

Rockchip

Thermal 开发指南

发布版本:1.01

日期:2017.04

前言

概述

本文档主要介绍 RK 平台 Thermal 配置的调试方法。

产品版本

产品名称	内核版本
RK3399	Linux4.4

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

日期	版本	作者	修改说明
2016-07-01	V1.0	XF,HYZ	初始发布
2017-04-28	V1.0.1	XF,HYZ	增加了 IPA 参数调试方法和解除频率的方法介绍

目录

前言	I
目录	II
1 概述	1-1
2 重要概念	2-1
3 配置方法	3-1
3.1 Tsadc 配置	3-1
3.1.1 Menuconfig 配置	3-1
3.1.2 DTS 配置	3-1
3.2 power_allocator 策略配置	3-3
3.2.1 默认使用 power_allocator 策略	3-3
3.2.2 CPU 开启温控	3-4
3.2.3 GPU 开启温控	3-5
3.2.4 thermal_zone 配置	3-6
3.2.5 参数调试方法总结	3-8
4 调试接口	4-1
4.1 关温控	4-1
4.2 获取当前温度	4-1

1 概述

本章主要描述 Thermal 的相关的重要概念、配置方法和调试接口。

2 重要概念

在 Linux 内核中，定义一套温控框架 linux Generic Thermal Sysfs Drivers，它可以通过不同的策略控制系统的温度，目前常用的有以下几种策略：

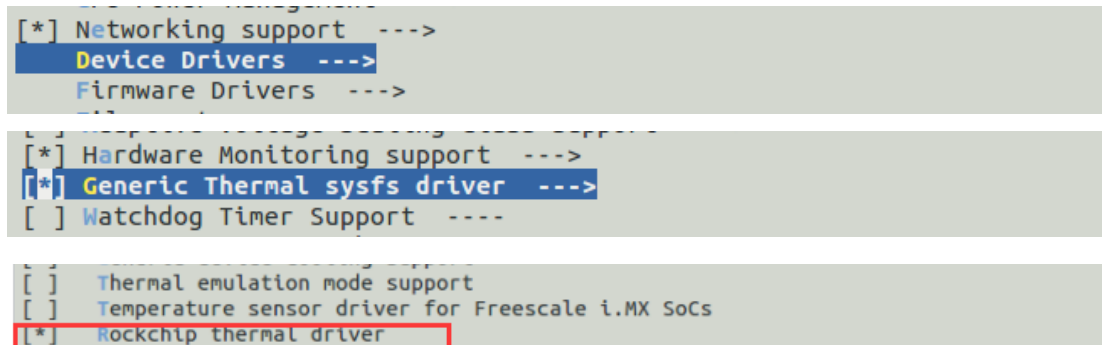
- **power_allocator**: 引入 PID（比例-积分-微分）控制，根据当前温度，动态给各模块分配 power，并将 power 转换为频率，从而达到根据温度限制频率的效果。
- **step_wise** : 根据当前温度，逐级限制频率。
- **userspace**: 不限制频率。

3 配置方法

3.1 Tsadc 配置

3.1.1 Menuconfig 配置

make ARCH=arm64 menuconfig



3.1.2 DTS 配置

如下是芯片级的 DTSI 的配置:

```
tsadc: tsadc@ff260000 {
    compatible = "rockchip,rk3399-tsadc";
    reg = <0x0 0xff260000 0x0 0x100>;
    interrupts = <GIC_SPI 97 IRQ_TYPE_LEVEL_HIGH>;
    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 = <95000>;
    status = "disabled";
};
```

```
tsadc: tsadc@ff260000 {
    compatible = "rockchip,rk3399-tsadc";
```

Tsadc 驱动加载的标识字符串

```
reg = <0x0 0xff260000 0x0 0x100>;
```

Tsadc 操作的寄存器基地址和寄存器地址总长度

```
interrupts = <GIC_SPI 97 IRQ_TYPE_LEVEL_HIGH>;
```

Tsadc 中断号

```
rockchip,grf = <&grf>;
```

Tsadc 引用 grf 模块，驱动会操作配置 grf

```
clocks = <&cru SCLK_TSADC>, <&cru PCLK_TSADC>;
clock-names = "tsadc", "apb_pclk";
```

Tsadc 模块的两个 clock，其中"tsadc"是 Tsadc 的工作时钟，"apb_pclk"是 Tsadc 的配置时钟

```
assigned-clocks = <&cru SCLK_TSADC>;
assigned-clock-rates = <750000>;
```

配置 Tsadc 的工作时钟是 750000，Tsadc 的配置时间周期，是以这个时钟为基准的。

```
resets = <&cru SRST_TSADC>;
reset-names = "tsadc-apb";
```

Tsadc 的 reset 控制，用于 resett Tsadc 模块

```
pinctrl-names = "init", "default", "sleep";
pinctrl-0 = <&otp_gpio>;
pinctrl-1 = <&otp_out>;
pinctrl-2 = <&otp_gpio>;
```

Tsadc 工作的 GPIO 口配置，这个配置与下面的配置对应。初始化 Init 和休眠 Sleep 的时候，是 GPIO 口功能，默认 Default 的时候，是输出 Out 功能。

```
tsadc {
    otp_gpio: otp-gpio {
        rockchip,pins = <1 6 RK_FUNC_GPIO &pcfg_pull_none>;
    };
    otp_out: otp-out {
        rockchip,pins = <1 6 RK_FUNC_1 &pcfg_pull_none>;
    };
};
#thermal-sensor-cells = <1>;
```

表示 Tsadc 可以向 Thermal Zone 注册，而且对应的 Thermal Zone 引用 Tsadc 的时候，带有一个参数。

```
如: thermal-sensors = <&tsadc 0>;
    rockchip,hw-tshut-temp = <95000>;
Tsadc 设定关机温度是 95000
    status = "disabled";
};
```

注:

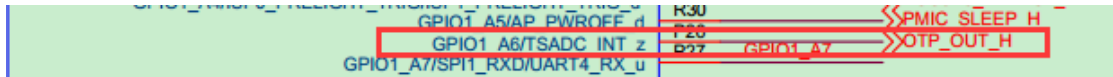
1. Tsadc 模块，默认在 DTSI 中是 Disabled 状态，要启用的时候，需要在板级的 DTS 中配置，如:

```
361 &tsadc {
362     rockchip,hw-tshut-mode = <1>; /* tshut mode 0:CRU 1:GPIO */
363     rockchip,hw-tshut-polarity = <1>; /* tshut polarity 0:LOW 1:HIGHIGH */
364     status = "okay";
365 };
```

2. rockchip,hw-tshut-mode = <1>

配置温度超过关机温度的复位方式，配置 0 是通过复位 SoC 的 CRU 模块，配置 1 是通过配置

上文提到的 `pinctrl-1 = <&otp_out>`; 来实现, 这个引脚功能 `otp_out` 一般会接入 `pmic` 的 `reset` 引脚 (如下图), 至于这个引脚是高有效还是低有效, 需要配置 `rockchip,hw-tshut-polarity` 来实现。

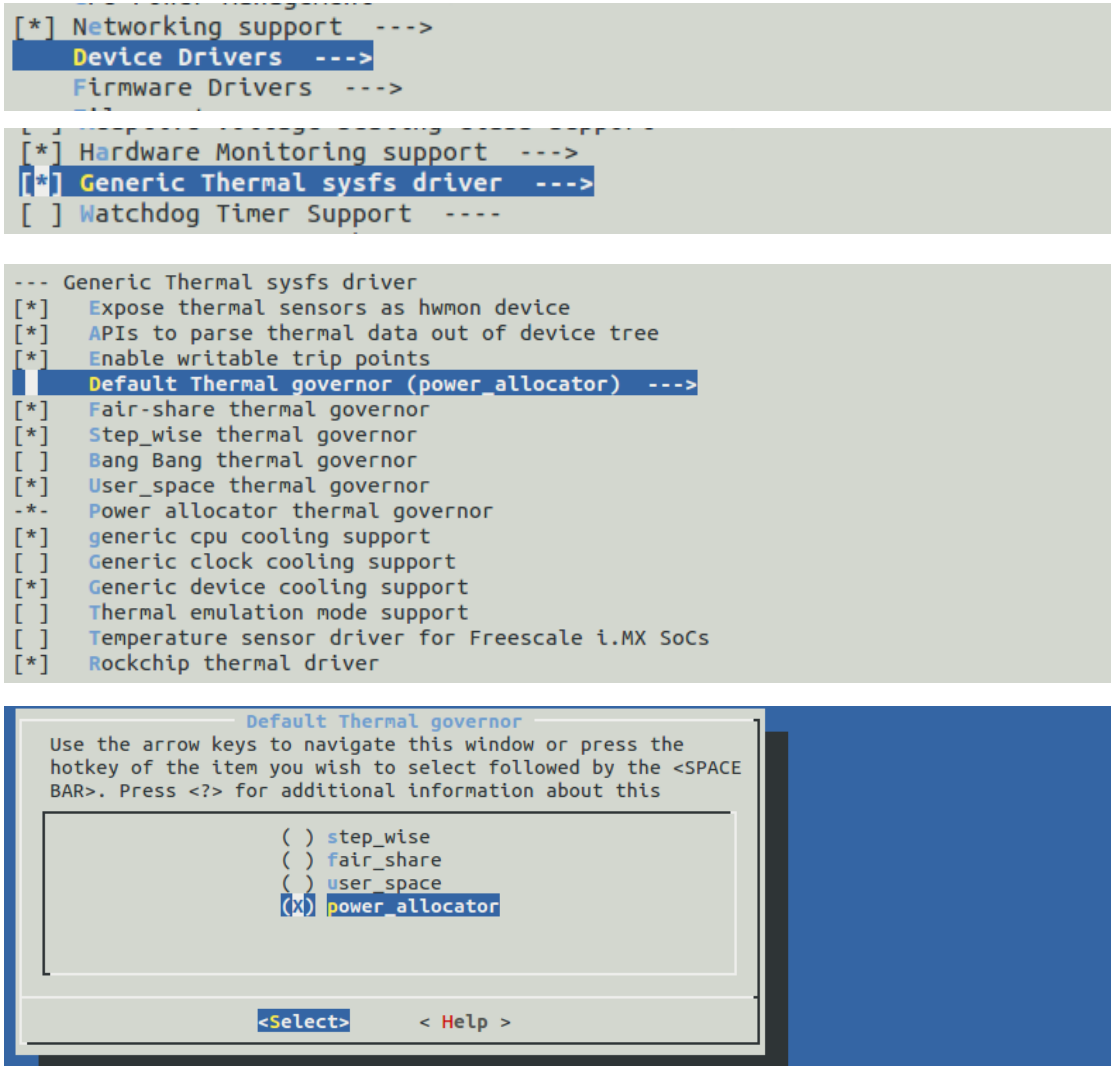


另外, 不管配置 0 还是配置 1, 这个复位方式都是硬件行为, 这与下文介绍的 Thermal 配置的 `soc_crit` 节点不同。下文 `soc_crit` 节点表示温度超过 95 度, 会自动重启系统, 这种重启系统是软件行为, 会走内核系统重启的标准流程。为防止系统重启丢失数据, `rockchip,hw-tshut-temp` 建议配置高于 `soc_crit` 节点配置的温度。

3.2 power_allocator 策略配置

3.2.1 默认使用 power_allocator 策略

make ARCH=arm64 menuconfig



3.2.2 CPU 开启温控

3.2.2.1 menuconfig 配置

```

--- Generic Thermal sysfs driver
[*] Expose thermal sensors as hwmon device
[*] APIs to parse thermal data out of device tree
[*] Enable writable trip points
[*] Default Thermal governor (power_allocator) --->
[*] Fair-share thermal governor
[*] Step_wise thermal governor
[ ] Bang Bang thermal governor
[*] User_space thermal governor
[*] Power_allocator thermal governor
[*] generic cpu cooling support
[ ] generic clock cooling support
[*] Generic device cooling support
[ ] Thermal emulation mode support
[ ] Temperature sensor driver for Freescale i.MX SoCs
[*] Rockchip thermal driver

```

3.2.2.2 DTS 配置

```

cpu_l0: cpu@0 {
    device_type = "cpu";
    compatible = "arm,cortex-a53", "arm,armv8";
    reg = <0x0 0x0>;
    enable-method = "psci";
    #cooling-cells = <2>; /* min followed by max */
    dynamic-power-coefficient = <121>;
    clocks = <&cru ARMCLKL>;
    cpu-idle-states = <&cpu_sleep>;
    operating-points-v2 = <&cluster0_opp>;
    sched-energy-costs = <&CPU_COST_A53 &CLUSTER_COST_A53>;
};

cpu_b0: cpu@100 {
    device_type = "cpu";
    compatible = "arm,cortex-a72", "arm,armv8";
    reg = <0x0 0x100>;
    enable-method = "psci";
    #cooling-cells = <2>; /* min followed by max */
    dynamic-power-coefficient = <1068>;
    clocks = <&cru ARMCLKB>;
    cpu-idle-states = <&cpu_sleep>;
    operating-points-v2 = <&cluster1_opp>;
    sched-energy-costs = <&CPU_COST_A72 &CLUSTER_COST_A72>;
};

```

#cooling-cells

表示该设备可以作为一个 cooling 设备；频率随温度变化时，会限制在最大值和最小值之间。

dynamic-power-coefficient

动态功耗公式为 $P = c * v^2 * f / 1000000$ ，dynamic-power-coefficient 是其中的参数 c，这个值跟芯片相关，由模块负责人实验数据得来，客户一般不需要修改。

3.2.3 GPU 开启温控

3.2.3.1 menuconfig 配置

```

--- Generic Thermal sysfs driver
[*] Expose thermal sensors as hwmon device
[*] APIs to parse thermal data out of device tree
[*] Enable writable trip points
[*] Default Thermal governor (power_allocator) --->
[*] Fair-share thermal governor
[*] Step_wise thermal governor
[ ] Bang Bang thermal governor
[*] User_space thermal governor
-* Power allocator thermal governor
[*] generic cpu cooling support
[ ] Generic clock cooling support
[*] Generic device cooling support
[ ] Thermal emulation mode support
[ ] Temperature sensor driver for Freescale i.MX SoCs
[*] Rockchip thermal driver

```

3.2.3.2 配置

```

gpu: gpu@ff9a0000 {
    compatible = "arm,mali860",
        "arm,mali86x",
        "arm,mali8xx",
        "arm,mali-midgard";

    reg = <0x0 0xff9a0000 0x0 0x10000>;

    interrupts = <GIC_SPI 19 IRQ_TYPE_LEVEL_HIGH>,
        <GIC_SPI 20 IRQ_TYPE_LEVEL_HIGH>,
        <GIC_SPI 21 IRQ_TYPE_LEVEL_HIGH>;
    interrupt-names = "GPU", "JOB", "MMU";

    clocks = <&cru ACLK_GPU>;
    clock-names = "clk_mali";
    #cooling-cells = <2>; /* min followed by max */
    operating-points-v2 = <&gpu_opp_table>;
    power-domains = <&power RK3399_PD_GPU>;
    power-off-delay-ms = <200>;
    status = "disabled";

    power_model {
        compatible = "arm,mali-simple-power-model";
        voltage = <900>;
        frequency = <500>;
        static-power = <300>;
        dynamic-power = <1780>;
        ts = <32000 4700 (-80) 2>;
        thermal-zone = "gpu-thermal";
    };
};

```

- #cooling-cells 属性表示该设备可以作为一个 cooling 设备；频率随温度变化时，会限制在最大值和最小值之间。
- voltage、frequency、static-power、dynamic-power 属性，这四个参数表示 GPU 在频率为 500MHz, 电压为 900mv 下, 全速运行时, 静态功耗 300mW, 动态功耗 1780mW, 由模块负责人的实验实测所得, 用于求出计算静态功耗和动态功耗所需的参数 c。客户一般不需要修改

动态功耗公式: $P(d) = c * v^2 * f / 1000000$

静态功耗公式: $t_scale = (2 * t^3) + (-80 * t^2) + (4700 * t) + 32000$

$v_scale = v^3 / 1000000$

$P(s) = c * t_scale * v_scale / 1000000$

- TS 属性, 公式 $t_scale = (2 * t^3) + (-80 * t^2) + (4700 * t) + 32000$ 中的常数。
- Thermal-Zone 属性, 指导 GPU 通过 gpu-thermal 获取温度, 在 RK3399 上是 tsadc 1。

3.2.4 thermal_zone 配置

以 RK3399 的配置为例

```
thermal-zones {
    soc_thermal: soc-thermal {
        polling-delay-passive = <20>; /* milliseconds */
        polling-delay = <1000>; /* milliseconds */
        sustainable-power = <1600>; /* milliwatts */

        thermal-sensors = <&tsadc 0>;

        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 = <95000>; /* millicelsius */
                hysteresis = <2000>; /* millicelsius */
                type = "critical";
            };
        };
    };
};
```

```

        cooling-maps {
            map0 {
                trip = <&target>;
                cooling-device =
                    <&cpu_l0 THERMAL_NO_LIMIT THERMAL_NO_LIMIT>;
                contribution = <4080>;
            };
            map1 {
                trip = <&target>;
                cooling-device =
                    <&cpu_b0 THERMAL_NO_LIMIT THERMAL_NO_LIMIT>;
                contribution = <1024>;
            };
            map2 {
                trip = <&target>;
                cooling-device =
                    <&gpu THERMAL_NO_LIMIT THERMAL_NO_LIMIT>;
                contribution = <4080>;
            };
        };
    };

    gpu_thermal: gpu-thermal {
        polling-delay-passive = <100>; /* milliseconds */
        polling-delay = <1000>; /* milliseconds */

        thermal-sensors = <&tsadc 1>;
    };
};

```

Thermal-Zones 下的第一级子节点，对应的是不同的 Tsadc，RK3399 上有两个 Tsadc，一个在 CPU 旁边，一个在 GPU 旁边，所以我们可以看到两个子节点。

```
polling-delay-passive = <20>;
```

温度超过阈值时，每隔 20ms 查询温度，并限制频率。

```
polling-delay = <1000>;
```

温度未超过阈值时，每隔 1000ms 查询温度。

```
sustainable-power = <1600>;
```

当前温度到达预设的最高值时，系统能分配给 cooling 设备的能量。

```
thermal-sensors = <&tsadc 0>;
```

当前 thermal_zone 的温度是通过 tsadc0 获取的。

- trips

- threshold 节点表示温度超过 70 度温控策略开始工作，但不一定马上限制频率，power 小到一定程度才开始限制频率，type 要设置成"passive"；
- target 节点表示系统期望的最高温度为 85 度，虑到芯片的寿命和可靠性，建议不超过 85，type 要设置成"passive"；
- soc_crit 节点表示温度超过 95 度，自动重启系统，type 要设置成"critical"。

- cooling-maps

- 子节点 map0、map1、map2 表示有 3 个设备作为 cooling 设备，即会根据温度被限制频率的设备。

- ◆ cooling-device 属性

以 RK3399 为例，子节点的 cooling-device 属性分别引用了 &cpu_b0、&cpu_l0、&gpu，表示小核 A53、大核 A72 和 GPU 会根据温度限制频率。后面两个参数表示该设备允许被限制的最低频率和最高频率，填 THERMAL_NO_LIMIT 表示按最小最大值是频率电压表中的最小值和最大值。

- ◆ trip 属性

对与 power_allocator 策略子节点的 trip 属性都设置 target。

注：

如果板子出现过温重启的情况，首先采集过温时候的温度，然后可以通过 2 种方式调整参数：

1. 调整 threshold 和 target 的温度参数，如由 70，85 度，同步调整为 65，80 度，建议 threshold 和 target 的温度保持 10-15 度的范围。
2. 调整 soc_crit 和上文中的 rockchip,hw-tshut-temp，如由 95 度调整为 100 度。

方法 1 会损失一些性能，方法 2 的芯片温度会较高

3.2.5 参数调试方法总结

温控有关的参数：thermal 配置里面的 polling-delay-passive, polling-delay, sustainable-power, threshold, target, soc_crit, rockchip,hw-tshut-temp, gpu 配置里面的 voltage, frequency, static-power, dynamic-power, ts, cpu 的 dynamic-power-coefficient;

根据产品要调试的参数：threshold, target, soc_crit, rockchip,hw-tshut-temp, sustainable-power, contribution。具体步骤如下：

- (1) 根据产品形态，确定 target 温度，考虑到芯片的寿命和可靠性，建议该值不高于 85 度。
- (2) soc_crit = target + 5 度左右。
- (3) rockchip,hw-tshut-temp = soc_crit + 5 度左右，并且要小于最高温度 120 度。
- (4) sustainable-power 需要根据板子的实际情况调试获得。假设 sustainable-power=800，芯片运行的最高温度是 76 度（target 是 80 度），增加 sustainable-power 到 1000，可以抬高芯片运行的最高温度到 78 度或者更高，微调 sustainable-power 直到最高温度满足性能需求。

- (5) 可以修改 contribution，调整 power 的分配比例，到达改变降频优先级的作用。

以 rk3399 evb 为例子：

- (1) 确定目标温度。

假设我们希望，70 度以上温控开始工作，最高温度不超过 85 度，超过 95 度系统重启。于是要做如下配置：

```
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 = <95000>; /* millicelsius */
            hysteresis = <2000>; /* millicelsius */
            type = "critical";
        };
};

tsadc: tsadc@ff260000 {
    rockchip,hw-tshut-temp = <95000>;
};

```

(2) 配置 sustainable-power。

以上设置了一个 70 度到 85 度的范围，表示系统在 70 度的时候会提供一个比较大的 power 值，随着温度的升高，power 逐渐减小，但并不意味着 70 度的时候就一定会降频，范围越小，温度每升高一度，power 降的越多。该什么时候降频可以通过修改 sustainable 的值，进行调整。

假如我们希望 80 度的时候才开始限制频率，那么可以先让 80 度时的 power 值等于 A72 的最大功耗，A53 的最大功耗，和 GPU 的最大功耗之和，然后适当减小调试，直到满足我们的需求。

A72 1800MHz 的最大功耗: $436 \times 1200 \times 1200 \times 1800 \times 2 / 1000000000 = 2260$

A53 1416MHz 的最大功耗: $100 \times 1125 \times 1125 \times 1416 \times 4 / 1000000000 = 716$

GPU 800MHz 的最大功耗: $944 + 502$ (动态功耗+静态功耗，见 3.2.7 中的公式)

$\text{Power}(80) = 2260 + 716 + 944 + 502 = 4422$

Target = 85

Threshold = 70

通过以下公式

$\text{sustainable} + 2 * \text{sustainable} / (\text{target} - \text{threshold}) * (\text{target} - 80) = \text{power}(80)$

得到

$\text{sustainable} = 2653$

假如实测发现，超过 85 度才开始限制频率，不符合预期。于是减小 Power(80)继续调试，如：

$\text{Power}(80) = 3526$

经公式转换得到

sustainable = 2116

实测发现，80 度开始温控，最高到 83 度，符合预期。

那么 sustainable 最终确定为 2116。

(3) contribution 调测。

默认情况（即使不填）contribution 为 1024:

A53: contribution = <1024>;

A72: contribution = <1024>;

GPU: contribution = <1024>;

假如在高温下，A53 和 A72 都满负载运行，发现 A53 更容易被降频，这时如果想让 A72 优先降频，可以增大 A53 的 contribution，比如修改为：

A53: **contribution = <1536>;**

A72: **contribution = <1024>;**

GPU: **contribution = <1536>;**

不建议调试的参数: gpu 配置里面的 voltage, frequency, static-power, dynamic-power, ts, cpu 的 dynamic-power-coefficient, 这些参数是根据 soc 内部 gpu 和 cpu 的动静态功耗，计算出来的系数

一般不需要调试的参数: polling-delay-passive, polling-delay。

如果 soc 的功耗很大，如果轮询的间隔太大，soc 的温度会飙升，这种情况需要微调

polling-delay-passive, polling-delay。一般情况不需要调试这个参数。

4 调试接口

4.1 关温控

一般来说，关温控其实是把策略切换到 `user_space`，并解除频率限制。以 RK3399 为例，策略切换到 `user_space`：

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

解除频率限制：

```
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
```

4.2 获取当前温度

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
```