# D1 Linux Thermal
# 开发指南

**版本号: 1.0**
**发布日期: 2021.04.13**

# 版本历史

| 版本号 | 日期 | 制/修订人 | 内容描述 |
| --- | --- | --- | --- |
| 1.0 | 2021.04.13 | AWA1442 | 1. 添加初始版本。 |

# 目　录

# 1 前言

## 1.1 文档简介

该使用文档介绍了 thermal 的温控策略配置方法，以及调试使用说明。

## 1.2 目标读者

thermal 模块开发、维护人员。

## 1.3 适用范围

表 1-1：适用产品列表

| 产品名称 | 内核版本 | 驱动文件 |
| --- | --- | --- |
| D1 | Linux-5.4 | drivers/thermal/* |

# 2 模块介绍

## 2.1 模块功能介绍

Thermal 俗称热控制系统，其功能是通过 temperature sensor 测量当前 CPU、GPU 等设备的温度值，然后根据此温度值，影响 CPU、GPU 等设备的调频策略，对 CPU、GPU 等设备的最大频率进行限制，最终实现对 CPU、GPU 等设备温度的闭环控制，避免 SOC 温度过高。

IPA(Intelligent Power allocator) 温控策略：引入 PID 控制，根据系统温度动态分配 power 给各个设备，并将 power 转化为频率限制。

## 2.2 相关术语介绍

表 2-1: 术语介绍

| 术语 | 说明 |
| --- | --- |
| Temperature sensor | 温度传感器。 |
| Thermal | CPU 温度控制系统。 |
| CPU | 中央处理器。 |
| GPU | 图像处理器。 |
| thermal zone | 将提供温度及 trip 点相关信息给 themal core 子系统。 |
| cooling device | themal core 子系统通过 cooling device 对 CPU、GPU 等设备最大频率进行限制。 |

## 2.3 模块配置介绍

### 2.3.1 Device Tree 配置说明

设备树中存在的是该类芯片所有平台的模块配置，设备树文件的路径为：kernel/linux-5.4/arch/riscv/boot/dts/sunxi/CHIP.dtsi(CHIP 为研发代号，如 sun20iw1p1 等)。

- of-thermal

在 thermal 模块开发中，只需要将 thermal zone、thermal Sensor、trip point、cooling Device 的关系在 DTS 文件内按照规定的格式描述，of-thermal 模块就会根据 DTS 将描述的内容自动注册，逻辑关系由 of-thermal 模块维护，使驱动代码量大大减少。

```
thermal-zones{
    cpu_thermal_zone{
        polling-delay-passive = <500>;      //温度超过阈值，轮询温度周期(ms)
        polling-delay = <1000>;             //温度未超过阈值，轮询温度周期(ms)
        thermal-sensors = <&ths 0>;
        sustainable-power = <1200>;         //温度达到预设温度最大值，系统可分配的最大power
        k_po = <25>;                        //超过预设最高温度时pid的p参数
        k_pu = <50>;                        //未超过预设最高温度时pid的p参数
        k_i = <0>;                          //pid的i参数

        cpu_trips: trips{
            cpu_threshold: trip-point@0 {
                temperature = <70000>;      //代表系统温控在70度左右开启
                type = "passive";
                hysteresis = <0>;
            };
            cpu_target: trip-point@1 {
                temperature = <80000>;      //代表系统最高温度是80度左右
                type = "passive";
                hysteresis = <0>;
            };
            cpu_crit: cpu_crit@0 {
                temperature = <110000>;     //代表系统到达110度就会过温关机
                type = "critical";
                hysteresis = <0>;
            };
        };

        cooling-maps {
            map0 {
                trip = <&cpu_target>;
                cooling-device = <&CPU0
                THERMAL_NO_LIMIT
                THERMAL_NO_LIMIT>;
                contribution = <1024>;      //cpu 分配power权重
            };
            map1{
                trip = <&cpu_target>;
                cooling-device = <&gpu
                THERMAL_NO_LIMIT
                THERMAL_NO_LIMIT>;
                contribution = <1024>;      //gpu 分配power权重
            };
        };
    };

    gpu_thermal_zone{
        polling-delay-passive = <500>;
        polling-delay = <1000>;
        thermal-sensors = <&ths 0>;
    };

    ve_thermal_zone{
        polling-delay-passive = <0>;
        polling-delay = <0>;
```

```
        thermal-sensors = <&ths 1>;
    };

    ddr_thermal_zone{
        polling-delay-passive = <0>;
        polling-delay = <0>;
        thermal-sensors = <&ths 3>;
    };
};

cpu_target节点中的temperature:
可根据产品温控规格，适当调整该参数。提高该参数，会允许系统在高温情况下运行更快，性能更好。
当然，也会让产品的温度更高，所以需要注意，修改该参数后能否满足产品温度要求和高温测试等。
同理，降低该参数就会在一定程度上降低高温情况下的性能，可以让产品运行在较低的温度。
```

- Thermal driver

```
ths: ths@02009400 {
    compatible = "allwinner,sun20iw1p1-ths";
    reg = <0x0 0x02009400 0x0 0x400>;
    clocks = <&ccu CLK_BUS_THS>;
    clock-names = "bus";
    resets = <&ccu RST_BUS_THS>;
    nvmem-cells = <&ths_calib>;
    nvmem-cell-names = "calibration";
    #thermal-sensor-cells = <1>;
};
```

- Cooling device

cpu device

```
CPU0: cpu@0  {
    device_type = "cpu";
    reg = <0>;
    status = "okay";
    compatible = "riscv";
    riscv,isa = "rv64imafdcvsu";
    mmu-type = "riscv,sv39";
    clocks = <&ccu CLK_RISCV>;
    clock-frequency = <24000000>;
    operating-points-v2 = <&cpu_opp_table>;
    cpu-idle-states = <&CPU_SLEEP>;
    #cooling-cells = <2>;

};

dynamic-power-coefficient: cpu动态功耗系数，由P = c * v * v * f / 1000000得来（参数c就是动态功耗系
    数）。
```

cpu opp table

```
cpu_opp_l_table: opp_l_table {
    compatible = "operating-points-v2";
    opp-shared;

    opp@408000000 {
        opp-hz = /bits/ 64 <408000000>;
        opp-microvolt = <820000>;
        clock-latency-ns = <244144>; /* 8 32k periods */
    };
    opp@816000000 {
        opp-hz = /bits/ 64 <816000000>;
        opp-microvolt = <880000>;
        clock-latency-ns = <244144>; /* 8 32k periods */
    };
    opp@1008000000 {
        opp-hz = /bits/ 64 <1008000000>;
        opp-microvolt = <940000>;
        clock-latency-ns = <244144>; /* 8 32k periods */
    };
    opp@1200000000 {
        opp-hz = /bits/ 64 <1200000000>;
        opp-microvolt = <1020000>;
        clock-latency-ns = <244144>; /* 8 32k periods */
    };
    opp@1416000000 {
        opp-hz = /bits/ 64 <1416000000>;
        opp-microvolt = <1100000>;
        clock-latency-ns = <244144>; /* 8 32k periods */
    };
};
opp-hz: 频率
opp-microvolt: 频率对应的电压
```

## gpu device

```
gpu: gpu@0x01800000 {
    device_type = "gpu";
    compatible = "arm,mali-midgard";
    reg = <0x0 0x01800000 0x0 0x10000>;
    interrupts = <GIC_SPI 95 IRQ_TYPE_LEVEL_HIGH>,
             <GIC_SPI 96 IRQ_TYPE_LEVEL_HIGH>,
             <GIC_SPI 97 IRQ_TYPE_LEVEL_HIGH>;
    interrupt-names = "JOB", "MMU", "GPU";
    clocks = <&clk_pll_gpu>, <&clk_gpu0>, <&clk_gpu1>;
    clock-names = "clk_parent", "clk_mali", "clk_bak";
    #cooling-cells = <2>;
    ipa_dvfs:ipa_dvfs {
        compatible = "arm,mali-simple-power-model";
        static-coefficient = <17000>;
        dynamic-coefficient = <750>;
        ts = <254682 9576 0xffffff98 4>;
        thermal-zone = "gpu_thermal_zone";
        ss-coefficient = <36>;
        ff-coefficient = <291>;
    };
};
static-coefficient:  gpu静态功耗计算系数
```

```
dynamic-coefficient:  gpu动态功耗计算系数
```

gpu dvfs

```
gpu dvfs表:
gpu: gpu@0x01800000 {
        gpu_idle = <1>;
        dvfs_status = <1>;
        operating-points = <
             /* KHz    uV */
            600000 950000
            576000 950000
            540000 950000
            504000 950000
            456000 950000
            420000 950000
            384000 950000
            360000 950000
            336000 950000
            306000 950000
         >;
};
```

### 2.3.2 board.dts 配置说明

board.dts 用于保存每一个板级平台的设备信息（如 demo 板，perf1 板等），里面的配置信息会覆盖上面的 Device Tree 默认配置信息。thermal 模块在 board.dts 中无用户可用配置。

### 2.3.3 sysconfig 配置说明

thermal 模块在 sysconfig 中无用户可用配置。

### 2.3.4 kernel menuconfig 配置说明

进入 longan 目录，执行./build.sh config 选择平台和版型；然后./build.sh menuconfig 进入配置界面。

首先，进入到 Device Drivers ->Generic Thermal sysfs driver，如下图所示：

```
-*- 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
<*>   QorIQ Thermal Monitoring Unit
<*>   Allwinner sunxi next genaration thermal driver
      ACPI INT340X thermal drivers  ----
      allwinner(SUNXI) thermal drivers  --->
```

图 2-1: thermal 配置

配置项说明:

```
Default Thermal governor 选择，默认为power_allocator
Thermal emulation mode support:支持thermal模拟温度功能
generic cpu cooling support :打开通用的cpu cooling
generic device cooling support:打开通用的device cooling
Allwinner sunxi next generation thermal driver:sunxi thermal驱动
```

## 2.4 源码结构介绍

```
kernel/
|-- drivers/thermal/sunxi_thermal-ng.c //thermal sensor驱动代码
|-- drivers/thermal/cpu_cooling.c //thermal cpu cooling代码
|-- drivers/thermal/devfreq_cooling.c //thermal devfreq cooling代码
```

## 2.5 驱动框架介绍

无。

# 3 模块使用范例

无。

# 4 FAQ

## 4.1 调试方法

### 4.1.1 调试工具

无。

### 4.1.2 调试节点

无。

## 4.2 常见问题

### 4.2.1 查看 sensor 温度

不同平台温度 sensor 的个数及温度监控区域 thermal_zone 是不一样的。多个温度监控区域在/sys/class/thermal 目录下就会有多个 thermal_zone。查看 thermal_zone 的温度，下面以 thermal_zone0 为例：

```
查看thermal_zone的类型

#cat sys/class/thermal/thermal_zone0/type
cpu_thermal_zone
```

```
查看thermal_zone温度

#cat sys/class/thermal/thermal_zone0/temp
36000
温度单位为mC,也就是36摄氏度
```

### 4.2.2 模拟温度

thermal 有温度模拟功能，可以通过模拟温度校验温度策略是否符合预期。

```
设置thermal_zone0的模拟温度

#echo 80000 > /sys/class/thermal/thermal_zone0/emul_temp
```

```
关闭thermal_zone0的模拟温度功能

#echo 0 > /sys/class/thermal/thermal_zone0/emul_temp
```

### 4.2.3 关闭温控

以关闭 thermal_zone0 温控为例。

```
关闭温控策略

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

解除所有cooling device的限制
#echo 0 > /sys/class/thermal/thermal_zone0/cdev*/cur_state
```

### 4.2.4 不同温控策略下芯片性能和温度的关系

传统的 stepwise 温控策略，通过 dts 配置芯片在不同温度下 cpu 运行频率、打开核数等性能限制。所以对于 stepwise 策略，芯片在特定温度下性能是确定的。

与 stepwise 温控策略不同，对于 IPA 温控策略，芯片在特定温度下性能是不确定的。若需要了解，可以通过实际测试得出。

### 4.2.5 如何设定过温不关机

修改 cpu_crit@0 节点的 temperature 为很大的值，就不会触发过温关机。

```
        cpu_trips: trips{

            ......

            cpu_crit: cpu_crit@0 {
                temperature = <110000>;      //代表系统到达110度就会过温关机
                type = "critical";
                hysteresis = <0>;
            };
        };
```

同时，在使用 PMIC 的方案上，可能也需要关闭 PMIC 的过温保护功能。详见《Linux_PMIC_开发指南》。

```
pmu0: pmu@0{

    ......

    overtemp_shutdown = <1>;     //过温保护，0为关闭
    overtemp_value = <145>;      //过温保护温度

    ......
};
```

## 4.2.6 如何修改温控策略的目标温度

可以根据方案需求，修改 trip-point@1 节点的 temperature 为温度策略的目标温度。
如若实测温度高于目标温度，可以适当改小 sustainable-power 和 trip-point@0 节点的 temperature；若修改后仍不起效，可以考虑瓶颈是否在硬件。

```
    cpu_thermal_zone{

        ......

        sustainable-power = <1000>;          //温度达到预设温度最大值，系统可分配的最大power

        ......

        cpu_trips: trips{
            cpu_threshold: trip-point@0 {
                temperature = <70000>;       //代表系统温控在70度左右开启
                type = "passive";
                hysteresis = <0>;
            };
            cpu_target: trip-point@1 {
                temperature = <80000>;       //代表系统最高温度是80度左右
                type = "passive";
                hysteresis = <0>;
            };

            ......

        };

    ......

    };
```

# 4.2.7 其他温控方法

- 使用更低功耗的 cpu 调频策略，如 ondemand、conservative、powersave 等。详见《D1_Linux_CPUFREQ_ 开发指南》。
- cpufreq 删除高频点、增加低频点。详细咨询方案硬件开发人员。

**著作权声明**



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

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

**商标声明**


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

**免责声明**

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

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

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