Tina Linux 温度控制
使用指南

版本号: 1.2
发布日期: 2024.10.23

版本历史

版本号	日期	制/修订人	内容描述
1.0	2021.04.07	AWA1610	初始版本
1.1	2023.11.03	AWA1812	**第一章 前言** 1.3 小节更新适用范围 1.4 小节增加文档约定 **第二章 模块配置** 2.1 小节增加 linux5.15 内核配置
1.2	2024.10.23	AWA1812	适配新的文档模板，提供新版内容

目　录

插　图

1 前言

1.1 文档简介

介绍 Tina 温度控制机制实现及相关模块的配置使用。

1.2 目标读者

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

1.3 适用范围

表 1-1: 适用产品列表

产品名称	内核版本
R58	Linux-3.4
R328	Linux-4.9
MR813	Linux-4.9
MR813B	Linux-4.9
R818	Linux-4.9
R818B	Linux-4.9
R329	Linux-4.9
R528	Linux-5.4
H133	Linux-5.4
T113	Linux-5.4
D1	Linux-5.4
MR527	Linux-5.15
AI985	Linux-5.15
MR536	Linux-5.15
V821	Linux-5.4

1.4 文档约定

1.4.1 标志说明

⚠ 注意

- **提醒操作中应注意的事项。不当的操作可能会损坏器件，影响可靠性、降低性能等。**

说明

为准确理解文中指令、正确实施操作而提供的补充或强调信息。

技巧

一些容易忽视的小功能、技巧。了解这些功能或技巧能帮助解决特定问题或者节省操作时间。

1.4.2 地址与数据描述方法约定

本文档在描述地址、数据时遵循如下约定：

表 1-2: 地址与数据描述方法约定

符号	例子	说明
0x	0x0200	地址或数据以 16 进制表示。
0b	0b010	数据采用二进制表示 (寄存器描述除外)。
X	00X	数据描述中，X 代表 0 或 1，例如，00X 代表 000 或 001。

1.4.3 数值单位约定

本文档在描述数据容量（如 NAND 容量）时，单位词头代表的是 1024 的倍数；描述频率、数据速率等时则代表的是 1000 的倍数。具体如下：

表 1-3: 数值单位约定

类型	符号	对应数值
数据容量	1 K	1024
数据容量	1 M	1 048 576
数据容量	1 G	1 073 741 824
频率，数据速率等	1 K	1000
频率，数据速率等	1 M	1 000 000
频率，数据速率等	1 G	1 000 000 000

1.5 相关术语介绍

表 1-4: 术语列表

术语	解释说明
thermal	温控系统
thermal zone	温度控制域

2 模块介绍

2.1 模块功能

温控（thermal）模块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。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，需要思考两个关键点：

（1）如何降低温度

（2）何时降低温度

在 Tina 平台上，thermal 模块通过在合适的时机对系统降频来控制温度。

2.2 软件框架

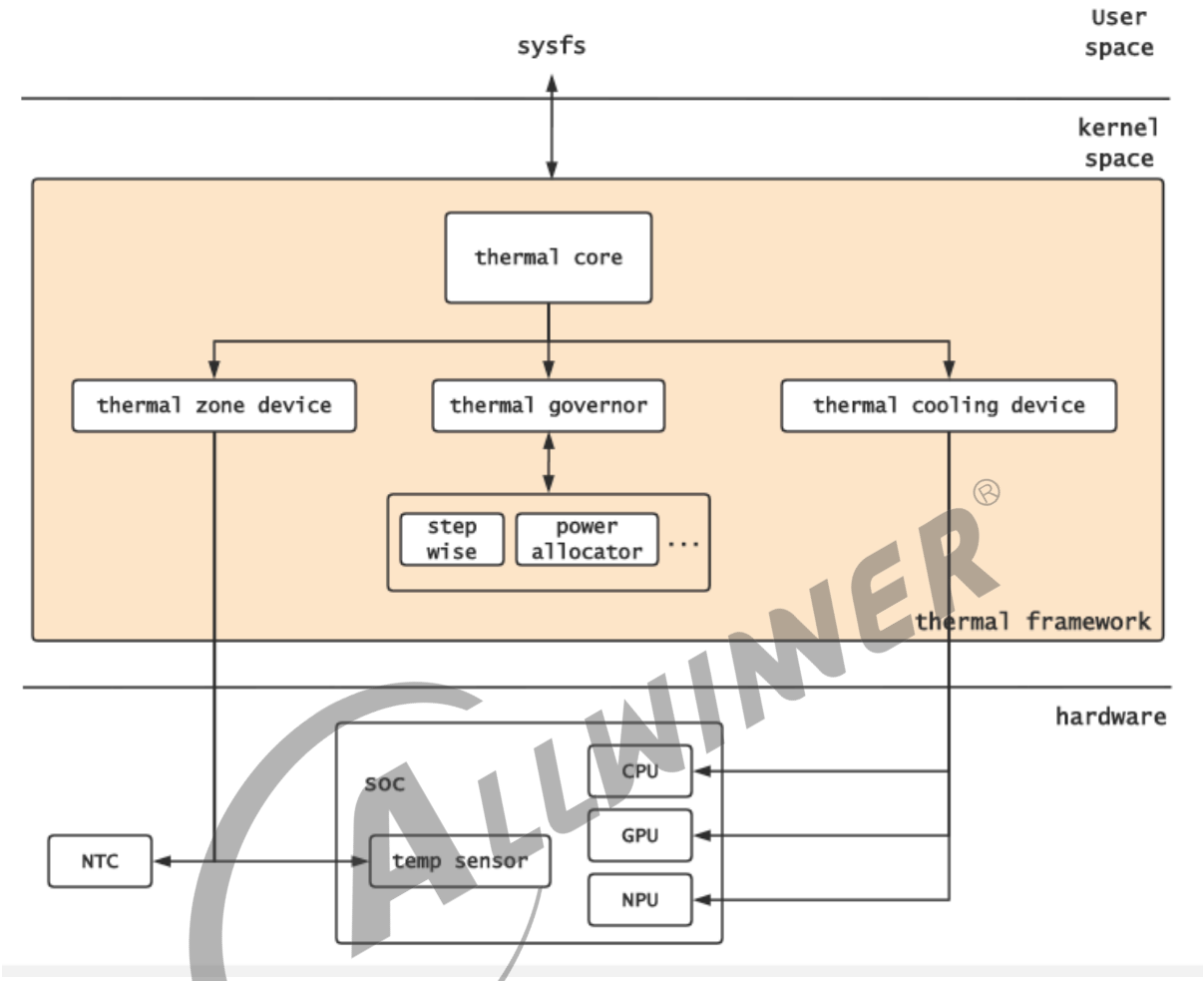

图 2-1: thermal framework

thermal framework 抽象出 Thermal Zone Device、Thermal Cooling Device、Thermal Governor、Thermal Core 四个部分。

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。

2.2.1 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，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。

2.2.2 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。

2.2.3 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，当前的温度升高速很快，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。

2.2.3.1 Thermal Governor 相关说明

- 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

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。

2.2.4 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。

2.3 模块配置

说明

早期平台（R58、R328）的 thermal 配置没有完全适配内核的设备树机制与标准 thermal 框架，会存在较多全志平台独有的配置项。
R58、R328 之外的其他平台已完成适配，采用标准方式配置设备树中 thermal 框架所需配置项。
为方便理解，在这里
（1）将 R58 平台的温控配置称为：Linux-3.4 配置；
（2）将 R328 平台的温控配置称为：Linux-4.x-old 配置。

Linux 标准温控框架的 Device tree 配置项主要包含以下三点：

（1）sensor device 配置：即用于获取温度的设备的配置；

（2）cooling device 配置：即 cpu、gpu 设备节点，thermal 一般只是引用它们的节点；

（3）of-thermal 配置：of-thermal 是 Linux thermal 框架的模块，通过 dts 配置为 Thermal Zone Device、Thermal Cooling Device、Thermal Governor 三者建立起描述关系。

使用不同的温控策略，of-thermal 配置会有所不同。全志平台主要支持 step-wise 以及 Power allocator 两种温控策略。

2.3.1 menuconfig 配置说明

2.3.1.1 Linux 3.4 配置

主要适用于 Linux 内核版本为 3.4 的 Tina 平台，例如 R58。

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：

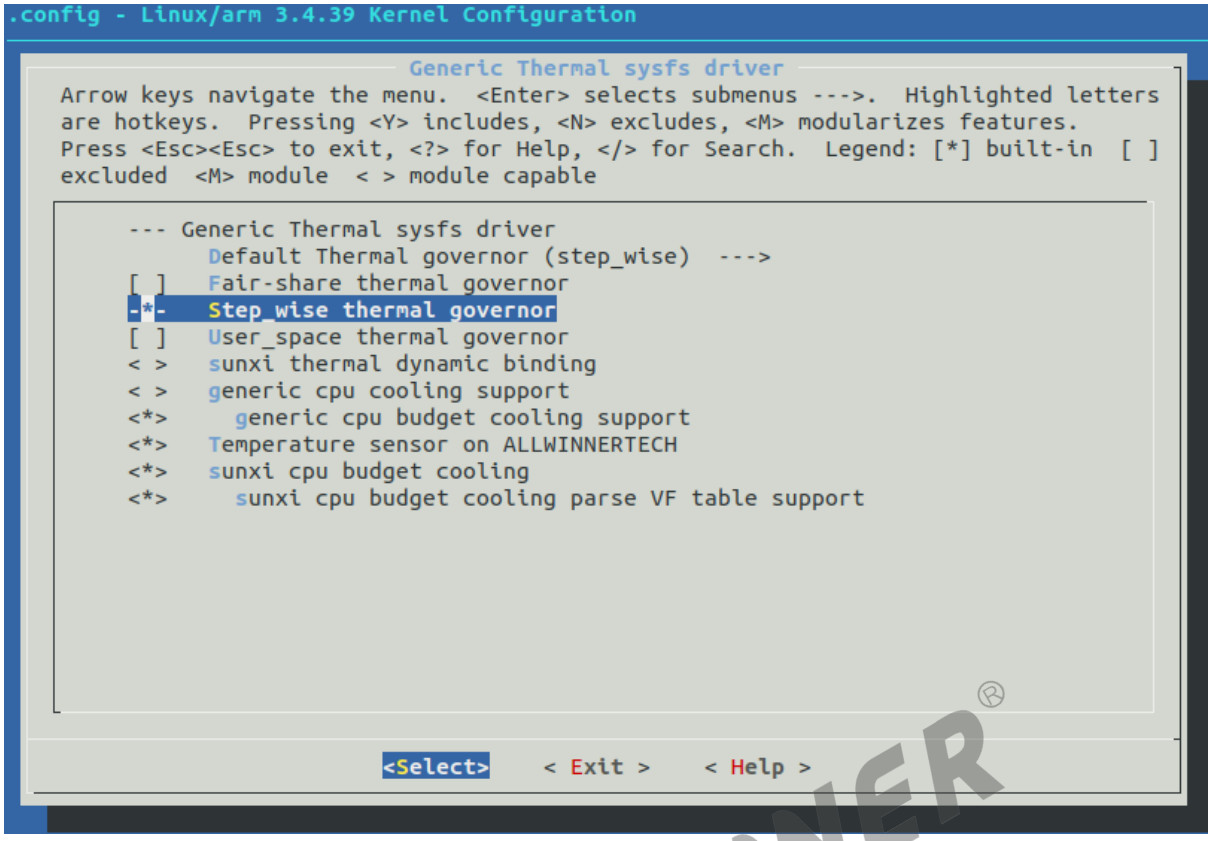

图 2-2: Thermal_menuconfig_R58_001.png

关键配置项：

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

2.3.1.2 Linux 4.x_old 配置

主要适用于 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 。

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：

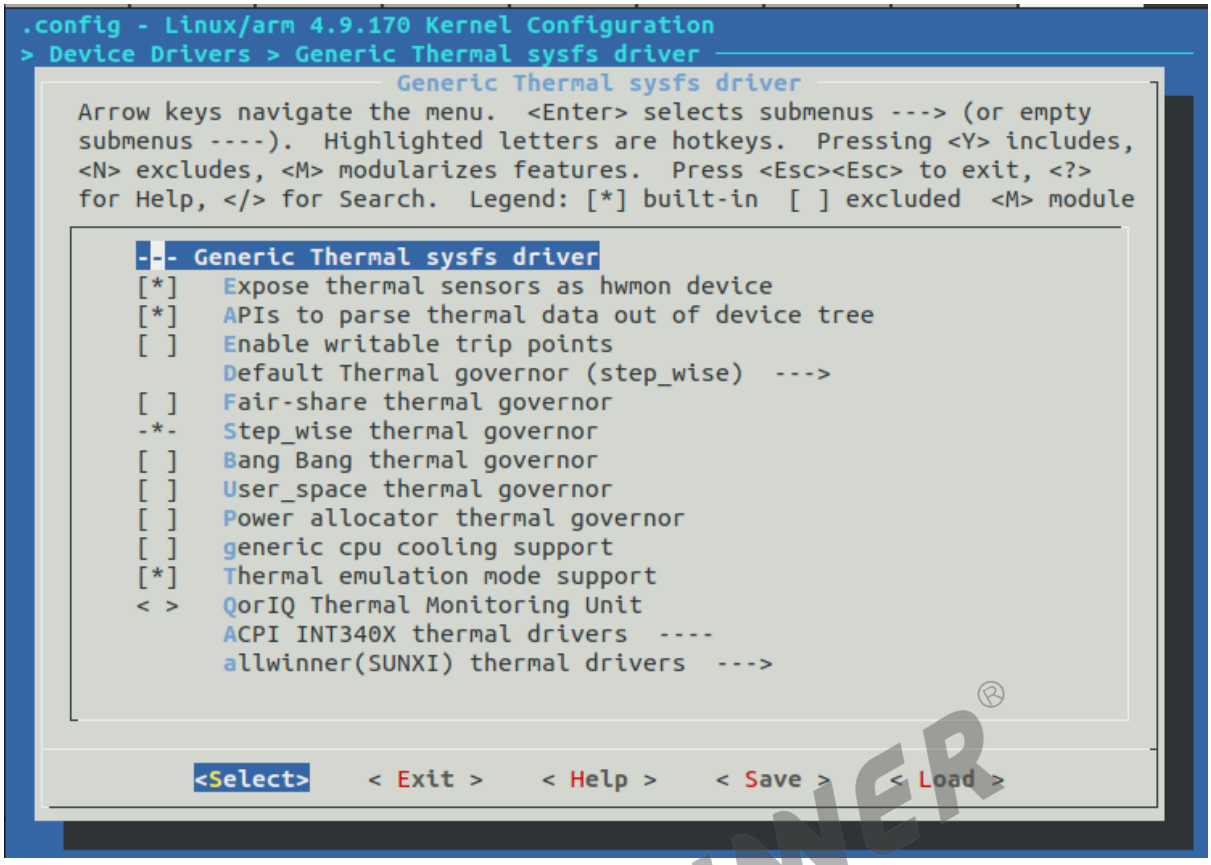

图 2-3: Thermal_menuconfig_R328_001.png

关键配置项：

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：

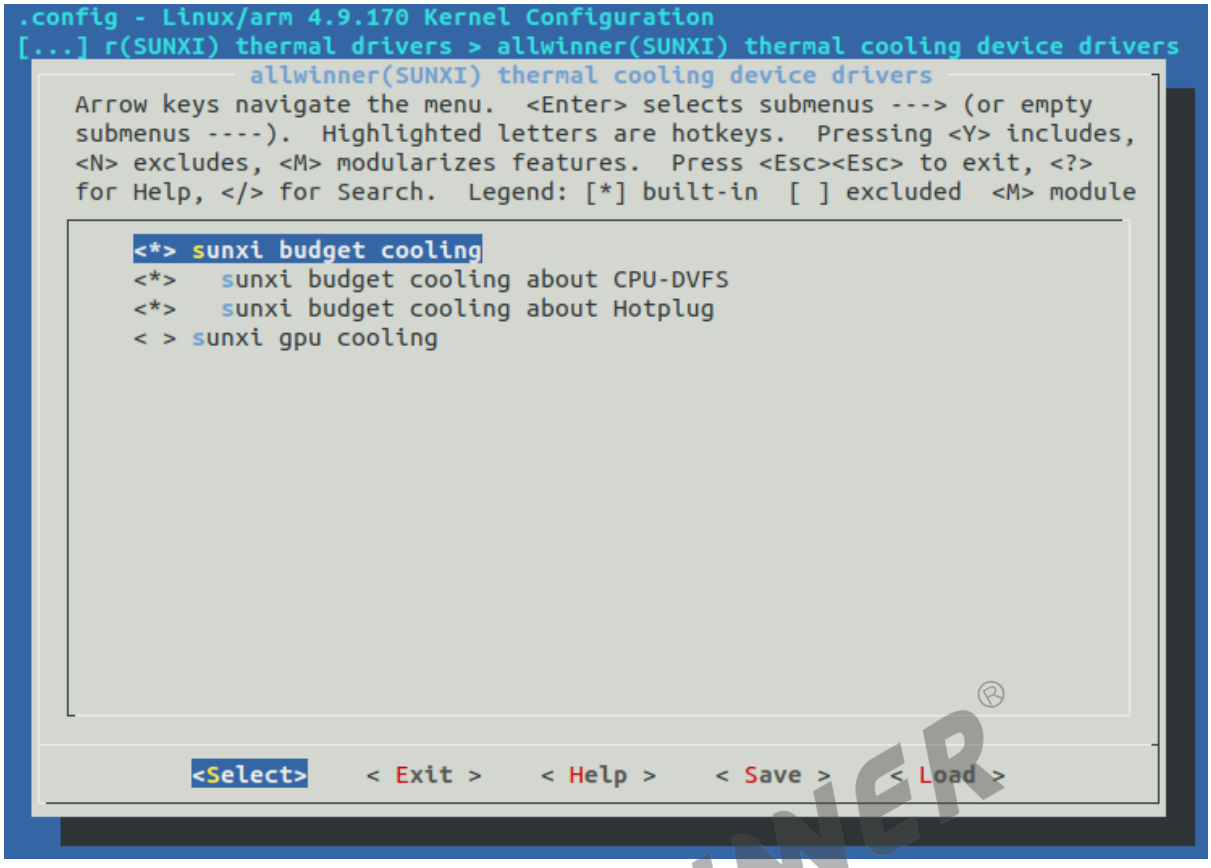

图 2-4: Thermal_menuconfig_R328_002

关键配置项：

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：

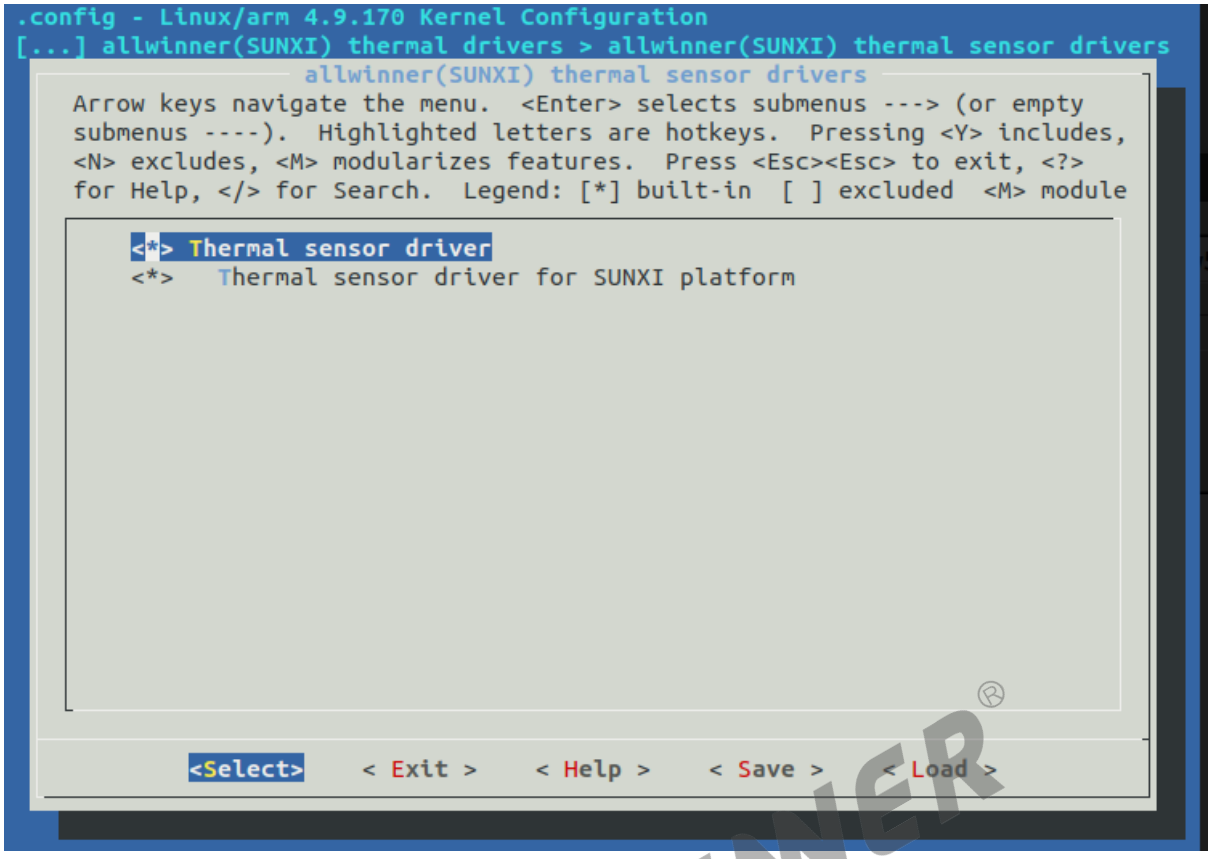

图 2-5: Thermal_menuconfig_R328_003

关键配置项：

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

2.3.1.3 Linux 4.9 配置

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

进入 tina 源码目录，执行 make kernel_menuconfig 进入配置主界面：

进入到Device Drivers –>Generic Thermal sysfs driver，参考配置项：

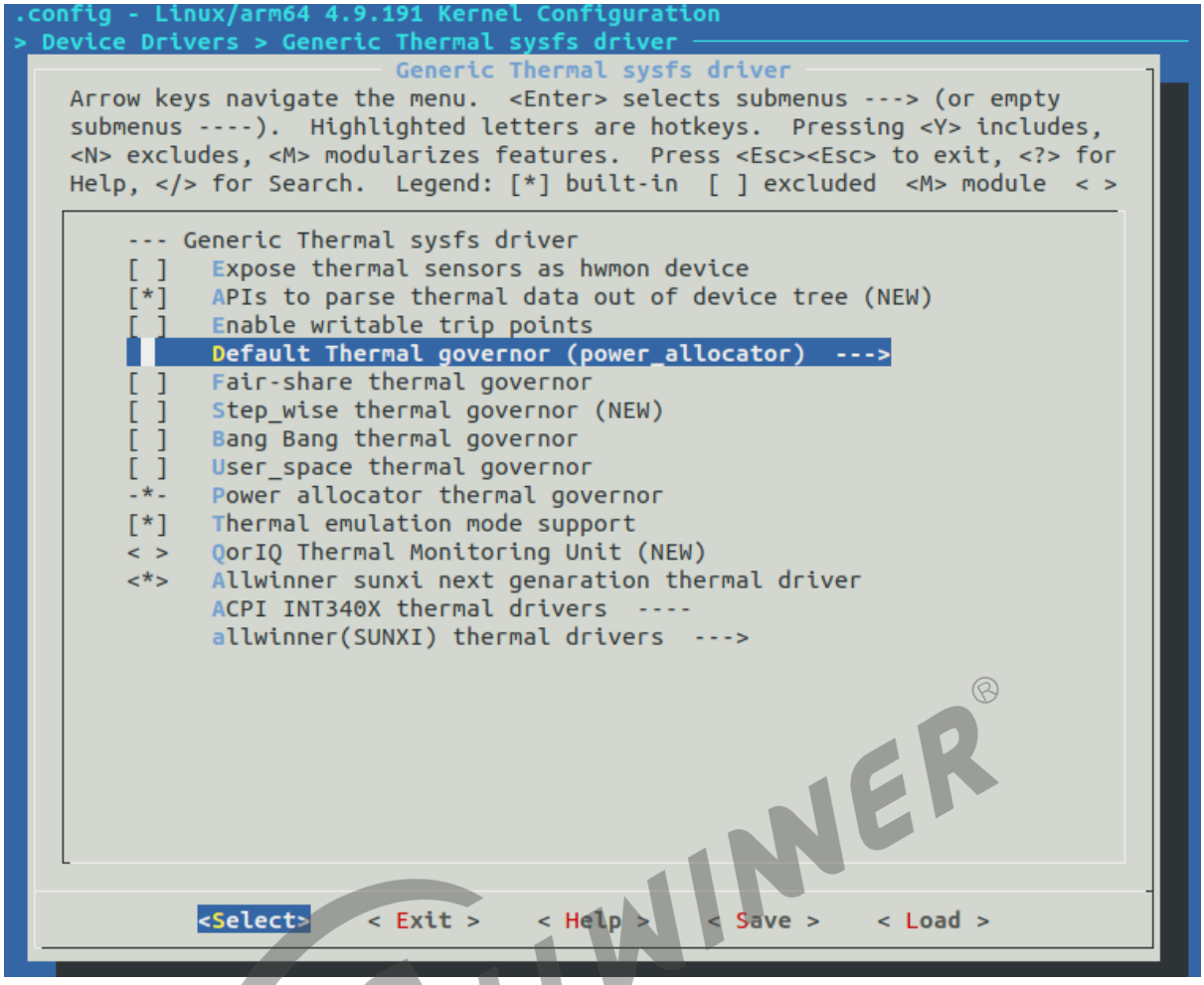

图 2-6: Thermal_menuconfig_R818

关键配置项：

APIs to parse thermal data out of device tree：使能解析设备树外的配置项功能
Default Thermal governor (power_allocator)：默认使用的thermal策略
Step_wise thermal governor：step-wise温控策略
Power allocator thermal governor：IPA温控策略
Allwinner sunxi next genaration thermal driver：Tina4.0 SDK中用于使能allwinner thermal driver，tina5.0不需要在此处选中

2.3.1.4 Linux 5.4 配置

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。

进入 tina 源码目录，执行 make kernel_menuconfig 进入配置主界面:

进入到Device Drivers –>Generic Thermal sysfs driver，参考配置：

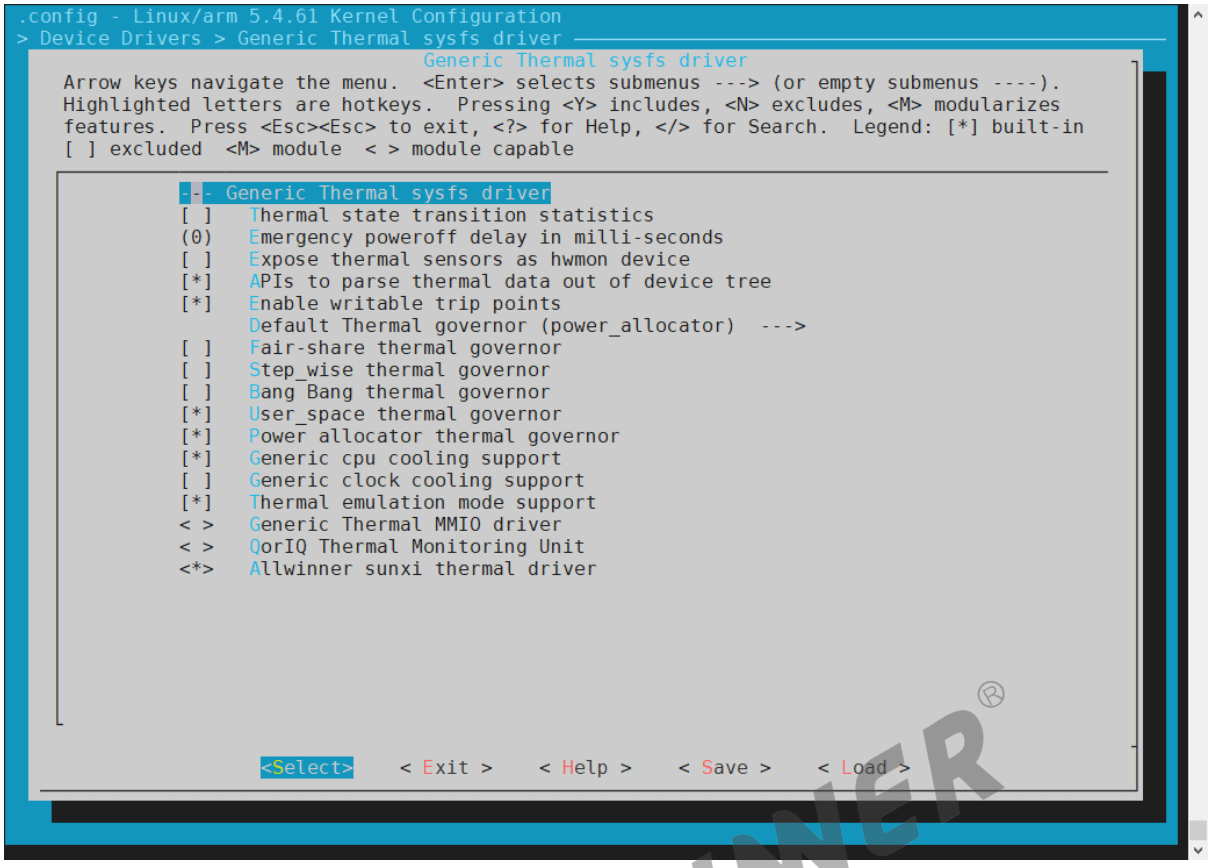

图 2-7: Thermal_menuconfig_R528

关键配置项：

```
[*] APIs to parse thermal data out of device tree： 使能解析设备树外的配置项功能
  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温控策略，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温控策略，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 # Tina4.0 SDK中用于使能allwinner thermal driver，tina5.0不需要在此处选中
```

2.3.1.5 Linux 5.15 配置

主要对应使用 Linux 内核版本为 5.15 ，且 DTS 描述中有"allwinner,sun**-ths 设备的平台。

进入 tina 源码目录，执行 make kernel_menuconfig 进入配置主界面：

进入到Device Drivers > Thermal drivers，参考配置项：

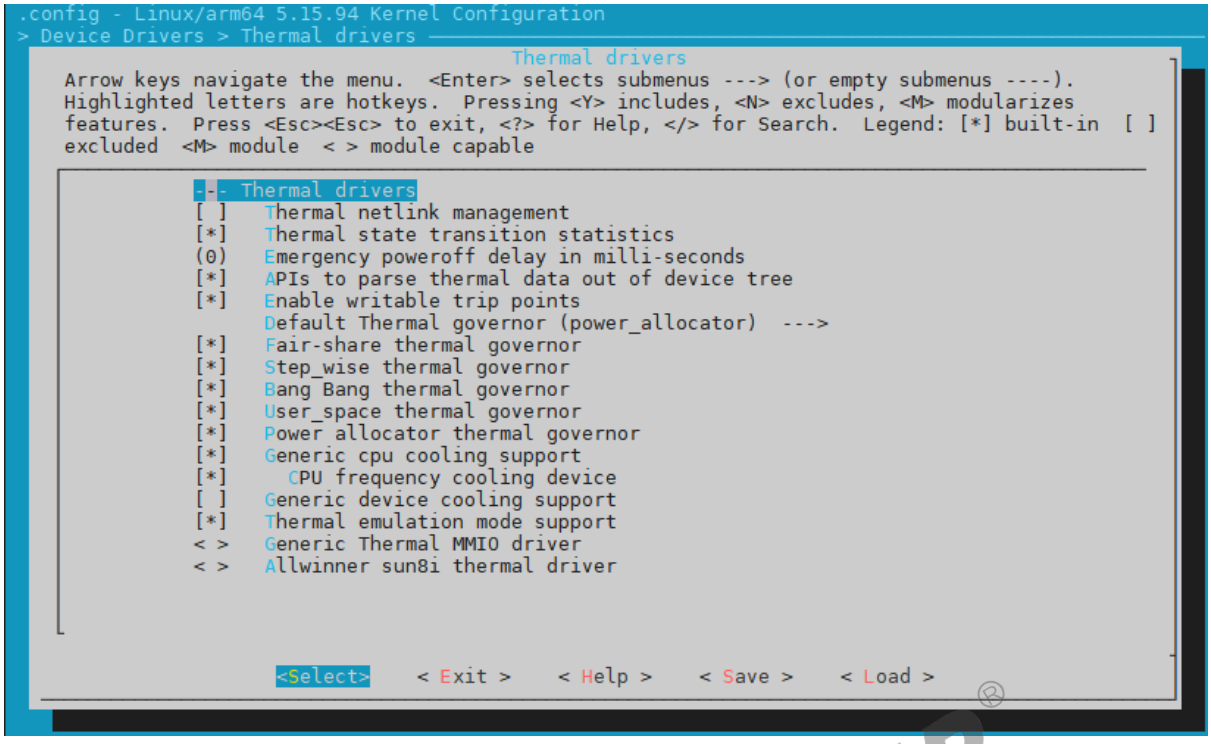

图 2-8: Thermal_menuconfig_AI985

关键配置项：

```
[*] APIs to parse thermal data out of device tree：使能解析设备树外的配置项功能
    Default Thermal governor (power_allocator) # 默认温控策略
[*] Step_wise thermal governor          # step_wise策略
[*] Power allocator thermal governor    # IPA策略
[*] Generic cpu cooling support         # 使能cpu降频
[*]  CPU frequency cooling device       # 使能cooling设备
[*] Thermal emulation mode support      # 使能模拟温度功能，可用于调试
```

2.3.2 step-wise 策略下的 Device tree 配置说明

2.3.2.1 Linux 3.4 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"
;--------------------------------
```

2.3.2.2 Linux 4.x_old 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，各模块配置已逐渐趋向于内核参考配置，但实现中仍采用了很多 Allwinner 特有的配置。

```
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
    };
};

thermal_sensor下的子节点（ths_combine0）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。

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，同时修改父节点thermal_sensor 的 combine_num 数值即可。
```

Linux 4.x-old thermal 由平台实现了一个 sunxi_cooling_device 模块，该模块是 cpu，gpu 等最高运行频率属性的一个抽象，用于设置 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。此外，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。

该模块的代码路径：drivers/thermal/sunxi_thermal/sunxi_cooling_device/

其功能主要分为两个部分：

1、对 cpu 属性的抽象，并将其注册成一个 Thermal Cooling Device。用于降低 cpu 的产热量，例如限制 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

2、对 gpu 属性的抽象，并将其注册成一个 Thermal Cooling Device。用于降低 gpu 的产热量，例如限制 gpu 运行频率。关键源码：sunxi_gpu_cooling.c

根据 sunxi_cooling_device 驱动的功能划分，dts 定义了两个虚拟设备节点以描述 cpu，gpu 降温属性，分别是 cpu_budget_cooling，gpu_cooling，需要在 dts 中分别配置：

- 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>;
};
```

除平台特定配置项外，还需完成 Linux thermal 框架的 of-thermal 模块配置，其配置示例如下。

```
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
        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
        bind4{
          contribution = <0>;
          trip = <&cpu_trip4>;
          cooling-device
          = <&cpu_budget_cooling 6 6>;
        };
      };
    };
};
```

2.3.2.3 Linux 4.x 以及 Linux 5.x 配置

使用 Linux thermal 的标准框架，配置示例如下：

- sensor device

```
ths: thermal_sensor{
  compatible = "allwinner,sun50iw10p1-ths";
  reg = <0x0 0x05070400 0x0 0x400>;
  clocks = <&clk_ths>;
  clock-names = "bus";
  #thermal-sensor-cells = <1>;
};
```

- of-thermal

```
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_trip0:t3{ # 最后一个 trip 点，critical 类型，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，并与cooling-veice映射
            contribution = <0>;
            trip = <&cpu_trip0>;
            cooling-device = <&cpu0 1 1>; #温度控制范围是1-1，具体对应到降频措施上，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，此处表示触发该trip时，频率可在序号1 ~ 1中选择
        };
        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>;
  };
};
```

2.3.3 Power allocator 策略下的 Device tree 配置说明

2.3.3.1 Linux 4.x 以及 Linux 5.x 配置

该策略在早期平台未支持，使用该策略的平台已适配 Linux thermal 的标准框架，配置示例：

- sensor device

```
ths: thermal_sensor{
  compatible = "allwinner,sun50iw10p1-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";
  #thermal-sensor-cells = <1>;
};
```

- of-thermal

```
thermal-zones {
  cpu_thermal_zone {
    polling-delay-passive = <500>;
    polling-delay = <1000>;
    thermal-sensors = <&ths 0>; # 配置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，由厂家测试给出，不建议更改
    k_pu = <48>;
    k_i = <0>;

    cpu_trips: trips {
      cpu_threshold: trip-point@0 {
        temperature = <70000>; # IPA的触发温度，超过此温度时，IPA算法可开始工作，有可能通过限制手段限制温度涨
      幅；
        type = "passive";
        hysteresis = <0>;
      };
      cpu_target: trip-point@1 {
        temperature = <80000>; # IPA的目标温度，超过此温度时，IPA会通过限制手段控制温度向目标温度靠近，例如通过
      降频来降低温度。
        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>; # cpu 分配功率权重
      };
    };
  };
  gpu_thermal_zone{
    polling-delay-passive = <500>;
    polling-delay = <1000>;
    thermal-sensors = <&ths 1>;# 配置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>;# 配置2号传感器对应的Thermal Zone Device为ddr_thermal_zone设备
    };
};
```

2.3.4 源码结构

thermal 源码位于内核根目录/drivers/thermal

```
drivers/thermal
├── cpu_cooling.c
├── of-thermal.c
├── power_allocator.c
├── step_wise.c
├── thermal_core.c
├── thermal_helpers.c
├── thermal_sysfs.c
└── user_space.c
```

2.3.5 温控策略介绍

2.3.5.1 step-wise

step-wise 是一种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：

```
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
a. 如果趋势是 THERMAL_TREND_RAISING，则使用这个 trip 对应的更高的降温状态；
b. 如果趋势是 THERMAL_TREND_DROPPING，则不做任何事情；
c. 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；
d. 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；
e. 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
a. 如果趋势是 THERMAL_TREND_RAISING，不做任何事；

b. 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；
c. 如果趋势是 THERMAL_TREND_RAISE_FULL，则不采取任何措施；
d. 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；

温度变化趋势通过比较这次测量值与上次测量值间确定，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。此外，THERMAL_TREND_RAISE_FULL，THERMAL_TREND_DROP_FULL 这两种趋势状态，在 Allwinner 平台未使用。

2.3.5.2 power allocator

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。

该策略是 linux 内核基于 PID 算法实现的一种温控策略，该策略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，调控结果是将设备维持在一个目标温度附近。

调温过程概述：

1、当设备负载高，温度上升达到设定的触发温度时，IPA 开始生效。但生效并不意味着会立即降频，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附近时，IPA 将根据温度变化情况调整频率，通过对历史温度数据的比例，积分，微分运算，来估计温度变化趋势，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，将设备温度维持在目标温度附近。

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。下降到低于触发温度时，IPA 就会关闭。

调节原理概述：

目前 allwinner 平台的 IPA 对象仅针对 CPU，理想环境下，设系统 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，则系统温度将保持为 T 不变。若此时 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，当再次平衡时，系统温度 Ta 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 Tb 小于 T。

IPA 机制通过一系列的 PID 闭环控制，使系统的温度尽量保持在 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 Po，此时调频模块将 CPU 频率升高，使 CPU 消耗的功耗为 Po，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 Po，调频模块将 CPU 频率降低，使 CPU 消耗的功耗为 Po，系统温度降低。

T 即 IPA 的目标温度，温控机制致力于将系统温度控制在目标温度附近，但并不能保证系统温度一直为 T，主要原因是：

（1）当系统负载很高时，IPA 计算输出的功耗值和实际控制 CPU 消耗的功耗不完全一致，且存在环境扰动；

（2）当系统负载低时，即使 CPU 以最高频率运行，也可能达不到 IPA 计算的期望功耗值 Po，这会导致系统温度下降。此外，如果负载持续为低，CPU 将一直保持最高频率运行。因此，一般会设置一个触发温度（触发温度必须小于 T，两者通常相差 10 摄氏度）,只有当系统温度超过触发温度时，才让温控策略生效。

IPA 的使用需要配置计算参数，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。这些参数主要在使用 IPA 策略的平台上测试与计算得到，默认配有一版参数，通常无需改动。

3 调试方法

3.1 调试节点

thermal core会在/sys/devices/virtual/thermal目录下创建调试节点，主要有thermal_zone和cooling_device两部分节点。

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

3.2 常见操作

当 dts 配置了多个 thermal zone 时，会相应产生多个/sys/devices/virtual/thermal/thermal_zone*目录，说明常见操作时将以 thermal_zone0 为例。

3.2.1 查看温度域温度

1、确认 thermal_zone0 的类型：

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

表示thermal_zone0的控制的是cpu温度
```

2、查看 thermal_zone0 的温度：

```
// 通常单位为千分之一摄氏度
#cat /sys/class/thermal/thermal_zone0/temp
24522

// 若仅输出两、三位数值，单位为摄氏度
#cat /sys/class/thermal/thermal_zone0/temp
36
```

3.2.2 使能、失能温控功能

```
（1）失能thermal_zone0的温控策略（开：enabled，关：disabled）：
#echo disabled > /sys/class/thermal/thermal_zone0/mode

（2）解除对cooling device的限制：
#echo 0 > /sys/class/thermal/thermal_zone0/cdev*/cur_state
```

3.2.3 修改过温关机触发温度

过温关机温度默认是 dts 中，of-thermal 模块配置的 critical 类型 trip，可直接修改 dts 配置

```
dts：
        crt_trip:t2{
           temperature = <110000>; # 达到110摄氏度时表示系统过温，触发保护动作
           type = "critical";
           hysteresis = <0>;
        };

或通过调试节点临时改写过温温度：
(1) 找到类型为critical的trip:
# cat /sys/devices/virtual/thermal/thermal_zone0/trip_point_2_type
critical
```

```
(2) 调高过温温度为150摄氏度：
echo 150000 > /sys/devices/virtual/thermal/thermal_zone0/trip_point_2_temp
```

3.2.4 设置一个模拟温度

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：

```
// 设置系统始终读到thermal_zone0的温度为80摄氏度
#echo 80000 > /sys/class/thermal/thermal_zone0/emul_temp
```

2、关闭 thermal_zone0 的模拟温度功能：

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

3.2.5 修改温度策略触发温度

修改 dts thermal_zone 节点下 trip 子节点的 temperature 配置值，详见“模块配置” 章节对 dts 配置项的描述

```
cpu_trip1:t1{
        temperature = <100000>; # 修改此处
        type = "passive";
        hysteresis = <0>;
    };
```

4 FAQ

4.1 power allocator 策略下为什么温度从高于目标温度降回低于目标温度一段时间后，系统性能还是被限制

IPA 策略使用了 PID 控制算法，对当前可分配能量的计算公式为：Power = sustainable-power + P + I + D，其中 “I” 作为累计误差只考虑了温度超过目标温度时候取值为负数的误差，且 I 值只有在温度低于策略启用温度时才会被清零。

温度长时间超过目标温度时，会导致 “I” 一直累积，当 “I” 累积值大于 sustainable-power + P 时，可分配的能量将一直为 0，受控的 cooling-device 将一直处于最低频率状态，直到温度低于策略启用温度后将 PID 参数复位，被限制的现象才会消失。

此类问题的优化办法是针对使用场景重新调整 IPA 参数，或想办法不让系统温度长时间超过目标温度。

4.2 如何选择温控策略

全志平台主要支持 IPA 温控策略和 step-wise 温控策略。

（1）IPA 温控策略：对温度的控制能力较强，但不能保证高温下的系统性能，在高温场景下温度较难过温，但系统性能是不确定的。

（2）step-wise 温控策略：对性能的控制能力较强，但不能保证不超温度，在高温场景下系统性能确定，但温度是不确定的。

若有明确的温度限制需求，建议选择 IPA 温控策略。若有明确的高温性能限制要求，建议选择 step-wise 温控策略。

4.3 FAQ 列表

更多 FAQ 请登录 Allwinner 一号通平台，搜索 “温控”、“thermal”、“温度” 等关键词查找。

典型 FAQ：

（1）【FAQ938】高温场景下出现卡顿、掉帧等性能不足

（2）【FAQ1894】SOC 的 thermal sensor 模块数据采样异常，温度显示为负值

著作权声明

商标声明

免责声明