Application Note

Dynamic Temperature Detection and Thermal Management of TDA4x and AM6x

李彪 Central FAE

ABSTRACT

This document was translated from a simplified Chinese source. (ZHCAFM6)

The Jacinto™ 7 TDA4 family and Sitara™ AM6x processors are the latest generation of processors launched by TI based on the Keystone architecture. They integrate different application cores based on an on-chip multi-core heterogeneous architecture, assigning different tasks to the corresponding cores for parallel processing to maximize the advantages of TI processors. Stronger processing capabilities are often accompanied by higher heat dissipation. Excessively high temperatures, or even continuously operating beyond the chip's thermal design point, can cause irreversible damage. How to fully unleash the powerful performance of TI processor chips without affecting the chip lifespan implies that thermal management must be performed on the processors. All TI processor chips integrate on-chip temperature sensors and integrate a dedicated module, VTM (Voltage Thermal Management), to perform thermal management on the chips. The basic working principle of the on-chip temperature sensors is to utilize the physical characteristics of semiconductor materials, where their resistance or potential changes as temperature changes, thereby causing changes in the output electrical signals. This change is read and processed by a microcontroller or other circuits to achieve temperature measurement. The VTM module is also responsible for functions such as chip voltage management, while this application note primarily focuses on its temperature management function.

This application note will introduce the features, working principles, and usage of the VTM integrated on TI processors, as well as the software and hardware protection schemes implemented through the VTM module that are enabled by default in actual chips. When involving software code and specific application examples, the AM62A processor is taken as an example for description. The corresponding code modifications and commands can be applied to all TDA4x and AM6x series processors; however, corresponding path modifications might be required due to SDK version updates.

Table of Contents

List of Figures

List of Tables

1 VTM module

The VTM (Voltage Thermal management) module provides control, status, and interrupt and event generation functions related to integrated temperature sensors and user-programmed thermal events. In chip thermal management, we primarily focus on its functions of temperature measurement, as well as issuing early warnings and interrupts. The figure below is the internal connection diagram of the VTM module within the chip. Since the VTM may directly initiate instructions to reset the entire chip, it also has a direct connection to the reset controller of the entire chip. At the same time, the VTM also contains a set of memory-mapped registers used to store device-specific operating voltages (AVSVNOM) set during device testing. These values can be used by the system AVS software to adjust the operating voltage of the device to achieve optimal operating conditions (balance between power consumption and performance).

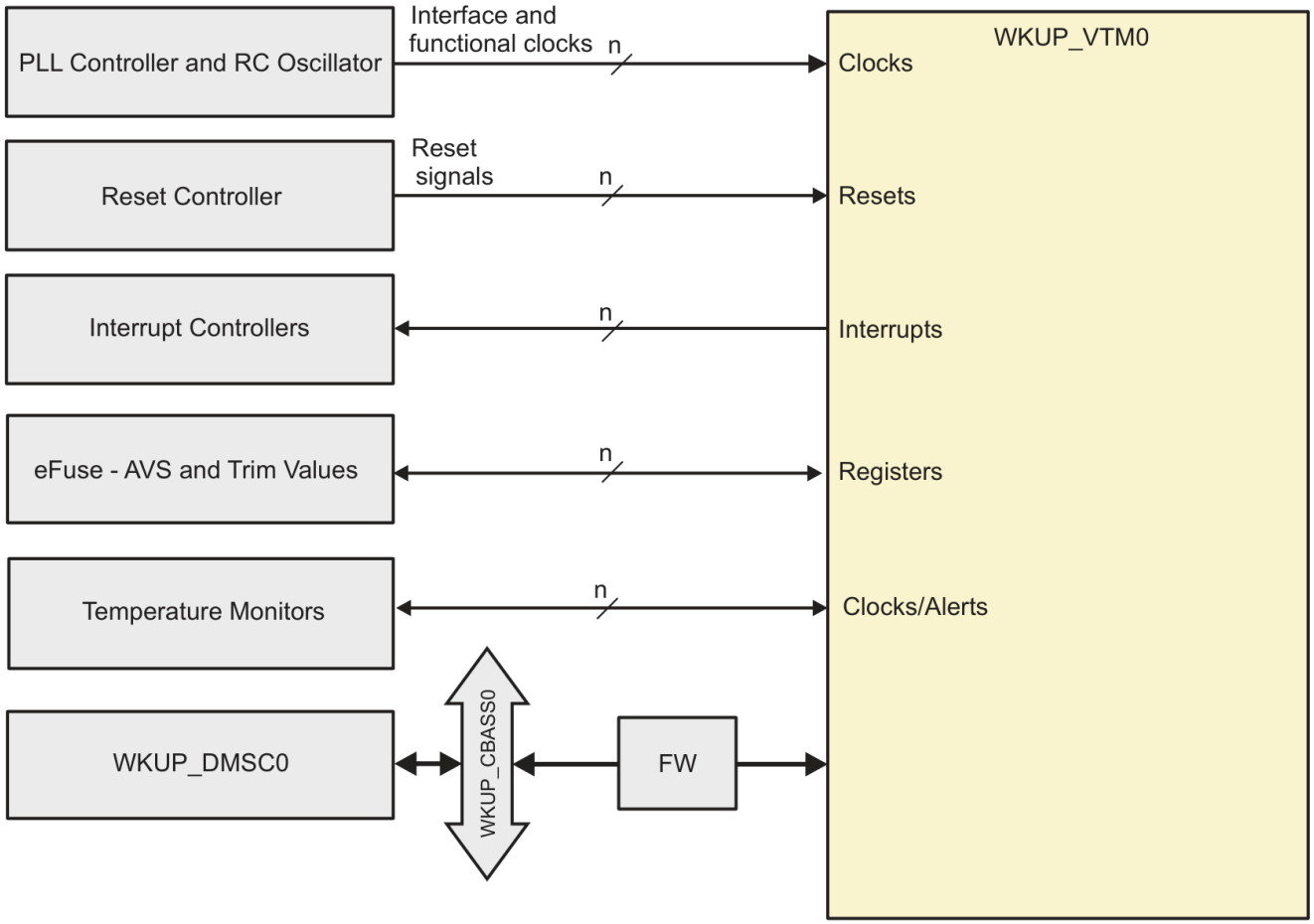

Figure 1-1. VTM System Connection Diagram

There are numerous PN series SoC chips within the entire TI SoC family, among which the system connection methods of the VTM are all similar. The VTM is also generally located in the wake-up domain (WKUP) of the chip. This domain is the first to start working when the chip boots up. Temperature management steps into operation immediately upon chip power-on to ensure the normal operation and lifespan of the entire chip.

1.1 Introduction to VTM Module

The typical layout of the VTM inside a TI SoC is shown in Figure 1-2, where the Temperature Monitor, namely the on-chip temperature sensor, will be placed close to the heat generation points. The VTM module controls the temperature monitors within the chip through internal connections. A single VTM can control up to 8 temperature sensors through its registers. Since temperature sensors do not spontaneously update periodically, the VTM periodically enables the temperature sensors to continuously update the reported temperature data. The temperature values returned by the temperature sensors will be captured by VTM registers and stored in the

corresponding registers inside the VTM. The VTM keeps the sensors in a reset state when they are not enabled to save power and reduce sensor utilization, thereby maximizing sensor lifespan.

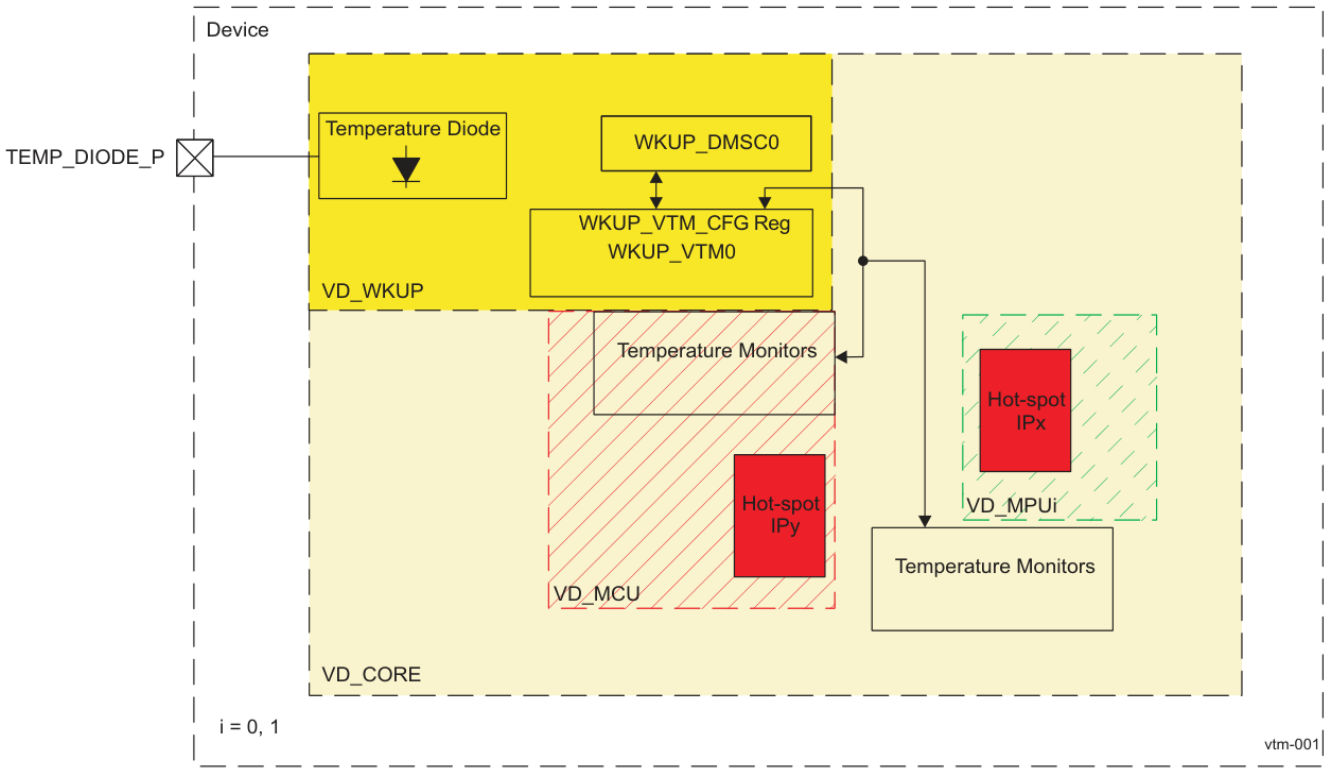

Figure 1-2. VTM Layout Within Chip

Different chips will generally be placed with different quantities of on-chip temperature sensors according to needs. The placement principle is to be as close to the heat generation points as possible and cover all hot spots. Table 1-1 statistics all the on-chip temperature sensors and locations contained in all TI processor products, which customers can consult based on the product used. The following table is statistically categorized based on the hot spots close to physical locations. Since most sensors may be located between two heat sources and at different power domain boundaries, making them difficult to define precisely, the following table can only serve as a rough reference for sensor locations. For details, please refer to the TRM (Technical Reference Manual) of the corresponding device.

Table 1-1. On-Chip Temperature Sensor Table

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	√	-	-	-	-	-	-

All chips will have temperature sensors placed near the Arm large cores, as this is generally the hottest spot of the entire chip. Since the DDR is responsible for the data throughput of the entire chip and generally operates at

high speeds, the risk of overheating is also relatively high. Generally, it is also necessary to place sensors at the DDR controller for monitoring.

1.2 VTM Working Principle and Usage

As mentioned above, each VTM module controls up to 8 temperature sensors, and the temperature values returned by the temperature sensors will be captured by VTM registers and stored in the corresponding registers inside the VTM. The register name is WKUP_VTM_TMPSENS_STAT_j[9-0] DATA_OUT, which stores the temperature value as a 10-bit binary number in the corresponding register. The table below lists several typical actual temperature values and their corresponding 10-bit temperature values.

Table 1-2. On-Chip Temperature Sensor Register Value and Actual Value Conversion Table

Actual Temperature	-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

The table shown above demonstrates a table lookup method, which can easily establish a rough correspondence between the value you read and the actual temperature value. This method is also applicable to converting temperature-related register values and actual values in the chip. Looking up by the table can only provide a rough range of the temperature. To obtain an accurate temperature value, the following formula needs to be used for calculation:

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

After plotting this formula using a Python script, the following image can be obtained, which is roughly a monotonically increasing function. You can input the read register value into the script, and after running, it will automatically calculate the corresponding accurate temperature value and label it. The script can be downloaded from here.

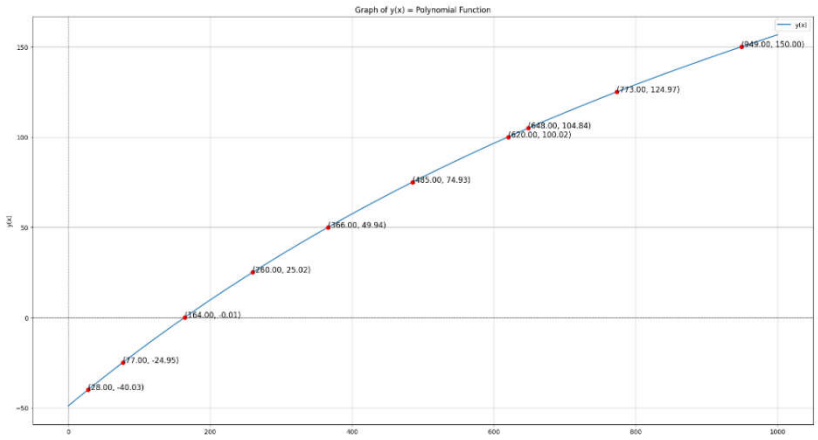

Figure 1-3. Register Value and Actual Value Calculation Formula Diagram

Being able to accurately calculate the corresponding actual temperature value allows for convenient modifications to the system's temperature protection logic. For example, modifying the temperature values encoded in the chip's maximum temperature protection thresholds WKUP_VTM_MISC_CTRL2[25-16] MAXT_OUTRG_ALERT_THR0 and WKUP_VTM_MISC_CTRL2[9-0] MAXT_OUTRG_ALERT_THR thresholds can modify the limit maximum value of the chip temperature protection.

In the default Linux software provided officially by TI, corresponding interfaces and commands are also provided to read the values of the on-chip temperature sensors. The commands used are all similar. Taking AM62A as an example, to read the values of three temperature sensors on the EVM, the following commands can be used to read the values of all temperature sensors in units of milliCelsius. As shown in the following printout results, the temperature values of sensor1, sensor2, and sensor3 are 45.2°C, 38.7°C, and 38.9°C respectively. Due to

different sensor locations, the read temperatures will vary, and a deviation of ±5°C from the average value is normal. This range is defined in the latest data manual of the corresponding device.

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

2 Hardware Temperature Protection of TI Processor Chips

The previous section mainly introduced the hardware management module responsible for temperature in TI processor chips. The values of the on-chip temperature sensors can be read in the VTM. After obtaining the temperature, the VTM module will perform hardware temperature protection for the entire chip based on the read temperature.

2.1 Overtemperature Protection Thresholds of VTM

Each temperature monitoring register set of the VTM can be configured to sample the temperature of its corresponding temperature monitor and trigger up to 3 alert (level) signals, which are: GT_TH1_ALERT (Overtemperature Alert Comparator 1 Result) - Generated by the 10-bit temperature threshold point 1 (THPT1). This value can be set in VTM_TMPSENSx_TH[25-16] TH1_VAL. GT_TH2_ALERT (Overtemperature Alert Comparator 2 Result) - The 10-bit delta point generated by threshold point 2 (THPT2). This value can be set in VTM_TMPSENSx_TH2[9-0] TH2_VAL. LT_TH0_ALERT (Undertemperature Alert Comparator Result) - The 10-bit delta point generated by threshold point 0 (THPT0). This value can be set in VTM_TMPSENSx_TH[9-0] TH0_VAL.

Generally speaking, the principle of THPT2 > THPT1 > THPT0 needs to be observed for settings, and its working principle is as follows:

TH1 is configured as an early alert to indicate that the temperature is higher than the threshold defined in VTM_VTM_TMPSENS_TH_j[25-16] TH1_VAL. TH2 is configured as a warning, indicating that the temperature is higher than the threshold defined in VTM_VTM_TMPSENS_TH2_j[9-0] TH2_VAL. The concept is that TH1 indicates the processor is getting hot; TH2 requires immediate attention. TH0 is configured to trigger when the temperature detected by the sensor is lower than the threshold defined in VTM_VTM_TMPSENS_TH_j[9-0] TH0_VAL. This interrupt indicates that thermal mitigation measures can be relaxed or the emergency state can be exited since the temperature has dropped below the original TH1 level. It should be noted that if LT_TH0_INT is enabled, LT_TH0_INT will be triggered when the read temperature is lower than TH0, regardless of whether the TH1 interrupt and TH2 interrupt are enabled or have ever been triggered. The above three interrupts can all be detected by software, and the thermal dissipation scheme of the chip can be designed based on these interrupts. Customers can design according to their needs; the above is the general logic for reference. Each sensor can independently set overtemperature or undertemperature points. As long as one sensor triggers a warning, the corresponding interrupt will eventually be generated. As shown in the figure below, all sensors will eventually generate three interrupts corresponding to the undertemperature and overtemperature interrupts of TH0, TH1, and TH2, and the customer software can perform subsequent processing on these interrupts. Extra attention is needed that the interrupts will only be active when the sensor is in continuous mode. A single sample of the sensor will not trigger any interrupts.

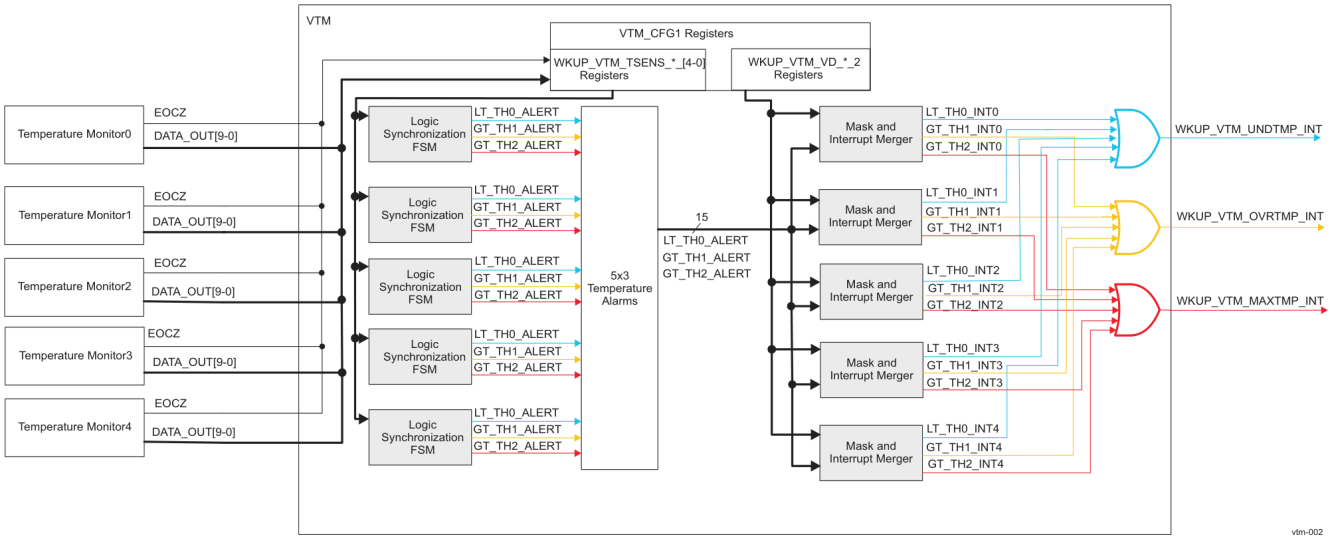

Figure 2-1. Schematic Diagram of VTM Temperature Protection Interrupt Mechanism

The ultimate threshold is MAXT_OUTRG_ALERT_THR. When this threshold is exceeded, the device will be forced into a reset state (and directly shut down the PLLs). When the device cools down sufficiently (< MAXT_OUTRG_ALERT_THR0), the internal reset will be released, and the device will restart. This temperature protection is hardware protection; the hardware performs a reset once triggered. Customers can modify the corresponding thresholds and whether to trigger, but cannot use this interrupt for software design because the chip will immediately enter a reset state once triggered, rendering any software unable to operate, and this protection is not restricted by the sensor's working mode.

2.2 Maximum Hardware Temperature Protection

To ensure that the chip does not operate under overtemperature conditions and to guarantee the normal operating lifespan of the chip, a hardware maximum temperature protection mechanism is set inside TI's processor chips. This mechanism does not require software involvement. By default, TI's processor chips all have this protection enabled. At the same time, the internal mechanism of this protection is introduced here. For a TI chip to trigger the VTM maximum temperature out-of-range alert, the following conditions must be met simultaneously:

1. A valid temperature input needs to be obtained. For the output of the VTM maximum temperature protection to be valid, the VTM must be configured to enable at least one temperature sensor, i.e., WKUP_VTM_TMPSENS_CTRL_j[11] MAXT_OUTRG_EN = 1;
2. In addition, WKUP_VTM_MISC_CTRL[0] ANYMAXT_OUTRG_ALERT_EN must be set to "1". This bit is used to control the switch for the alert signal output of the VTM maximum temperature protection;
3. Under the condition of ensuring the input and output paths, i.e., properly configuring and enabling one temperature sensor and enabling the maximum temperature protection output, if the chip temperature exceeds the programmed value in WKUP_VTM_MISC_CTRL2 [9-0] MAXT_OUTRG_ALERT_THR, the VTM output will be driven to "1" once the sensor detects the programmed maximum temperature;
4. At the device level, the VTM output THERM_MAXTEMP_OUTRANGE_ALERT will drive the PLL controller into a thermal reset state. Meanwhile, the PLL controller enters the reset state, and all PLLs also enter clock bypass mode simultaneously;
5. Through the operations described in point 4, the device will significantly reduce power consumption and lower the device temperature within a few seconds;
6. The VTM continuously reads the temperature, and once it detects the code corresponding to the programmed value in WKUP_VTM_MISC_CTRL2[25-16] MAXT_OUTRG_ALERT_THR0, the VTM will drive THERM_MAXTEMP_OUTRANGE_ALERT = 0;
7. After item 6 is satisfied, the PLL controller brings the SoC out of the reset state and restarts the boot sequence. At the same time, the PLL exits bypass mode, starts the entire chip at its target frequency, and begins normal operation.

The flow chart of the maximum temperature hardware protection is shown in the figure below:

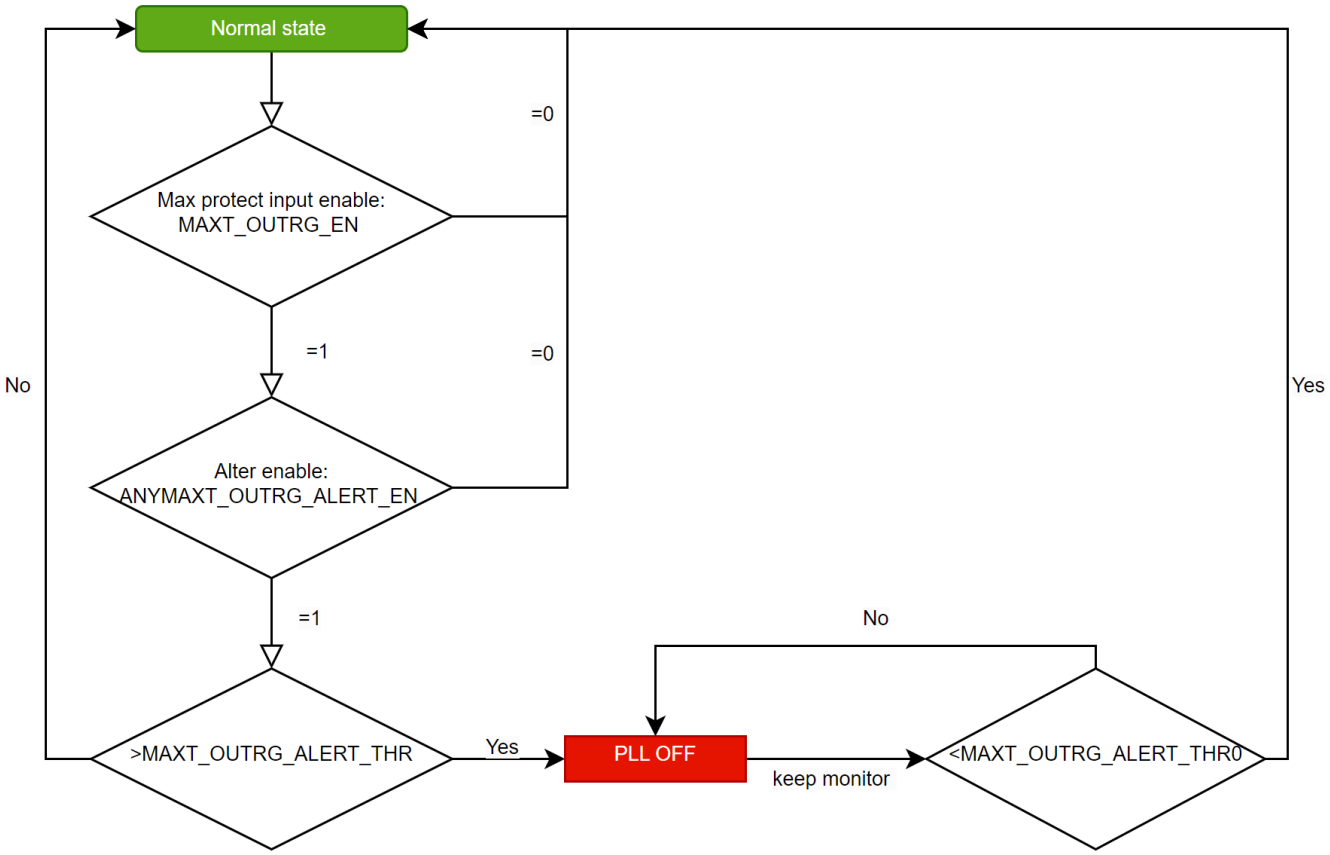

Figure 2-2. Schematic Diagram of VTM Maximum Temperature Hardware Protection

In all series of TI chips, this maximum temperature protection is enabled by default. Since running the chip beyond the maximum temperature will permanently damage the chip lifespan, this impact is irreversible. The temperature range within which different chips can operate normally can be found in the data manual of the corresponding device. In the process of triggering the maximum temperature protection, the chip's clock PLL will be put into bypass mode, meaning the clock signals of the entire chip are turned off. At this time, the VTM will use the clock from the Sensor.

Taking AM62A as an example below, we can obtain the current maximum temperature protection value set for the chip by reading registers. The specific register addresses can be found by flipping through the TRM. Among them, the encoded value of WKUP_VTM_MISC_CTRL2[25-16] MAXT_OUTRG_ALERT_THR0 is 0x288. Using a Python script, it can be calculated that this value corresponds to an actual temperature of 105°C. The encoded value of WKUP_VTM_MISC_CTRL2 [9-0] MAXT_OUTRG_ALERT_THR is 0x2F8, and the corresponding actual temperature value is 123°C. Therefore, the default maximum hardware temperature protection value of AM62A is 123°C, and it will only restart again after the temperature drops below 105°C.

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

3 Software Temperature Protection Strategy

The VTM module is a hardware module inside the chip, whose main task is to manage the temperature of the entire chip. Although the VTM module includes the hardware protection function for the maximum temperature, as well as flexibly settable temperature protection thresholds and overtemperature protection interrupts for users to perform flexible designs. However, hardware-based temperature protection has limitations. Except for the maximum temperature protection which directly stops the chip clock, the hardware module cannot directly control the chip to take other thermal protection measures such as frequency reduction. Moreover, since TI processor chips all adopt multi-core heterogeneous architectures, and the number of temperature sensors contained inside each processor model and their suitable operating temperatures vary, all VTM temperature protection thresholds are set to 0 by default. That is, other temperature protection interrupts except for the maximum temperature protection are not enabled by default. This part needs to be designed by users according to system requirements. Therefore, the current VTM module lacks any temperature protection strategies other than stopping operation. As a supplement, TI has added a software-level temperature protection strategy. After the software obtains the temperature value through the VTM module, it generates the corresponding interrupt or flag bit on its own, allowing the entire system to enter measures such as reducing frequency and stopping high-load work to reduce system load. It can also be flexibly designed differently for different temperatures, which is extremely important for the functional safety of the system.

3.1 Optional Software Temperature Protection Measures

TI processors can provide power estimation electronic spreadsheets for corresponding devices. This spreadsheet provides power estimations based on measurement and simulation data. Although the operating power consumption of a chip depends on many factors such as electrical parameters, silicon process variations, environmental conditions, and the processor's runtime use cases. However, it can also serve as a rough reference for chip power consumption. Therefore, for measures that software can adopt, such as frequency reduction and reducing processor load rates, to affect power consumption, this tool can be used as a rough reference. The temperature and power consumption of a chip are positively correlated. The greater the power consumption, the faster its temperature rises. Therefore, the main goal of software temperature management is to reduce the power consumption of the chip.

The following table takes the AM62A processor as an example, and the corresponding tool can be downloaded from here. This method can also be applied to other TI processors.Table 3-1 The results are all statistical data obtained under the condition that the junction temperature is 125°C and the frequencies of all cores are at their maximum.

Table 3-1. Power Consumption of AM62A Under Different Workloads

Case	Power/mW	A53	DDR	C7x	R5F	DM_R5	ΔPower/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

The Table 3-2 results are all statistical data obtained under the condition that the junction temperature is 125°C and the workloads of all cores are at 80%.

Table 3-2. Power Consumption of AM62A Under Different Operating Frequencies

Case	Power/mW	A53	DDR	C7x	R5F	DM_R5	ΔPower/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

Table 3-2. Power Consumption of AM62A Under Different Operating Frequencies (continued)

Case	Power/mW	A53	DDR	C7x	R5F	DM_R5	ΔPower/mW
DM Freq down	5084	1400MHz	3200MHz	1000MHz	800MHz	400MHz	52
All Freq down	4220	700MHz	1600MHz	500MHz	400MHz	400MHz	916

From the above calculation results, it can be seen that the core that has the greatest impact on power consumption in AM62A is C7x. Therefore, when designing the system's temperature control scheme, reducing the frequency and load of C7x should be considered first after reaching the set temperature.

3.2 Linux Temperature Protection Logic

Starting from SDK11.0 (or earlier), the Linux SDK began to incorporate the Linux VTM driver. The Linux kernel VTM and the chip's hardware VTM are two unrelated concepts. The kernel VTM framework uses the configuration in the device tree (k3-am62-thermal.dtsi, defined by the kernel thermal binding documentation) to monitor sensor temperatures and take corresponding actions, such as reducing CPU frequency, shutting down, or rebooting Linux. The sensor temperature used by Linux is acquired by the AM62x hardware VTM module.

The definition/configuration in the kernel device tree k3-am62a-thermal.dtsi is only an example, and customers can modify it according to project requirements, such as shutting down or rebooting Linux. Using the default settings in the kernel device tree k3-am62a-thermal.dtsi, the kernel will trigger a shutdown sequence when the temperature reaches 105°C. Taking Linux SDK11.0 as an example, the Linux VTM driver also defines interrupt thresholds similar to the hardware VTM. Up to 3 threshold temperatures can be programmed and set, of which 2 are used for greater-than thresholds and 1 is used for a less-than threshold, so that the VTM can alert the kernel to take action. For example, when the first threshold is exceeded, the kernel can be alerted to start reducing the CPU's voltage and clock speed, thereby stabilizing the overall temperature of the SoC. If the SoC temperature continues to rise, we can use the second threshold to take more aggressive measures. For example, we can issue a poweroff command after exceeding the second threshold to completely turn off the device. The threshold temperatures can be set in the kernel using the values we define in the device tree. This can be used to: set a "critical" temperature at which the kernel will shut down. The SoC issues a "passive" alert to the kernel when overheating, and the MPU frequency can be reduced by registering the cpufreq driver as a cooling device. The threshold for software thermal protection can be changed by modifying the following code, such as changing the shutdown temperature from 105°C to 125°C, correspondingly switching the software protection from the industrial-grade shutdown temperature to the automotive-grade temperature. The following code indicates a critical temperature of 55°C with a 5°C hysteresis control. Upon reaching 55°C, it enters a "passive" state to initiate heat dissipation measures, and immediately protects and shuts down upon reaching 125°C (with a 2°C hysteresis control).

```
/* 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>;
                        };
                };
        };
```

When the kernel enters the "passive" state, it can adopt the method of killing low-priority processes to reduce the load of the corresponding core. For some critical tasks, it can adopt the method of reducing the operating frequency to lower power consumption. Dynamic Frequency Selection (DFS) is an excellent way to reduce CPU frequency based on needs. For DFS details, please refer to this tutorial.

The DFS method can currently easily change the frequency of the A core. The method for changing the frequencies of other cores is provided below. First, identify the clock source/PLL and divider associated with the core from the chip's TRM (clocking section). Modify the divider according to the PLL to make the final output clock conform to the speed grade.

The following is an example of how to change the C7x core clock of AM62A. The PLL is identified as MAIN_PLL7 HSDIV0, and this content can be obtained in the TRM of the corresponding device. Therefore, the changes made are as follows. This modification divides the previous 1GHz main frequency clock to become 500MHz. Frequency changes for other cores can also be modified using this method. After the modifications are completed, the corresponding clock frequency can be verified in Linux using the K3conf command. Similarly, customers can use the K3conf set clock $CLOCKID $FREQ command to change the clock. The Device IDs corresponding to different devices may need to be modified. For example, the IDs corresponding to C7x in AM62A are 208 and 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,
```

It should be noted that after modifying the above code, a clean build of the lib files in the SDK is required. Furthermore, since the modified code runs on the DM (Device Management) core, it is also necessary to re-clean build the firmware corresponding to the DM core.

4 Summary

As the latest generation of TI processor chips, TDA4x and AM6x contain a powerful internal VTM temperature management module, helping TI's processors become efficient and powerful multi-core heterogeneous processors and exert their maximum capabilities within a safe temperature range. Stronger processor computing capabilities imply higher heat dissipation. Customers need to fully consider the design of the device's thermal dissipation scheme during the overall hardware design stage of the board. During the actual operation of the system, the system should always remain within the operating temperature specified by the device. Measures such as the temperature protection mentioned in this article are supplementary measures under abnormal conditions and must not become routine thermal dissipation schemes. The VTM only supports measuring on-chip temperatures. For temperature management at other locations of the system board, such as DDR components and eMMC, customers need to install temperature measurement resistors or other sensors on their own for management. This article systematically introduces the internal VTM working mechanism of TI processors and the corresponding software and hardware temperature protection strategies currently enabled on the devices.

5 References

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

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale, TI’s General Quality Guidelines, or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Last updated 10/2025