TDA4x 和 AM6x 的动态温度检测与热管理

李彪 ***Central FAE***

摘要

Jacinto™ 7 TDA4 系列和 Sitara™ AM6x 处理器是 TI 公司基于 Keystone 架构推出的最新一代处理器， 基于片上多核异构架构进行集成不同的应用核，将不同的任务分配给对应的核进行并行处理，最大限度的发挥 TI 处理器的优势。处理器的能力越强往往伴随着越高的发热，过高的温度甚至超过芯片的热设计点并持续工作会造成不可逆转的损坏，如何充分发挥 TI 处理器芯片的强大性能，并且不影响芯片寿命，这意味着需要对处理器进行热管理，TI 所有的处理器芯片都集成了片上的温度传感器，并且集成专门的模块 VTM (Voltage Thermal Management) 对芯片进行热管理，片上温度传感器的基本工作原理是利用半导体材料的物理特性，在温度变化时，其电阻或电势会发生变化，从而导致输出的电信号发生变化，这个变化由微控制器或其他电路读取和处理，从而实现温度的测量。VTM 模块同时也负责芯片的电压管理等功能，本应用手册主要关注其温度管理功能。

本应用手册将介绍关于 TI 处理器片上集成的 VTM 的功能及其工作原理和如何使用以及结合实际芯片中默认使能的通过 VTM 模块实现的软硬件保护的方案。涉及到软件代码以及具体应用实例时以 AM62A 处理器为例进行描述，对应代码更改及命令可以对应应用到所有 TDA4x 和 AM6x 系列处理器上，可能存在由于 SDK 的版本更新需要更改对应路径。

目录

图

表

1. VTM 模块

VTM (Voltage Thermal management) 模块提供与集成温度传感器和用户编程的热事件相关的控制、状态以及中断和事件生成功能。在芯片热管理中我们主要关注其温度测量以及发出预警和中断的功能。下图是 VTM 模块在芯片内部的连接图，由于 VTM 可能会直接发起 reset 整个芯片的指令，因此它也和整个芯片的 reset controller 有直接的连接。同时 VTM 还包含一组内存映射寄存器，用于存储在设备测试期间设置的设备特定工作电压 (AVSVNOM)。这些值可用于系统 AVS 软件调整设备的工作电压，以实现最佳工作状态（功耗和性能平衡）。

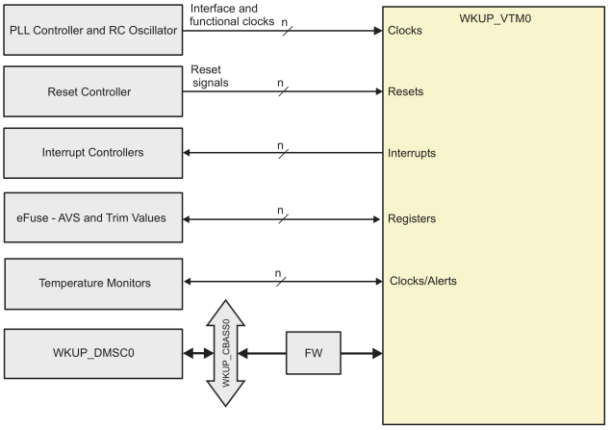

图 1 VTM 系统连接图

TI 整个 SOC 家族中有众多的 PN 系列 SoC 芯片，其中 VTM 的系统连接方式都是类似的。VTM 一般也都是位于芯片的唤醒域（WKUP），该域在芯片启动时是最先开始工作的，温度管理在芯片上电起就立即介入工作，以保证整个芯片的正常运行和寿命。

1.1 VTM 模块简介

在 TI SOC 上 VTM 在芯片内部典型的布局如图 2 所示，其中 Temperature Monitor 即片上温度传感器会被放置在靠近发热点的地方， VTM 模块通过内部连接控制芯片内的温度监视器。单个 VTM 可通过其寄存器控制最多 8 个温度传感器。由于温度传感器不会自发的周期性更新，因此 VTM 会定期启用温度传感器，来持续的更新报告的温度数据。温度传感器返回的温度值会由 VTM 寄存器捕获并存在 VTM 内部对应的寄存器中。VTM 在未启用传感器时会将传感器保持复位状态，以节省功耗并减少传感器使用率，从而最大限度地延长传感器寿命。

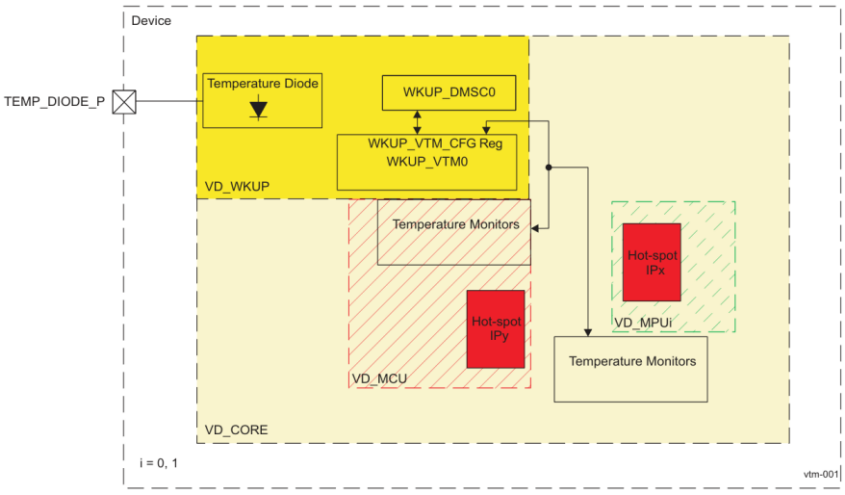

图 2 VTM 芯片内布局

不同的芯片根据需要一般会放置不同数量的片上温度传感器，放置的原则是尽量靠近发热点，并且涵盖所有的热点，表 1 对所有 TI 的处理器产品包含的片上温度传感器和位置进行了统计，客户可以根据使用产品进行查阅。下表根据物理位置靠近的热点为分类统计，由于传感器大部分可能位于两个热源之间，以及不同的电源域边界，无法很好的界定，下表仅可作为传感器位置的大致参考，详细请参阅对应器件的 TRM（Technical Reference Manual）。

表 1 片上温度传感器表

Onchip Sensor	Total num	A72/A53	DDR controller	C7x	R5F	GPU	CODEC	DPHYs
TDA4VH	7	√	√	√	√	√	√	√
TDA4VE/VL/AL	7	√	√	√	√	√	√	√
TDA4VM	5	√	√	√	√	√	-	-
TDA4VEN	3	√	√	-	-	√	-	-
DRA821	3	√	√	-	√	-	-	-
AM62A	3	√	√	√	-	-	-	-
AM62P	3	√	√	-	-	√	-	-
AM62x	2	√	√	-	-	-	-	-
AM64/AM24	2	√	√	-	-	-	-	-
AM62L	1	√	-	-	-	-	-	-

所有芯片都会在 Arm 大核附近放置温度传感器，由于这一般是整个芯片最热点。由于 DDR 负责整个芯片的数据吞吐，并且一般都是高速运行，过热的风险也较大，一般也需要在 DDR 控制器处放置传感器进行监控。

1.2 VTM 工作原理及使用

上文提到每个 VTM 模块最多控制 8 个温度传感器，并且会将温度传感器返回的温度值会由 VTM 寄存器捕获并存在 VTM 内部对应的寄存器中。该寄存器名称为 WKUP_VTM_TMPSENS_STAT_j[9-0] DATA_OUT，将温度值以 10bit 二进制数储存在对应寄存器，下表列出了几个典型实际温度值以及其对应的 10bit 温度值。

表 2 片上温度传感器寄存器值和实际值转换表

实际温度	-40	-25	0	25	50	75	100	105	125	150
10bit Dec	28	77	164	260	366	485	620	648	773	949
10bit Hex	01C	04D	0A4	104	16E	1E5	26C	288	305	3B5

上表展示的是一种表查找方法，该表格可以方便的将你所读取的值与实际的温度值有一个大致的对应关系，该方式同样也适用于转换芯片中和温度相关的寄存器值与实际值。按表查找只能得到温度的大致区间，需要得到准确的温度值，需要使用以下公式进行计算：

$$y=6.0373*10^{-8}*x^3 - 1.7058*10^{-4} * x^2 + 0.32512x - 49.002 - 9.2627 *10^{-12} * x^4$$

将该公式使用 Python 脚本进行绘图后可以得到以下的图像，大致是一个单调上升的函数，并且可以在脚本中输入读取到的寄存器值，运行后会进行自动计算出对应的准确对应温度值并进行标注。脚本可以从此处下载。

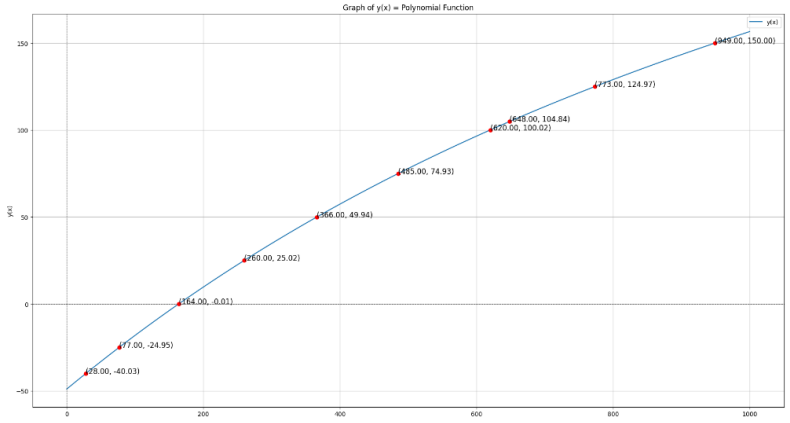

图 3 寄存器值与实际值计算公式图

能够准确计算对应的实际温度值可以让方便的对系统的温度保护逻辑进行修改，如修改芯片的最大温度保护阈值 WKUP_VTM_MISC_CTRL2[25-16] MAXT_OUTRG_ALERT_THR0 和 WKUP_VTM_MISC_CTRL2[9-0] MAXT_OUTRG_ALERT_THR 阈值中编码的温度值可以修改芯片温度保护的极限最大值。

在 TI 官方提供默认的 Linux 软件中也提供了对应的接口和命令读取片上温度传感器的值，所使用的命令都是类似的，以 AM62A 为例，在 EVM 上读取三个温度传感器的值可以使用以下命令，读取所有的温度传感器的值，单位是 milliCelsius，如下打印结果 sensor1，sensor2，sensor3 的温度值分别是 45.2℃, 38.7℃,38.9℃。由于传感器位置不同，所以读取的温度会有差异与均值间±5℃是正常的，该范围在对应器件的最新数据手册中进行定义。

```
root@am62axx-evm:/opt/edgeai-gst-apps# cat /sys/class/thermal/thermal_zone*/temp
43209
38749
38983
```

2. TI 处理器芯片的硬件温度保护

前文主要介绍了 TI 处理器芯片的负责温度的硬件管理模块，在 VTM 中可以对片上的温度传感器的值进行读取，得到温度之后，VTM 模块会根据读取到的温度进行整个芯片的硬件的温度保护。

2.1. VTM 的过温保护阈值

VTM 的每个温度监测寄存器组均可配置为对其对应的温度监测器进行温度采样，并触发最多 3 个警报（级别）信号，分别是：GT_TH1_ALERT（过温报警比较器 1 结果）- 由 10 位温度阈值点 1 (THPT1) 生成。该值可以设置于 VTM_TMPSENSx_TH[25-16] TH1_VAL。GT_TH2_ALERT（过温报警比较器 2 结果）- 由阈值点 2 (THPT2) 生成的 10 位增量点。该值可以设置于 VTM_TMPSENSx_TH2[9-0] TH2_VAL。LT_TH0_ALERT（欠温报警比较器结果）- 由阈值点 0 (THPT0) 生成的 10 位增量点。该值可以设置于 VTM_TMPSENSx_TH[9-0] TH0_VAL。

一般来说需要遵守 THPT2 > THPT1 > THPT0 的原则来设置，其工作原理如下：

TH1 配置为早期警报，以指示温度高于 VTM_VTM_TMPSENS_TH_j[25-16] TH1_VAL 中定义的阈值。TH2 配置为警告，表示温度高于 VTM_VTM_TMPSENS_TH2_j[9-0] TH2_VAL 中定义的阈值。其概念是，TH1 表示处理器正在变热；TH2 需要立即引起注意。TH0 配置为当传感器检测到的温度低于 VTM_VTM_TMPSENS_TH_j[9-0] TH0_VAL 中定义的阈值时触发。此中断表示由于温度已降至原始 TH1 水平以下，因此可以放松热缓解措施或退出紧急状态。需要注意的是如果启用 LT_TH0_INT，则无论 TH1 中断和 TH2 中断是否已启用或曾经触发，当读取的温度低于 TH0 时，LT_TH0_INT 都会被触发。以上三个中断均是软件可以进行检测并根据该中断来设计芯片的散热方案。客户可以根据需要进行设计，以上为一般逻辑供参考。每个传感器都可以独立进行设置过温点或欠温点，只要有一个传感器触发警告，最后都会生成对应的中断，如下图所有的传感器最后是会生成三个中断对应 TH0，TH1，TH2 的欠温和过温中断，客户软件可以对该中断进行后续处理。需要额外注意的是仅当传感器处于连续模式时，中断才会激活。传感器的单次采样不会触发任何中断。

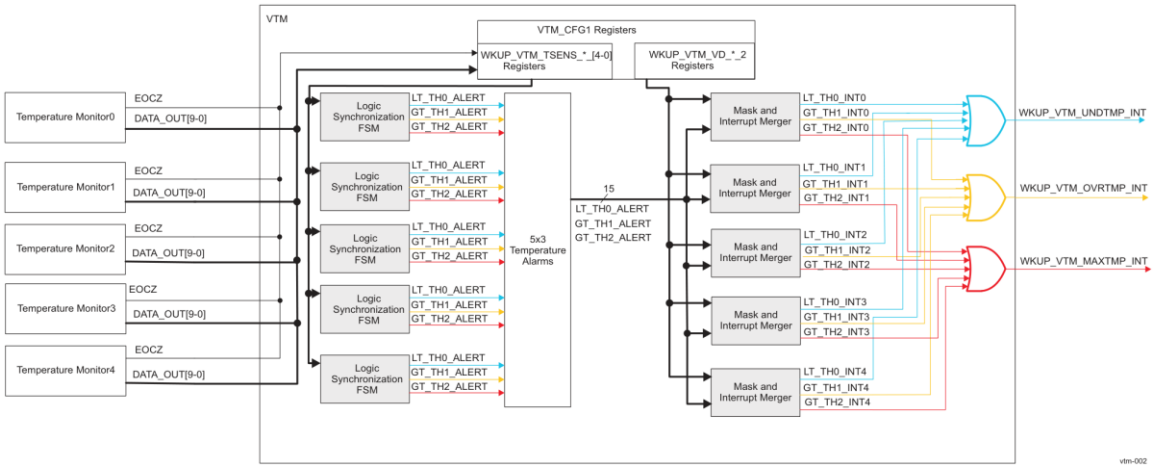

图 4 VTM 温度保护中断机制示意图

最终阈值为 MAXT_OUTRG_ALERT_THR。超过此阈值时，设备将被强制进入复位状态（并直接关闭 PLL）。当设备充分冷却（< MAXT_OUTRG_ALERT_THR0）时，内部复位将被释放，设备将重新启动。该温度保护是硬件保护，触发后硬件进行复位。客户可以修改对应的阈值和是否触发，无法使用该中断进行软件设计，因为触发后芯片会立即进入复位状态，任何软件都无法进行操作，并且该保护不受传感器工作模式限制。

2.2. 最高硬件温度保护

为了确保芯片不在过温的情况下进行工作，保证芯片的正常工作寿命，TI 的处理器芯片内部设置了硬件的最高温度保护机制，该机制不需要软件进行参与，默认情况下 TI 的处理器芯片都是使能了该保护，同时在这里介绍该保护内部的机制，TI 芯片触发 VTM 最高温度超出范围警报需要同时满足以下几个条件：

1. 需要获得有效的温度输入，要使 VTM 的最高温度保护的输出有效，则 VTM 必须配置为至少启用一个温度传感器，即 WKUP_VTM_TMPSENS_CTRL_j[11] MAXT_OUTRG_EN = 1；

2. 此外，WKUP_VTM_MISC_CTRL[0] ANYMAXT_OUTRG_ALERT_EN 必须设置为“1”。该位是用来控制 VTM 的最高温度保护的警告信号输出的开关；

3.在保证输入和输出通路情况下，即正确配置并启用一个温度传感器且使能最高温度保护输出，如果芯片温度超过 WKUP_VTM_MISC_CTRL2 [9-0] MAXT_OUTRG_ALERT_THR 中的编程值，则一旦传感器检测到编程的最高温度，VTM 输出将被驱动为“1”；

4. 在设备层面，VTM 输出 THERM_MAXTEMP_OUTRANGE_ALERT 将驱动 PLL 控制器进入热复位状态，同时 PLL 控制器进入 复位状态，所有 PLL 也同时进入时钟旁路模式；

5. 通过第 4 点所述的操作，设备将显著降低功耗，并在几秒钟内降低设备温度；

6. VTM 会持续读取温度，一旦检测到与 WKUP_VTM_MISC_CTRL2[25-16] MAXT_OUTRG_ALERT_THR0 中编程值对应的代码，VTM 就会驱动 THERM_MAXTEMP_OUTRANGE_ALERT = 0；

7. 满足第 6 项后， PLL 控制器使 SoC 退出复位状态，并重新启动启动序列。同时，PLL 退出旁路模式，并以其目标频率启动整个芯片并开始正常工作。

最高温度硬件保护流程图如下图所示：

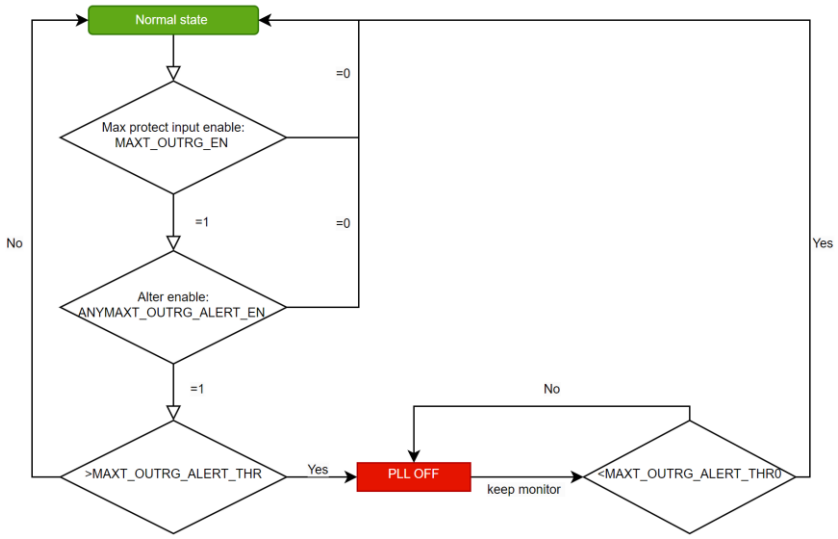

图 5 VTM 最高温度硬件保护示意图

在 TI 所有系列的芯片中，该最大温度保护都是默认使能的，由于芯片在超过最大温度运行是会永久损害芯片寿命，该影响是不可逆转的。不同芯片可以正常工作的温度范围可以在对应器件的数据手册中进行查找。在触发最高温度保护的过程中，会将芯片的时钟 PLL 进入旁路模式，即将整个芯片的时钟信号关闭，此时 VTM 会使用来自 Sensor 的

下面以 AM62A 为例，我们可以通过读取寄存器的方式得到目前芯片设置的最大温度保护值，具体的寄存器地址可以翻阅 TRM 得到。其中 WKUP_VTM_MISC_CTRL2[25-16] MAXT_OUTRG_ALERT_THR0 的编码值为 0x288，使用 python 脚本可以计算出该值对应实际温度为 105℃，WKUP_VTM_MISC_CTRL2 [9-0] MAXT_OUTRG_ALERT_THR 的编码值为 0x2F8，对应的实际温度值为 123℃，因此默认的 AM62A 的最大硬件温度保护值为 123℃，并且只有在温度降低到 105℃以下后才会再次启动。

```
root@am62axx-evm:/opt/edgeai-gst-apps# devmem2 0x00b01010
/dev/mem opened.
Memory mapped at address 0xffff8488e000.
Read at address  0x00B01010 (0xffff8488e010): 0x028802F8
```

3. 软件温度保护策略

VTM 模块是芯片内部的硬件模块，主要任务是负责管理整个芯片的温度，虽然 VTM 模块包含最高温度的硬件保护功能，以及可以灵活设置的温度保护阈值，以及过温保护中断，供使用者进行灵活的设计。但硬件上的温度保护是有局限性的，除最高温度保护直接将芯片时钟停止外，硬件模块无法直接控制芯片进行其他的如降频等热保护措施。并且由于 TI 处理器芯片都是多核异构的架构，并且每个型号处理器内部包含的温度传感器的数量以及其适宜的工作温度不同，因此默认所有的 VTM 温度保护阈值设置是 0，即默认没有使能除最高温度保护外的其他温度保护中断，该部分需要使用者根据系统需要，自行进行设计。因此目前 VTM 模块是缺少除停止工作外的其他任何温度保护策略的。作为补充，TI 增加了软件层面的温度保护策略，软件通过 VTM 模块获得温度值后，自行生成对应的中断或标志位，让整个系统进入如降低频率，停止高负荷工作降低系统负载等措施，并且可以灵活的针对不同的温度进行不同的设计，这个对于系统的功能安全是十分重要的。

3.1. 可选的软件温度保护措施

TI 处理器可以提供对应器件的功耗估算电子表格，该表格是基于测量和仿真数据提供功耗估算，虽然芯片的运行功耗取决于电气参数、硅片工艺变化、环境条件以及处理器运行时的用例等很多因素。但也可以作为芯片功耗的一个大致参考，因此对于软件可以采取的措施如降频以及降低处理器负载率方式对功耗的影响，可以使用该工具作为一个大致参考。芯片的温度和功耗的大小是正相关，功耗越大，其温度上升越快，因此软件温度管理的主要目标是降低芯片的功耗。

下表以 AM62A 处理器为例，对应的工具可以由此处下载。该方式也可以适用其他的 TI 处理器。表 3 结果均在结温 125℃以及所有核的频率均为最高情况下进行统计。

表 3 AM62A 不同工作负载下的功耗

Case	Power/mW	A53	DDR	C7x	R5F	DM_R5	Δ Power/mW
Normal status	5136	80%	80%	80%	80%	80%	0
A53 loading down	4936	20%	80%	80%	80%	80%	200
DDR loading down	4689	80%	20%	80%	80%	80%	447
C7x loading down	4181	80%	80%	20%	80%	80%	955
R5F loading down	5113	80%	80%	80%	20%	80%	23
DM loading down	5109	80%	80%	80%	80%	20%	27
All loading down	3127	20%	20%	20%	20%	20%	2009

表 4 结果均在结温 125℃以及所有核的负载均为 80%情况下进行统计。

表 4 AM62A 不同工作频率下的功耗

Case	Power/mW	A53	DDR	C7x	R5F	DM_R5	Δ Power/mW
Normal status	5136	1400MHz	3200MHz	1000MHz	800MHz	800MHz	0
A53 Freq down	4837	700MHz	3200MHz	1000MHz	800MHz	800MHz	299
DDR Freq down	5091	1400MHz	1600MHz	1000MHz	800MHz	800MHz	45
C7x Freq down	4473	1400MHz	3200MHz	500MHz	800MHz	800MHz	663
R5F Freq down	5109	1400MHz	3200MHz	1000MHz	400MHz	800MHz	27
DM Freq down	5084	1400MHz	3200MHz	1000MHz	800MHz	400MHz	52
All Freq down	4220	700MHz	1600MHz	500MHz	400MHz	400MHz	916

从以上的计算结果，可以看到在 AM62A 中对功耗影响最大的核是 C7x，因此在进行系统的温度控制方案设计时，在达到设定温度后应考虑首先降低 C7x 的频率和负载。

3.2. Linux 温度保护逻辑

从 SDK11.0（或更早）开始，Linux SDK 开始加入 Linux VTM 驱动，Linux 内核 VTM 和 芯片的硬件 VTM 是两个互不相关的概念。内核 VTM 框架使用设备树中的配置（k3-am62-thermal.dtsi，由内核热绑定文档定义）来监控传感器温度，并采取相应的措施，例如降低 CPU 频率、关闭或重启 Linux。Linux 所使用的传感器温度由 AM62x 硬件 VTM 模块获取。

内核设备树 k3-am62a-thermal.dtsi 中的定义/配置只是一个示例，客户可以根据项目需求进行修改，例如，关闭或重启 Linux。使用内核设备树 k3-am62a-thermal.dtsi 中的默认设置，当温度达到 105C 时，内核将触发关机序列。Linux SDK11.0 为例，Linux 的 VTM 驱动程序也定义了类似硬件 VTM 的中断阈值，可以编程设置最多 3 个阈值温度，其中 2 个用于大于阈值，1 个用于小于阈值，以便 VTM 能够提醒内核采取行动。例如，当超过第一个阈值时，可以提醒内核开始降低 CPU 的电压和时钟速度，从而使 SoC 的整体温度稳定下来。如果 SoC 温度持续升高，我们可以使用第二个阈值采取更积极的措施。例如，我们可以在超过第二个阈值后发出 poweroff 命令，完全关闭设备。阈值温度可以在内核中使用我们在设备树中定义的值进行设置。这可以用于：设置“临界”温度，内核将在此温度下关闭 。SoC 在过热时向内核发出“被动”警报，并且可以通过将 cpufreq 驱动程序注册为冷却设备来降低 MPU 频率。 通过修改下列代码可以更改软件热保护的阈值，如将关机温度由 105℃更改到 125℃，对应将软件保护从工规的关机温度切换到车规温度。以下代码表示临界温度为 55℃，5℃的滞后控制，在达到 55℃时，进入“被动”状态启动散热措施，在达到 125℃（2℃的滞后控制）时立即保护并关机。

```
/* From arch/arm64/boot/dts/ti/k3-am62a-thermal.dtsi */
thermal_zones: thermal-zones {
        main0_thermal: main0-thermal {
                polling-delay-passive = <250>;          /* milliSeconds */
                polling-delay = <500>;                  /* milliSeconds */
                thermal-sensors = <&wkup_vtm0 0>;
                trips {
                        main0_alert: main0-alert {
                                temperature = <95000>;
                                hysteresis = <2000>;
                                type = "passive";
                        };
                        main0_crit: main0-crit {
                                temperature = <105000>;     /* milliCelsius */
                                hysteresis = <2000>;        /* milliCelsius */
                                type = "critical";
                        };
                };
                cooling-maps {
                        map0 {
                                trip = <&main0_alert>;
                                cooling-device =
                                        <&cpu0 THERMAL_NO_LIMIT THERMAL_NO_LIMIT>,
                                        <&cpu1 THERMAL_NO_LIMIT THERMAL_NO_LIMIT>,
                                        <&cpu2 THERMAL_NO_LIMIT THERMAL_NO_LIMIT>,
                                        <&cpu3 THERMAL_NO_LIMIT THERMAL_NO_LIMIT>;
                        };
                };
        };
```

内核在进入“被动”状态时，可以采用杀掉优先级低的进程的方式来降低对应核的负载，对于一些关键的任务，则可以采用降低运行频率的方式来降低功耗，动态频率选择 DFS（Dynamic Frequency Selection）就是很好的根据需要降低 CPU 频率的方式。DFS 详情可以参阅该教程。

DFS 的方式目前可以方便的改变 A 核的频率，下文提供更改其他核的频率的方式。首先从芯片的 TRM（时钟部分）中识别与该核关联的时钟源/PLL 和分频器。 根据 PLL 修改分频器，使最终输出时钟符合速度等级。

以下是如何更改 AM62A 的 C7x 内核时钟的示例，PLL 被标识为 MAIN_PLL7 HSDIV0，该内容可以在对应器件的 TRM 中得到，因此所做的更改如下。该修改将之前的 1GHz 的主频时钟，通过分频成为 500MHz，其他核的频率更改同样可以使用该方式进行更改。在更改完成后，可以在 Linux 中使用 K3conf 命令对对应的时钟频率进行验证。同样客户可以使用 K3conf set clock $CLOCKID $FREQ 命令对时钟进行更改。不同设备的对应的 Device ID 可能需要进行修改，如 AM62A 中 C7x 对应 ID 为 208，211。

```
k3conf dump clock 208
k3conf dump clock 211
output:
|-------------------------------------------------------------------------------------|
| Device ID | Clock ID | Clock Name                | Status        | Clock Frequency |
|-------------------------------------------------------------------------------------|
|  208   |    0   | DEV_C7X256V0_C7XV_CORE_0_C7XV_CLK | CLK_STATE_READY | 500000000   |
|-------------------------------------------------------------------------------------|
|  211   |    0   | DEV_C7X256V0_CLK_C7XV_CLK             | CLK_STATE_READY | 500000000     |
|  211   |    7   | DEV_C7X256V0_CLK_PLL_CTRL_CLK           | CLK_STATE_READY | 500000000     |
```

```
diff --git a/source/drivers/device_manager/rm_pm_hal/rm_pm_hal_src/pm/soc/am62ax/clocks.c
b/source/drivers/device_manager/rm_pm_hal/rm_pm_hal_src/pm/soc/am62ax/clocks.c
index 2122081..7438db5 100644
--- a/source/drivers/device_manager/rm_pm_hal/rm_pm_hal_src/pm/soc/am62ax/clocks.c
+++ b/source/drivers/device_manager/rm_pm_hal/rm_pm_hal_src/pm/soc/am62ax/clocks.c
@@ -2440,7 +2440,7 @@ static const struct clk_data_div_reg clk_data_hsdiv0_16fft_main_7_hsdiv0 = {
 static const struct clk_data_div_reg clk_data_hsdiv0_16fft_main_7_hsdiv0 = {
            .data_div            = {
                        .n                  = 128,
-                       .default_div        = 2,
+                       .default_div        = 4,
            },
            .reg                            = 0x00680000UL + (0x1000UL * 7UL) + 0x80UL + (0x4UL * 0UL),
            .bit                            = 0,
```

需要注意的是，在更改上述代码后需要对 SDK 里的 lib 文件进行 clean build，并且由于更改的代码是运行在 DM（Device Management）核上的，所以也需要重新 clean build DM 核对应的固件。

4. 总结

TDA4x 和 AM6x 作为最新一代的 TI 处理器芯片，内部包含一个功能强大的 VTM 温度管理模块，帮助 TI 的处理器成为一个高效且强大的多核异构处理器，并在安全的温度范围内发挥其最大的能力，处理器算力能力越强意味着发热越高，客户需要在板卡的整体硬件设计阶段充分考虑器件的散热方案的设计，在系统实际运行过程中，系统应该始终处于器件规定的工作温度之内的，本文提到的温度保护等措施是在异常情况下补充措施，不可成为常规的散热方案。VTM 仅支持测量片上的温度，在系统板卡其他位置如 DDR 颗粒和 eMMC 的温度管理，客户需要自行安装测温电阻或其他传感器进行管理。本文系统介绍了 TI 处理器的内部 VTM 的工作机制和对应目前在器件上使能的软硬件的温度保护策略。

5. 参考文献

1. AM62Ax Sitara™ Processors datasheet (Rev. C)
2. TDA4VM Processors datasheet (Rev. K)
3. AM62Ax Sitara Processors Technical Reference Manual (Rev. B)
4. J721E DRA829/TDA4VM Processors Silicon Revision 2.0, 1.1 Technical Reference Manual (Rev. D)
5. (+) [FAQ] TDA4VM/TDA4VL/TDA4AL/TDA4VH/DRA821: How can we make the Jacinto SDK compatible for device variants? YXZ - Processors forum - Processors - TI E2E support forums
6. (+) [FAQ] AM625 / AM623 / AM620-Q1 / AM62Ax / AM62D-Q1 / AM62Px / AM62L / AM64x / AM243x (ALV, ALX) Custom board hardware design – Voltage and Thermal Manager (VTM) - Processors forum - Processors - TI E2E support forums
7. (+) [FAQ] TDA4VM: How to read on-die temperatures on J7 family of SoCs using Linux - Processors forum - Processors - TI E2E support forums
8. (+) AM625: Linux Thermal shut down - Processors forum - Processors - TI E2E support forums
9. How to Boost Your CPU, GPU, and SoC Performance Through Thermal Accuracy
10. (+) [FAQ] SK-AM62: How do I measure power and temperature on the AM62A and the AM62X? - Processors forum - Processors - TI E2E support forums

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。 严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265