密级状态：绝密(　　)　　秘密(　　)　　内部(　)　　公开(√)

RK3399_功耗优化

（技术部，第二系统产品部）

文件状态： [　] 正在修改 [√] 正式发布	当前版本：	V1.1
	作　　者：	王剑辉
	完成日期：	2019-07-25
	审　　核：	陈亮、邓训金、张文平、魏建兴
	完成日期：	2019-07-26

福州瑞芯微电子股份有限公司

Fuzhou　Rockchips　Electronics　Co．, Ltd

版 本 历 史

版本号	作者	修改日期	修改说明	备注
V1.0	王剑辉	2018.11.21	发布初版	
V1.1	王剑辉	2019.07.25	完善功耗排查方法 增加功耗 debug 手段介绍 增加功耗优化说明 增加休眠唤醒问题排查 增加温控介绍 增加系统自排查 checklist	

目 录

前 言

概述

本文档主要介绍 Rockchip RK3399 Android7.1 行业版本功耗初步排查、优化、debug 方法，旨在帮助软件开发工程师优化 RK3399 平台 Android7.1 行业版本的功耗，达到性能和功耗兼顾。

读者对象

本文档主要适用于以下工程师：

技术支持工程师

软件开发工程师

1 RK3399 Android7.1 行业 SDK 功耗数据

1.1 RK3399 Android7.1 行业 SDK--挖掘机功耗数据

表 1-1 RK3399 挖掘机功耗数据

Power consumption of RK3399 EXCAVATOR (ANDROID 7.1 REMOVE PANEL)																						
TEST SCENE		BAT		VCC3V3_SYS		VDD_CPU_B		VDD_CPU_L		VDD_GPU		VDD_LOG		VDD_0V9		VDD_CENTER		VCC_DDR		VCC_1V8		note
		Voltage	Current	Voltage	Current	Voltage	Current	Voltage	Current	Voltage	Current	Voltage	Current	Voltage	Current	Voltage	Current	Voltage	Current	Voltage	Current	
Deep sleep power		8.00	5.80	3.32	12.30	0.00	0.00	0.00	0.00	0.00	0.00	0.90	15.90	0.92	1.30	0.00	0.00	1.27	4.80	1.86	1.30	①③
Desktop	Static	8.00	146.80	3.32	314.30	0.80	8.10	0.82	4.60	0.79	27.70	1.03	231.90	0.92	5.30	0.83	158.70	1.26	118.80	1.86	111.30	②④⑨
Video local	1080P	8.00	185.40	3.32	395.80	0.80	17.70	0.81	49.90	0.79	29.20	1.03	254.40	0.92	5.30	0.86	277.80	1.26	164.70	1.86	115.90	②⑤⑨
	4K	8.00	297.30	3.32	644.60	0.80	21.50	0.82	77.80	0.79	37.20	1.03	270.70	0.92	5.30	0.91	723.50	1.26	357.90	1.86	133.10	②⑥⑨
	1080P	8.00	187.80	3.32	401.30	0.80	22.50	0.82	50.90	0.79	29.80	1.03	254.40	0.92	5.30	0.86	283.50	1.26	170.90	1.86	114.90	②⑦⑨
Game local	Fish2	8.00	256.40	3.32	548.30	0.80	16.00	0.84	87.70	0.80	273.20	1.03	277.20	0.92	5.30	0.86	364.60	1.26	254.90	1.86	118.10	②⑧⑨
Video online	aiqiyi	8.00	259.60	3.32	556.40	0.80	25.80	0.82	79.20	0.80	265.30	1.03	256.90	0.92	5.40	0.86	301.20	1.26	223.40	1.86	117.10	②⑧⑨

Test Item Description: default remove screen
①: just sapphire socboard，remove L900、R901、R90016，powered at VCC_SYS，PCIE_VDD_0V9 can not be switched off when deep sleep.
②: remove excavator board U1401、R9015、FB2602、FB2604、disabled HDMI IN. powered at VCC_BATT.
③: Test time:60s. Waiting 10s,when the TP is standby,then Record data.
④: Test time:60s. Waiting 10s,when the TP is standby,then remove screen and record data.
⑤: Test time:5min. Source file:bbb_full.ffmpeg.1920x1080.mp4.libx264_10000kbps_30fps.libfaac_stereo_192kbps_48000Hz.mp4.
⑥: Test time:5min. Source file:4K_60fps_H265.mp4,Full Scrreen.
⑦: Test time:5min. Source file:1920x1080,27.0 Mbps,23.976fps,AVC,DTS AC3 AC3,1536 Kbps,变形金刚2：卷土重来sample.mkv.
⑧: Test time:5min. Fish2_v1_1_3.apk.
⑨: Test time:5min. aiqiyi_8.8.0.apk play 车手，1080p.
⑩: Used EDP panel，VCC_1V8 will increase 78.8mA.

表 1-1 是室温环境下（25℃），RK3399 挖掘机的功耗数据。DDR 使用的是 lpddr3 颗粒，客户当前项目 DDR 使用的是 lpddr3 颗粒的，请参考表 1-1 的功耗数据进行对比排查。表 1-1 有 Deep Sleep 场景、静态桌面场景、播放 1080p 视频场景、播放 4K 视频场景、Fish2 拷机场景、播放爱奇艺在线视频场景功耗数据。其中表 1-1 填写的测试片源和 apk 可以通过联系我司 fae 窗口邮箱 fae@rock-chips.com 获取。

1.2 RK3399 Android7.1 行业 SDK--EVB 板功耗数据

表 1-2 RK3399 EVB 板功耗数据

Power consumption of RK3399 solutions																						
Power consumption of RK3399 LPDDR4 (400-800M负载变频)EVB																						
TEST SCENE		BAT		VCC3V3_SYS		VDD_CPU_B		VDD_CPU_L		VDD_GPU		VDD_LOG		VDD_0V9		VDD_CENTER		VCC_DDR		VCC_1V8		note
		Voltage	Current	Voltage	Current	Voltage	Current	Voltage	Current	Voltage	Current	Voltage	Current	Voltage	Current	Voltage	Current	Voltage	Current	Voltage	Current	
Deep sleep power		12.0	12.0	3.3	17.7	0.0	0.0	0.0	0.0	0.0	0.0	0.9	6.0	0.9	4.2	0.0	0.0	1.1	3.3	1.8	5.8	③
Desktop	Static	12.0	114.6	3.3	317.2	0.8	5.2	0.8	4.2	0.8	23.2	0.9	201.4	0.9	38.4	0.9	297.5	1.1	144.0	1.8	75.1	④
Video local	1080P	12.0	141.6	3.3	383.9	0.8	14.8	0.8	47.7	0.8	21.9	0.9	206.9	0.9	38.1	0.9	349.7	1.1	193.9	1.8	84.7	⑤
	4K	12.0	232.4	3.3	688.7	0.8	12.8	0.8	80.2	0.8	29.5	0.9	218.1	0.9	39.0	0.9	829.8	1.1	400.6	1.8	117.8	⑥
	1080P	12.0	141.2	3.3	385.9	0.8	19.7	0.8	48.2	0.8	25.7	0.9	208.5	0.9	38.5	0.9	357.1	1.1	195.6	1.8	82.3	⑦
Game local	Fish2	12.0	161.5	3.3	422.8	0.8	15.7	0.8	73.8	0.8	225.8	0.9	213.0	0.9	38.3	0.9	350.0	1.1	232.0	1.8	85.0	⑧

③: Test time:60s. Waiting 10s,when the TP is standby,then Record data.
④: Test time:60s. Waiting 10s,when the TP is standby,then remove screen and record data.
⑤: Test time:5min. Source file:bbb_full.ffmpeg.1920x1080.mp4.libx264_10000kbps_30fps.libfaac_stereo_192kbps_48000Hz.mp4.
⑥: Test time:5min. Source file:4K_60fps_H265.mp4,Full Scrreen.
⑦: Test time:5min. Source file:1920x1080,27.0 Mbps,23.976fps,AVC,DTS AC3 AC3,1536 Kbps,变形金刚2：卷土重来sample.mkv.
⑧: Test time:5min. Fish2_v1_1_3.apk.

表 1-2 是室温环境下（25℃），RK3399 EVB 板的功耗数据。DDR 使用的是 lpddr4 颗粒，客户当前项目 DDR 使用的是 lpddr4 颗粒的，请参考表 1-2 的功耗数据进行对比排查。表 1-2 有 Deep Sleep 场景、静态桌面场景、播放 1080p 视频场景、播放 4K 视频场景、Fish2 拷机场景、功耗数据。其中测试片源和 apk 可以通过联系我司 fae 窗口邮箱 fae@rock-chips.com 获取。

2 功耗初步排查步骤

2.1 对比总功耗

首先，对比同一场景下的总功耗，初步确认客户板子功耗是否正常。表 1-1 和表 1-2 中 BAT 这一列是 RK3399 Android7.1 行业 SDK 板各种场景的总功耗数据。假如客户板子总功耗和 RK 提供的 SDK 板参考数据差异较大，建议进一步分解功耗，请参阅第 2.2 章节。

注意：测试功耗数据时，确保只有硬件差异，其他测试条件一致。例如对比 1080p 视频播放场景的功耗数据，Android 版本相同、播放器 APK 相同、视频片源相同、去除 edp 屏和 hdmi 显示。

2.2 分解各路功耗

为了详细分解功耗，熟悉项目 Power Tree 也是需要的。图 1-1 是 RK3399 挖掘机 Power Tree。在项目功耗优化过程中，可以根据硬件原理图列出 Power Tree，然后在对应电路上串一定阻值的电阻（一般推荐串 20mR 电阻），测量出对应电路的电流。根据 1.1 章节的功耗数据表，RK3399 挖掘机分解了以下几路功耗：VCC3V3_SYS、VDD_CPU_B、VDD_CPU_L、VDD_GPU、VDD_LOG、VDD_0V9、VDD_CENTER、VCC_DDR、VCC_1V8。

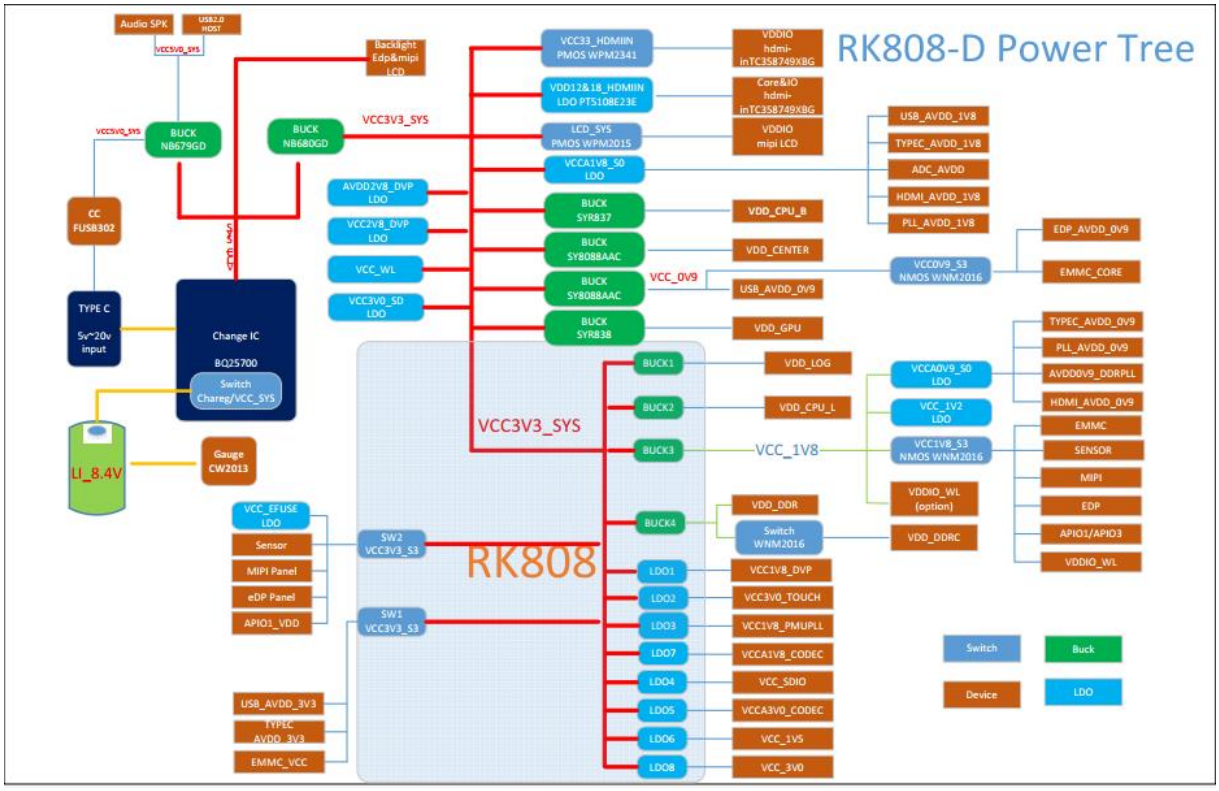

图 1-1 RK3399 挖掘机 Power Tree

分解功耗后，对比数据。先对比 CPU 大小核、GPU、DDR 功耗是否正常，确认 CPU 变频、GPU 变频、DDR 变频是否正常。假如有抬电压，确认抬的电压是否为 12.5mv 的整数倍。变频策略正常的情况下，则需要进一步对比分析其他外设功耗。

2.3 分解各路电源的关联介绍（以挖掘机的 Power Tree 为例）

VCC3V3_SYS 为 VDD_CPU_B、VDD_CPU_L、VDD_GPU、VDD_LOG、VDD_0V9、VDD_CENTER、VCC_DDR、VCC_1V8（VCC3V3_S3）等供电。由图 1-1 可以看出，Audio SPK、USB2.0 HOST、FUSB302、Backlight、EDP/MIPI LCD 是单独供电的，所以可以粗略估算出单独供电的这几路功耗为：BTA 功耗 - VCC3V3_SYS 功耗。

VDD_CPU_B 为 CPU 大核供电，当这路功耗差异较大时，需要查下 CPU 大核是否一直跑在高频率的档位，CPU 大核每档电压是否合理。

VDD_CPU_L 为 CPU 小核供电，当这路功耗差异较大时，需要查下 CPU 小核是否一直跑在高频率的档位，CPU 小核每档电压是否合理。

VDD_GPU 为 GPU 供电，这路功耗差异较大时，需要查下 GPU 是否定频到高频率的档位，同时确认 GPU 每档电压是否合理。

VDD_LOG 为 MIPI CSI、ISP、CLK 等供电，当功耗数据较大时，查看下 clk 是否正确配置、power domain 是否正确配置。VDD_LOG 电压建议在 0.94V-1V。

VDD_0V9 为 usb2.0 USB_AVDD_0V9、EDP_AVDD_0V9、EMMC_CORE 等供电。

VDD_CENTER 为 phy 和 ddr control 供电，这路功耗数据差异较大时，查看下 DDR 的频率是否有问题，确认 phy 和 ddr 控制器是否正常。

VDD_DDR 为 ddr 颗粒供电，这路功耗数据差异较大时，确认 ddr 是否有变频，每档电压是否设置对了，有可能是 ddr 一直跑高频。对于 lpddr4 的颗粒，目前只有 416M 和 856M 两档，所以功耗会比较大。

VCC_1V8 为 TYPEC_AVDD_0V9 、 PLL_AVDD_0V9 、 AVDD0V9_DDRPLL 、HDMI_AVDD_0V9、EMMC、SENSOR、APIO1/APIO3 等供电。这路功耗较大时，需要检查外设功耗是否正常、gpio 电平是否配对。

3 运行功耗 debug 介绍

3.1 实时查询系统负载

RK3399 Android7.1 行业 sdk V2.1.0 后，集成了 hardware_monitor.sh 脚本，可以查询以下信息：CPU 当前频率、负载、温度；GPU 当前频率、负载、温度；DDR 当前频率、负载；UI 刷新率；信息运行时间等。在使用 hardware_monitor.sh 脚本时，需要 su 下。图 1-2 是 hardware_monitor.sh 运行时的输出信息。

注意：

1.有些客户使用 lpddr4，没有打开 ddr 变频功能，需要在 hardware_monitor.sh 把 ddr 信息获取部分删掉才能使用。

2.user 版本固件，这个脚本没办法使用。

```
su
rk3399_all:/ # hardware_monitor.sh
hardware_monitor.sh

Hardware Monitor for RK3399 , Version: 1.7
        F - Freq(MHz)
        L - Load(%)
        T - Temperature(C)

[Model]: rk3399-all
[Firmware]: rk3399_all-userdebug 7.1.2 NHG47K eng.zwp.20190723.174420 test-keys
[Kernel]: Linux version 4.4.126 (zwp@rk-intel-1) (gcc version 6.3.1 20170404 (Linaro GCC 6.3-2017.05) ) #394 SMP PREEMPT Tue Jul 23 17:41:04 CST 2019


UPTIME(s)       CPU(LF/BF/L/T)  GPU(F/L/T)      VPU/HEVC(F)     DDR(F/L)        FPS
5:05,   1008/408/01/32  200/00/32      000/000         528/19          0.0
5:05,   1200/816/18/33  200/46/32      000/000         800/19          58.0
5:05,   1200/816/27/33  200/10/32      000/000         528/24          59.9
5:05,   1416/408/27/32  200/43/32      000/000         800/39          58.2
5:05,   1416/408/32/33  200/32/32      000/000         800/25          58.3
5:05,   1416/408/26/33  200/55/32      000/000         528/30          57.9
5:05,   1200/408/25/33  200/46/33      000/000         800/27          60.0
5:06,   1416/408/21/33  200/38/32      000/000         600/31          58.0
5:06,   1200/408/18/33  200/00/32      000/000         528/22          57.9
```

图 1-2 hardware_monitor.sh 信息

参数说明：

UPTIME:系统运行时间

CPU L：小核 CPU 频率，CPU B：大核 CPU 频率，CPU L：CPU 负载，CPU T：CPU 温度

GPU F：GPU 频率，GPU L：GPU 负载，GPU T：GPU 温度

VPU/HEVC（F）：目前没用

DDR F：DDR 频率，DDR L：DDR 负载

FPS：UI 刷新率

3.2 实时查询 CPU 占用

RK3399 Android7.1 行业 sdk V2.1.7 后，替换了系统原生 top 工具，可以通过命令：top -t -m 10 来查看 CPU 占用情况，可以查看线程运行在哪个 CPU 核上，占用率多少。图 1-3 是 top 打印的系统线程信息。

参数：-t 表示打印线程占用信息，-m 10 表示打印 cpu 占用率前 10 的线程。

```
  PID   TID PR CPU% S     VSS     RSS PCY UID      Thread          Proc
 1607 14639  0  10% S 1833472K 163616K  fg u0_a45   CameraThread    com.alipay.zoloz.smile
 1607 14599  5   8% R 1833472K 163616K  fg u0_a45   pool-10-thread- com.alipay.zoloz.smile
 1607 14641  2   2% S 1833472K 163616K  fg u0_a45   Thread-134      com.alipay.zoloz.smile
 1607 14633  1   1% S 1833472K 163616K  fg u0_a45   CameraThread    com.alipay.zoloz.smile
 1607  1623  3   0% R 1835368K 165540K  fg u0_a45   HeapTaskDaemon  com.alipay.zoloz.smile
 1557  1653  3   0% S 1796780K 169904K  fg u0_a43   RenderThread    zoloz.phone.android.alipay.com.dragonfly
 1607  1668  1   0% S 1835368K 164592K  fg u0_a45   Thread-2        com.alipay.zoloz.smile
 1557  1557  3   0% S 1796780K 169904K  fg u0_a43   y.com.dragonfly zoloz.phone.android.alipay.com.dragonfly
  229   229  3   0% S 301132K  56780K  fg system   surfaceflinger  /system/bin/surfaceflinger
15008 15008  4   0% R    796K    400K  fg root     top             top
 2329  2329  2   0% S   7716K   2596K  fg root     hardware_monito /system/bin/sh
 1692  1736  0   0% S 1712524K  98944K  bg u0_a45   Thread-2        com.alipay.zoloz.smile:push
  229   257  3   0% S 301132K  56780K  fg system   Binder:229_1    /system/bin/surfaceflinger
  464 14577  1   0% S 2218096K 187188K  fg system   Binder:464_E    system_server
 1557  1666  3   0% S 1796780K 169904K  fg u0_a43   mali-cmar-backe zoloz.phone.android.alipay.com.dragonfly
```

图 1-3 top 系统信息

打印信息说明：

PID：进程 ID，TID：线程 ID

PR：当前线程跑在哪个 CPU 上

CPU%：当前线程 CPU 占用率，当某个线程占用 CPU 接近 17%（6 个 CPU 核，100/6=17%），表示当前线程已经在这个 CPU 核上跑满。

Thread：线程名称，Proc：进程名称

注意：

通过 top 工具查看某个应用对于系统负载情况，假如 apk 的某个线程对 CPU 占用较大，而且 apk 线程较少，其他线程 CPU 占用率低，可以考虑把这个线程的任务拆分到多个线程里面去处理，可以降低 CPU 负载，达到降低功耗目的。

3.3 CLK 检查

查看各个子系统 clk 状态，可以看到时钟频率、enable_count、prepare_count、父子时钟关系等。常用的 clk 命令如下（以 saradc 为例）：

查询 CLK Tree 命令：

cat /d/clk/clk_summary

设置频率（单位 HZ）：

echo 24000000 > /sys/kernel/debug/clk_saradc/clk_rate

获取频率：

cat /sys/kernel/debug/clk_saradc/clk_rate

打开 clk：

echo 1 > /sys/kernel/debug/clk_saradc/clk_enable_count

关闭 clk：

echo 0 > /sys/kernel/debug/clk_saradc/clk_enable_count

注意：引用计数 enable_count,是驱动主动申请开启时钟后（clk_enable）引用计数就会+1。但是有一些默认常开的时钟，引用计数可能是 1 也可能是 0，这种查功耗可以不用太关心。主要关心没有使用的设备的 CLK 的 enable_cnt 是否为 0，如果不为 0，可以手动用关闭 CLK 的命令去关闭调试。

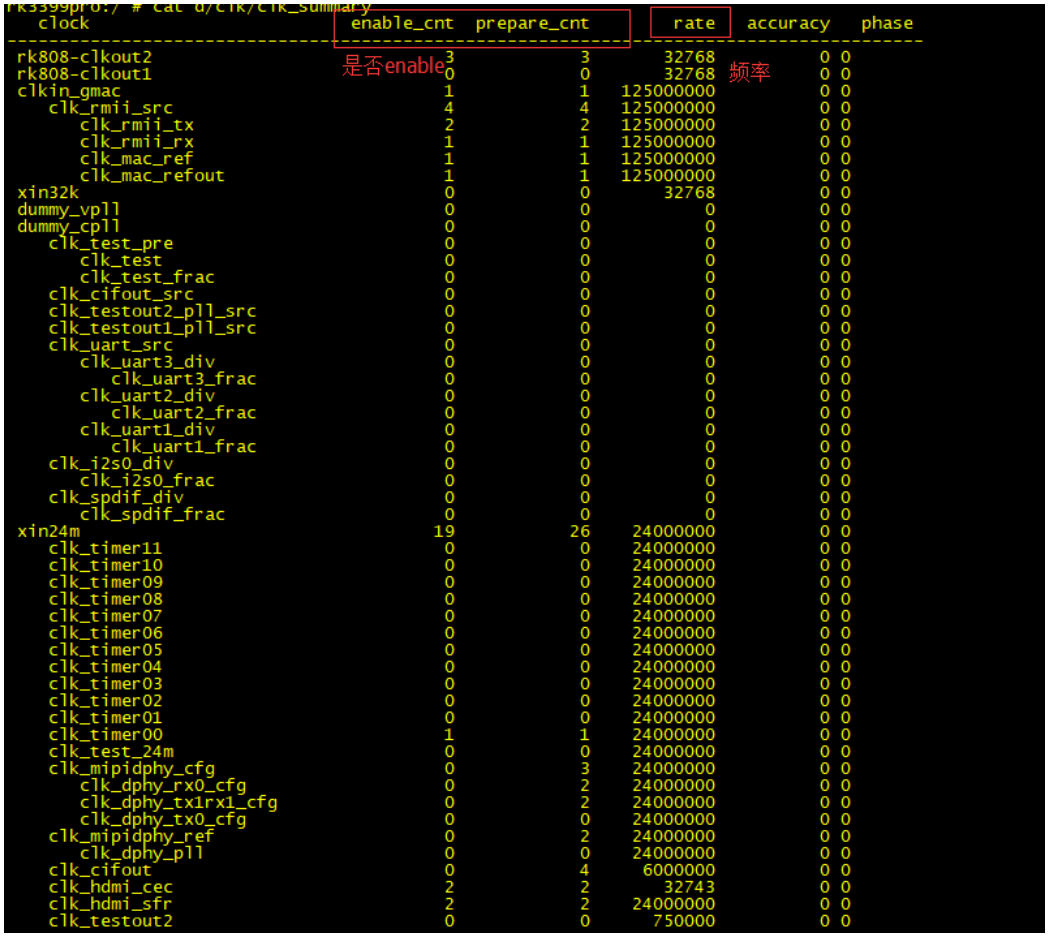

图 1-4 CLK Tree

dts 配置 clk，以 vopb 配置为例：

```
vopb: vop@ff900000 {
        clocks = <&cru ACLK_VOP0>, <&cru DCLK_VOP0>, <&cru HCLK_VOP0>,
<&cru DCLK_VOP0_DIV>;
        clock-names = "aclk_vop", "dclk_vop", "hclk_vop", "dclk_source";
};
```

3.4 PD 检查

确认各个 PD 当前状态，没有使用的模块，其状态要是 suspend，然后这个 pd 下面所有的 devices 都 suspend 后，这个 PD 就会关闭，详细说明见图 1-5。

查询 PD summary 命令：

cat /d/pm_genpd/pm_genpd_summary

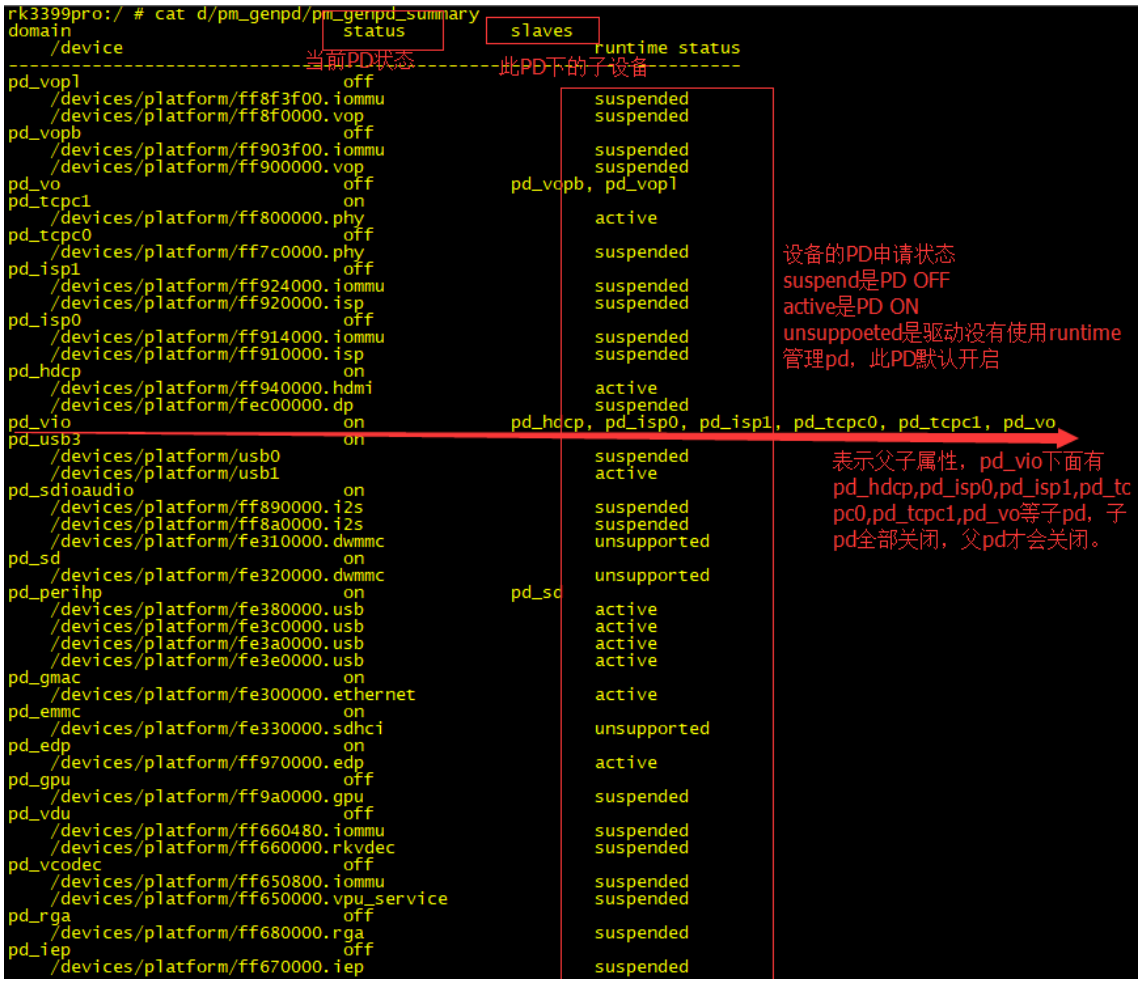

图 1-5 PD summary 信息

dts 配置（以 vopb 为例）：

```
vopb: vop@ff900000 {
        power-domains = <&power RK3399_PD_VOPB>;
};
```

注意：如果 dts 节点中没有引用 PD，这个 PD 认为没有 devices 使用，开机后会被框架关闭。如果上面 dts 节点中，增加了 pd 的引用，但是 vop 驱动没有 runtime 的操作，这个 pd 是常开的（因为认为这个 device 是不支持的 runtime，状态会变成 unsupported）。如果 dts 节点中增加 pd 引用，驱动中也有 runtime 操作，那么这个 pd 的状态就是看驱动主动申请是开还是关。

降低运行功耗总结：查询 pd summary 和 clk tree，要求没有使用的模块，PD 和 CLK 需要关闭，避免内部 mos 管的漏电。

举例说明：一级待机时，vop0 模块没有使用。那么这个模块对应的 clk 和 pd 是需要关闭的。clk（dclk_vop0 和 aclk_vop0 的引用计数都是 0）如下：

```
rk3399pro:/ # cat d/clk/clk_summary | grep vop
       dclk_vop1_div                0        2     6000000          0 0
          dclk_vop1                 0        1     6000000          0 0
          dclk_vop1_frac            0        0      300000          0 0
          aclk_vop1_pre             2        3   400000000          0 0
             aclk_vop1_noc          1        1   400000000          0 0
             aclk_vop1              0        4   400000000          0 0
             hclk_vop1_pre          1        2   100000000          0 0
                hclk_vop1_noc       1        1   100000000          0 0
                hclk_vop1           0        4   100000000          0 0
          clk_vop1_pwm              0        0   100000000          0 0
          clk_vop0_pwm              0        0   100000000          0 0
          aclk_vop0_pre             2        3   400000000          0 0
             aclk_vop0_noc          1        1   400000000          0 0
             aclk_vop0              0        4   400000000          0 0
             hclk_vop0_pre          1        2   100000000          0 0
                hclk_vop0_noc       1        1   100000000          0 0
                hclk_vop0           0        4   100000000          0 0
          dclk_vop0_div             0        2   200000000          0 0
             dclk_vop0              0        1   200000000          0 0
             dclk_vop0_frac         0        0    10000000          0 0
```

图 1-6 vop0 clk 信息

pd（pd_vop0 是状态是 off）

```
172.16.12.236 (1) | 10.10.10.110 | 172.16.12.236 (1) (1) | Serial-COM8 | 172.16.12.236 (1) (2) | 172.16.12.236 (1) (3) | 10.10
domain                          status          slaves
    /device                                             runtime status
------------------------------------------------------------------------
pd_vopl                         off
    /devices/platform/ff8f3f00.iommu                    suspended
    /devices/platform/ff8f0000.vop                      suspended
pd_vopb                         off
    /devices/platform/ff903f00.iommu                    suspended
    /devices/platform/ff900000.vop                      suspended
pd_vo                           off             pd_vopb, pd_vopl
pd_tcpc1                        on
    /devices/platform/ff800000.phy                      active
pd_tcpc0                        off
    /devices/platform/ff7c0000.phy                      suspended
```

图 1-7 vop0 PD 信息

4 待机功耗 debug 介绍

4.1 休眠唤醒问题排查

无法进入休眠：

1. 确认是否有 wake_lock，查看命令如下

cat /sys/power/wake_lock（只能看 Android 的锁无法看到 kernel 中设置的锁）

cat /sys/kernel/debug/wakeup_sources（有锁都可以查看）

```
1|rk3399_all:/ #
1|rk3399_all:/ # echo xxxx > /sys/power/wake_lock
rk3399_all:/ # cat /sys/kernel/debug/wakeup_sources
name                                    active_count    event_count     wakeup_count    expire_count    active_since
revent_suspend_time
xxxx                                    1               1               0               0               12877
usb@fe800000                            0               0               0               0               0
wlan_wd_wake                            3130458         3130458         0               0               0
wlan_wake                               2625896         2625896         0               0               0
5               0
PowerManagerService.WakeLocks           12              12              0               0               312437400
0646            0
PowerManagerService.Broadcasts          3               3               0               0               0
event1                                  0               0               0               0               0
event0                                  104             104             0               0               0
event2                                  4               4               0               0               0
eventpoll                               108             108             0               0               0
KeyEvents                               10480           10480           0               0               0
[timerfd]                               0               0               0               0               0
[timerfd]                               0               0               0               0               0
[timerfd]                               12457           12457           0               0               0
[timerfd]                               95              95              0               0               0
```

图 1-8 wakeup_sources 信息

2. 休眠过程中，有中断产生，结合 log 查看是哪个模块的中断引起。

3. 确认是否有设置 2S 内会有闹钟中断产生

```
static int alarmtimer_suspend(struct device *dev)
{
    struct rtc_time tm;
    ktime_t min, now;
    unsigned long flags;
    struct rtc_device *rtc;
    int i;
    int ret;

    spin_lock_irqsave(&freezer_delta_lock, flags);
    min = freezer_delta;
    freezer_delta = ktime_set(0, 0);
    spin_unlock_irqrestore(&freezer_delta_lock, flags);

    rtc = alarmtimer_get_rtcdev();
    /* If we have no rtcdev, just return */
    if (!rtc)
        return 0;

    /* Find the soonest timer to expire*/
    for (i = 0; i < ALARM_NUMTYPE; i++) {
        struct alarm_base *base = &alarm_bases[i];
        struct timerqueue_node *next;
        ktime_t delta;

        spin_lock_irqsave(&base->lock, flags);
        next = timerqueue_getnext(&base->timerqueue);
        spin_unlock_irqrestore(&base->lock, flags);
        if (!next)
            continue;
        delta = ktime_sub(next->expires, base->gettime());
        if (!min.tv64 || (delta.tv64 < min.tv64))
            min = delta;
    }
    if (min.tv64 == 0)
        return 0;

    if (ktime_to_ns(min) < 2 * NSEC_PER_SEC) {
        __pm_wakeup_event(ws, 2 * MSEC_PER_SEC);
        return -EBUSY;
    }

    /* Setup an rtc timer to fire that far in the future */
    rtc_timer_cancel(rtc, &rtctimer);
    rtc_read_time(rtc, &tm);
    now = rtc_tm_to_ktime(tm);
```

2S内就唤醒就不让睡。

唤醒时间过长：

以 rk3399 为例，从按下 power 键到背光 pwm 有波形输出的时间在 972ms，如图 1-9。

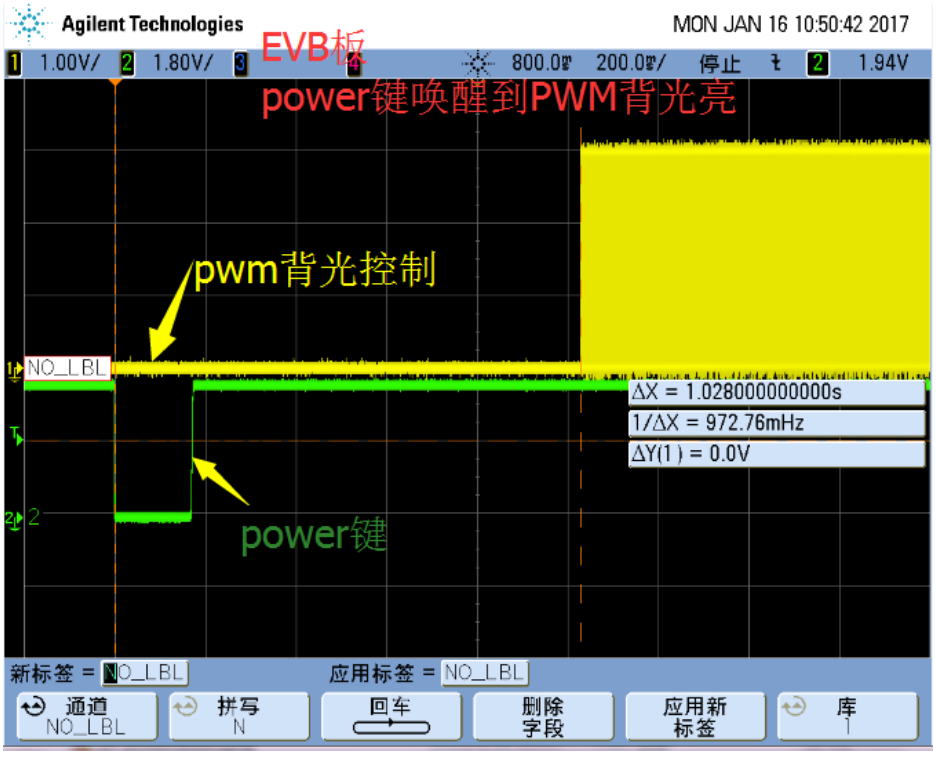

图 1-9 唤醒流程波形图

若发现休眠唤醒时间太长可以通过以下方法定位：

1. 命令的形式：

echo N > /sys/module/printk/parameters/console_suspend

echo 1 > /sys/power/pm_print_times

打印出每个设备休眠的耗时

```
[   70.672395] calling  rfkill2+ @ 277, parent: mmc2:0001:2, cb: rfkill_suspend
[   70.673053] call rfkill2+ returned 0 after 1 usecs
[   70.673511] calling  rfkill1+ @ 277, parent: phy0, cb: rfkill_suspend
[   70.674100] call rfkill1+ returned 0 after 0 usecs
[   70.674576] calling  phy0+ @ 180, parent: mmc2:0001:2, cb: wiphy_suspend
[   70.674604] call phy0+ returned 0 after 6 usecs
[   70.675607] calling  mmc2:0001:3+ @ 277, parent: mmc2:0001, cb: pm_generic_suspend
[   70.675628] call mmc2:0001:3+ returned 0 after 0 usecs
[   70.675646] calling  mmc2:0001:2+ @ 277, parent: mmc2:0001, cb: pm_generic_suspend
[   70.675658] bcmsdh_sdmmc_suspend Enter func->num=2
[   70.675669] bcmsdh_sdmmc_suspend Exit
[   70.675679] call mmc2:0001:2+ returned 0 after 20 usecs
[   70.675695] calling  mmc2:0001:1+ @ 277, parent: mmc2:0001, cb: pm_generic_suspend
[   70.675706] bcmsdh_sdmmc_suspend Enter func->num=1
[   70.675716] call mmc2:0001:1+ returned 0 after 8 usecs
[   70.675736] calling  mmc2:0001+ @ 277, parent: mmc2, cb: mmc_bus_suspend
[   70.675752] call mmc2:0001+ returned 0 after 4 usecs
[   70.675775] calling  input1+ @ 277, parent: rockchip-key, cb: input_dev_suspend
[   70.675790] call input1+ returned 0 after 2 usecs
[   70.675821] calling  es8316-sound+ @ 277, parent: platform, cb: platform_pm_suspend
[   70.680107] call es8316-sound+ returned 0 after 4171 usecs
[   70.680130] calling  dmc+ @ 277, parent: platform, cb: platform_pm_suspend
[   70.680154] call dmc+ returned 0 after 11 usecs
[   70.680173] calling  fe310000.dwmmc+ @ 277, parent: platform, cb: pm_generic_suspend
[   70.680187] call fe310000.dwmmc+ returned 0 after 1 usecs
[   70.680297] calling  usb4+ @ 180, parent: fe3e0000.usb, cb: usb_dev_suspend
[   70.736885] call usb4+ returned 0 after 55234 usecs
[   70.736970] calling  fe3e0000.usb+ @ 277, parent: platform, cb: pm_generic_suspend
[   70.736999] call fe3e0000.usb+ returned 0 after 14 usecs
[   70.737102] calling  usb3+ @ 180, parent: fe3a0000.usb, cb: usb_dev_suspend
[   70.793279] call usb3+ returned 0 after 54834 usecs
[   70.793409] calling  fe3a0000.usb+ @ 277, parent: platform, cb: pm_generic_suspend
[   70.793438] call fe3a0000.usb+ returned 0 after 14 usecs
[   70.793606] calling  usb2+ @ 180, parent: fe3c0000.usb, cb: usb_dev_suspend
```

2. 打开 DPM_WATCHDOG_TIMEROUT

```
Symbol: DPM_WATCHDOG_TIMEOUT [=5]
Type  : integer
Range : [1 120]
Prompt: Watchdog timeout in seconds
  Location:
    -> Power management options
      -> Device power management core functionality (PM [=y])
        -> Power Management Debug Support (PM_DEBUG [=y])
(2)       -> Device suspend/resume watchdog (DPM_WATCHDOG [=y])
  Defined at kernel/power/Kconfig:211
  Depends on: DPM_WATCHDOG [=y]
```

超时时间可配置，但只能精确到秒。

```
        [ ]     Extra PM attributes in sysfs for low-level debugging/testing
        [ ]     Test suspend/resume and wakealarm during bootup
        [*]     Device suspend/resume watchdog
        (60)      watchdog timeout in seconds (NEW)
        [ ] Enable workqueue power-efficient mode by default
```

```
--- a/drivers/mfd/rk808.c
+++ b/drivers/mfd/rk808.c
@@ -842,12 +842,12 @@ err_irq:
        regmap_del_irq_chip(client->irq, rk808->irq_data);
        return ret;
 }
-
+#include <linux/delay.h>
 static int rk808_suspend(struct device *dev)
 {
        int i, ret;
        struct rk808 *rk808 = i2c_get_clientdata(rk808_i2c_client);
-
+       mdelay(8000);
        for (i = 0; i < suspend_reg_num; i++) {
                ret = regmap_update_bits(rk808->regmap,
                                         suspend_reg[i].addr,
(END)
```

```
[   31.897864] rk808 0-001c: **** DPM device timeout ****
[   31.898102] [<c010f608>] (unwind_backtrace) from [<c010b800>] (show_stack+0x10/0x14)
[   31.898248] [<c010b800>] (show_stack) from [<c0504c9c>] (dpm_watchdog_handler+0x20/0x4c)
[   31.898388] [<c0504c9c>] (dpm_watchdog_handler) from [<c018cb90>] (call_timer_fn+0xa0/0x20c)
[   31.898514] [<c018cb90>] (call_timer_fn) from [<c018cf3c>] (run_timer_softirq+0x240/0x2cc)
[   31.898636] [<c018cf3c>] (run_timer_softirq) from [<c012a45c>] (__do_softirq+0x138/0x354)
[   31.898750] [<c012a45c>] (__do_softirq) from [<c012a900>] (irq_exit+0x88/0xf8)
[   31.898870] [<c012a900>] (irq_exit) from [<c017b884>] (__handle_domain_irq+0x8c/0xb0)
[   31.898986] [<c017b884>] (__handle_domain_irq) from [<c010142c>] (gic_handle_irq+0x44/0x74)
[   31.899086] [<c010142c>] (gic_handle_irq) from [<c010c314>] (__irq_svc+0x54/0x90)
[   31.899153] Exception stack(0xdc90bbf0 to 0xdc90bc38)
[   31.899235] bbe0:                                     dc90bc50 00000000 fd640800 c1298c80
[   31.899341] bc00: dc90bc50 3835d96f 00005dbf c05239d4 ddb158d0 ddb34e20 c0ee31e3 c1258f7c
[   31.899434] bc20: c05239d4 dc90bc40 c03c86c0 c010edf4 a00f0113 ffffffff
[   31.899543] [<c010c314>] (__irq_svc) from [<c010edf4>] (arch_timer_read_counter_long+0x8/0x18)
[   31.899674] [<c010edf4>] (arch_timer_read_counter_long) from [<c03c86c0>] (read_current_timer+0x20/0x38)
[   31.899794] [<c03c86c0>] (read_current_timer) from [<c03c8708>] (__timer_delay+0x30/0x6c)
[   31.899918] [<c03c8708>] (__timer_delay) from [<c0523a04>] (rk808_suspend+0x30/0xac)
[   31.900043] [<c0523a04>] (rk808_suspend) from [<c0504d80>] (dpm_run_callback+0xb8/0x23c)
[   31.900154] [<c0504d80>] (dpm_run_callback) from [<c0505b5c>] (__device_suspend+0x254/0x340)
[   31.900263] [<c0505b5c>] (__device_suspend) from [<c0507694>] (dpm_suspend+0x12c/0x38c)
[   31.900373] [<c0507694>] (dpm_suspend) from [<c0176df0>] (suspend_devices_and_enter+0x78/0x350)
[   31.900484] [<c0176df0>] (suspend_devices_and_enter) from [<c0177734>] (pm_suspend+0x66c/0x78c)
[   31.900585] [<c0177734>] (pm_suspend) from [<c017591c>] (state_store+0x40/0x68)
[   31.900691] [<c017591c>] (state_store) from [<c0297b18>] (kernfs_fop_write+0x148/0x1ac)
[   31.900804] [<c0297b18>] (kernfs_fop_write) from [<c0234558>] (__vfs_write+0x2c/0xf4)
[   31.900907] [<c0234558>] (__vfs_write) from [<c0234d50>] (vfs_write+0xac/0x17c)
[   31.901002] [<c0234d50>] (vfs_write) from [<c0235588>] (SyS_write+0x4c/0xa4)
[   31.901104] [<c0235588>] (SyS_write) from [<c0107180>] (ret_fast_syscall+0x0/0x3c)
[   31.901322] Kernel panic - not syncing: rk808 0-001c: unrecoverable failure
[   31.901322]
[   32.134735] CPU: 0 PID: 547 Comm: system_server Not tainted 4.4.77 #623
[   32.141381] Hardware name: Rockchip (Device Tree)
[   32.146180] [<c010f608>] (unwind_backtrace) from [<c010b800>] (show_stack+0x10/0x14)
[   32.153998] [<c010b800>] (show_stack) from [<c03ca7d4>] (dump_stack+0x7c/0x9c)
[   32.161311] [<c03ca7d4>] (dump_stack) from [<c01ec0d4>] (panic+0x94/0x208)
[   32.168265] [<c01ec0d4>] (panic) from [<c0504cc0>] (dpm_watchdog_handler+0x44/0x4c)
[   32.175999] [<c0504cc0>] (dpm_watchdog_handler) from [<c018cb90>] (call_timer_fn+0xa0/0x20c)
[   32.184515] [<c018cb90>] (call_timer_fn) from [<c018cf3c>] (run_timer_softirq+0x240/0x2cc)
```

休眠唤醒排查技巧：

1. 确认系统有没有进入休眠：pmic_sleep 高电平系统处在休眠模式；

2. 确认休眠时各路电压是不是正常，在 pmic_sleep 脚为高的情况下，量各路的电压；

3. 若是高概率问题，可以把 rockchip,sleep-debug-en 赋值为 1，休眠唤醒能打印出更多的 log，便于确认问题；

4. 确认休眠时，断某一路电是否引起系统不稳定，排查方法：去掉 rockchip,sleep-mode-config 中 RKPM_SLP_AP_PWROFF 的配置。

4.2 待机功耗排查

设备状态检查：

确认是否有设备没有休眠或者频繁休眠唤醒。

电源断电检查：

对于很多外设在二级待机，没有使用的时候电源可以关闭。

关闭方法：

（1）设备主动关闭

如 LCD 背光的电源或者 LCD 的 3.3V 的 IO 电源，在灭屏的时候就会在 VOP 驱动中使用 regulator_disable 或者拉高拉低使用 PIN 脚的方式，关闭输出。

（2）使用 PMIC 的，依赖二级待机 SLEEP 脚拉高关闭

PMIC 的节点中，配置 regulator-off-in-suspend;属性，就会在二级待机 sleep 脚拉高后关闭输出。

如下图：

```
vdd_cpu_l: DCDC_REG2 {
        regulator-always-on;
        regulator-boot-on;
        regulator-min-microvolt = <750000>;
        regulator-max-microvolt = <1350000>;
        regulator-ramp-delay = <6001>;
        regulator-name = "vdd_cpu_l";
        regulator-state-mem {
            regulator-off-in-suspend;
        };
    };
```

具体哪些可以关闭，要结合实际硬件。

5 温控策略介绍

5.1 Thermal 概述

Thermal 是内核开发者定义的一套支持根据指定 governor 控制系统温度，以防止芯片过热的框架模型。Thermal framework 由 governor、core、cooling device、sensor driver 组成，软件架构如下：

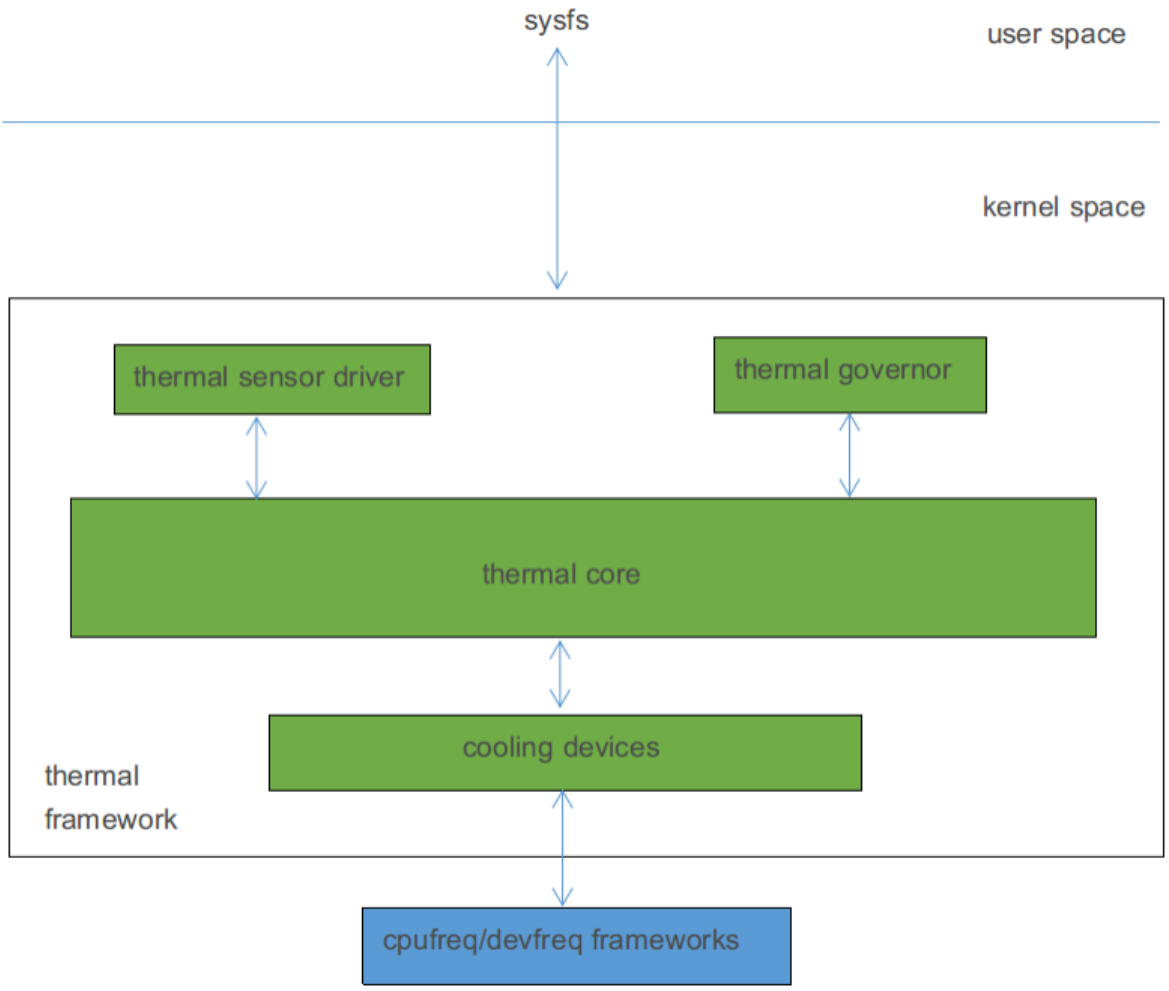

图 1-10 Thermal 框架图

5.2 温控参数调整

假设我们希望 70 度以上温控开始工作（更频繁地获取温度），最高温度不超过 85 度，超过 115 度系统重启。于是要做如下配置：

```
thermal_zones: thermal-zones {
    soc_thermal: soc-thermal {
        ....
        trips {
            threshold: trip-point-0 {
                /*
                 * 70度以上温控开始工作，缩短了获取温度的是时间间隔，但不一定马上降频，
                 * 还跟sustainable-power有关
                 */
                temperature = <70000>; /* millicelsius */
                hysteresis = <2000>; /* millicelsius */
                type = "passive";
            };
            target: trip-point-1 {
                /* 期望最高温度不超过85度 */
                temperature = <85000>; /* millicelsius */
                hysteresis = <2000>; /* millicelsius */
                type = "passive";
            };
            soc_crit: soc-crit {
                /* 超过115度系统重启 */
                temperature = <115000>; /* millicelsius */
                hysteresis = <2000>; /* millicelsius */
```

5.3 用户态接口介绍

用户态接口在/sys/class/thermal/目录下，具体内容和 dtsi 中 thermal zone 节点的配置对应。RK3399 平台有两个子节点，对应/sys/class/thermal/目录下就会有 thermal_zone0 和 thermal_zone1 子目录。通过用户态接口可以切换温控策略，查看当前温度等。

以 RK3399 为例子，/sys/class/thermal/thermal_zone0/目录下包含如下常用的信息：

```
temp                    /* 当前温度 */
available_policies      /* 支持的温控策略 */
policy                  /* 当前使用的温控策略 */
sustainable_power       /* 期望的最高温度下对应的power值 */
integral_cutoff         /* PID算法中I的触发条件：当前温度-期望的最高温度<integral_cutoff */
k_d                     /* PID算法中计算D的时候用的参数 */
k_i                     /* PID算法中计算I的时候用的参数 */
k_po                    /* PID算法中计算P的时候用的参数 */
k_pu                    /* PID算法中计算P的时候用的参数 */
mode                    /* enabled：自带定时获取温度，判断是否需要降频。disabled关闭该功能 */
type                    /* 当前thermal zone的类型 */
/* 不同的温度阀值，对应trips节点的配置 */
trip_point_0_hyst
trip_point_0_temp
trip_point_0_type
trip_point_1_hyst
trip_point_1_temp
trip_point_1_type
trip_point_2_hyst
trip_point_2_temp
trip_point_2_type
/* 不同cooling devic的状态，对应cooling-maps节点的配置 */
cdev0                   /* 代表一个cooling devic，有的平台还有cdev1、cdev2等 */
    cur_state           /* 该cooling device当前频率的档位 */
    max_state           /* 该cooling device最多有几个档位 */
    type                /* 该cooling device的类型 */
cdev0_weight            /* 该cooling devic在计算power时扩大的倍数 */
```

5.4 关温控

方法一：menuconfifig 中默认温控策略设置为 user_space。

```
<*> Generic Thermal sysfs driver  --->
    --- Generic Thermal sysfs driver
    [*]   APIs to parse thermal data out of device tree
    [*]   Enable writable trip points
          Default Thermal governor (user_space)  --->  /* power_allocator改为user_space */
```

方法二：开机后通过命令关温控。

首先，把温控策略切换到 user_space，即把用户态接口下的 policy 节点改成 user_space；或者把 mode 设置成 disabled 状态；然后，解除频率限制，即将用户态接口下的所有 cdev 的 cur_state 设置为 0。

以 RK3399 为例，策略切换到 user_space：

```
echo user_space > /sys/class/thermal/thermal_zone0/policy
```

或者把 mode 设置成 disabled 状态：

```
echo disabled > /sys/class/thermal/thermal_zone0/mode
```

解除频率限制：

```
/* 具体有多少个cdev，根据实际情况修改 */
echo 0 > /sys/class/thermal/thermal_zone0/cdev0/cur_state
echo 0 > /sys/class/thermal/thermal_zone0/cdev1/cur_state
echo 0 > /sys/class/thermal/thermal_zone0/cdev2/cur_state
```

5.5 获取当前温度

直接查看用户态接口 thermal_zone0 或者 thermal_zone1 目录下的 temp 节点即可。

以 RK3399 为例，获取 CPU 温度，在串口中输入如下命令：

```
cat /sys/class/thermal/thermal_zone0/temp
```

获取 GPU 温度，在串口中输入如下命令：

```
cat /sys/class/thermal/thermal_zone1/temp
```

6 系统稳定性和功耗自排查 CheckList

序号	问题类别	排查点	备注	结果反馈

1	稳定性	检查各路电压是否正常	以太网断线、camera 拷机死机、hdmi in 拷机死机等问题	
2	功耗/稳定性	vdd_log 实测 0.95V		
3	稳定性	LPDDR4：VDD_CENTER 实测 0.9V		
4	稳定性	VDD_CPU_B，VDD_CPU_L，VDD_GPU，VDD_LOG,VDD_CENTER 电源纹波小于 10%（建议±50mV 以内）		
5	稳定性	CPU/GPU/DDR 尝试定频验证		
6	稳定性	CPU/GPU 尝试按 12.5mV 步进抬压验证		
7	稳定性	PMUIO2 电压配置为 3.0V 时，PMUIO2_VOLSEL 脚需上拉 10K	Pin V30 PMUIO2_VOLSEL 脚是配置 PMUIO2 电压的管脚	
8	稳定性	dts 的 io-domain 配置跟原理图是否匹配	参考 RKDocs/rk3399/rk3399_android7.1_软件开发指南_vxxx.pdf 第 9.1.2 节内容	
9	功耗	检查 DTS 里没用到的模块是否关闭	结合实际项目把一些没用到的模块 disabled，例如 ISP、EDP、pcie 等。适当降低 lcd 显示屏的刷新率，可以降低功耗	
10	功耗/稳定性	用 TOP 指令检查 APK 的 CPU 占用情况	运行客户自己开发的 apk，出现 cpu 某个核负载特别大的情况，可以尝试把 apk 任务拆分成多个线程，这样均摊到各个 cpu 核上降低 CPU 的频率，从而降低功耗。	
11	功耗/稳定性	散热方案是否给 RK review 过		
12	性能/功耗/稳定性	原理图/PCB 是否给 RK review 过		
13	性能/功耗/稳定性	反馈 SDK 软件版本		
14	性能/功耗/稳定性	用 LPDDR4 时，检查 DDR 版本	确认更新到 V2.3（含）之后	

7 功耗的优化点

基本功耗优化点有：背光亮度，屏幕刷新率降低，关闭一些没用的外设，调整 target load，根据温度限 CPU 频率等。LPDDR4 机器，确认 ddr 变频是否有生效。

8 FAQ

8.1 CPU 调频异常

设置的 CPU 电压不支持，RK pmic 步进电压是 12.5mv，所以抬电压时，只能是 12.5mv 的整数倍。

```
[    3.132492] core: _opp_supported_by_regulators: OPP minuV: 880000 maxuV: 880000, not supported by regulator
[    3.132587] cpu cpu0: _opp_add: OPP not supported by regulators (408000000)
[    3.132903] core: _opp_supported_by_regulators: OPP minuV: 880000 maxuV: 880000, not supported by regulator
[    3.133048] cpu cpu0: _opp_add: OPP not supported by regulators (600000000)
[    3.133344] core: _opp_supported_by_regulators: OPP minuV: 880000 maxuV: 880000, not supported by regulator
[    3.133495] cpu cpu0: _opp_add: OPP not supported by regulators (816000000)
[    3.133820] core: _opp_supported_by_regulators: OPP minuV: 930000 maxuV: 930000, not supported by regulator
[    3.133941] cpu cpu0: _opp_add: OPP not supported by regulators (1008000000)
[    3.134245] core: _opp_supported_by_regulators: OPP minuV: 1005000 maxuV: 1005000, not supported by regulator
[    3.134388] cpu cpu0: _opp_add: OPP not supported by regulators (1200000000)
```

8.2 待机功耗高

dts 里面 ddr 变频没开，用的是最高频率档位在跑。

```
&dmc {
        center-supply = <&vdd_logic>;
        status = "okay";
};
```

8.3 整体运行功耗高

在客户支持过程中，遇到一些客户因为硬件板子纹波较大，把 cpu、gpu、ddr 电压抬的较多，导致功耗比 RK sdk 板子高比较多，每档按照 12.5mv 的倍数加。在系统稳定的前提下，抬的越少越好。

8.4 图形合成策略：用 HWC 合成会比 GPU 合成功耗低

查看系统属性 sys.hwc.compose_policy，0 是用 GPU 合成，6 是用 hwc 合成。建议能用 hwc 合成的场景，都用 hwc 合成。播放 4K 视频场景，用 hwc 合成会比 gpu 合成功耗小 4.2V-200mA 左右。

8.5 休眠功耗较大

检查 regulators 里面的电压配置，例如下面是一个 vdd_gpu 电压配置问题，会导致休眠功耗降不下来。建议客户尽量不要随意修改 regulators 里面的电压。

```
vdd_gpu: DCDC_REG2 {
        regulator-name = "vdd_gpu";
        regulator-always-on;
        regulator-boot-on;
        regulator-min-microvolt = <800000>;
        regulator-max-microvolt = <1250000>;
        regulator-ramp-delay = <6001>;
        regulator-state-mem {
                regulator-off-in-suspend;
                regulator-suspend-microvolt = <950000>;
        };
};
```

正确配置

```
vdd_2v3: DCDC_REG2 {
        regulator-name = "vdd_2v3";
        regulator-always-on;
        regulator-boot-on;
        regulator-min-microvolt = <1100000>;
        regulator-max-microvolt = <1100000>;
        regulator-ramp-delay = <6001>;
        regulator-state-mem {
                regulator-off-in-suspend;
                regulator-suspend-microvolt = <1100000>;
        };
};
```

8.6 60℃温箱拷机过不了，机器重启

检查过温保护温度是否正确，配置在 kernel/arch/arm64/boot/dts/rockchip/rk3399.dtsi。

```
trips {
        threshold: trip-point-0 {
                temperature = <70000>; /* millicelsius */
                hysteresis = <2000>; /* millicelsius */
                type = "passive";
        };
        target: trip-point-1 {
                temperature = <85000>; /* millicelsius */
                hysteresis = <2000>; /* millicelsius */
                type = "passive";
        };
        soc_crit: soc-crit {
                temperature = <115000>; /* millicelsius */
                hysteresis = <2000>; /* millicelsius */
                type = "critical";
        };
};
```

软件过温保护

```
tsadc: tsadc@ff260000 {
        compatible = "rockchip,rk3399-tsadc";
        reg = <0x0 0xff260000 0x0 0x100>;
        interrupts = <GIC_SPI 97 IRQ_TYPE_LEVEL_HIGH 0>;
        rockchip,grf = <&grf>;
        clocks = <&cru SCLK_TSADC>, <&cru PCLK_TSADC>;
        clock-names = "tsadc", "apb_pclk";
        assigned-clocks = <&cru SCLK_TSADC>;
        assigned-clock-rates = <750000>;
        resets = <&cru SRST_TSADC>;
        reset-names = "tsadc-apb";
        pinctrl-names = "init", "default", "sleep";
        pinctrl-0 = <&otp_gpio>;
        pinctrl-1 = <&otp_out>;
        pinctrl-2 = <&otp_gpio>;
        #thermal-sensor-cells = <1>;
        rockchip,hw-tshut-temp = <120000>;
        status = "disabled";
};
```

硬件保护关机温度

8.7 高温拷机，CPU 温度到 80℃左右，机器重启

软件过温保护已经配置为 115℃，硬件过温保护已经配置为 120℃，但是出现 CPU 温度在 80℃时出现硬件关机，log 里面没有任何打印。这种现象可能是因为下图的 MOS 管在 80℃时直接导通，把 RK808 的 reset 脚拉高，导致硬件重启。解决办法：换好的 mos 管物料。