



Tina Linux 温度控制 使用指南

版本号: 1.0
发布日期: 2021.04.07

版本历史

版本号	日期	制/修订人	内容描述
1.0	2021.04.07	AWA1610	1. 初始版本



目 录

1 概述	1
1.1 编写目的	1
1.2 适用范围	1
1.3 相关人员	1
2 Linux 温控框架简介	2
2.1 基础框架	2
2.1.1 Thermal Zone Device——获取温度的设备	3
2.1.2 Thermal Cooling Device——控制温度的设备	4
2.1.3 Thermal Governor——温控系统的调度	4
2.1.3.1 thermal governor 相关的基础知识	4
2.1.4 Thermal Core——内核及用户空间接口	5
2.2 温控策略	5
2.2.1 step-wise policy	5
2.2.2 power allocator policy	6
2.3 平台差异	7
2.3.1 Linux 3.4 温控框架	7
2.3.2 Linux 4.x_old 温控框架	7
2.3.3 Linux 4.x_new 温控框架	8
2.3.4 Linux 5.x 温控框架	8
3 Step-Wise 温控机制参数配置	9
3.1 Linux 3.4 温控框架中的 step-wise 参数配置	9
3.2 Linux 4.x_old 温控框架中的 step-wise 参数配置	10
3.2.1 获取温度模块	10
3.2.2 控制温度模块	11
3.2.3 温度控制策略	12
3.3 Linux 4.x_new 及 Linux 5.x 温控框架中的 step-wise 参数配置	14
3.3.1 获取温度模块	14
3.3.2 控制温度模块	14
3.3.3 温度控制策略	14
4 Power allocator 温控机制参数配置	16
4.1 Linux 4.x_new 及 Linux 5.x 温控框架中的 Power allocator 参数配置	16
4.1.1 温度控制策略	16
5 温控机制的 menuconfig 配置	18
5.1 Linux 3.4 温控框架配置	18
5.1.1 R58 配置	18
5.2 Linux 4.x_old 温控框架配置	19
5.2.1 R328 配置	19
5.3 Linux 4.x_new 及 Linux 5.x 温控框架配置	21

5.3.1 R818/MR813/R329 配置	21
5.3.2 R528 配置	24
6 温控框架调试	27
6.1 基础说明	27
6.1.1 sysfs 节点	27
6.2 基础操作	28



插 图

2-1 thermal framework	3
5-1 Thermal_menuconfig_R58_001.png	18
5-2 Thermal_menuconfig_R328_001.png	19
5-3 Thermal_menuconfig_R328_002	20
5-4 Thermal_menuconfig_R328_003	21
5-5 Thermal_menuconfig_R818_001	22
5-6 Thermal_menuconfig_R818_002	23
5-7 Thermal_menuconfig_R818_003	24
5-8 Thermal_menuconfig_R528_001	25
5-9 Thermal_menuconfig_R528_002	26



1 概述

1.1 编写目的

介绍 Tina 温度控制机制实现及相关的模块的实现与配置。

1.2 适用范围

适用于 R58、R328、MR813、R818、R329、R528、D1 平台。

1.3 相关人员

适用 Tina 平台的广大客户和对 Linux thermal 框架感兴趣的同事。

2 Linux 温控框架简介

2.1 基础框架

温控（thermal）系统的核心功能就是将目标温度控制在一个合理的范围。温度过高，则会快速消耗寿命，体验不佳，严重时还会带来安全隐患。

而对于嵌入式系统来说，降温的一般方法是降频，温度降低越厉害，系统频率越低，对应的性能也越差。而如果不降温，则系统可能温度会越来越高，导致系统寿命降低，体验不好。因此选择合适的降温状态是非常重要的一点。

要达到此目的，首先需要思考两个关键点：

（1）如何降低温度

（2）何时降低温度

为了实现上述功能需求，thermal framework 抽象出 Thermal Zone Device、Thermal Cooling Device、Thermal Governor、Thermal Core 四个部分。

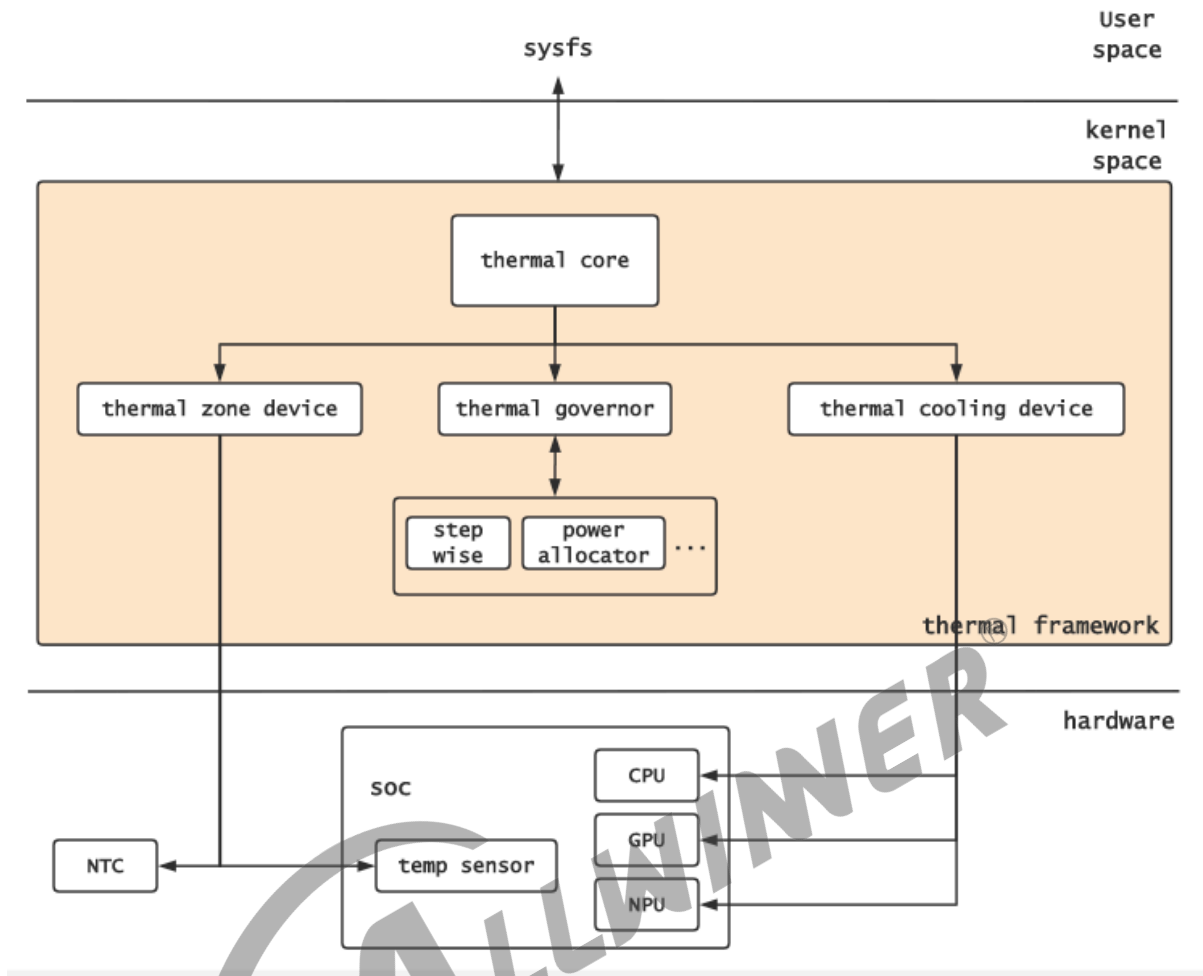


图 2-1: thermal framework

如何降低温度是由 Thermal Cooling Device 处理，它负责向 thermal 框架提供不同级别的降温状态，级别越大降温越深，一般对系统性能功耗的影响就会越大。

何时降低温度是由 Thermal Zone Device 和 Thermal Governor 模块负责，前者负责对系统温度采样，为 thermal 框架提供实时数据反馈，而后者根据这些数据计算并选择一个合适的降温状态。

2.1.1 Thermal Zone Device——获取温度的设备

Thermal Zone Device 是在 thermal framework 中充当测温设备。由于获取温度的设备有很多，例如 cpu, gpu, ddr 内部集成的 sensor，还有电池温敏电阻，外置温度传感器等等。这些获取温度的设备形态，功能，原理，分布的位置等各有差异，有的内嵌在 IC 中，有的独立于 IC 之外，有的是数字的，有些又是模拟的，甚至有的只是一个虚拟出来的设备。为了方便管理他们，所以内核将获取温度的特性抽象出来统一管理，至于对温度获取的具体实现由对应的设备驱动完成。只有当驱动将设备注册到 thermal framework 中成为 Thermal Zone Device 后，该设备才能参与温控过程。

2.1.2 Thermal Cooling Device——控制温度的设备

与测温设备相同，控制温度的设备也各式各样，常见的有散热片，风冷，水冷等等，他们都是通过增加散热效率降温的。而在嵌入式系统中，一般采用降低产热量的方式，例如降低设备运行频率等等，这种方式通过对某种设备属性的操作，达到温度控制的目的，因此内核也需要将其特性抽象出来，这便是 Thermal Cooling Device。同样，各厂商需要实现自己的降温设备驱动，然后将降温功能注册成 Thermal Cooling Device。

值得注意的是由于在电子系统中，发热是固有的属性，设备常常会过热，而且高温比低温对系统的危害要高得多，因此大多数产品对温度的控制都是专注于如何降温。同样地，在本文中，对温度的控制都是指降低温度。

2.1.3 Thermal Governor——温控系统的调度

有了测温和控温的设备，那么何时控温，怎样控温，采取何种策略控温就成了 Thermal 框架最重要的问题了。于是内核提供了 Thermal Governor 模块来抽象这些工作。

1，Thermal Governor 负责对温度系统的调用，如何时采样，采样后根据什么判断是否更新降温状态，如果更新降温状态，采用什么样的策略来计算最终的降温状态等等，这些均由 Thermal Governor 负责管理，但 Governor 只能提供框架，并不实现具体的计算操作。

2，Thermal policy 温控策略主要是根据当前温度来选择合适的降温状态的计算方法。举个简单的例子，当前的温度升高很快，选择风扇 3 档风，温度升高不快，选择 1 档风，这就是一个温控策略。其中，policy 负责为 Governor 提供具体的计算工作，Governor 可通过切换不同的 policy 达到不同的计算结果。

2.1.3.1 thermal governor 相关的基础知识

- 常用policy说明

用户态可以通过 `cat /sys/devices/virtual/thermal/thermal_zone0/available_policies` 查看支持的温控策略

`user_space`: 用户模式，将温控区间的控制权移交给用户空间，用户可以客制化相关的热功耗参数。

`step_wise`: 逐步调整模式，该策略属于开环控制，基于温度阈值和趋势，逐步浏览每个cooling设备的cooling等级。

`power_allocator`: 功率分配模式，该策略属于闭环控制，基于每个相关设备的功率，温度和当前功耗通过PID算法进行闭环控制。

- trips 和 binds 说明

什么是 trips? 和 binds? 在 thermal policy 的实现中, 它需要了解哪一个测温设备, 对应哪一个降温状态, 以及温度和降温状态间的转换关系才能正确的计算。为了简化这个关系的描述, Linux thermal 提供了 trips 和 cooling-maps 的概念。policy 的实现需要依赖于 trips 和 binds 配置。

trips 一般理解为触发点, 当测温设备达到一定的温度时, 就需要更新降温状态, 而此时对应的温度值就是一个 trip。一个测温设备可以有多个触发点, 这个 trip 集合就称为 trips。

cooling-maps 有时也称做 binds, 一般理解为绑定。他的含义是为一个 trip 绑定一个或多个降温状态的集合 (这个集合下文称为 states), 当测温设备温度达到 trip 时, 会触发 thermal governor 调度一次 thermal policy, 而 thermal policy 将会从这个降温状态的集合 (states) 中, 计算出一个合适的降温状态 (state) 返回给 thermal governor 去设置, 这样就完成一次温控动作。

另外指出的是, 并不是每一个 trip 都会触发 thermal governor 调度, 因为 Linux 定义了四种 trip 类型, 如下:

- * active: 激活主动冷却
- * passive: 启动被动冷却
- * hot: 系统过热
- * critical: 系统不可靠

只有 active, passive 类型的 trip 会触发 thermal governor 调度一次 thermal policy, 完成降温状态的更新。而 hot, critical 类型的 trip 属于过热保护的系统警告, 触发他们后, 不会执行更新降温状态的操作, 而是发送 notify 到 thermal_zone_device 中, 由 thermal_zone_device 驱动执行过温保护的操作来保护系统。而且, critical 类型的 trip 会在 notify 发送完成后, 直接执行关机操作。因此可以通过配置 trip 的类型为 critical, 来设置过温关机保护。

2.1.4 Thermal Core——内核及用户空间接口

有了测温控温设备, 有了控制策略, 剩下的问题就是如何将其联系在一起了。内核抽象了 Thermal core, 这是 Thermal 的核心。主要功能是关联测温控温设备和控制策略, 并请求内核调度执行; 向其他模块提供统一的注册, 使用接口, 并向用户空间开放 sysfs 调试接口。

2.2 温控策略

2.2.1 step-wise policy

在 Tina 系统中, 一般我们采用的温控策略是 step-wise, 这是一种最基础, 也是最简单有效的温控策略。它的核心思想是根据当前温度的趋势 (上升、下降、平稳), 以及当前温度是否高于 trip(触发点), 来决定 cpu 下一次的 Cooling Device 状态。

Linux thermal 定义的温度趋势有以下五种, 都是根据上一次与这一次温度的变化关系而变化

的。

```
enum thermal_trend {  
    THERMAL_TREND_STABLE, /* temperature is stable */  
    THERMAL_TREND_RAISING, /* temperature is raising */  
    THERMAL_TREND_DROPPING, /* temperature is dropping */  
    THERMAL_TREND_RAISE_FULL, /* apply highest cooling action */  
    THERMAL_TREND_DROP_FULL, /* apply lowest cooling action */  
};
```

step-wise 主要逻辑：

如果当前温度比一个 trip 温度高，且

- 如果趋势是 THERMAL_TREND_RAISING，则使用这个 trip 对应的更高的降温状态；
- 如果趋势是 THERMAL_TREND_DROPPING，则不做任何事情；
- 如果趋势是 THERMAL_TREND_RAISE_FULL，则使用这个 trip 对应的最高的降温状态；
- 如果趋势是 THERMAL_TREND_DROP_FULL，则使用这个 trip 对应的最低的降温状态；
- 如果趋势是 THERMAL_TREND_STABLE，则不做任何事情；

如果当前温度比一个 trip 温度低，且

- 如果趋势是 THERMAL_TREND_RAISING，不做任何事；
- 如果趋势是 THERMAL_TREND_DROPPING，则使用此触发点对应的较低的降温状态，如果冷却状态已经等于下限，则停用任何降温措施；
- 如果趋势是 THERMAL_TREND_RAISE_FULL，则不采取任何措施；
- 如果趋势是 THERMAL_TREND_DROP_FULL，则使用下限，如果冷却状态已经等于下限，则停用任何降温措施；

温度变化趋势是由这次测量值与上次测量值间的比较确定，若本次测量值高于上次，则温度为 THERMAL_TREND_RAISING 趋势；若本次测量值低于上次，则温度为 THERMAL_TREND_DROPPING 趋势；若两次相同，则为 THERMAL_TREND_STABLE 趋势。另外，THERMAL_TREND_RAISE_FULL，THERMAL_TREND_DROP_FULL 这两种趋势状态，在 Allwinner 平台未使用。

2.2.2 power allocator policy

power_allocator 温控策略属于闭环控制，也称 IPA (Intelligent Power Allocator)，其核心是利用 PID 控制器，以 Thermal Zone 的温度作为输入，计算可分配功耗值作为输出，然后调节每个 Allocator 的频率和电压值，使每个 Allocator 的功耗值与所分配的一致。

举个例子，假如系统 CPU 消耗 P 的功耗，会产生 Q 的热量，且此时产生的热量刚好使系统产热和散热平衡，即系统温度保持为 T 不变。如果此时 CPU 消耗的功耗增加，产热增加，温度便会上升，温度增加的同时散热会增加，设当再次平衡时，系统温度为 T_a ，此时 T_a 必定大于 T 。相反，如果 CPU 消耗的功耗减少，则系统再次平衡时温度 T_b 必定小于 T 。

IPA 机制的核心就是通过一系列的 PID 闭环控制，使系统的温度一直保持在 T 状态。若系统温度低于 T ，则 PID 计算得出一个大于 P 的功耗值 P_o ，此时调频模块将 CPU 频率迅速升高，使

CPU 消耗的功耗为 P_o ，系统温度升高。相反，若系统温度高于 T ，则 PID 计算得出一个小于 P 的功耗值 P_o ，调频模块将 CPU 频率迅速降低，使 CPU 消耗的功耗为 P_o ，系统温度降低。

在此示例中， T 就是目标温度，也是 IPA 的核心，IPA 温控机制致力于将系统温度控制在目标温度附近。注意，IPA 并不能总是保证系统温度一直为 T 。

原因一，当系统负载很高时，IPA 计算输出的功耗值和实际控制 CPU 消耗的功耗不可能一致，且环境也会扰动，因此系统温度会在 T 上下波动；

原因二，当系统负载低时，即使 CPU 以最高频率运行，其消耗的功率也可能达不到 IPA 计算输出的功耗值 P_o ，所以系统温度会下降，如负载持续很低，那么 CPU 可以一直保持最高频率运行。

也正因此，我们一般会设置一个触发温度 T_{trip0} (T_{trip0} 必须小于 T ，两者一般相差 10 摄氏度)，只有当系统温度超过 T_{trip0} 时，才生效 IPA 温控策略。

如上，这便是 Allwinner 平台上使用的 IPA 机制的运行过程和基本原理。事实上，IPA 机制远不止于此，它还考虑了系统中不止 CPU 产热，GPU，DRAM 等设备也可以调频控温的情况，但在 Allwinner 平台上未使用，因此本文中均不涉及。

另外，在 IPA 使用时，我们需要配置很多参数，如 PID 参数，CPU 频率和功耗的关系，以及当系统在目标温度 T 上动态平衡时的功耗等。这些参数有些需要测试，有些需要计算得到，目前我们还没有统一配置说明，所以只在 DTS 中提供了一套完整的参数，供客户评估。如果不能满足您得要求，请联系我们。而且，由于这套机制配置和 step-wise 的配置会冲突，二者只能选配其一，所以目前只在 R818，MR813，R329 等平台 SDK 上有推广试用（R528，D1 待推广），如果你正在使用这些平台，但却不想使用这套机制，也可以联系我们修改回 step-wise 配置。

2.3 平台差异

2.3.1 Linux 3.4 温控框架

主要对应 Linux 内核版本为 3.4 的 Tina 平台。

例如：R58

2.3.2 Linux 4.x_old 温控框架

主要对应 Linux 内核版本 4.4/4.9，且 DTS 描述中有 "allwinner,thermal_sensor" "allwinner,budget_cooling" 设备的平台。

例如：R328

2.3.3 Linux 4.x_new 温控框架

主要对应 Linux 内核版本 4.9，且 DTS 描述中有 "allwinner,sun**-ths 设备的平台。

例如：MR813、R818、R329

2.3.4 Linux 5.x 温控框架

主要对应使用 Linux 内核版本为 5.4，且 DTS 描述中有 "allwinner,sun**-ths 设备的平台。

例如：R528、D1



3 Step-Wise 温控机制参数配置

3.1 Linux 3.4 温控框架中的 step-wise 参数配置

在 Tina Linux-3.4 的平台中，对于使用 step-wise 所需的配置由 sysconfig 描述。然后由 allwinner 实现驱动和文件的解析，然后完成注册操作，不依赖与内核 devicetree 机制。

```
;-----  
;thermal configuration  
;-----  
[ths_para]  
ths_used          = 1 ; 1: 使用 0: 不使用  
ths_trend         = 0 ;  
ths_trip1_count   = 6 ; 正常触发点trip1数量  
ths_trip1_0       = 50 ; 触发点0对应的温度(摄氏度),下同  
ths_trip1_1       = 60  
ths_trip1_2       = 70  
ths_trip1_3       = 85  
ths_trip1_4       = 95  
ths_trip1_5       = 105  
ths_trip1_6       = 0  
ths_trip1_7       = 0  
ths_trip1_0_min   = 0 ; 触发点0对应的最小cooling设备状态,下同  
ths_trip1_0_max   = 1 ; 触发点0对应的最大cooling设备状态,下同  
ths_trip1_1_min   = 1  
ths_trip1_1_max   = 2  
ths_trip1_2_min   = 2  
ths_trip1_2_max   = 3  
ths_trip1_3_min   = 3  
ths_trip1_3_max   = 5  
ths_trip1_4_min   = 5  
ths_trip1_4_max   = 7  
ths_trip1_5_min   = 7  
ths_trip1_5_max   = 7  
ths_trip1_6_min   = 0  
ths_trip1_6_max   = 0  
ths_trip2_count   = 1 ; 关机触发点trip2数量  
ths_trip2_0       = 105 ; trip2描述的为关机温度  
;-----  
;cooler_table cooler_count <=32  
;-----  
[cooler_table]  
cooler_count = 8  
; 依次表示freq(culster0) [cpu0-cpu3] freq(culster1) [cpu4-cpu7]  
; 非常大的数4294967295, 表示0xffffffff, 表示culster*四核全关  
cooler0 = "0 4 0 4"  
cooler1 = "1 4 1 4"  
cooler2 = "2 4 2 4"  
cooler3 = "3 4 3 4"  
cooler4 = "3 4 3 2"
```

```
cooler5 = "3 4 4294967295 0"  
cooler6 = "3 2 4294967295 0 1"  
cooler7 = "3 1 4294967295 0 1"  
;-----
```

3.2 Linux 4.x_old 温控框架中的 step-wise 参数配置

在本版驱动中，逐渐开始使用内核的 devicetree 机制完成部分配置的解析，但在实现中采用了很多 Allwinner 特有的配置，但各模块配置已逐渐趋向于内核参考配置。

3.2.1 获取温度模块

我们可以将一个获取温度模块，主要分为两个部分：

- 一、对温度传感器的统一配置和管理；
- 二、将温度获取功能注册到 Linux Thermal 框架中；

```
sunxi_thermal_sensor:thermal_sensor{  
    compatible = "allwinner,thermal_sensor"; # 匹配驱动  
    reg = <0x0 0x05070400 0x0 0x100>; # 寄存器地址描述  
    interrupts = <GIC_SPI 49 IRQ_TYPE_NONE>; # 中断描述  
    clocks = <&clk_hosc,&clk_ths>; # 时钟描述  
    sensor_num = <1>; # 描述系统中sunxi_thermal_sensor支持的所有传感器的数量  
    combine_num = <1>; # 其值与子节点ths_combine*个数对应，描述注册为Thermal Zone Device的分组个数  
    alarm_temp = <100000>; # 告警温度，未使用  
    shut_temp = <110000>; # 关机温度，超出此温度，会打印提示信息  
    status = "okay"; # 使能该驱动  
  
    # 描述 注册为Thermal Zone Device的分组信息  
    # 一般来说，Allwinner平台都是按照sensor所属的cpu,gpu域来进行分组的  
    ths_combine0:ths_combine0{  
        compatible = "allwinner,ths_combine0"; # 匹配驱动  
        #thermal-sensor-cells = <1>;  
        combine_sensor_num = <1>; # 本分组中的传感器数量  
        combine_sensor_type = "CPU"; # 本分组所属域类型，一般有cpu,gpu等  
        combine_sensor_temp_type = "max"; # 温度值类型，一般有max, min,avg  
        combine_sensor_id = <0>; # 本分组中的传感器在所有传感器中的编号  
    };  
};
```

首先描述了一个 sunxi_thermal_sensor:thermal_sensor 设备节点，通过匹配 “allwinner,thermal_sensor” 驱动来初始化该设备的各项参数，这是获取温度模块的第一部分功能。

然后，定义了一个子设备节点 ths_combine0，匹配 “allwinner,ths_combine0” 驱动，将初始化好的温度传感器按照位置分组，并将其注册到 Linux Thermal 中，这是获取温度模块的第二部分功能。每一个子节点将注册成为一个 Thermal Zone Device，同时也代表一个温度域，例如 cpu 温度域，gpu 温度域等等。

在 R328 平台中，由于不存在 gpu，因此这里没有描述 gpu 域的相关分组。如需要添加，则只需添加一个 gpu 域对应的 ths_combine1 的子设备节点，并修正父节点的 combine_num 计数即可。

3.2.2 控制温度模块

在 Linux 4.x 温控模块旧版的实现中，实现了一个 sunxi_cooling_device 模块，该模块仅仅是 cpu, gpu 等最高运行频率属性的一个抽象，通过对该设备的操作，即可设置 cpu 运行时的最高频率，从而降低 CPU 热量的产生，达到温控的目的。当然，cpu 关核也是降低产热的一种方式，因此在驱动中也抽象了 cpu hotplug 的属性。

该模块也可大致分为二个部分：

代码路径：drivers/thermal/sunxi_thermal/sunxi_cooling_device/

一、对 cpu 属性的抽象，并将其注册成一个 Thermal Cooling Device。主要降低 cpu 的产热量，其包括限制 cpu 最高运行频率和限制 cpu 核运行数量。关键源码：sunxi-budget-cooling.c, sunxi-budget-cooling-dvfs.c, sunxi-budget-cooling-hotplug.c

二、对 gpu 属性的抽象，并将其注册成一个 Thermal Cooling Device。主要是降低 gpu 的产热量，包括限制 gpu 运行频率。关键源码：sunxi_gpu_cooling.c

根据 sunxi_cooling_device 驱动的划分，在这些平台中，其 dts 定义了两个虚拟设备节点，分别描述 cpu, gpu 降温属性，分别是 cpu_budget_cooling, gpu_cooling，因此需要在 dts 中分别描述它。

- cpu_budget_cooling

这个节点主要描述该平台 cpu 降温等级数量，以及每个级别对应的 cpu 频率和能使能的最大 cpu core 数。例如：达到 state5 级别，表示该 cpu cluster 最高运行 648000kHz，最多使用 2 个 cpu core 工作。

```
cpu_budget_cooling:cpu_budget_cool{
    compatible = "allwinner,budget_cooling"; # 匹配驱动
    #cooling-cells = <2>;
    status = "okay"; # 使能该驱动
    state_cnt = <7>; # 状态级别数量
    cluster_num = <1>; # cpu cluster数量

    # 每种状态对应的限制频率，及运行核数
    state0 = <1008000 4>;
    state1 = <912000 4>;
    state2 = <816000 4>;
    state3 = <720000 4>;
    state4 = <648000 4>;
    state5 = <648000 2>;
    state6 = <648000 1>;
};
```


如果当前平台 cpu cluster 不止一个，则需要在每种状态中，指定每个 cluster 的运行频率和最大运行 cpu core 数，如下（仅供参考）：

```
cluster_num = <2>;
#stateN = <cluster0(freq maxnum) cluster1(freq maxnum) ... >
state0 = <1104000 3 1104000 3>;
state1 = <1008000 3 1008000 3>;
state2 = <912000 3 912000 3>;
state3 = <912000 3 912000 0>;
state4 = <720000 2 720000 0>;
state5 = <600000 1 600000 0>;
```

• gpu_cooling

这个节点主要描述该平台 gpu 降温等级数量，以及每个降温状态对应的 gpu 频率，描述方式与 cpu 类似，不再赘述。需要说明的是：这里的state0 = <4>，表示降温状态 0，对应 gpu 频率为 gpu dvfs 表中的第 4 项。

```
gpu_cooling:gpu_cooling{
    compatible = "allwinner,gpu_cooling"; # 匹配驱动
    device_type = "gpu_cooling";
    reg = <0x0 0x0 0x0 0x0>;
    #cooling-cells = <2>;
    status = "okay";
    state_cnt = <4>; # 状态级别数量

    state0 = <4>; # state0 对应gpu频率在gpu dvfs表中的标号
    state1 = <3>;
    state2 = <2>;
    state3 = <1>;
};
```

3.2.3 温度控制策略

上文说到，在 Linux thermal 中，我们需要在配置文件中描述 trips 和 binds，才能使 policy 正确工作。为了简化这个描述，Linux thermal 提供了 of-thermal 模块，只需要在 DTS 文件内，按照 of-thermal 规定的格式定义不同的 thermal-zone 关系域即可。of-thermal 模块就会根据 DTS 描述的内容自动注册控制关系，使驱动代码量大大减少。具体配置说明详见 Linux document，下文仅简述部分 Allwinner 平台的配置。

首先描述的每个 thermal-zone 关系域，都必须先绑定一个 Thermal Zone Device，并指明该设备轮询温度的间隔时间。

如下文 cpu_thermal_zone 关系域，其绑定了 ths_combine0 这个 Thermal Zone Device，并定义了普通状态下轮询温度间隔为 2000ms，被动散热状态下轮询温度间隔减小到 1000ms。另外，由于在目前 Linux thermal 框架中，每个 thermal-zone 关系域仅支持一个 sensor，所以在thermal-sensors = <&ths_combine0 0>;属性中，还需要指定使用 ths_combine0 的哪个传感器（即 sensor ID），例如此处的 sensor 0。

然后，需要描述这个 thermal-zone 关系域中所有的温度 trips。

最后，为每个 trip 绑定一个降温状态的集合，如下 cooling-maps。

```
thermal-zones{
    cpu_thermal_zone{
        polling-delay-passive = <1000>; # 被动冷却状态下，轮询温度间隔
        polling-delay = <2000>; # 普通状态下，轮询温度间隔
        thermal-sensors = <&ths_combine0 0>; # 绑定的Thermal Zone Device

        # 温度触发点
        trips{
            cpu_trip0:t0{
                temperature = <65000>; # 对应65摄氏度时触发
                type = "passive"; # 类型为 passive
                hysteresis = <0>; # 滞后切换延时，用来执行绑定动作的延时。一般用于风扇延时启停，但此处
                # 为切换频率，不需要延时。
            };
            cpu_trip1:t1{
                temperature = <75000>;
                type = "passive";
                hysteresis = <0>;
            };
            ... # 省略一些
            crt_trip:t8{
                temperature = <110000>;
                type = "critical";
                hysteresis = <0>;
            };
        };

        # 绑定关系
        cooling-maps{
            bind0{
                contribution = <0>;
                trip = <&cpu_trip0>; # 绑定的温度trips
                cooling-device
                = <&cpu_budget_cooling 1 1>; # 数值标号 对应在cpu_budget_cooling节点中配置的各种
                # 降温状态，1 1 表示最大和最小都是该状态。
            };
            ... # 省略一些
            bind4{
                contribution = <0>;
                trip = <&cpu_trip4>;
                cooling-device
                = <&cpu_budget_cooling 6 6>;
            };
        };
    };
};
```

3.3 Linux 4.x_new 及 Linux 5.x 温控框架中的 step-wise 参数配置

3.3.1 获取温度模块

在新版本实现中，获取温度模块不再分为两个部分，而是直接配置成一个设备。这个设备包含 cpu, gpu 等所有的温度传感器，并为所有的传感器执行初始化，注册为 Thermal Zone Device 的操作。

```
ths: thermal_sensor{
    compatible = "allwinner,sun50iw10pl-ths"; # 匹配驱动
    reg = <0x0 0x05070400 0x0 0x400>; # 寄存器资源
    clocks = <&clk_ths>; # 时钟资源
    clock-names = "bus";
    nvmem-cells = <&ths_calib>; # 一个nvmem设备引用，用于读取校准值
    nvmem-cell-names = "calibration"; # 校准值的引用名
    #thermal-sensor-cells = <1>;
};
```

以上描述由 Allwinner 配置，均不需要修改。

3.3.2 控制温度模块

在新版实现中，控制温度的方式仍然是使用 cpu 降频降压等方式。但新实现中直接使用 Linux 通用的 cpu-cooling, dev-cooling 设备驱动，不再使用 Allwinner 实现的 sunxi_cooling_device 模块。

因此，不需要在 dts 中描述并抽象 cpu_budget_cooling, gpu_cooling 设备。这一部分内容也由 Allwinner 配置，且由于调频调压涉及系统稳定性，客户不可修改。

如需知道当前系统支持哪些调频的频率点，可通过读取 sysfs 节点获取，如下：

```
# 各平台频点存在差异，因此该示例仅供参考，请以实际为准
root@TinaLinux:/# cat /sys/devices/system/cpu/cpu0/cpufreq/policy0/scaling_available_frequencies
408000 720000 1008000 1152000 1320000 1440000 1560000
```

3.3.3 温度控制策略

在新版本中，温度控制策略描述仍然使用 Linux Thermal 提供的 of_thermal 模块来构建关系，故此处配置，仍然可参考上文旧版说明或 Linux Document 说明。

```

thermal-zones {
    cpu_thermal_zone {
        polling-delay-passive = <500>; # 当温控策略激活时，即触发第一个trip后，温度采样间隔
        polling-delay = <1000>; # 当温控策略未激活时，温度采样间隔
        thermal-sensors = <&ths 0>; # 此 cpu_thermal_zone 对应的sensor

        trips{
            cpu_trip0:t0{ #第一个 trip 点
                temperature = <95000>;
                type = "passive";
                hysteresis = <0>;
            };
            cpu_trip1:t1{
                temperature = <100000>;
                type = "passive";
                hysteresis = <0>;
            };
            cpu_trip2:t2{
                temperature = <110000>;
                type = "passive";
                hysteresis = <0>;
            };
            crt_trip:shutdown{ # 最后一个 trip 点， 用于过温关机保护
                temperature = <120000>;
                type = "critical";
                hysteresis = <0>;
            };
        };
        cooling-maps {
            bind0{ # trip0 对应的 降温状态
                contribution = <0>;
                trip = <&cpu_trip0>;
                cooling-device = <&cpu0 1 1>; #cpufreq 的频点从最高频率到最小频点排序，从0开始标
注，0对应最高频率
            };
            bind1{
                contribution = <0>;
                trip = <&cpu_trip1>;
                cooling-device = <&cpu0 2 2>;
            };
            bind2{
                contribution = <0>;
                trip = <&cpu_trip2>;
                cooling-device = <&cpu0 3 3>;
            };
        };
    };
    gpu_thermal_zone{
        polling-delay-passive = <500>;
        polling-delay = <1000>;
        thermal-sensors = <&ths 1>;
        sustainable-power = <1100>;
    };
    ddr_thermal_zone{
        polling-delay-passive = <0>;
        polling-delay = <0>;
        thermal-sensors = <&ths 2>;
    };
};

```

4 Power allocator 温控机制参数配置

4.1 Linux 4.x_new 及 Linux 5.x 温控框架中的 Power allocator 参数配置

获取温度模块和控制温度模块的配置和实现与调频策略无关，因此这两部分内容与 step-wise 配置相同，可参考上文 step-wise 中对该部分的说明。

4.1.1 温度控制策略

在新版本中，温度控制策略描述仍然使用 Linux Thermal 提供的 of_thermal 模块来构建关系，故此处配置仍然可参考上文旧版说明或 Linux Document 说明。

不过由于新版采用了 IPA 降温策略，在描述 thermal-zones 关系域时还是有一定差异的，这里简要说明一下。

IPA 是 linux 内核基于 PID 算法实现的一种温控策略，其核心是根据当前温度和预设目标温度的偏差来调节分配给设备的功率，进而调节设备温度。其调控结果会将设备维持在一个目标温度附近。

调温过程简述：

- 1、当设备负载高，温度上升达到设定的触发温度 Ttrip0 时（如下文配置中的 cpu_threshold，70 度），IPA 开始生效。生效并不意味着立即降频，而是开始 PID 计算，是否降频由 PID 的计算结果而定。
- 2、当温度继续上升到目标温度 T 附近时（如下文配置中的 cpu_target，80 度），IPA 将根据温度变化情况调整频率，IPA 温控涵盖对历史温度数据的比例，积分，微分运算，会估计温度变化趋势，若温度变化趋势上升，IPA 考虑降频，相反，IPA 会提高设备运行频率，将设备温度维持在 80 度附近。
- 3、若一段时间后设备散热情况变好或 cpu 负载降低，cpu 进入 idle 的时间片增多，即使运行在最高频率，设备温度也会下降。下降到触发温度 Ttrip0 之下时，IPA 就会关闭了。

```
thermal-zones {
    cpu_thermal_zone {
        polling-delay-passive = <500>;
        polling-delay = <1000>;
        thermal-sensors = <&ths 0>; # 绑定0号传感器对应的Thermal Zone Device为cpu_thermal_zone
        设备
```

```

sustainable-power = <800>; #当前温度到达预设的最高值时，系统能分配给 cooling 设备的功率。
k_po = <24>; #PID控制器的参数，不建议更改
k_pu = <48>;
k_i = <0>;

cpu_trips: trips {
    cpu_threshold: trip-point@0 {
        temperature = <70000>; # 当温度高于此温度，IPA开始起作用；
        type = "passive";
        hysteresis = <0>;
    };
    cpu_target: trip-point@1 {
        temperature = <80000>; # IPA的control_temp，也即高于此温度，IPA就会考虑降低可分配
功耗，以达到降低温度的目的。
        type = "passive";
        hysteresis = <0>;
    };
    cpu_crit: cpu_crit@0 {
        temperature = <110000>;
        type = "critical"; # 该类型说明该trip为关机温度触发点
        hysteresis = <0>;
    };
};

cooling-maps {
    map0 {
        trip = <&cpu_target>;
        cooling-device = <&cpu0
THERMAL_NO_LIMIT
THERMAL_NO_LIMIT>; # 绑定的降温设备为cpu0，由cpu-cooling调频
        contribution = <1024>;
    };
};

gpu_thermal_zone{
    polling-delay-passive = <500>;
    polling-delay = <1000>;
    thermal-sensors = <&ths 1>;# 绑定1号传感器对应的Thermal Zone Device为 gpu_thermal_zone
设备
    sustainable-power = <1100>;
};

ddr_thermal_zone{
    polling-delay-passive = <0>;
    polling-delay = <0>;
    thermal-sensors = <&ths 2>;# 绑定2号传感器对应的Thermal Zone Device为ddr_thermal_zone设
备
};
};

```

5 温控机制的 menuconfig 配置

5.1 Linux 3.4 温控框架配置

5.1.1 R58 配置

进入 tina 源码目录，执行 make kernel_menuconfig 进入配置主界面，并按以下步骤操作：

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

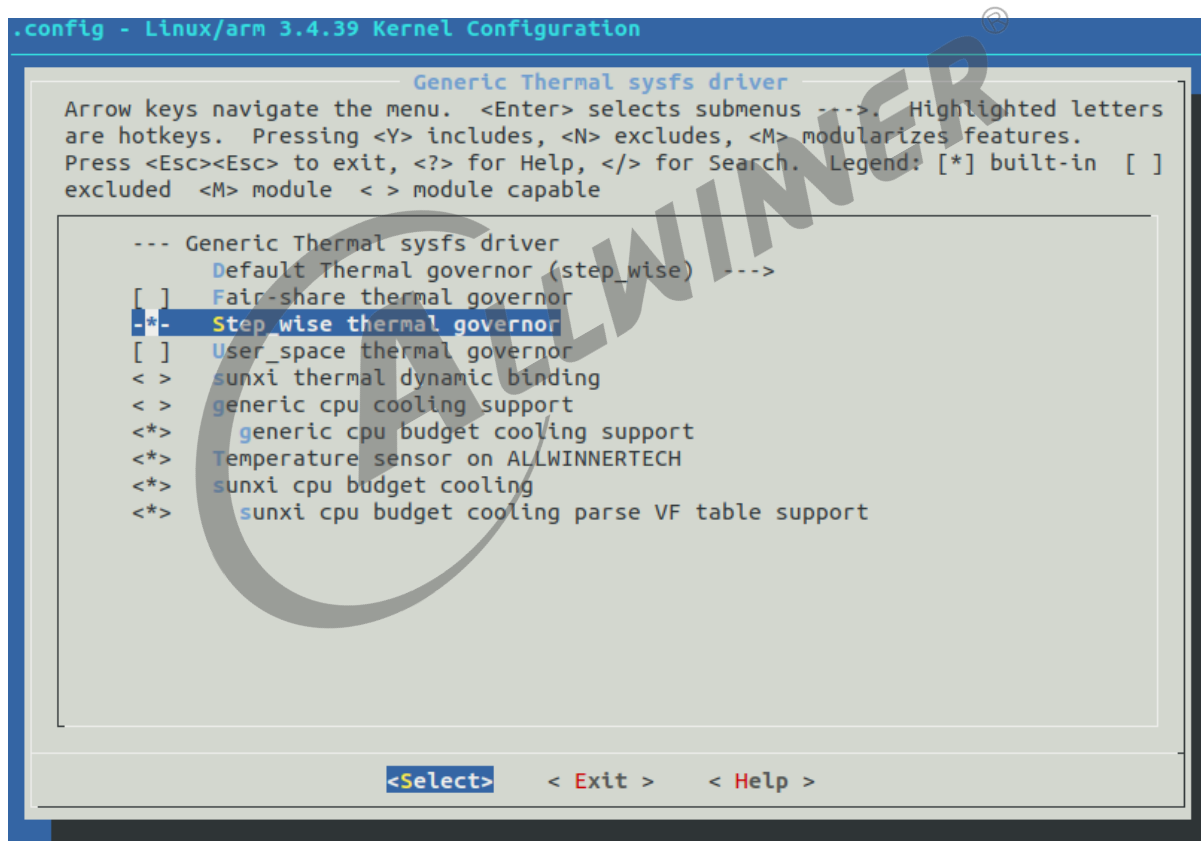


图 5-1: Thermal_menuconfig_R58_001.png

配置项说明：

generic cpu budget cooling support:使能cpu cooling支持
Temperature sensor on ALLWINNERTECH:使能温度传感器
sunxi cpu budget cooling:使能sunxi cpu cooling driver支持
sunxi cpu budget cooling parse VF table support :使能sunxi cpu dvfs driver支持

5.2 Linux 4.x_old 温控框架配置

5.2.1 R328 配置

进入 tina 源码目录，执行 make kernel_menuconfig 进入配置主界面，并按以下步骤操作：

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

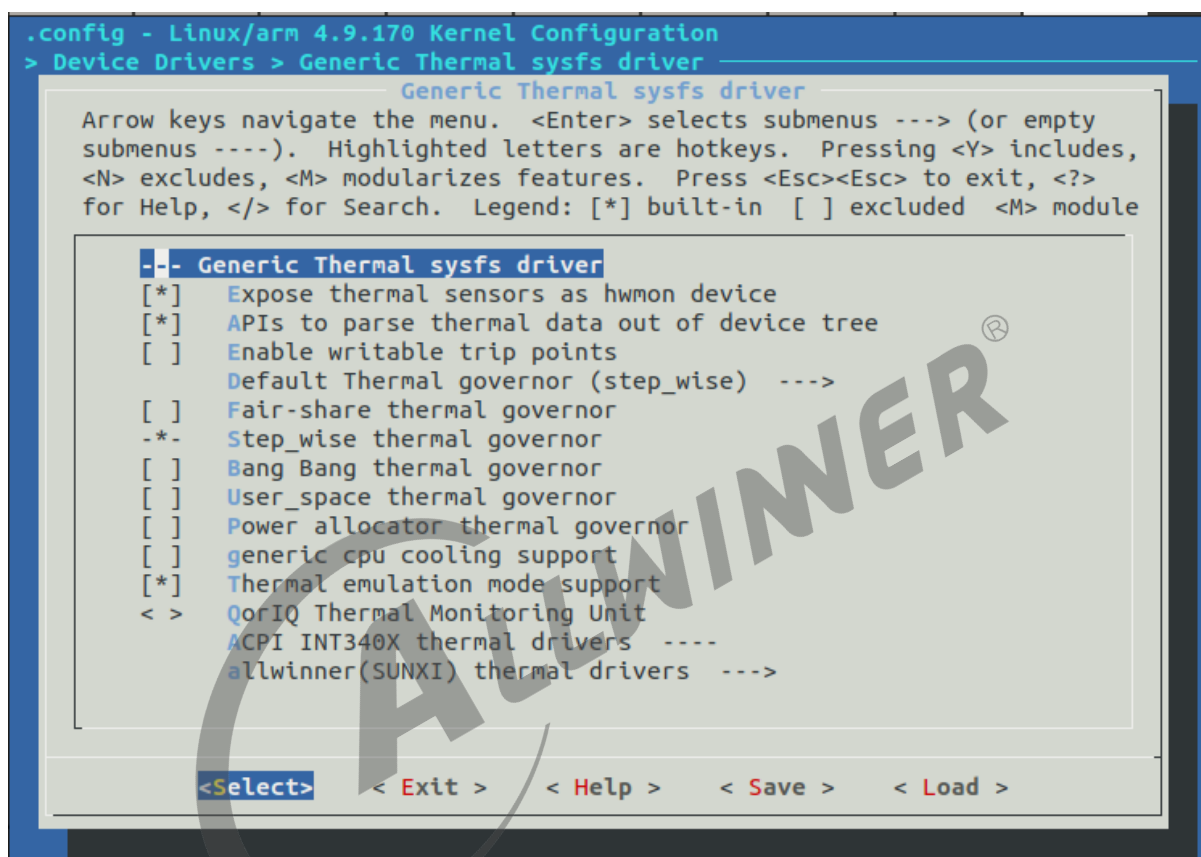


图 5-2: Thermal_menuconfig_R328_001.png

配置项说明：

Default Thermal governor 选择，默认为step-wise
Thermal emulation mode support:支持thermal模拟温度功能

2、thermal cooling drivers 配置，进入到Device Drivers ->Generic Thermal sysfs driver->allwinner(SUNXI)thermal drivers->allwinner(SUNXI)thermal cooling device drivers，如下图所示：

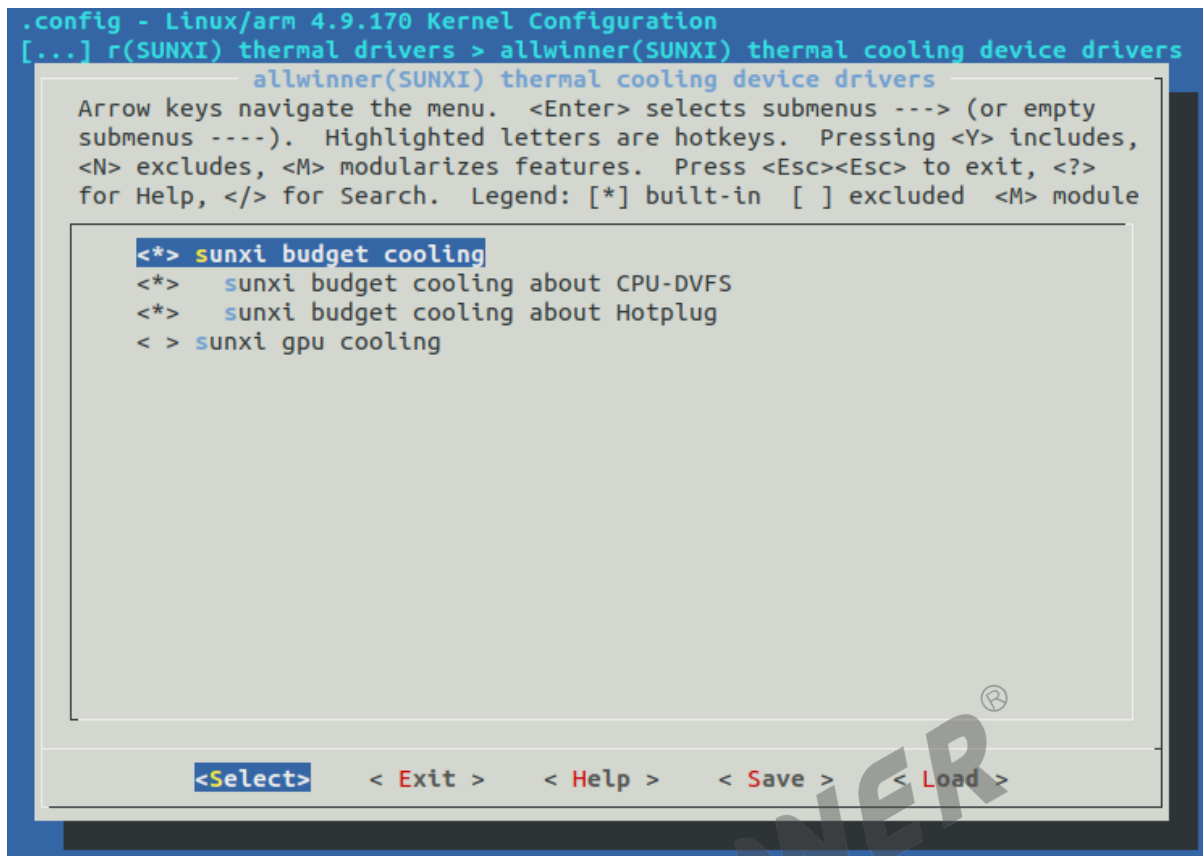


图 5-3: Thermal_menuconfig_R328_002

配置项说明:

Sunxi budget cooling about CPU-DVFS:支持cpu dvfs功能做cooling
Sunxi budget cooling about Hotplug:支持cpu hotplug功能做cooling
Sunxi budget cooling about GPU FS:支持gpu做cooling

3、thermal sensor drivers 配置, 进入到 Device Drivers ->Generic Thermal sysfs driver->allwinner(SUNXI)thermal drivers->allwinner(SUNXI)thermal sensor drivers, 进行 thermal sensor 驱动配置, 如下图所示:



图 5-4: Thermal_menuconfig_R328_003

配置项说明：

Thermal sensor driver:支持thermal sensor驱动
Thermal sensor drive for SUNXI platformr:支持sunxi thermal sensor驱动

5.3 Linux 4.x_new 及 Linux 5.x 温控框架配置

5.3.1 R818/MR813/R329 配置

进入 tina 源码目录，执行 make kernel_menuconfig 进入配置主界面，并按以下步骤操作：

1、首先，进入到Device Drivers -> NVMEM Support，如下图所示：

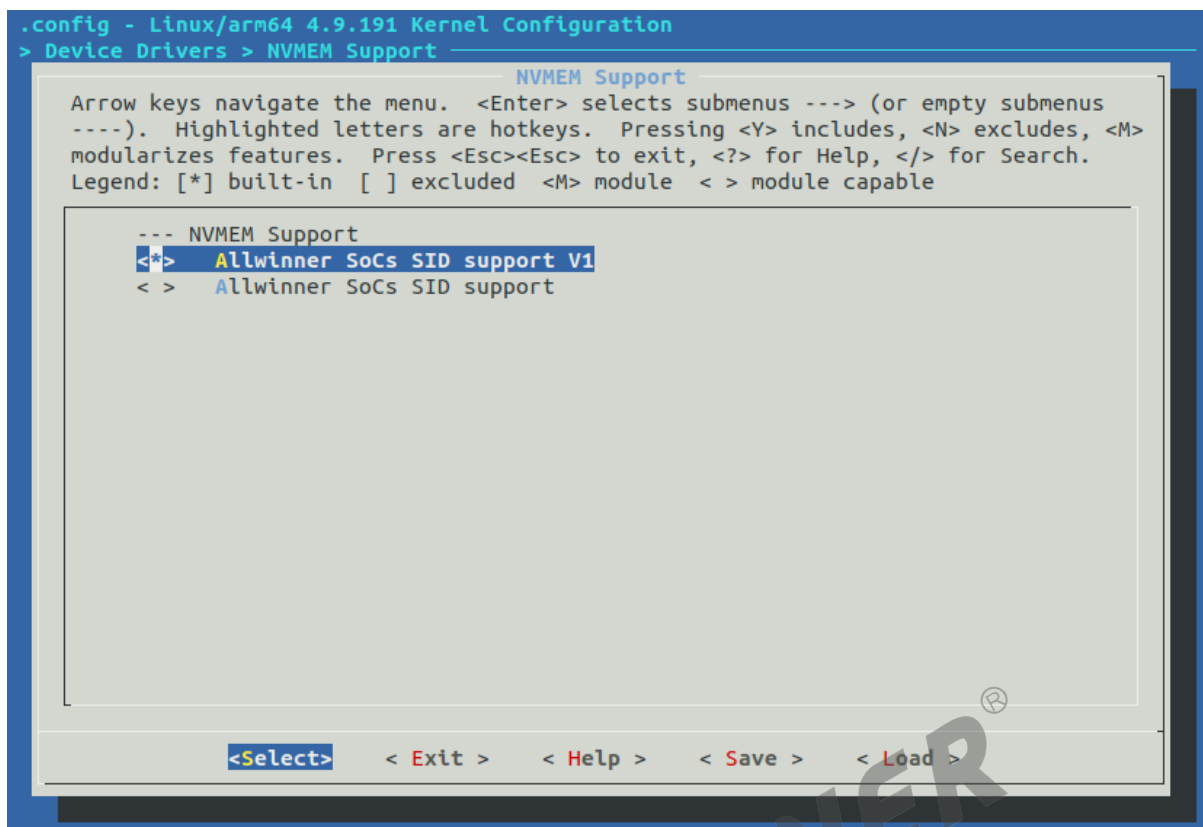


图 5-5: Thermal_menuconfig_R818_001

配置项说明：

Allwinner SoCs SID support V1:使能SID V1驱动支持，主要用于读取校准值
Allwinner SoCs SID support

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

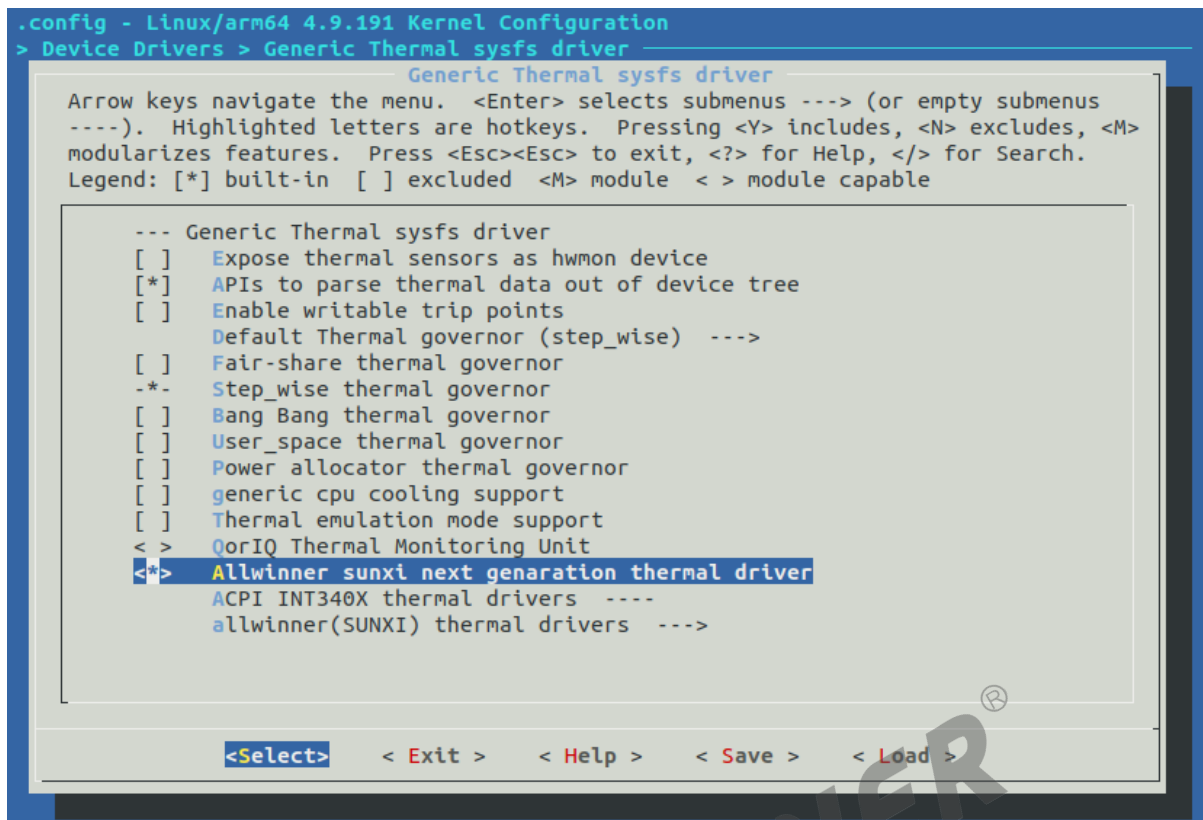


图 5-6: Thermal_menuconfig_R818_002

配置项说明：

Allwinner sunxi next generation thermal driver:使能新版本thermal driver

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

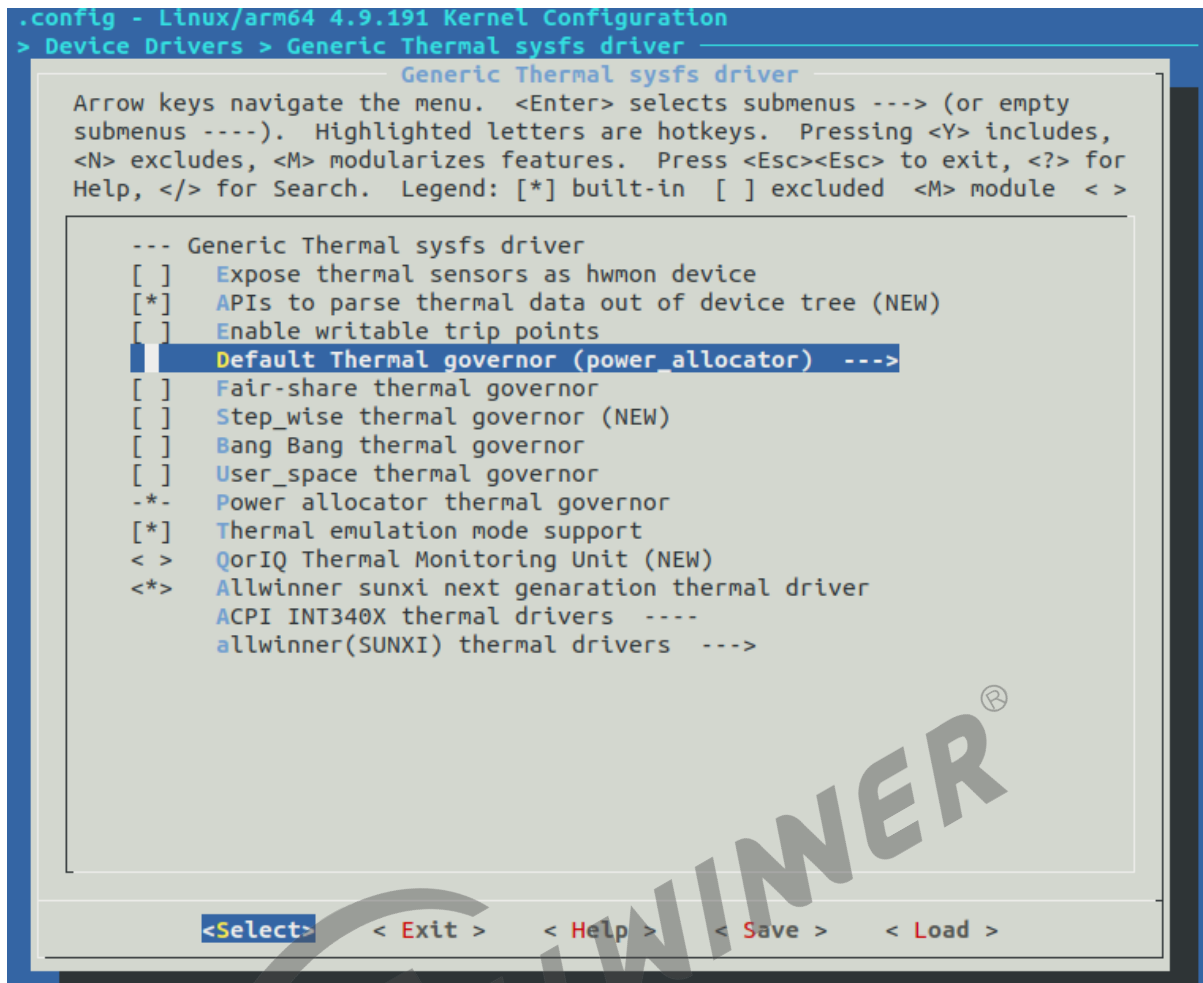


图 5-7: Thermal_menuconfig_R818_003

配置项说明：

Default Thermal governor (power_allocator): 选择使用新的thermal策略

5.3.2 R528 配置

进入 tina 源码目录，执行 make kernel_menuconfig 进入配置主界面，并按以下步骤操作：

1、首先，进入到Device Drivers -> NVMEM Support，如下图所示：

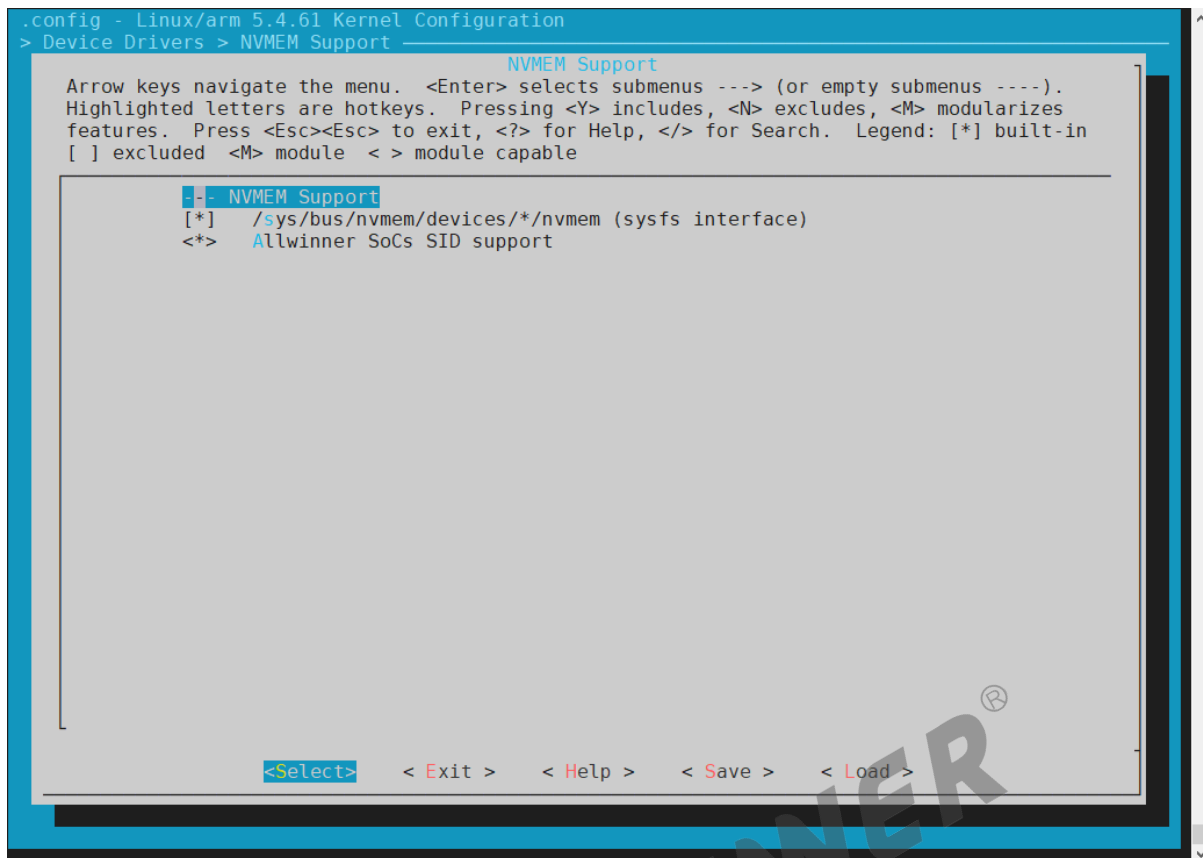


图 5-8: Thermal_menuconfig_R528_001

配置项说明：

[*]	/sys/bus/nvmem/devices/*/nvmem (sysfs interface) #sysfs 调试节点
<*>	Allwinner SoCs SID support #使能 SID 驱动支持，主要用于读取校准值

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

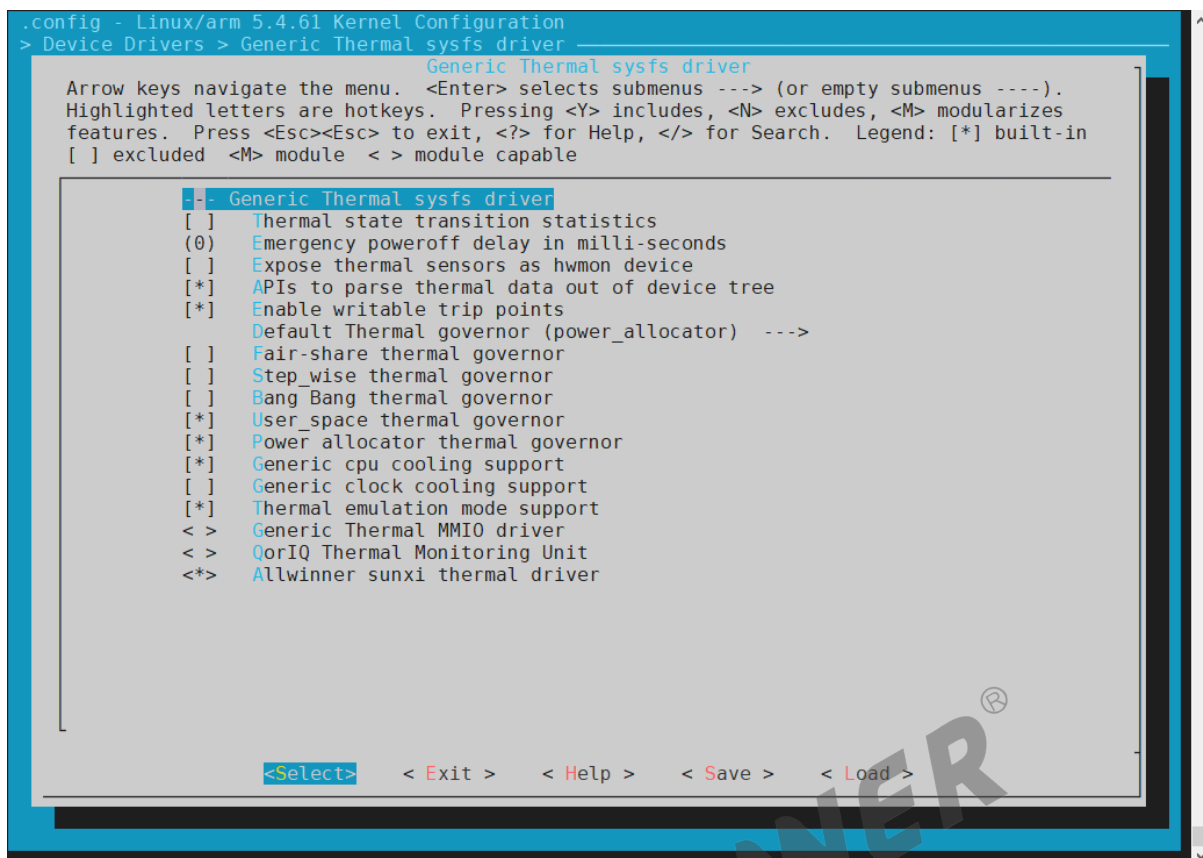


图 5-9: Thermal_menuconfig_R528_002

配置项说明：

Default Thermal governor (power_allocator) ---> # 默认配置项，需要根据DTS配置选择

[*] Step_wise thermal governor # step-wise调频策略，需要根据DTS配置选择，选中后，默认配置才能选择此策略

[*] Power allocator thermal governor # IPA调频策略，需要根据DTS配置选择，选中后，默认配置项才能选择此策略

[*] Generic cpu cooling support # 使能cpu降频

[*] Thermal emulation mode support # 使能 模拟温度，调试用

Allwinner sunxi thermal driver #使能 thermal driver

6 温控框架调试

6.1 基础说明

6.1.1 sysfs 节点

thermal core会在/sys/devices/virtual/thermal目录下创建相应的节点，主要有thermal_zone和cooling_device两部分。

cooling_device的主要节点含义如下：

- type：降温设备的类型（处理器/风扇/...）
- max_state：降温设备的最大降温状态（以处理器为例，将 VF 表的数量划分成对应等级，最大降温等级就是 cpu 的最低电压和频率所在的等级）
- cur_state：降温设备的当前降温状态（以处理器为例，即当前的 cpu 电压和频率所在的等级）

thermal_zone的主要节点含义如下：

- type：温控区间的类型，比如cpu、gpu、ddr等
- temp：温控区间的当前温度
- mode：温控区间的工作状态，比如 enabled、disabled
- policy：温控区间的当前策略
- available_policies：温控区间支持的策略
- trip_point[0-*]_temp：触发点[0-*]的温度
- trip_point[0-*]_type：触发点[0-*]的类型，active、passive、hot、critical（没有风扇等主动设备的情况下一般使用passive类型表示被动降频降压降温，以及使用critical类型表示需要 shutdown 的温度）
- trip_point[0-*]_hyst：触发点[0-*]的热滞系数，环境快速变化时的温度数据滞后现象，热滞系数和测温元件的质量，元件的比热容，热交换系数，元件的有效表面积有关，通常配置为 0
- emul_temp：模拟温度的节点，默认值是 0（设置成 0 就关闭模拟温度的功能）
- sustainable_power：可持续且稳定的能量值
- k_po：power allocator 策略的当前温度超过期望温度的比例系数（PID 算法的 P）
- k_pu：power allocator 策略的当前温度未达期望温度的比例系数（PID 算法的 P）
- k_i：power allocator 策略的积分系数（PID 算法的 I）
- k_d：power allocator 策略的微分系数（PID 算法的 D）
- integral_cutoff：累计误差的偏移量
- slope：线性外推法的斜率
- offset：线性外推法的偏移

6.2 基础操作

- 查看 thermal_zone 参数

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*。当然，这些目录是/sys/devices/virtual/thermal/thermal_zone*的软连接。

下面以 thermal_zone0 为例，介绍几种常用的操作：

1、查看 thermal_zone0 的类型 (cpu_thermal_zone 为 cpu 温度域，gpu_thermal_zone 为 gpu 温度域)

查看 cpu、gpu 温度时，先通过此节点确定其与thermal_zone*设备的对应关系，然后查看对应thermal_zone*设备的temp值即可。

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

2、查看 thermal_zone0 的温度

```
// 一些平台是两位数值，单位 摄氏度
#cat /sys/class/thermal/thermal_zone0/temp
36
// 新平台一般是5位数值，单位 千分之一摄氏度
#cat /sys/class/thermal/thermal_zone0/temp
24522
```

3、thermal_zone0 的温控 (开：enabled；关：disabled)

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

4、过温关机温度配置

配置关机温度只需要建立一个 trip，并将其类型配置为 critical 即可，具体可查看上文trips和 binds说明一节描述。

```
    crt_trip:t8{
        temperature = <110000>;
        type = "critical";
        hysteresis = <0>;
    };
```

- 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。

1、设置 thermal_zone0 的模拟温度

```
#echo 80 > /sys/class/thermal/thermal_zone0/emul_temp  
// 注： 这里的数值单位与/sys/class/thermal/thermal_zone0/temp节点单位一致
```

2、关闭 thermal_zone0 的模拟温度功能

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






著作权声明

版权所有 © 2021 珠海全志科技股份有限公司。保留一切权利。

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

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

商标声明

、 全志科技、（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。