw-supply = <&reg_dldo1>;
        vdmmc33sw-supply = <&reg_dldo1>;
        vqmmc18sw-supply = <&reg_aldo1>;
        vdmmc18sw-supply = <&reg_aldo1>;
        status = "okay";
};
```

图 3-12: dts

## 3.5 sd/eMMC/sdio 卡死分析

问题的方法通用适用于 sd/eMMC/sdio 等情形。

**问题现象**

sd 卡在某次插拔之后，反复插拔也无任何反应，并且不再带有 sd 的识别的正常 log 输出。

**问题分析**

1、确定卡死的线程现场通过 echo t > /proc/sysrq-trigger，确定是否存在 mmc 相关的线程出现超时调度现象。

```
[ 3153.827565] Showing busy workqueues and worker pools:
[ 3153.833200] workqueue events_freezable: flags=0x4
[ 3153.838444]   pwq 6: cpus=3 node=0 flags=0x0 nice=0 active=2/256
[ 3153.845158]     in-flight: 459:mmc_rescan mmc_rescan
[ 3153.850761] pool 6: cpus=3 node=0 flags=0x0 nice=0 hung=8s workers=3 idle: 741 25
```

图 3-13: sysrq 堆栈

2、确认了卡死线程状态后，通过 sd 寄存器节点，dump 出 sd 的寄存器 cat /sys/devices/platform/soc/sdc0/sunxi_dump_host_register

确认当前 sd 当前的动作，

```
0xffffff8009e30000 : 20000010 00010000 ffffffff 00000000
0xffffff8009e30010 : 00000200 00000200 1000014b 00000000
0xffffff8009e30020 : 00000320 00000000 00000000 00000000
0xffffff8009e30030 : 0000bfc6 00000000 00010000 00005d06
0xffffff8009e30040 : 200700f8 00000000 00000000 00000000
0xffffff8009e30050 : 00000001 00000022 00000000 81710000
0xffffff8009e30060 : 00000088 00000000 00000000 00000000
0xffffff8009e30070 : 00000000 00000000 00000001 00000000
0xffffff8009e30080 : 00000200 1e011000 00000000 00000000
0xffffff8009e30090 : 40000000 40000000 00000000 00000000
0xffffff8009e300a0 : 00000000 00000000 00000000 00000000
0xffffff8009e300b0 : 00000000 00000000 00000000 00000000
0xffffff8009e300c0 : 00000000 00000000 00000000 00000000
0xffffff8009e300d0 : 00000000 00000000 00000000 00000000
0xffffff8009e300e0 : 00000000 00000000 00000000 00000000
0xffffff8009e300f0 : 00000000 00000000 00000000 00000000
0xffffff8009e30100 : 00000000 00000000 00000000 00000000
0xffffff8009e30110 : 0000005e 00000000 00000000 00000000
0xffffff8009e30120 : 00000000 00000000 00000000 00000000
0xffffff8009e30130 : 00000000 00000000 00000000 00000000
0xffffff8009e30140 : 00010000 00002000 00002000 00000000
```

图 3-14: mmc 寄存器

3、确认当前偏移 0x18 的寄存器的 5：0bit 的值，如当前为 cmd11。

4、根据当前寄存器分析卡死的原因。

5、捕获卡死现场的当前 cmd 的，clk,cmd，dat0 线的波形。

**解决方法**

这类问题比较复杂，所以只需要向模块负责人提供上面分析需要的信息。

## 3.5.1 插 sd 卡启动不识别

**问题现象**

样机设备插入 sd 卡的情况下启动，系统无法识别到 sd 卡

**问题分析**

1. 启动后插拔 sd 卡是否识别，重新插拔能识别进行步骤 2，不能识别进行步骤 3
2. 则使用示波器跟踪启动过程中，sd 卡相关的电源是否正常。如果启动过程，sd 卡的供电不正常，则由硬件主导问题查询。
3. 确定 sd 卡启动识别不到是否是个别行为，如果是个别行为，windows 上进行 sd 卡操作，确定 sd 卡是否损坏。

**解决方法：**

1. 确定硬件上的供电异常问题，并且解决。
2. 更新确定电路图是否有如下 sd 卡供电开关，如果不存在，只能使用步骤 1 进行解决。
3. 确定环境是否打上了如下补丁，如果没有，请更新到该补丁，并且配置上如下开关属性（具体电源管脚参考电路图），否则只能使用步骤 1 进行解决。

   电源开关电路：

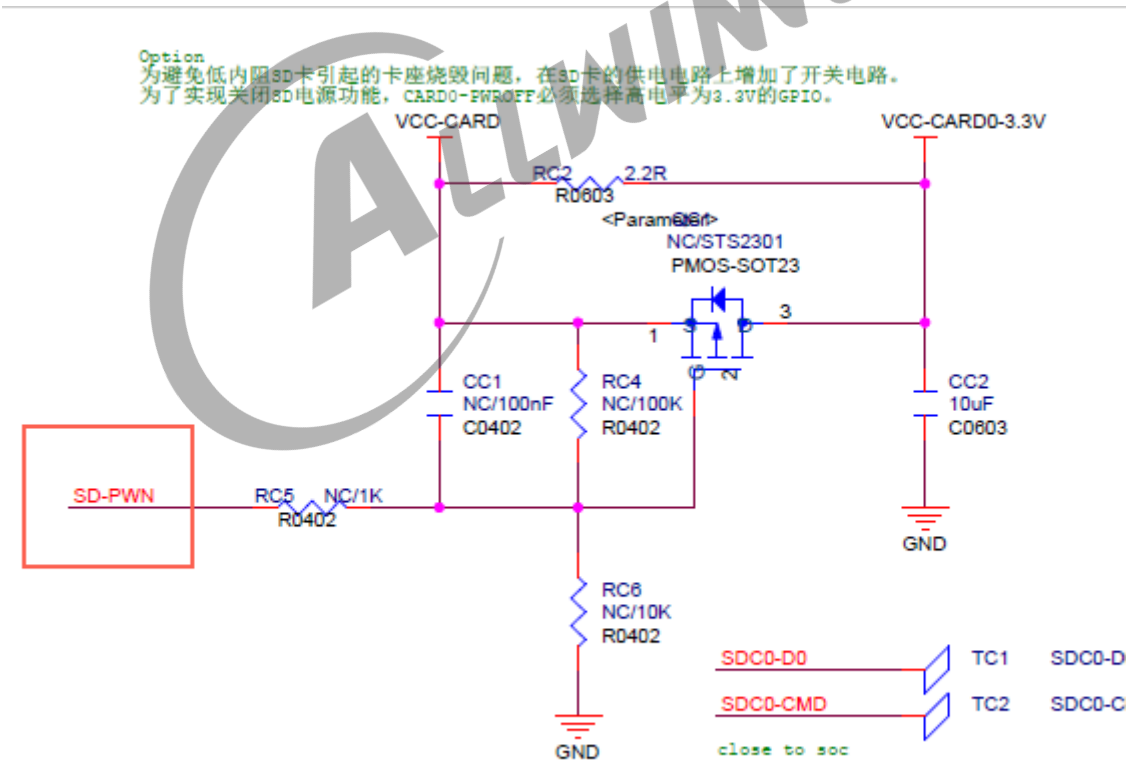


图 3-15: sd 卡电源开关

补丁：

```
commit ab9c3d15f5a9ae4cdfcbf0484f21980dae16e423
Author: [redacted]
Date:   Tue Oct 22 09:37:10 2019 +0800

    sdmmc:all platform:add delay to ensure voltage stability

    Change-Id: Id0de8a7c527ebc9b6e77fd6069d065354e8b5d99

commit f401c0247b10415017783c3cfb40989baa6a8e98
Author: [redacted]
Date:   Tue Sep 24 16:01:08 2019 +0800

    sdmmc:sun8iw19p1:power-off before power-up and break sleep_pin into sleep and uart_jtag

    Change-Id: I78d4489630863a5d9f07aa5d7edb1577a4376c17

commit 89dd0ed05f78a9d8bc8c9d6c978e52231721901a
Author: [redacted]
Date:   Fri Sep 20 17:34:48 2019 +0800

    sdmmmc:sun8iw19p1:add the min-frequency getting from dts

    Change-Id: I453effb6b993698653c03828aa5545026d4113ac
```

图 3-16: 电源开关补丁

配置：

```
->-------->--------card-pwr-gpios = <&pio PC 4 1 1 2 0xffffffff>;
```

图 3-17: 电源开关 gpio 配置

# 3.6 sd/eMMC/sdio 运行过程中读写错误

当设备正常上电启动后并且初始化完成了，但是出现如下常见错误：

## 3.6.1 写出错

**问题现象：**

出现如下 cmd25 的错误：

```
card erase all
[6.783]successed in erasing flash
[mmc]: mmc 2 data timeout 80
[mmc]: mmc 2 close bus gating and reset
[mmc]: mmc 2 mmc cmd 25 err 0x00000080
mbr write fail
sunxi_flash - sunxi_flash sub-system
```

图 3-18: cmd25 错误

**解决方法：**

**情况一：eMMC bias 设置**

由于历史原因，设置 bias 的方式有几种，这里为了保证设置正确，排查问题的时候，几种方式都设置

先确定 eMMC 对应 io 口是哪个，一般是 pc。确认硬件电路上 eMMC 对应 io 供电是多少，一般是 1.8v

涉及 uboot 和内核配置的地方，需要都配置上

（1）uboot2014:

检测 bias 配置是否和对应 io 电压一致，例如 pc 口 io 电压为 1.8V，sys_config.fex 设置如下

```
;------------------------------------------------------------------------------
;
; normal config: eg. px_bias = "pmu_name:supply_name:voltage"
;
;                   pmu_name = axp809, axp806
;                   supply_name = dcdc1, dcdc2, aldo1, gpio1ldo, etc...
;------------------------------------------------------------------------------
[gpio_bias]
pc_bias         = "axp803:eldo1:1800"
pg_bias         = "axp803:gpio0:1800"
```

图 3-19: bias 配置

（2）uboot2018:

1. 首先，需要根据要 io 使用的电压设置 uboot.dts 中 bias 的值，例如此处用 pc 口则设置 pc_bias，如果 io 电压为 3.3V，则设置 pc_bias = <3300>，如果 io 电压为 1.8V，则设置 pc_bias = <1800>，如果用到 pf 口则设置 pf_bias。

然后继续设置步骤 2 和 3，

```
gpio_bias@4 {
        /*
         * Avoid dtc compiling warnings.
         * @TODO: Developer should modify this to the actual value
         */
        reg = <0x0 0x4 0x0 0x0>;
        device_type = "gpio_bias";
        pc_bias = <1800>;
};
```

图 3-20: pc_bias 配置

2. 如果对应 io 电压为 3.3V，则设置如下图所示。如果 io 电压为 1.8V，则应该在 card2_boot_para@3 节点添加 sdc_io_1v8 = <0x1>。（如果是 sdc0 接 eMMC，则对应修改 card0_boot_para）

```
>-------card2_boot_para@3 {
>-------->-------/*
>-------->------- * Avoid dtc compiling warnings.
>-------->------- * @TODO: Developer should modify this to the actual value
>-------->------- */
>-------->-------reg = <0x0 0x3 0x0 0x0>;
>-------->-------device_type = "card2_boot_para";
>-------->-------card_ctrl = <0x2>;
>-------->-------card_high_speed = <0x1>;
>-------->-------card_line = <0x8>;
>-------->-------pinctrl-0 = <&card2_pins_a>;
>-------->-------/*sdc_ex_dly_used = <0x2>;*/
>-------->-------/*sdc_io_1v8 = <0x0>;*/
>-------->-------/*sdc_type = "tm4";*/
>-------};
```

图 3-21: uboot2018-bias

具体参考 sd/eMMC/sdio 模组供电检查

3. 如果 io 口是 pc，通过 “pc_bias” 关键字来设置 bias。如果 io 电压为 3.3V，设置 “pc_bias = <0>”。如果 io 电压为 1.8V，则设置 pc_bias = <1800>。

如果 io 不是 pc，这里不用设置。

```
card2_boot_para@3 {
        /*
         * Avoid dtc compiling warnings.
         * @TODO: Developer should modify this to the actual value
         */
        reg = <0x0 0x3 0x0 0x0>;
        device_type = "card2_boot_para";
        card_ctrl = <0x2>;
        card_high_speed = <0x1>;
        card_line = <0x8>;
        pinctrl-0 = <&card2_pins_a &card2_pins_b>;
        sdc_ex_dly_used = <0x2>;
        pc_bias = <1800>;
        sdc_tm4_hs200_max_freq = <200>;
        sdc_tm4_hs400_max_freq = <150>;
        sdc_tm4_win_th = <0x08>;
        /*tm4_tune_r_cycle = <1>;*/
        sdc_type = "tm5";
};
```

图 3-22: uboot2018-bias

4.linux5.4 以上的版本内核:

内核核进行耐压模式的配置，在 board.dts 的 pio 节点增加你对应使用引脚的耐压模式配置 vcc-px-supply = <&reg_piox_x>，如下所示：

```
&pio {
    vcc-pg-supply = <&reg_pio1_8>;
    vcc-pc-supply = <&reg_pio1_8>;
    vcc-pf-supply = <&reg_pio1_8>;
    vcc-pfo-supply = <&reg_pio3_3>;

  ........
```

vcc-px-supply 则对应你要配置的 gpio 口 px 的耐压值，后面的 reg_piox_x 则需要根据实际的硬件方案确定该口的耐压值，如 PC 口，如果你的 VCC-PC 是 1.8V 供电，则配置为 vcc-pc-supply = <&reg_pio1_8>，如果是 3.3V 供电则配置为 vcc-pc-supply = <&reg_pio3_3>

注意，上述的 PF 口比较特殊，因为存在切换电压的过程，所以会多一个 vcc-pfo-supply 的关键字。对于 PF 口配置耐压模式的配置不用修改，始终如下配置：

```
  vcc-pf-supply = <&reg_pio1_8>;
  vcc-pfo-supply = <&reg_pio3_3>;
```

PF 口的电压切换会根据使用引脚的 dts 配置 pin 的 power-source(如下) 来决定，如果是 3.3V 就配置 power-source 等于 3300,1.8V 则为 1800。

pin 的 power-source 软件会根据使用的 sd 速度模式自动切换使用，正常情况下这个不需要额外配置。

```
sdc0_pins_a: sdc0@0 {
                    pins = "PF0", "PF1", "PF2",
                           "PF3", "PF4", "PF5";
                    function = "sdc0";
                    drive-strength = <40>;
                    bias-pull-up;
            ----->    power-source = <3300>;
              };

sdc0_pins_b: sdc0@1 {
                    pins = "PF0", "PF1", "PF2",
                           "PF3", "PF4", "PF5";
                    function = "sdc0";
                    drive-strength = <40>;
                    bias-pull-up;
             ----->   power-source = <1800>;
              };
```

**情况二：sdio bias 设置**

注：由于历史原因，设置 bias 的方式有几种，这里为了保证设置正确，排查问题的时候，内核和 uboot 都要设置 bias

涉及 uboot 和内核配置的地方，需要都配置上

（1）uboot2014:

检测 bias 配置是否和对应 io 电压一致，例如 pg 口 io 电压为 1.8V，sys_config.fex 设置如下

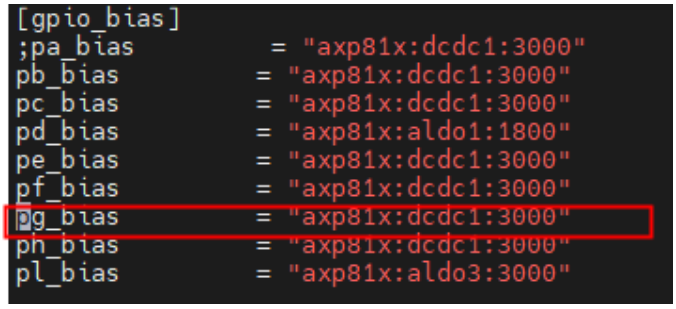


图 3-23: bias 配置

（2）uboot2018:

根据要 io 使用的电压设置 uboot.dts 中 bias 的值，例如此处用 pg 口则设置 pg_bias，如果 io 电压为 3.3V，则设置 pg_bias = <3300>。如果 io 电压为 1.8V，则设置 pg_bias = <1800>，如果用到 pf 口则设置 pf_bias。

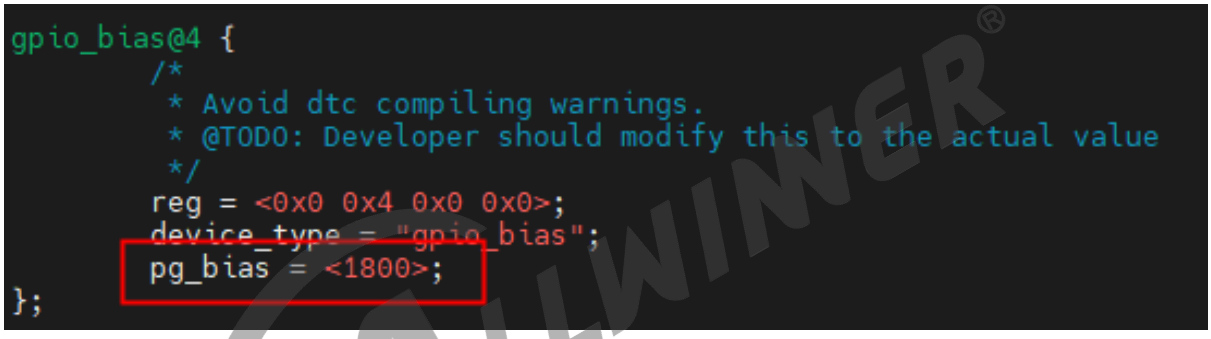


图 3-24: pc_bias 配置

（3）kernel-4.9:

如果 io 电压是 1.8 加上 sdio-used-1v8 以设置 1.8v bias，io 电压是 3.3v 就不用加上 sdio-used-1v8。

（4）linux5.4 以上的版本:

内核核进行耐压模式的配置，在 board.dts 的 pio 节点增加你对应使用引脚的耐压模式配置 vcc-px-supply = <&reg_piox_x>，，如下所示：

```
&pio {
    vcc-pg-supply = <&reg_pio1_8>;
    vcc-pc-supply = <&reg_pio1_8>;
    vcc-pf-supply = <&reg_pio1_8>;
    vcc-pfo-supply = <&reg_pio3_3>;

    ........
```

vcc-px-supply 则对应你要配置的 gpio 口 px 的耐压值，后面的 reg_piox_x 则需要根据实际的硬件方案确定该口的耐压值，如 PC 口，如果你的 VCC-PC 是 1.8V 供电，则配置为 vcc-pc-supply = <&reg_pio1_8>，如果是 3.3V 供电则配置为 vcc-pc-supply = <&reg_pio3_3>

注意，上述的 PF 口比较特殊，因为存在切换电压的过程，所以会多一个 vcc-pfo-supply 的关键字。对于 PF 口配置耐压模式的配置不用修改，始终如下配置：

```
vcc-pf-supply = <&reg_pio1_8>;
vcc-pfo-supply = <&reg_pio3_3>;
```

PF 口的电压切换会根据使用引脚的 dts 配置 pin 的 power-source(如下) 来决定，如果是 3.3V 就配置 power-source 等于 3300,1.8V 则为 1800。

pin 的 power-source 软件会根据使用的 sd 速度模式自动切换使用，正常情况下这个不需要额外配置。

```
sdc0_pins_a: sdc0@0 {
                pins = "PF0", "PF1", "PF2",
                       "PF3", "PF4", "PF5";
                function = "sdc0";
                drive-strength = <40>;
                bias-pull-up;
          ----->    power-source = <3300>;
            };

sdc0_pins_b: sdc0@1 {
                pins = "PF0", "PF1", "PF2",
                       "PF3", "PF4", "PF5";
                function = "sdc0";
                drive-strength = <40>;
                bias-pull-up;
           ----->   power-source = <1800>;
            };
```

**方法二：IO 驱动能力**

提高或者降低驱动能力，以卡 2 的内核设置为例，修改图中红色的地方，范围是 0-3

```
992
993 [sdc2]
994 sdc2_used                 = 1
995 non-removable             =
996 bus-width                 = 8
997 sdc2_ds                   = port:PC01<3><1><3><default>
998 sdc2_clk                  = port:PC04<3><1><3><default>
999 sdc2_cmd                  = port:PC05<3><1><3><default>
000 sdc2_d0                   = port:PC06<3><1><3><default>
001 sdc2_d1                   = port:PC07<3><1><3><default>
002 sdc2_d2                   = port:PC08<3><1><3><default>
003 sdc2_d3                   = port:PC09<3><1><3><default>
004 sdc2_d4                   = port:PC10<3><1><3><default>
005 sdc2_d5                   = port:PC11<3><1><3><default>
006 sdc2_d6                   = port:PC12<3><1><3><default>
007 sdc2_d7                   = port:PC13<3><1><3><default>
008 ;warn becasue fpga not connect rst,so not set it
009 sdc2_emmc_rst             = port:PC14<3><1><3><default>
010 cd-gpios                  =
011 sunxi-power-save-mode     =
012 sunxi-dis-signal-vol-sw =
```

图 3-25: 驱动能力配置

**方法三：输出相位**

卡 2 即是 sdc2:

```
188     int frq_index = 0;
189     u32 cmd_drv_ph = 1;
190     u32 dat_drv_ph = 0;
191     u32 sam_dly = 0;
192     u32 ds_dly = 0;
193     struct sunxi_mmc_clk_dly *mmc_clk_dly =
194         ((struct sunxi_mmc_ver_priv *)host->version_priv_dat)->mmc_clk_dly;
195
196     if (!mmc->parent || !mmc->parent->of_node) {
197         dev_err(mmc_dev(host->mmc),
198             "no dts to parse clk dly,use default\n");
199         return;
200     }
201
202     np = mmc->parent->of_node;
203
204     switch (timing) {
205     case MMC_TIMING_LEGACY:
206     case MMC_TIMING_UHS_SDR12:
207         speed_mod = SM0_DS26_SDR12;
208         break;
209     case MMC_TIMING_MMC_HS:
210     case MMC_TIMING_SD_HS:
211     case MMC_TIMING_UHS_SDR25:
212         speed_mod = SM1_HSSDR52_SDR25;
213         break;
214     case MMC_TIMING_UHS_DDR50:
215         speed_mod = SM2_HSDDR52_DDR50;
216         dat_drv_ph = 1;
217         break;
218     case MMC_TIMING_UHS_SDR50:
219     case MMC_TIMING_UHS_SDR104:
220     case MMC_TIMING_MMC_HS200:
221         speed_mod = SM3_HS200_SDR104;
222         break;
223     case MMC_TIMING_MMC_HS400:
224         speed_mod = SM4_HS400;
 40% drivers/mmc/host/sunxi-mmc-v4p5x.c
```

图 3-26: 输出相位配置

卡 0/卡 1，即是 sdc0 和 sdc1:

卡 0,1 可以改 dtsi，以 150MHz 的频点为例，其他频率点可以类推

```
/*sunxi-dly-400k = <1 0 0 0 0 0>; */
/*sunxi-dly-26M  = <1 0 0 0 0 0>;*/
/*sunxi-dly-52M  = <1 0 0 0 0 0>;*/
sunxi-dly-52M-ddr4  = <1 0 0 0 0 2>;
/*sunxi-dly-52M-ddr8  = <1 0 0 0 0 0>;*/
sunxi-dly-104M  = <1 1 0 0 0 1>;
sunxi-dly-208M  = <1 1 0 0 0 0>;
/*sunxi-dly-104M-ddr  = <1 0 0 0 0 0>;*/
/*sunxi-dly-208M-ddr  = <1 0 0 0 0 0>;*/
```

图 3-27: sdc0/sdc1 输出相位配置

sunxi-dly-208M=<1(cmd driver phase) 1(data driver phase) 0 0 0(data sample phase) 0 (cmd sample phase)>

cmd 错就修改 cmd driver phase，数据错就修改 data driver phase

最佳输出相位的配置方法如下表所示：

| 速度模式 | cmd driver phase | data driver phase |
|---|---|---|
| SDR | 1 | 1 |
| DDR8 | 1 | 1 |
| DDR4 | 1 | 0 |
| DDR4(2X) | 1 | 1 |
| HS400 | 1 | 0 |
| HS400(2X) | 1 | 1 |

**方法四：retry**

> 如果上述都不行，可以考虑使用 retry 的方式解决，这个可以通过 sys_config.fex 来使能

在 sys_config.fex/board.dts 的 sdcX 位置中：加上 “ctl-sepc-cap =”

```
ctl-spec-caps = <0x8>;
```

图 3-28: retry 配置

**方法五：将频降模式**

卡 0/卡 1, 即 sdc0/sdc1

```
sdc0: sdmmc@04020000 {
        pinctrl-0 = <&sdc0_pins_a>;
        bus-width = <4>;
        cd-gpios = <&pio PF 6 0 1 3 0xffffffff>;
        card-pwr-gpios = <&pio PC 4 1 1 2 0xffffffff>;
        sunxi-power-save-mode;
        sunxi-signal-vol-sw-without-pmu;
        sd-uhs-sdr50;
        sd-uhs-ddr50;
        sd-uhs-sdr104;
        /*sunxi-dis-signal-vol-sw;*/
        max-frequency = <150000000>;
        min-frequency = <100000>;
        cd-used-24M;
        no-sdio;
        /*broken-cd;*/
        ctl-spec-caps = <0x8>;
        status = "okay";
};
```

图 3-29: 降频降模式

1. 把 sd-uhs-sdr50,sd-uhs-ddr50,sd-uhs-sdr104 等属性逐级去除，并且可以吧 max-frequency 属性在 [150000000,100000000,50000000,25000000] 范围, 逐级降低。
2. 如果降频降模式可行，检查时钟配置。

3. 请检查硬件。

   卡 2，即 sdc2

```
sdc2: sdmmc@04022000 {
        non-removable;
        bus-width = <8>;
        mmc-ddr-1_8v;
        mmc-hs200-1_8v;
        mmc-hs400-1_8v;
        no-sdio;
        no-sd;
        cap-mmc-highspeed;
        sunxi-power-save-mode;
        sunxi-dis-signal-vol-sw;
        max-frequency = <100000000>;
        vmmc-supply = <&reg_dcdc1>;
        /*emmc io vol 3.3v*/
        vqmmc-supply = <&reg_aldo1>;
        /*emmc io vol 1.8v*/
        /*vqmmc-supply = <&reg_eldo1>;*/
        status = "disabled";
    };
```

图 3-30: sdc2 降频降模式

1. 把 mmc-ddr-1_8v,mmc-hs200-1_8v,mmc-hs400-1_8v 等属性逐级去除。并且可以吧 max-frequency 属性在 [150000000,100000000,50000000,25000000] 范围, 逐级降低。
2. 如果降频降模式可行，请检查时钟配置。
3. 检查硬件。

## 3.6.2 读出错

**问题现象**

出现 cmd18 的错误。

**问题分析**

卡 2, 即 sdc2

**方法一：**

1. 确定 card2_boot_ param sys_config.fex 上，是否配置如下

```
sdc_ex_dly_used = 2
```

图 3-31: 自动采样

1) 如果不是，平台是非软件自动采样点 ic，请进行下面修改

这里以 50MHz 左右的频率点为例，找到对应的 dtsi，如下图所示

```
/*-- speed mode --*/
/*sm0: DS26_SDR12*/
/*sm1: HSSDR52_SDR25*/
/*sm2: HSDDR52_DDR50*/
/*sm3: HS200_SDR104*/
/*sm4: HS400*/
/*-- frequency point --
/*f0: CLK_400K*/
/*f1: CLK_25M*/
/*f2: CLK_50M*/
/*f3: CLK_100M*/
/*f4: CLK_150M*/
/*f5: CLK_200M*/
sunxi-dly-52M  = <1 0 0 0 1>;

sdc_tm4_sm0_freq0 = <0>;
sdc_tm4_sm0_freq1 = <0>;
sdc_tm4_sm1_freq0 = <0x00000000>;
sdc_tm4_sm1_freq1 = <0>;
sdc_tm4_sm2_freq0 = <0x00000000>;
sdc_tm4_sm2_freq1 = <0>;
sdc_tm4_sm3_freq0 = <0x05000000>;
sdc_tm4_sm3_freq1 = <0x00000005>;
sdc_tm4_sm4_freq0 = <0x00050000>;
sdc_tm4_sm4_freq1 = <0x00000004>;

/*vmmc-supply = <&reg_3p3v>;*/
/*vqmc-supply = <&reg_3p3v>;*/
```

图 3-32: 非自适应采样

sunxi-dly-52M=<1(driver phase) 0 0 0 1(sample phase)> (根据不同平台的寄存器 spec 去调式，并不是固定位置和意义不变的)

要修改的是 sample phase，值的范围 0-2（0:90 度，1:180 度，2:270 度）改了之后再看

**方法二：**

cmd25_method_1\cmd25_method_4\cmd25_method_5。

卡 0/1，即 sdc0/sdc1

参考 cmd25_method_1\cmd25_method_3\cmd25_method_4\cmd25_method_5。

## 3.6.3 DTO(data timeout)

**问题现象：**

对于打印 err 0x00000200 表示 Data timeout，或者直接打印 DTO

**问题排查：**

（1）数据线不导通，或者 eMMC/sd 卡数据线有问题，有某根数据线无输出

这个一般只会出现在卡 0 和卡 1，这种情况可以尝试改成 1 线模式，看看是不是正常，如果是正常的话，很可能就是这个原因，后面再找硬件的同事抓波形，就可以确定了

例 1：问题单 #47122 曾出现此问题，由于卡座不兼容导致部分 tf 卡无法识别。无法识别的卡报 DTO，硬件同事抓波形发现 D1 无数据输出。遇到此类现象，可以考虑硬件更换卡座。

例 2：问题单 #30323 曾出现此问题，eMMC 本身损坏，导致只支持 1 线模式，4 线情况下识别不到，现象是 T3 3.10 内核能识别，T5 4.9 内核 mmc 子系统不支持容错处理，所以 4 线情况下识别不到。

如何设置成 1 线模式

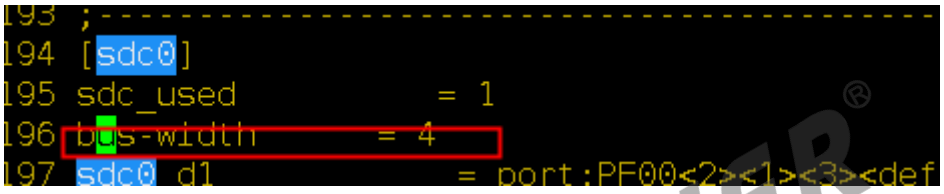


图 3-33: 线宽配置

(2) eMMC 读写时，eMMC 信号电压拉低。

在 sun50iw1p1 平台上出现过，reboot 启动时，eMMC 出现 dto 问题，并且最后查明原因是，eMMC 在读写时，eMMC 的 io 信号电压给拉低了（io 电压为 1.8, 给拉低到了 1.5V 左右），刚好在 eMMC 低电压检测范围徘徊。导致 eMMC 不响应，并且查明是由于 codec 模块影响导致，所以出现 dto 问题，需先检测，eMMC 使用过程中，动态的 io 和供电电压是否正常。具体检测步骤，参考如下：eMMC 供电检查

(3)sdio 与 wifi 设备交互读写时，wifi 里面的 fw 未加载

如果上述（1）（2）的连接和电压问题都排查过了没问题，并且该问题出现在 sdio 上面，则导致 DTO 的原因可能是 wifi device 的 fw 没有被正常加载，导致设备端没有回复。

T5平台出现过（问题单#79009）休眠唤醒之后，直接进行读写报 “cmd 53, RD DTO！！”，当时的结论是正常休眠wifi是不掉电的，所以休眠只需要通知一下模组，T5这个super standby 会给wifi模组掉电，掉电后829里面跑的程序就丢失了，唤醒就没法响应host的命令，所以唤醒后需要重新加载一下wifi驱动和fw，走一下reinit流程

## 3.6.4 STO

**问题现象**：

Log 中出现 STO 的打印：

```
[mmc]: mmc 2, cmd 8(0x80002348), arg 0x00000000
[mmc]: mmc 2 trans data 512 bytes
[mmc]: frag 0, remain 512, des[0](42d011c0): [0] = 8000003c, [1] = 00000200, [2] = 42ae5b80, [3] = 00000000
[mmc]: mmc 2 cacl timeout ffffff
[mmc]: mmc 2 data timeout ffffffff
[mmc]: smc 2 err, cmd 8, STO
[mmc]: mmc 2 close bus gating and reset
[mmc]: mmc 2 mmc cmd 8 err 0xffffffff
[mmc]: invald cache after read complete
[mmc]: mmc send ext csd failed
[mmc]: +++mmc_mmc_update_timeout
```

图 3-34: STO 错误

**问题排查**：

目前遇到这种问题多在 cmd8,cmd18,cmd25 等存在数据传输的命令，

**boot0 阶段报错时：**

如果烧录没问题，到启动阶段 boot0 出现 STO 报错，请更新代码后再测试。

boot0 mmc 驱动路径：spl/drivers/mmc/sun50iw9p1（对用平台的名字）

**Uboot 阶段的错误时：**

如果是拆机片等二手物料则可先进行步骤 4

1. 修改成 cpu 搬运数据，屏蔽掉使用 dma 搬运的宏，如下：

```
//#define CONFIG_MMC_SUNXI_USE_DMA
```

图 3-35: cpu 搬运配置

2. 使用 cpu 搬运数据后，相应的错误消失，则可直接使用 cpu 搬运，或者进行步骤 3
3. 改回使用 dma 搬运，在 mmc 驱动找到 mmc_trans_data_by_dma 这个函数，核对 ftrglevel（一般为基地址 +0x40, 具体 spec 为准）寄存器的设置是否为 spec 推荐值，并且进行修改，以下以非 h3 平台为例：

```
#if defined(CONFIG_MACH_SUN8IW7)
>-------writel((2U<<28)|(7<<16)|8, &priv->reg->ftrglevel);
#else
>-------if (priv->cfg.host_no == 2) {
>-------->-------writel((0x3<<28)|(15<<16)|240, &priv->reg->ftrglevel);
>-------} else {
>-------->-------writel((0x2<<28)|(7<<16)|248, &priv->reg->ftrglevel);
>-------}
```

图 3-36: fifo 阈值配置

如果还不行则进行步骤 4

4. 如果是二手或者损坏物料有可能 dat0 拉低时间过长，一直处于 busy，导致 STO 的发生，可以按照如下补丁修改，增加等待 busy 时间

```
diff --git a/drivers/mmc/sunxi_mmc.c b/drivers/mmc/sunxi_mmc.c
index 0682ceec57..8d51ed5ab5 100755
--- a/drivers/mmc/sunxi_mmc.c
+++ b/drivers/mmc/sunxi_mmc.c
@@ -909,7 +909,7 @@ static int sunxi_mmc_do_send_cmd_common(struct sunxi_mmc_priv *priv,
                                        (((cmd->cmdarg >> 16) & 0xFF) == EXT_CSD_SANITIZE_START)))
                        timeout_msecs = 0x1fffffff;
                else
-                       timeout_msecs = 2000;
+                       timeout_msecs = 5000;

                do {
                        status = readl(&priv->reg->status);
```

图 3-37: mmc patch

**Kernel 阶段的错误时：**

1. 直接核对 spec，ftrglevel 寄存器设置是否为推荐值，这里以 sun8iw19p1 为例：

```
>-------host->dma_tl = SUNXI_DMA_TL_SDMMC_V4P5X;
```

图 3-38: kernel-fifo 阈值配置

## 3.6.5 DCE/RCE(eMMC only)

**问题现象**：Log 中出现如下 RCE 或者是 DCE 的打印：

```
[    1.304443] sunxi-mmc sdc2: smc 0 p2 err, cmd 18, RD RCE !!
[    2.310499] sunxi-mmc sdc2: send  manual stop command failed
[    2.316968] sunxi-mmc sdc2: smc 0 p2 err, cmd 18, RD RTO !!
[    2.489558] sunxi-mmc sdc2: smc 0 p2 err, cmd 18, RD DTO !!
[    2.502199] sunxi-mmc sdc2: smc 0 p2 err, cmd 18, RD RCE !!
[    3.507164] sunxi-mmc sdc2: send  manual stop command failed
```

图 3-39: RCE 错误

**问题排查**：目前遇到这类 CRC 出错的问题，优先考虑 tuning 点的问题。

（1）多块 tuning 的补丁是否已经打上


图 3-40: 多块 tuning 的补丁

（2）烧录阶段 uboot 进行 tuning 的结果有没有正确传递到内核

烧录时的 tuning 结果如下：

```
[04.091][mmc]: DS26/SDR12: 0xffffffff 0xffffffff
[04.095][mmc]: HSSDR52/SDR25: 0xff00ffff 0xffffffff
[04.099][mmc]: HSDDR52/DDR50: 0xff0001ff 0xffffffff
[04.104][mmc]: HS200/SDR104: 0x0001ffff 0xffff0000
[04.108][mmc]: HS400: 0x1717ffff 0xffffff27
```

图 3-41: 烧录时候的 tuning 点

uboot 中查询 tuning 结果如下：

```
=> fdt list mmc2
sdmmc@4022000 {
        compatible = "allwinner,sunxi-mmc-v4p6x";
        device_type = "sdc2";
        reg = <0x00000000 0x04022000 0x00000000 0x00001000>;
        interrupts = <0x00000000 0x00000029 0x00000004>;
        clocks = <0x0000000f 0x00000002 0x00000006 0x00000002 0x0000003d 0x00000002 0x00000040>;
        clock-names = "osc24m", "pll_periph", "mmc", "ahb";
        resets = <0x00000002 0x00000015>;
        reset-names = "rst";
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <0x0000003f 0x00000040>;
        pinctrl-1 = <0x00000041>;
        bus-width = <0x00000008>;
        req-page-count = <0x00000002>;
        cap-mmc-highspeed;
        cap-cmd23;
        mmc-cache-ctrl;
        non-removable;
        max-frequency = <0x08f0d180>;
        cap-erase;
        mmc-high-capacity-erase-size;
        no-sdio;
        no-sd;
        sdc_tm4_sm0_freq0 = <0xffffffff>;
        sdc_tm4_sm0_freq1 = <0xffffffff>;
        sdc_tm4_sm1_freq0 = <0xff00ffff>;
        sdc_tm4_sm1_freq1 = <0xffffffff>;
        sdc_tm4_sm2_freq0 = <0xff0001ff>;
        sdc_tm4_sm2_freq1 = <0xffffffff>;
        sdc_tm4_sm3_freq0 = <0x0001ffff>;
        sdc_tm4_sm3_freq1 = <0xffff0000>;
        sdc_tm4_sm4_freq0 = <0x1717ffff>;
        sdc_tm4_sm4_freq1 = <0xffffff27>;
        sdc_tm4_sm4_freq0_cmd = <0x00000000>;
        sdc_tm4_sm4_freq1_cmd = <0x00000000>;
        mmc-ddr-1_8v;
        mmc-hs200-1_8v;
        mmc-hs400-1_8v;
        ctl-spec-caps = <0x00000308>;
        sunxi-power-save-mode;
        sunxi-dis-signal-vol-sw;
        status = "okay";
        phandle = <0x0000006d>;
};
```

图 3-42: uboot 中的 tuning 点

上图中，sm0-4 分别对应 DS26/SDR12、HSSDR52/SDR25、HSDDR52/DDR50、HS200/SDR104、HS400。每种速度模式都对应一个 freq0 和 freq1，从 freq0 的低位开始，每 8 位

代表一个 freq 等级，分表表示 tuning 的结果对应的频率值为 400000、25000000、50000000、100000000、150000000、200000000。即 freq0 的 bit8-0 对应的频率是 400000，freq1 的 bit8-0 对应的频率是 150000000。

校对 uboot 中查询 tuning 结果和烧录时是否相同，如果不同则需要检查 uboot 代码或者找 uboot 相关负责人协助。

（3）如果检查完以上的配置都是正常的，则检查 pin 的电压以及 bias 配置是否正确

# 3.7 sdio 相关的章节

- 这里单独列出 sdio 相关的章节，出现相应问题直接跳转参考。
- sd/eMMC/sdio 运行过程中读写错误
- sd/eMMC/sdio 卡死分析
- sdio 供电检查
- RTO（response timeout）
- DTO(data timeout)

# 3.8 sd/eMMC/sdio 模组供电检查

## 3.8.1 eMMC 供电检查

根据硬件原理图来排查模组供电是否异常：

eMMC:

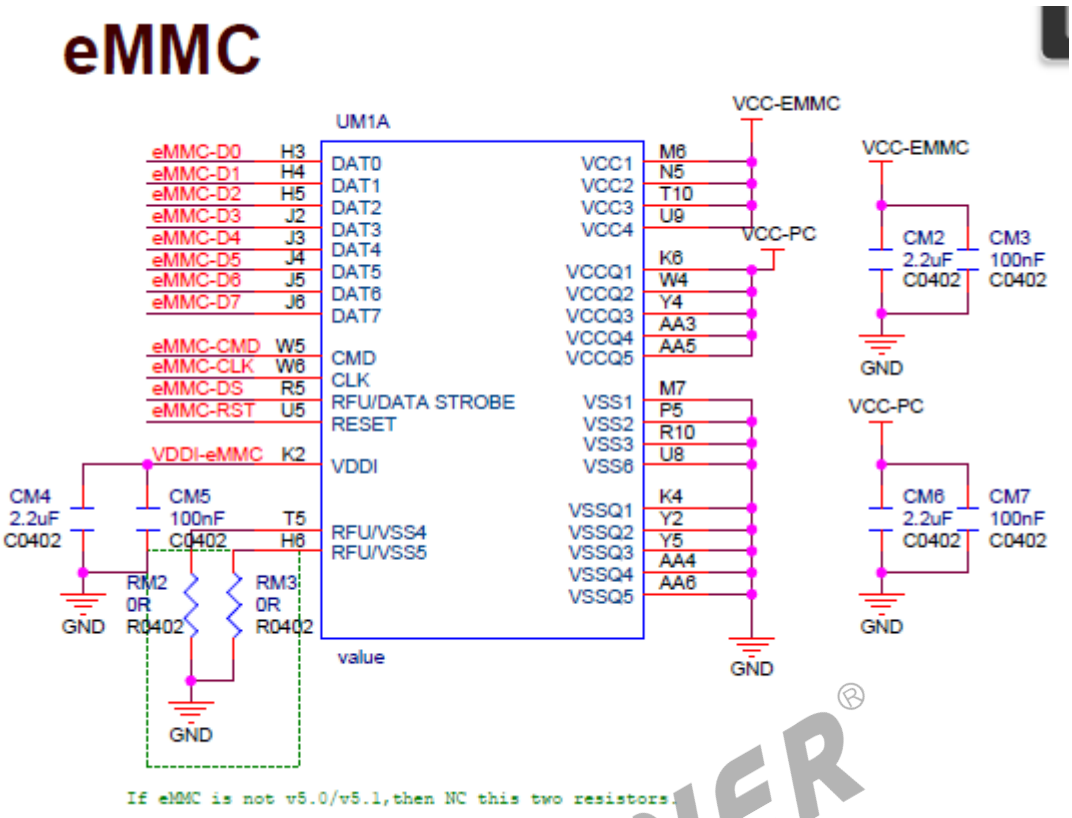


图 3-43: eMMC 供电

需要硬件工程师检查 VCC-EMMC 以及 VCC-PC（一般我们平台以 PC 脚作为 eMMC 的 pin 脚，所以检查 VCC-PC）电源是否正常。VCC-EMMC 和 VCC-PC 情况如下：
VCC-EMMC:3.3V
VCC-PC：
1、3.3 V: 如果设置为 3.3V，则不支持 HS200/HS400 等模式。
2、1.8V: 设置成 1.8V，则可以正常支持 HS200/HS400 等模式（具体能否支持还跟 eMMC 有关）。

VCC-PC 电压得与 eMMC datasheet 相匹配，部分 eMMC 不支持 3.3V 或 1.8V。

步骤二：

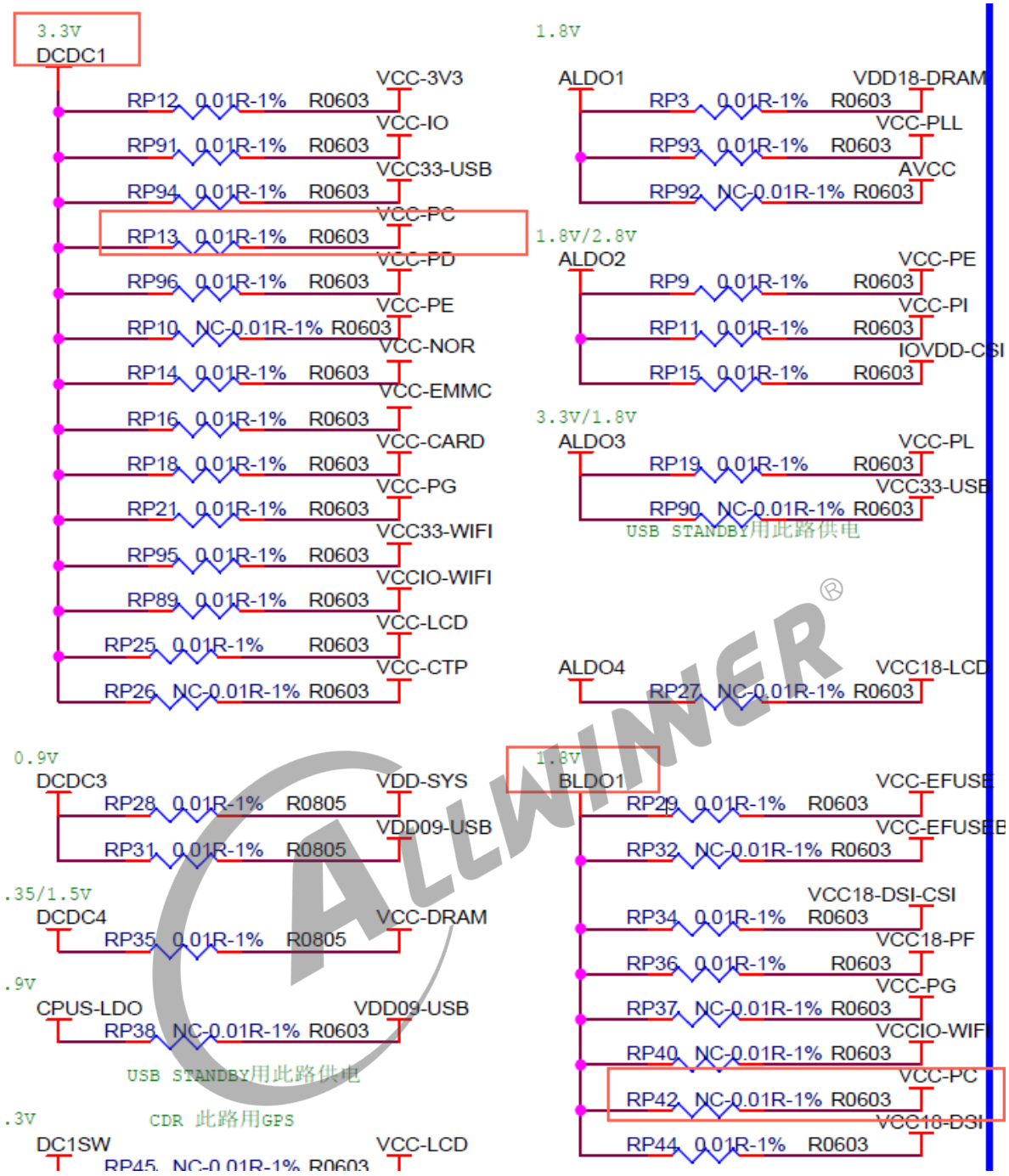


图 3-44: vcc-pc 供电连接

检查 sys_config.fex/board.dts 等配置文件：

> 核对 Sdc2 的相关电源配置是否和电路图相对应。一般对应 sdc2 节点的电源。下图中 vmmc-supply/vqmmc-supply 分别对应步骤一 VCC-EMMC/VCC-PC。

```
>-------->------->-------vmmc-supply = <&reg_dcdc1>;
>-------->------->-------/*1.8V select bldo1*/
>-------->------->-------/*vqmmc-supply = <&reg_bldo1>;*/
>-------->------->-------/*3.3V select dcdc1*/
>-------->------->-------vqmmc-supply = <&reg_dcdc1>;
```

图 3-45: sys_config 电源配置

vqmmc-supply 的选择须与外部电路对应，以下配置对应下图电路。一般通过接上电阻选择对应的电源，只能二选一。

步骤三：

核对 port 的 spec，确定是否支持 bias 设置，一般 bias 地址：（port 基地址 +340），如果支持 bias 设置，sys_config.fex 上应该根据 vcc-pc 的电压值，设置属性：比如说 vcc-pc 为 1.8V，使用的是 bldo5 供电。则设置如下：(如果使用 vcc-pc 为 3.3V 以下节点可以不设置)

Uboot-2014：

```
;-----------------------------------------------------------------------------
;gpio_bias pc_bias for emmc high speed mode use
;-----------------------------------------------------------------------------
[gpio_bias]
pc_bias         = "axp858:bldo5:1800"
```

图 3-46: bias 配置

Uboot-2018:

```
[gpio_bias]
pc_bias>>--------= 1800
```

图 3-47: uboot2018-bias 配置

如果使用 vcc-pc 为 3.3V 则以下节点可以使用，屏蔽或者不设置，如下图：

Uboot-2014：

```
;-----------------------------------------------------------------------------
;gpio_bias pc_bias for emmc high speed mode use
;-----------------------------------------------------------------------------
;[gpio_bias]
;pc_bias         = "axp858:bldo5:1800"
```

图 3-48: 3.3Vbias 配置

Uboot-2018:

```
;[gpio_bias]
;sdc_bias>-------->-------= 1800
```

图 3-49: uboot20183.3Vbias 配置

## 3.8.2 sd 供电检查

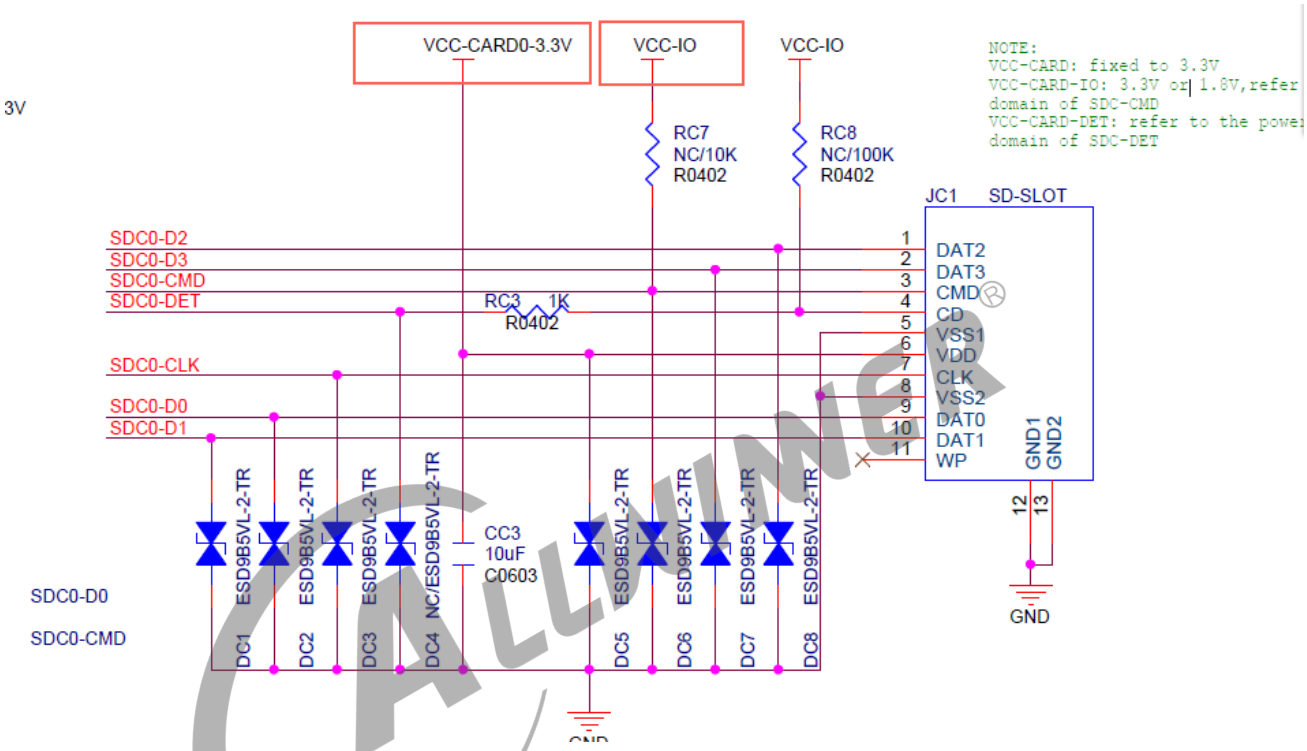


图 3-50: sd 供电

**步骤 1**

检查 VCC-CARD（VCC-CARD0-3.3V）以及 VCC-PF（VCC-IO）（一般我们平台都是以 PF 口作为 sd 卡口）IO 电压是否正常（通过测量信号线 cmd/data[0:3] 电压是否与 VCC-PF 相匹配）。
VCC-CARD：3.3V
VCC-PF：
1、3.3V:sd1.0，sd2.0 的卡以及 sd3.0 ds 模式都应该是 3.3V
2、1.8V:sd3.0 卡的 sdr12\sdr25\sdr50\sdr104\ddr50 等速度模式。电压值应该为 1.8V，检查这路电是否是 3.3,1.8V 可切换的。

**步骤 2**

如果使用如下配置：

确认 vcc-pf 是否配置正确

参考电路如下：


图 3-51: pf 供电

参考配置如下

```
885 vmmc="vcc-sdc"
886 vqmmc="vcc-pf"
887 vdmmc="vcc-pf"
```

图 3-52: sd 卡供电配置

```
regulator16     = "pmu1736_bldo4 none vdd-csi dvdd-csi"
regulator17     = "pmu1736_bldo5 none vcc-dsi vcc-efuse vcc-pc vcc-cpvin vcc-lvds vcc-mcsi"
regulator18     = "pmu1736_cldo1 none vcc-ctp"
regulator19     = "pmu1736_cldo2 none vcc-pf"
regulator20     = "pmu1736_cldo3 none vcc-motor vcc-mipi"
```

图 3-53: regulator 供电配置

**步骤 3**

如果使用如下配置：

参考电路如下：（参考电路 vcc33-pf 已经固定接到 dcdc1）

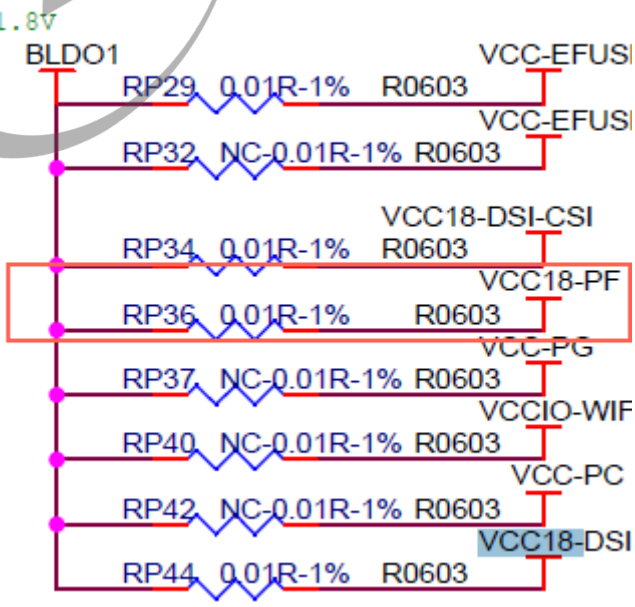


图 3-54: pf-io 供电

确认 vqmmc33sw 和 vqmmc18sw 对应的 regulator 通路是否分别为 3.3V 和 1.8V。

```
>--------vmmc-supply = <&reg_dcdc1>;
>--------/*vqmmc-supply = <&reg_dcdc1>;*/
>--------/*vdmmc-supply = <&reg_dcdc1>;*/
>--------vqmmc33sw-supply = <&reg_dcdc1>;
>--------vdmmc33sw-supply = <&reg_dcdc1>;
>--------vqmmc18sw-supply = <&reg_bldo1>;
>--------vdmmc18sw-supply = <&reg_bldo1>;
```

图 3-55: pf-io 供电配置

### 3.8.3 sdio 供电检查

> 一般 sdio 主要用作 wifi 通信，所以主要按照 wifi 量产指南，检查 wifi 供电以及 IO 电源。

## 3.9 eMMC 数据异常

### 3.9.1 Boot0 数据奇偶交换

**问题现象**：OTA 升级，U 盘升级之后或者使用一段时间突然启动不了，接上 usb，进入烧写模式

**问题分析**：先通过 dump 固件，把 boot0 dump 出来和原始的 boot0，或者正常机器的 boot0 做对比，如果出现字节间出现奇偶交换（参考下图），请更联系 mmc 同事新 mmc 驱动代码

图 3-56: 奇偶交换数据

## 3.9.2 Uboot load kernel img 失败

**问题现象一：**ota 等之后重启，出现启动不了，显示下面加粗部分错误

[2020-07-28-19-48-08]U-Boot 2014.07 (Jun 13 2020 - 20:55:58) Allwinner Technology

[2020-07-28-19-48-08]……...

[2020-07-28-19-48-09]show hash of file

[2020-07-28-19-48-09]f5 34 ee 4b 92 86 2c f8 cd 4c e8 f1 62 9c 9f ab

[2020-07-28-19-48-09]3d 69 92 e0 e6 8e 0c 84 8b 2a 63 4f e4 4c 8b e2

[2020-07-28-19-48-09]

[2020-07-28-19-48-09]find cert name optee

[2020-07-28-19-48-09]find cert name scp

[2020-07-28-19-48-09]find cert name u-boot

[2020-07-28-19-48-09]find cert name boot_a

[2020-07-28-19-48-09]show hash in certif

[2020-07-28-19-48-09]58 fd 98 e4 9a 99 d7 13 ba 7c 00 8b f3 6a 98 00

[2020-07-28-19-48-09]8d 9f 96 36 ab 52 44 f5 22 23 20 ac 4b c6 cf c1

[2020-07-28-19-48-09]

[2020-07-28-19-48-09]**]hash compare is not correct**

[2020-07-28-19-48-09]>>>>>>>hash of file<<<<<<<<<<

[2020-07-28-19-48-09]f5 34 ee 4b 92 86 2c f8 cd 4c e8 f1 62 9c 9f ab

[2020-07-28-19-48-09]3d 69 92 e0 e6 8e 0c 84 8b 2a 63 4f e4 4c 8b e2

[2020-07-28-19-48-09]

[2020-07-28-19-48-09]>>>>>>>hash in certif<<<<<<<<<<

[2020-07-28-19-48-09]58 fd 98 e4 9a 99 d7 13 ba 7c 00 8b f3 6a 98 00

[2020-07-28-19-48-09]8d 9f 96 36 ab 52 44 f5 22 23 20 ac 4b c6 cf c1

现象分析：

这种现象其中一种原因：

boot.img 有两个备份 a，b，每一个备份和特定的 toc1（uboot）对应，如果启动的时候对应不上，就会报这个错误。

出问题的一种情况是 ota 升级过程当中，出现失败，但是还是 ota 流程还是切换了备份，导致 toc1 和对应的 boot.img 对应不上，导致出错，这种情况需要找 ota 相关的同事分析

**问题现象二：** uboot 加载内核的时候，mmc 退出报错

[5.746]show bmp on ok
Hit any key to stop autoboot: 0
Kernel load addr 0x40008000 size 15888 KiB
Kernel command line: selinux=1 androidboot.selinux=permissive, use it to update bootargs
RAM disk load addr 0x42000000 size 2281 KiB
the chip id is 0x81
[mmc]: mmc 2 cmd 1 timeout, err 100
[mmc]: smc 2 err, cmd 1, RTO
[mmc]: mmc 2 close bus gating and reset
[mmc]: mmc 2 mmc cmd 1 err 0x00000100
[mmc]: mmc 2 cmd 1 timeout, err 100
[mmc]: smc 2 err, cmd 1, RTO
[mmc]: mmc 2 close bus gating and reset
[mmc]: mmc 2 mmc cmd 1 err 0x00000100

mmc 这种形式的报错，不会影响启动内核。遇到这种问题，最好先连接 DS5 判断程序目前卡在哪里。有可能代码已经运行到内核的时候卡住，但是从 log 结果来看是卡在 mmc。

例：问题单 #46484，出问题时 uboot 的最后 log 如上出现 eMMC 退出时报错，但是用 DS5 debug 发现 PC 指针已经跳转到内核了。该问题查明最后的原因为 dram 出问题导致代码运行出错。

## 3.9.3 其他情况

需要通过 dump 固件等方式把问题数据 dump 出来，同时 dump 出正常机器数据，两个对比，把对比结果发给 mmc 的同事。Dump 固件使用 SDK 可以生成特殊固件：dump 固件，用于把 flash 上的内容读到电脑上，并以文件的形式保存

tina 上使用 pack -m 可生成 dump 固件

Homlet 上使用 cd lichee ./build.sh pack_dump
可生成 linux dump 固件

在 windows 上，D 盘下建立一个名为 “test” 的文件夹

使用 phoenixsuit 选中 dump 固件，进行烧写。phoenixsuit 上面的 dump 复选框，不用勾选，注意是不用勾选。

此时不会烧写到 Flash 中。最终烧录工具会提示失败，是正常的，此时已经把 flash 中的内容读取到 D:/test 文件夹下。

注意 nand 方案 dump 出的 boot0,boot1 数据是无效的，请忽略。

# 3.10 其他问题

## 3.10.1 eMMC 速度慢或与设置不符合

**问题现象**：（1）同一个固件，有的 eMMC 跑的频率比设置慢，或者不是设定的速度模式（2）客户用 eMMC 在别的平台可以跑到设置频率，但是在全志平台跑不到

**问题原因**：

1. 此现象可能是不同的板子 tuning 的时候得到结果不同，可以先尝试减小 sys_config.fex 的 sdc_tm4_win_th 的值，如由 12 改成 10，由 10 改成 8，都可以进行尝试。
2. 没有打开 tuning 功能，确认 sys_config.fex [card2_boot_para] 下配置了 sdc_ex_dly_used = 2（如下图中 sdc_ex_dly_used 的配置）

**tips**：在进行 tuning 的时候，会将连续 tuning 成功的频点放到一个区间中，这个 window 值含义是区间的长度要大于该值才有效，否则会忽略该区间的结果。这个 window 值是我们进行多次测试多出的经验值，但是不同的物料也可能会有偏差，如果进行修改要进行一定量的压力测试。

```
[card2_boot_para]
card_ctrl         = 2
card_high_speed = 1
card_line         = 8
sdc_clk           = port:PC5<3><1><3><default>
sdc_cmd           = port:PC6<3><1><3><default>
sdc_d0            = port:PC10<3><1><3><default>
sdc_d1            = port:PC13<3><1><3><default>
sdc_d2            = port:PC15<3><1><3><default>
sdc_d3            = port:PC8<3><1><3><default>
sdc_d4            = port:PC9<3><1><3><default>
sdc_d5            = port:PC11<3><1><3><default>
sdc_d6            = port:PC14<3><1><3><default>
sdc_d7            = port:PC16<3><1><3><default>
sdc_emmc_rst      = port:PC1<3><1><3><default>
sdc_ds            = port:PC0<3><2><3><default>
sdc_tm4_hs400_max_freq = 150
sdc_tm4_win_th = 8
sdc_ex_dly_used = 2
sdc_io_1v8        = 1
;sdc_dis_host_caps = 0x100
;sdc_type         = "tm4"
```

图 3-57: tuning 点速度值

另外，如果是 linux-5.4 的内核，sys_config.fex 的配置移到了 uboot-board.dts，配置方法和含义不变。

## 3.10.2 brom 启动 boot0 失败

1. 请 FAE 检查 boot_select 配置，还有 uboot pin 电平是否正常
2. 如果掉电后可以恢复正常，可以去掉 dts 键字 “sunxi-power-save-mode” 再测试看看

```
&sdc2 {
        non-removable;
        bus-width = <8>;
        mmc-ddr-1_8v;
        mmc-hs200-1_8v;
        mmc-hs400-1_8v;
        no-sdio;
        no-sd;
        ctl-spec-caps = <0x308>;
        cap-mmc-highspeed;
        sunxi-power-save-mode;
        sunxi-dis-signal-vol-sw;
        max-frequency = <150000000>;
        /*vmmc-supply = <&reg_dcdc1>;*/
        /*emmc io vol 3.3v*/
        /*vqmmc-supply = <&reg_aldo1>;*/
        /*emmc io vol 1.8v*/
        /*vqmmc-supply = <&reg_eldo1>;*/
        status = "disabled";
};
```

图 3-58: dts config

## 3.10.3 SDC 的 CLK 不连续

请检查 dts 中是否配置了 sunxi-power-save-mode（一般作为 sd 卡和 eMMC 时才需要配置该属性），设置该属性之后会在保持在没有指令和数据的时候，不进行 clock 的输出。如果遇到这种疑问，可以先考虑将 dts 中的该字段去掉再进行 clk 测试。

## 3.10.4 mmc 设备擦除慢

该问题可能是由于 eMMC 设备处理不同的命令时间不同导致，可以参考问题单 #43247，该案例如下：

**问题现象**：Sandisk eMMC 在 data 分区格式为 f2fs 的时候，擦出异常慢（恢复出厂设置超过 15 分钟）。

**问题分析：**先 f2fs 文件系统格式化工具发送的是 secure discard ioctl 命令，这个命令这款 eMMC 执行一次要 11 秒。ext4 文件系统格式化工具发送的是 discard ioctl 这款 eMMC 执行一次 20ms。这个问题要根本解决，需要 eMMC 原厂优化 secure discard 的执行时间。

**解决方法：** 1. 可修改 mkf2fs 源码，关闭 BLKSECDISCARD。2. 应用层确定是这款 eMMC 时，mkf2fs 时不使能 t 参数。以上 2 点选其一。

## 3.10.5 eMMC 引脚信号弱，设置 eMMC 驱动能力

**解决方法**：设置 dts 修改内核 eMMC 驱动能力，配置关键字方法如：

fixed-emmc-driver-type = <1>

fixed-emmc-driver-type 表示设置驱动能力属性，后面的值为要设置的 Driver Type，如下图所示（此图为示例，具体的驱动能力表参考 eMMC 原厂提供的 datasheet）

For HS400, when tested with the reference load defined in page 27 HS400 reference load figure, Driver Type-0 or Driver Type-1 or Driver Type-4 shall meet all AC characteristics and HS400 Device output timing requirements.

| Driver Type | TOSHIBA e-MMC | Nominal Impedance (Driver strength) | Approximated driving capability compared to Type-0 | Remark |
|---|---|---|---|---|
| 0 | Supported | 50 Ω (18mA) | x1 | Default Driver Type |
| 1 | Supported | 33 Ω (27mA) | x1.5 | Recommendation at HS400 under the condition of JEDEC standard reference load. |
| 2 | Supported | 66 Ω (14mA) | x0.75 | |
| 3 | Supported | 100 Ω (9mA) | x0.5 | |
| 4 | Supported | 40 Ω (23mA) | X1.2 | Recommendation at HS400 under the condition of JEDEC standard reference load. |

1) Nominal impedance is defined by I-V characteristics of output driver at 0.9V when $V_{CCQ}$ = 1.8V.

图 3-59: dirver strength table

**tips**：需要先确定客户分支是否支持设置驱动能力代码（搜索 fixed-emmc-driver-type），不支持则需要从 sunxi-dev 移植

## 3.10.6 eMMC 速度模式设置

**uboot**：

```
card2_boot_para@3 {
        /*
         * Avoid dtc compiling warnings.
         * @TODO: Developer should modify this to the actual value
         */
        reg = <0x0 0x3 0x0 0x0>;
        device_type = "card2_boot_para";
        card_ctrl = <0x2>;
        card_high_speed = <0x1>;
        card_line = <0x8>;
        pinctrl-0 = <&card2_pins_a &card2_pins_b>;
        sdc_ex_dly_used = <0x2>;
        sdc_io_1v8 = <0x1>;
        sdc_tm4_hs200_max_freq = <200>;
        sdc_tm4_hs400_max_freq = <150>;
        sdc_tm4_win_th = <0x08>;
        /*tm4_tune_r_cycle = <1>;*/
        sdc_type = "tm5";
};
```

图 3-60: uboot-dts

如图，uboot 可以通过设置 uboot-board.dts 中的 sdc_freq、sdc_tm4_hs200_max_freq、sdc_tm4_hs400_max_freq 来对应设置 hs、hs200、hs400 速度模式的最大频率 (sys_config.fex 同理)

可通过 uboot-board.dts 中变量 sdc_dis_host_caps 关闭一些 Host 特性，以此来改变 eMMC 的速度模式。

| sdc_dis_host_caps | 定义 |
| --- | --- |
| Bit1 | 不支持 HS-SDR |
| Bit3 | 不支持 8 线模式 |
| Bit6 | 不支持 HS-DDR |
| Bit7 | 不支持 HS200 |
| Bit8 | 不支持 HS400 |
| 其它 bit | 未定义，不要使用 |

uboot-board.dts 配置 sdc_dis_host_caps 方法如图所示

```
card2_boot_para@3 {
        /*
         * Avoid dtc compiling warnings.
         * @TODO: Developer should modify this to the actual value
         */
        reg = <0x0 0x3 0x0 0x0>;
        device_type = "card2_boot_para";
        card_ctrl = <0x2>;
        card_high_speed = <0x1>;
        card_line = <0x4>;
        pinctrl-0 = <&sdc2_pins_a>;
        /*pinctrl-0 = <&sdc0_pins_a>;*/
        /*sdc_ex_dly_used = <0x2>;*/
        sdc_io_1v8 = <0x1>;
        /*sdc_type = "tm4";*/
        sdc_tm4_hs200_max_freq = <150>;
        sdc_tm4_hs400_max_freq = <100>;
        sdc_ex_dly_used = <2>;
        /*sdc_tm4_win_th = <8>;*/
        sdc_dis_host_caps = <0x180>;
};
```

图 3-61: sdc_dis_host_caps config

**kernel**:

```
>-------sdc2: sdmmc@04022000 {
>-------->-------non-removable;
>-------->-------bus-width = <8>;
>-------->-------mmc-ddr-1_8v;
>-------->-------mmc-hs200-1_8v;
>-------->-------mmc-hs400-1_8v;
>-------->-------no-sdio;
>-------->-------no-sd;
>-------->-------cap-mmc-highspeed;
>-------->-------sunxi-power-save-mode;
>-------->-------sunxi-dis-signal-vol-sw;
>-------->-------max-frequency = <100000000>;
>-------->-------vmmc-supply = <&reg_dcdc1>;
>-------->-------/*emmc io vol 3.3v*/
>-------->-------vqmmc-supply = <&reg_aldo1>;
>-------->-------/*emmc io vol 1.8v*/
>-------->-------/*vqmmc-supply = <&reg_eldo1>;*/
>-------->-------status = "disabled";
>-------};
```

图 3-62: sdc2 降频降模式

eMMC 速度模式设置修改 dts 即可，上图红圈第一处为设置速度模式，下面是设置频率大小。代码会使用已经设置速度模式的较为高速模式，例如上图三种速度模式都设置了则会使用 HS400，如果去掉 mmc-hs400-1_8v，则会使用 hs200。

另外，内核最终速度模式和频率是 uboot 和内核的交集，内核设定的速度模式和频率需要其在 uboot tuning 出合适的点才可以，也就是改 uboot eMMC 的频率和速度模式，这种方式也会影响内核的频率和速度模式，建议设置速度模式和频率在 uboot 中进行设置。

## 3.10.7 开机卡住

**问题现象**：烧录完固件后机器无法开机，排查的 log 发现卡在 boot0 或者 uboot

**解决方法**：如果判断和 mmc 有关系，但是又在以上未找到解决方法，可以尝试更新 uboot mmc 驱动，驱动所在路径：

uboot2018：u-boot-2018/drivers/mmc

uboot2014：u-boot-2014.07/arch/arm/cpu/armv7/sunxwxxpx/mmc/mmc_bsp.c

## 3.10.8 brom 耗时久

**问题现象**：从 brom 到 boot0 的时间比较久，如下的时间戳为 236，如果 brom 出现 eMMC 类的异常可能导致时间戳增大。

```
[236]HELLO! SBOOT is starting!
```

**问题示例**：曾出现过此问题，boot0 启动的时间戳增大到 800 多毫秒。后面发现问题卡在 eMMC 这里，最终抓波形发现为 eMMC 的 cmd1 一直 busy 耗时 730ms。由于耗时也是在协议规定时间内，原厂回复为正常现象。

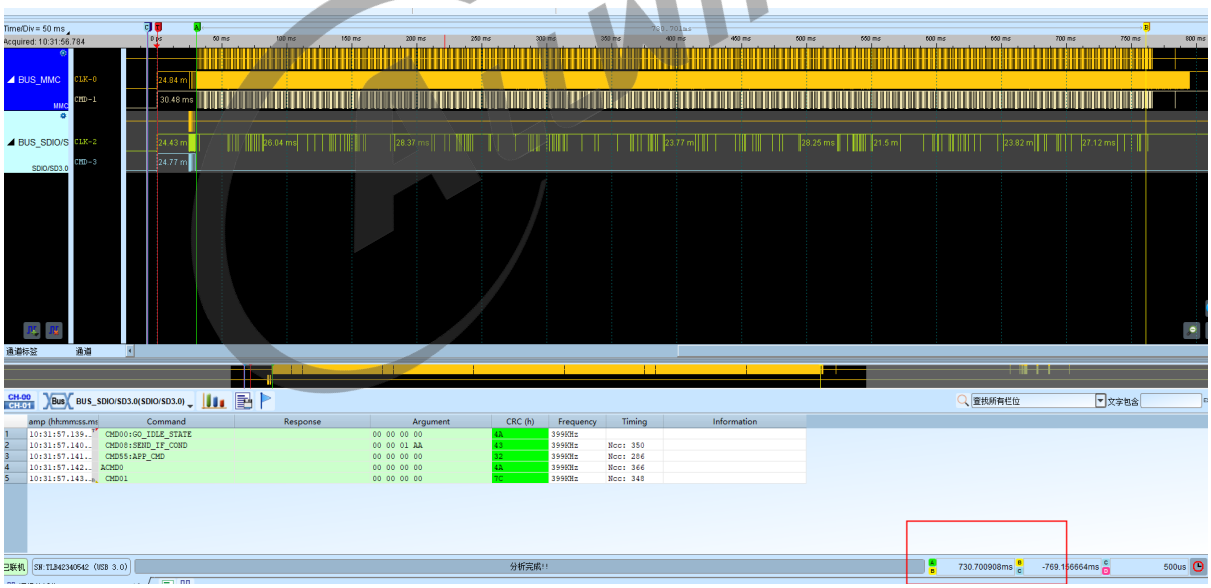


图 3-63: cmd1 busy 730ms

## 3.10.9 其他报错

**问题现象一**：

识别过程报如下 fail，该错误只是提示信息，不代表出现严重错误

```
sunxi-mmc sdc2: sdc set ios:clk 400000Hz bm PP pm ON vdd 21 width 8 timing LEGACY dt B
sunxi-mmc sdc2: sdc set ios:clk 400000Hz bm PP pm ON vdd 21 width 8 timing MMC-HS200 dt B
sunxi-mmc sdc2: sdc set ios:clk 100000000Hz bm PP pm ON vdd 21 width 8 timing MMC-HS200 dt B
sunxi-mmc sdc2: sdc set ios:clk 100000000Hz bm PP pm ON vdd 21 width 8 timing MMC-HS dt B
sunxi-mmc sdc2: sdc set ios:clk 52000000Hz bm PP pm ON vdd 21 width 8 timing MMC-HS dt B
sunxi-mmc sdc2: sdc set ios:clk 50000000Hz bm PP pm ON vdd 21 width 8 timing MMC-HS400 dt B
sunxi_mmc_get_hs400_cmd_dly,216
sunxi-mmc sdc2: failed to get HS400_cmd used default
sunxi-mmc sdc2: sdc set ios:clk 100000000Hz bm PP pm ON vdd 21 width 8 timing MMC-HS400 dt B
sunxi_mmc_get_hs400_cmd_dly,216
sunxi-mmc sdc2: failed to get HS400_cmd used default
mmc0: new HS400 MMC card at address 0001
mmcblk0: mmc0:0001 FFFFFF 29.1 GiB
mmcblk0boot0: mmc0:0001 FFFFFF partition 1 4.00 MiB
mmcblk0boot1: mmc0:0001 FFFFFF partition 2 4.00 MiB
mmcblk0rpmb: mmc0:0001 FFFFFF partition 3 4.00 MiB
sunxi-mmc sdc1: sdc set ios:clk 400000Hz bm PP pm ON vdd 21 width 1 timing LEGACY dt B
mmcblk0: p1 p2
```

图 3-64: get_dts_fail

# 4 排查失效

1. 按照以上步骤，仍然没排查到问题，参考 Sd/eMMC/sdio 日志信息调试，在**错误打印发生时（第一现场）**，打印出 pio,ccmu 以及 host 控制器等寄存器信息，提供给全志 FAE。
2. 使用逻辑分析仪，捉取出现问题时候第一现场的波形，提供给全志 FAE。（可以在第一现场前，添加延时或者添加 pin 脚反转，更好捉取波形。）

**著作权声明**



**商标声明**



**免责声明**