Application Note

Thermal Management of TDA4x and AM6x

Biao Li, Kevin Peng

ABSTRACT

Jacinto™ 7 TDA4 series and Sitara™ AM6x processors are the latest generation of processors introduced by TI based on the Keystone architecture. They are based on a heterogeneous multi-core architecture, integrating different application cores, assigning different tasks to corresponding cores for parallel processing, thereby maximizing the advantages of TI processors. The more computation power integrated in processor will generate a higher heating, which may cause irreversible damage if exceeding the thermal design point and continuing to operate. Hence, thermal management becomes significant to fully utilize the computation power without affecting processor's lifespan. All of TI processors integrate on-chip temperature sensors and a dedicated module VTM (Voltage Thermal Management) for thermal management. The basic working principle of the on-chip temperature sensor is to utilize the physical properties of semiconductor materials. When the temperature changes, the resistance or potential of the semiconductor material changes, leading to a change in the output electrical signal. This change is read and processed by a microcontroller or other circuit, enabling temperature measurement.

The VTM module also handles voltage management functions. This application note introduces the functions of the VTM module integrated in TI processors, including its working principles, usage methods, and the software and hardware protection schemes implemented. When discussing software code and specific application examples, the AM62A processor is used as a case study. Corresponding code changes and commands can be applied to all TDA4x and AM6x series processors. However, due to SDK version updates, the corresponding coding path modifications may be necessary.

Table of Contents

Trademarks

1 VTM Module

VTM (Voltage Thermal management) module provides control, status, and interrupt and event generation functionality related to integrated temperature sensors and user-programmed thermal events. In chip thermal management, we primarily focus on its temperature measurement, warning emission, and interrupt functionality. Figure 1-1 shows the block diagram of VTM inside the processor. VTM has the direct connection with reset controller of the processor because it could directly send the command to reset the processor. Additionally, the VTM includes 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 working conditions (power consumption and performance balance).

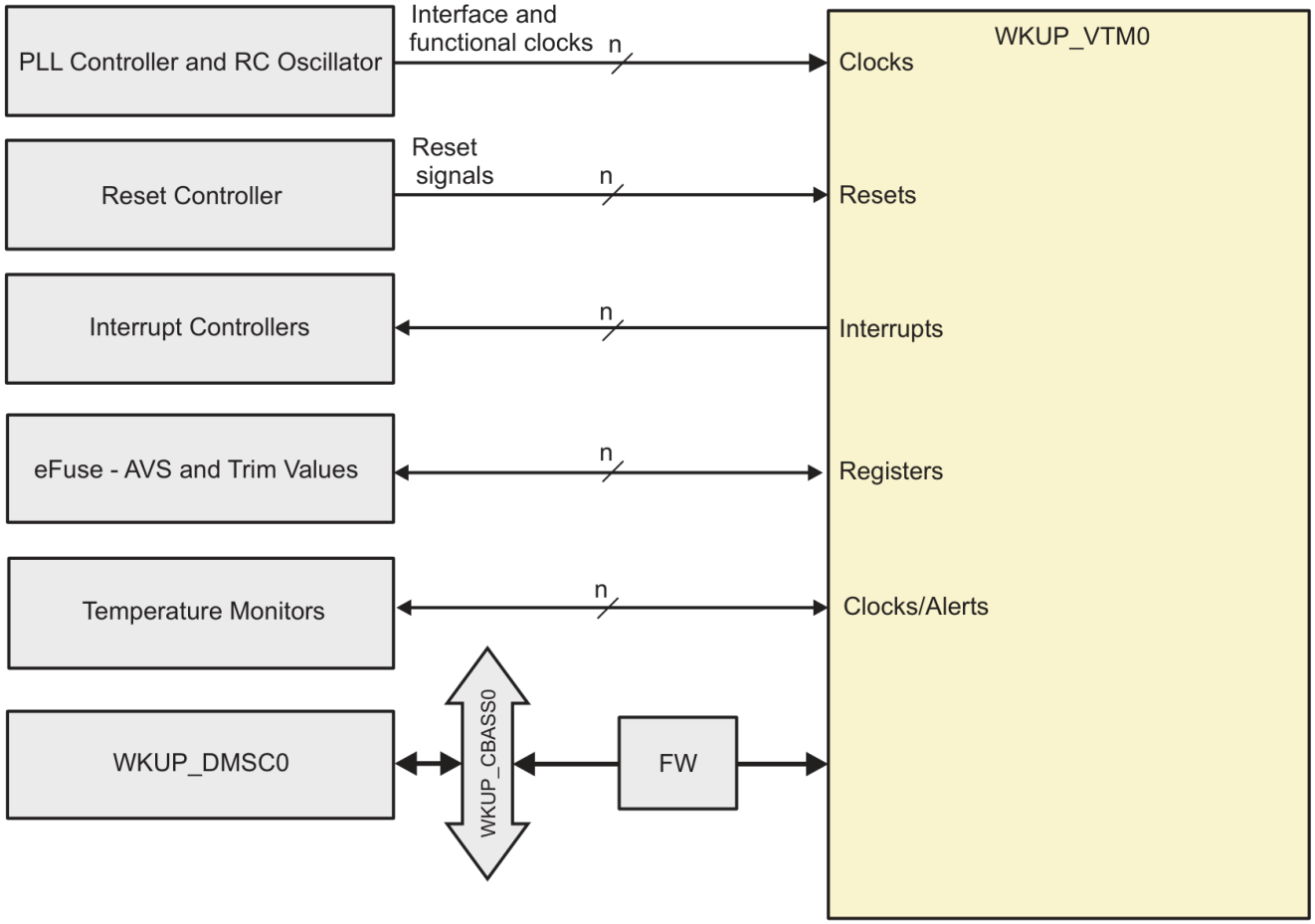

Figure 1-1. VTM System Diagram

TI has plenty of SOC series, and the VTM system connection is quite similar among them. TI has plenty of SOC series, and the VTM system connection is quite similar among them. VTM is generally located in the wake-up domain (WKUP), which is the first to start working when the SOC powers up. Temperature management immediately intervenes upon power-up to ensure the normal operation and longevity of the entire SOC.

1.1 VTM Module Description

The typical layout of VTM on TI SOC is shown in Figure 1-2. This shows that the temperature monitor, for example, the on-chip temperature sensor, is placed near the heat-generating areas. The VTM module controls the temperature monitors within the chip via internal connections. A single VTM can control up to eight temperature sensors through the registers. Since the temperature sensors do not update periodically on their own, the VTM periodically enables the temperature sensors to continuously update the reported temperature data. The temperature values returned by the temperature sensors are captured by the VTM registers and stored in corresponding registers within the VTM. When the sensors are not enabled, the VTM keeps them in a reset state to save power and reduce sensor usage, which maximizes the sensor lifespan.

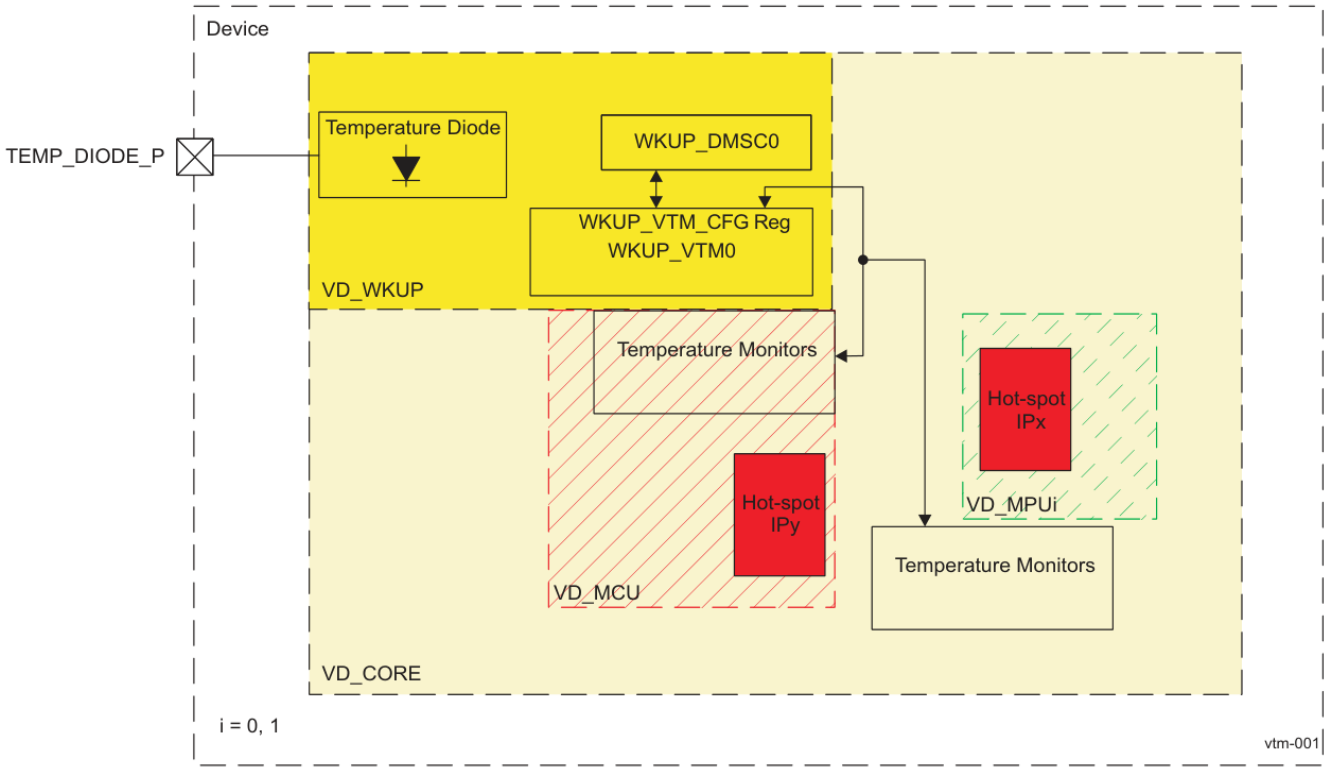

Figure 1-2. VTM Layout

Different SOC normally requires different number of temperature sensors, and the principle is to try to put the sensor close to the heat sources, and also cover all the heating sources. Table 1-1 collects the temperature sensors associating with their position in all of TI processors. Customers can loop up the result according to the corresponding SOC being applied. The table is classified based on the physical proximity of hotspots. Since sensors are mostly located between two heat sources and different power domain boundaries, it is difficult to define clearly. The table can only serve as a rough reference for sensor locations. For detailed information, please refer to the TRM (Technical Reference Manual).

Table 1-1. On-Chip Temperature Sensor

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 the SOCs place a sensor around the Arm cores, because the core is the hottest point in the SOC. Since DDR is responsible for the data throughput of the entire SOC and operates at high speeds, DDR carries a significant risk of overheating. Therefore, it is also generally required to place sensors in the DDR controller for monitoring.

1.2 VTM Working Principle and Usage

The previous section mentions that each VTM can control eight temperature sensors at maximum, and VTM registers can capture the data from the temperature sensors and store into the register. This register is named as WKUP_VTM_TMPSENS_STAT_j[9-0] DATA_OUT. The temperature values are stored in the corresponding registers as 10-bit binary numbers. Table 1-2 lists several typical actual temperature values and their corresponding 10-bit temperature value.

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

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 previous table lookup method can provide a rough correspondence between the values read and the actual temperature values. This method also can be applied for the conversion between the temperature data stored in the register and the actual values. While this table lookup can only provide a rough temperature range, to obtain an accurate temperature value, the following formula must be used for calculation.

$$y = 6.0373 \times 10^{-8} \times x^3 - 1.7058 \times 10^{-4} \times x^2 + 0.32512x - 49.002 - 9.2627 \times 10^{-12} \times x^4 \quad (1)$$

Using a Python script to plot the above formula can generate the following image, which is approximately a monotonically increasing function. The script allows inputting the register values read from the SOC, and after execution, the script automatically calculates the corresponding accurate temperature values and annotates them. The script can be downloaded from here.

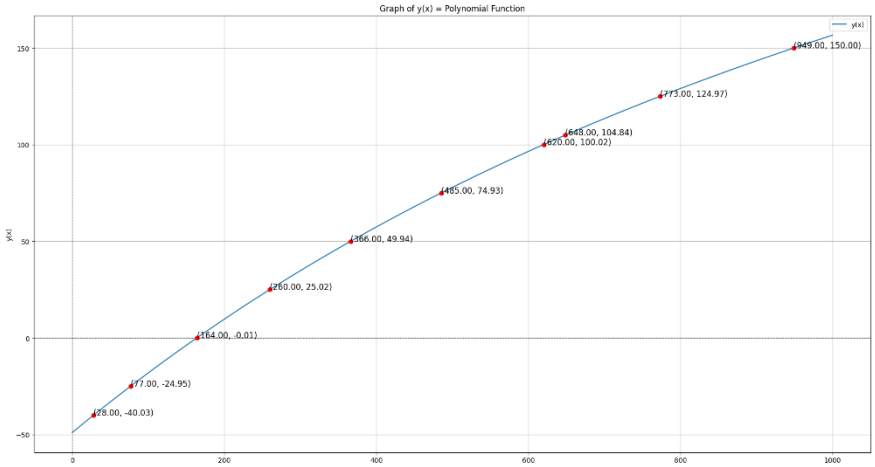

Figure 1-3. Register Value and Actual Value Calculation Formula Diagram

This can precisely calculate the corresponding actual temperature which makes modifying the system temperature protection logic more convenient. For instance, modifying WKUP_VTM_MISC_CTRL2[25-16] MAXT_OUTRG_ALERT_THR0 and WKUP_VTM_MISC_CTRL2[9-0] MAXT_OUTRG_ALERT_THR can change the threshold value of the maximum temperature protection of SOC.

In the default Linux software provided by TI, corresponding interfaces and commands are available to read the values of on-chip temperature sensors. The commands used are similar. For example, on the AM62A EVM, the values of three temperature sensors can be read using the following command. The values of all temperature sensors are read in milliCelsius, with the results displayed as follows: sensor1, sensor2, and sensor3 show temperatures of 45.2°C, 38.7°C, and 38.9°C, respectively. Due to differences in sensor locations, the measured temperatures can vary, with a ±5°C deviation from the average being normal. This range is defined in the latest datasheets of the corresponding SOC.

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

2 Hardware Temperature Protection of TI Processors

Previous sections primarily introduced the hardware temperature management module in TI processors, which is responsible for managing temperature. The VTM module can read the values from on-chip temperature sensors. After obtaining the temperature, VTM performs thermal protection for the entire SOC hardware based on the measured temperature.

2.1 Over-Temperature Protection Threshold of VTM

Each temperature monitoring register group of VTM can be configured to perform temperature sampling on its corresponding temperature monitor and trigger up to 3 alert signals. The first one is a 10-bit incremental point GT_TH1_ALERT (Over-temperature Alert Comparator 1 Result), which is generated by the threshold point THPT1. This value can be configured in VTM_TMPSENSx_TH[25-16] TH1_VAL. The second one is a 10-bit incremental point GT_TH2_ALERT (Over-temperature Alert Comparator 2 Result), which is generated by the threshold point THPT2. This value can be configured in VTM_TMPSENSx_TH2[9-0] TH2_VAL. The third one is a 10-bit incremental point LT_TH0_ALERT (Under-temperature Alert Comparator Result), which is generated by the threshold point THPT0. This value can be configured in VTM_TMPSENSx_TH[9-0] TH0_VAL.

Normally we need follow the principle THPT2 > THPT1 > THPT0 for the configuration, the working principle is explained in the following text.

TH1 is configured as an early alert to indicate a temperature higher than the threshold defined by VTM_VTM_TMPSENS_TH_j[25-16] TH1_VAL. TH2 is configured as a warning to indicate a temperature higher than the threshold defined by VTM_VTM_TMPSENS_TH2_j[9-0] TH2_VAL. The concept is that TH1 indicates the processor is heating up, and TH2 requires immediate attention. TH0 is configured to trigger when the sensor detects a temperature lower than the threshold defined by VTM_VTM_TMPSENS_TH_j[9-0] TH0_VAL. This interrupt indicates that the temperature has dropped below the original TH1 level, allowing for the relaxation of thermal mitigation measures or exiting the emergency state. Note that if LT_TH0_INT is enabled, then regardless of whether the TH1 and TH2 interrupts have been enabled or triggered, when the read temperature is below TH0, the LT_TH0_INT will be triggered. These three interrupts can be detected by software and applied to design SOC thermal management scheme. Customers can design according to their needs, and this is a general logic for reference. Each sensor can independently set over-temperature or under-temperature thresholds. As long as one sensor triggers a warning, the corresponding interrupt will be generated. As shown in the below figure, all sensors can generate three interrupts corresponding to the under-temperature and over-temperature interrupts for TH0, TH1, and TH2. The customer software can handle the interrupt afterward. Note that interrupts are only activated when the sensor is in continuous mode. A single sampling does not trigger any interrupt.

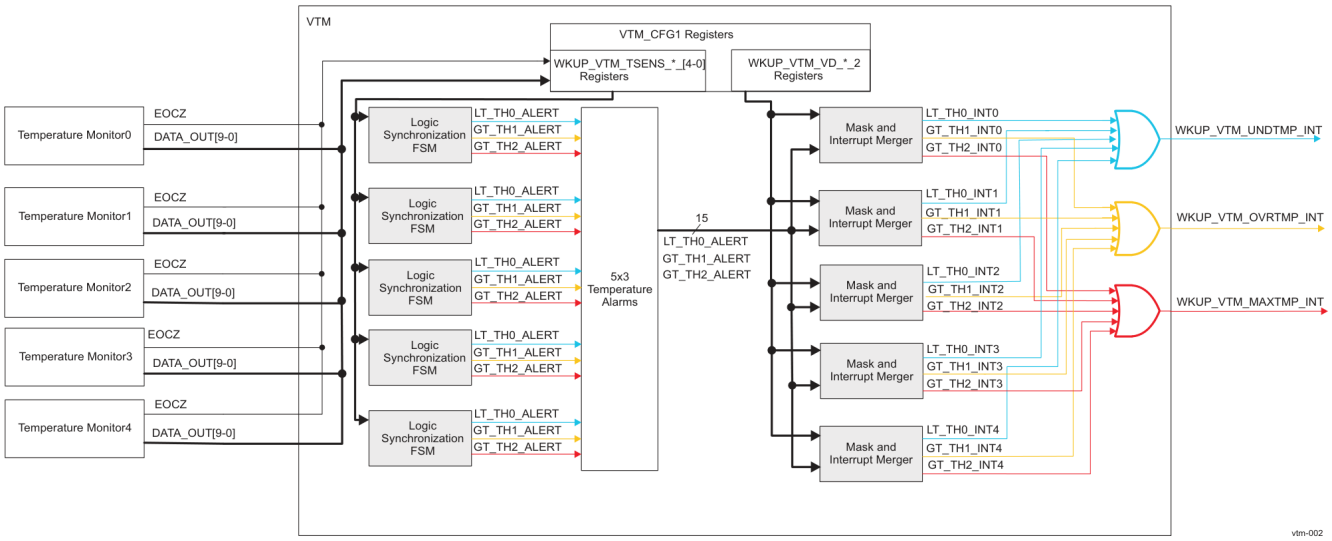

Figure 2-1. VTM Temperature Protection Interrupt Mechanism Diagram

The final threshold is MAXT_OUTRG_ALERT_THR. When this threshold is exceeded, the device will be forcibly reset (and the PLL will be directly disabled). Once the device cools down sufficiently (below MAXT_OUTRG_ALERT_THR0), the internal reset will be released, and the device restarts. This temperature protection is a hardware-based protection mechanism. Once triggered, the hardware performs a reset. Customers can modify the corresponding threshold and whether the trigger is enabled. However, it is not possible to use this interrupt for software design, as the SOC immediately enters a reset state, making any software operations impossible. Additionally, this protection is not limited by the operating mode of the sensor.

2.2 Maximum Hardware Temperature Protection

To ensure that SOC operates without exceeding its maximum operating temperature and to guarantee its normal lifespan, TI processors include a hardware-based maximum temperature protection mechanism. This mechanism does not require software participation and is enabled by default on TI SOC. This section introduces the internal mechanism of this protection. For a TI SOC to trigger a VTM alert for a temperature exceeding the maximum threshold, the following conditions must be met.

1. To ensure the effective output of the VTM's maximum temperature protection, the VTM must be configured to enable at least one temperature sensor, specifically by setting WKUP_VTM_TMPSENS_CTRL_j[11] MAXT_OUTRG_EN = 1.
2. Additionally, the bit WKUP_VTM_MISC_CTRL[0] ANYMAXT_OUTRG_ALERT_EN must be set to '1'. This bit controls the switch for the warning signal output of the maximum temperature protection for VTM.
3. In the context of ensuring input and output pathways, for example, after correctly configuring and enabling a 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, once the sensor detects the programmed maximum temperature, the VTM output will be driven to "1".
4. At the SOC level, the VTM outputs THERM_MAXTEMP_OUTRANGE_ALERT, which drives the PLL controller into a thermal reset state. Meanwhile, the PLL controller enters a reset state, and all PLLs simultaneously enter bypass mode.
5. Based on Point 4, the device will significantly reduce power consumption and lower the device temperature within a few seconds.
6. VTM continuously monitors the temperature. Once VTM detects a code corresponding to the programmed value in WKUP_VTM_MISC_CTRL2[25-16] MAXT_OUTRG_ALERT_THR0, VTM drives THERM_MAXTEMP_OUTRANGE_ALERT to 0.
7. After the completion of point 6, the PLL controller exits the reset state and restarts the boot sequence. Meanwhile, the PLL exits bypass mode and starts the entire chip at the target frequency, beginning normal operation。

The maximum temperature hardware protection flow chart is shown in Figure 2-2

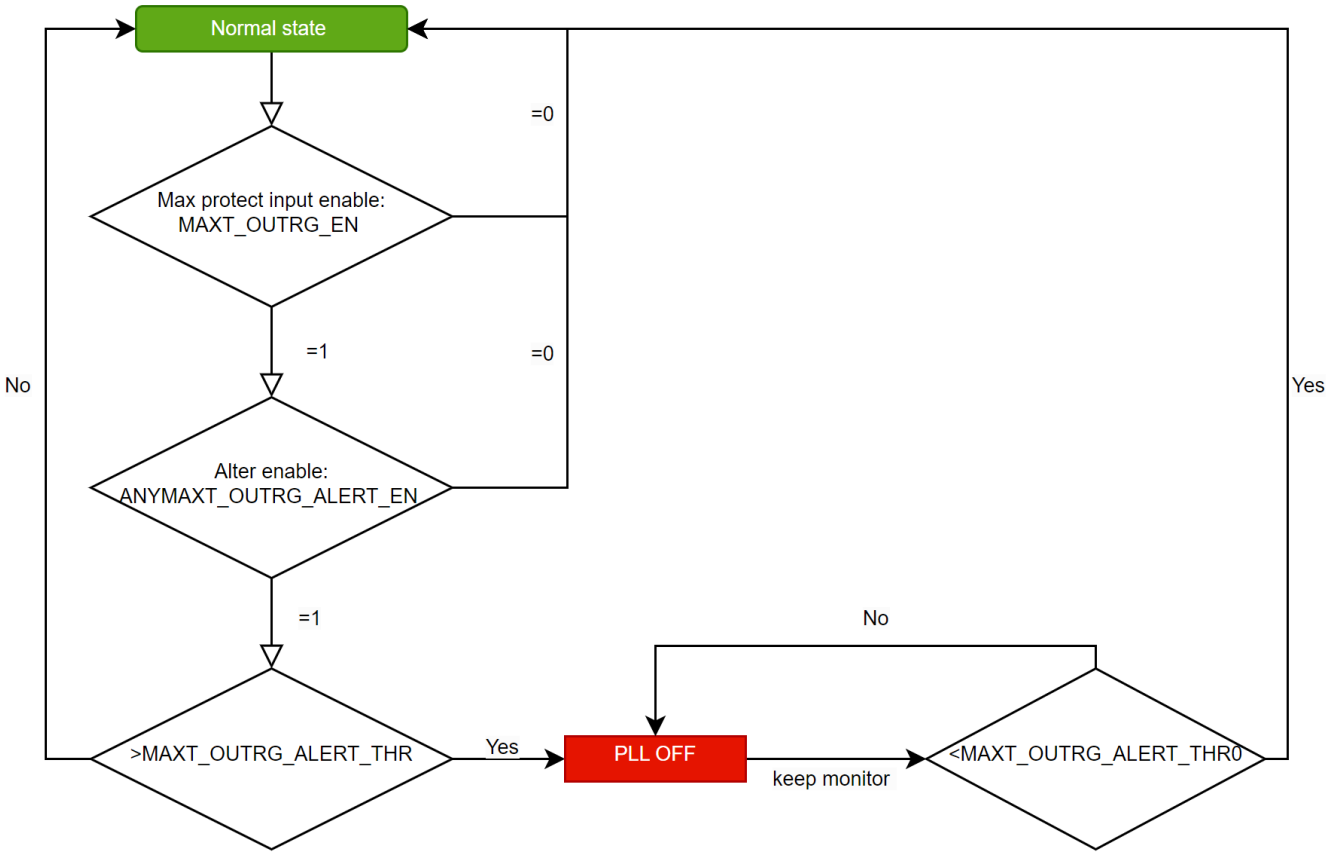

Figure 2-2. VTM Maximum Temperature Hardware Protection Diagram

In all series of TI SOC, the maximum temperature protection is enabled by default. Operating SOC beyond the maximum temperature will permanently damage SOC lifespan, and this effect is irreversible. The normal operating temperature range for different SOC can be found in the corresponding device datasheets. During the activation of the maximum temperature protection, the SOC's clock PLL enters bypass mode, effectively disabling the entire clock signal of the SOC. At this point, the VTM uses signals from the sensor.

The following example uses AM62A as a case study. We can obtain the SOC maximum temperature protection value by reading the corresponding registers, with specific register addresses detailed in the TRM. Specifically, the encoded value of WKUP_VTM_MISC_CTRL2[25-16] MAXT_OUTRG_ALERT_THR0 is 0x288, and using a Python script, this value corresponds to an actual temperature of 105°C. Similarly, the encoded value of WKUP_VTM_MISC_CTRL2[9-0] MAXT_OUTRG_ALERT_THR is 0x2F8, corresponding to an actual temperature of 123°C. Therefore, the default maximum hardware temperature protection value for the AM62A is 123°C. The system only restarts 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 which primarily manages the temperature of the entire SOC. Although the VTM module includes the highest temperature hardware protection function, flexible temperature protection thresholds, and over-temperature protection interrupts for user-configurable design, the hardware-based temperature protection is limited. Beyond stopping the SOC clock when the maximum temperature is reached, the hardware module cannot directly control other thermal protection measures such as frequency reduction. Additionally, since TI processors are multi-core heterogeneous architectures, and each processor model contains a different number of temperature sensors and preferred operating temperatures, the default VTM temperature protection thresholds are set to 0, meaning only the maximum temperature protection is enabled by default. Other temperature protection interrupts require user configuration based on system requirements. Therefore, the VTM module currently lacks any temperature protection strategies beyond stopping the SOC. As a supplement, TI has added software-level temperature protection strategies. Software can obtain temperature values through the VTM module and generate corresponding interrupts or flags to enable the system to take measures such as reducing frequency, stopping high-load operations, and lowering system load. This is crucial for system functional safety.

3.1 Optional Software Temperature Protection Measures

TI processors can provide a power consumption estimation spreadsheet for corresponding devices, which is based on measured and simulation data to estimate power consumption. Although the operating power consumption of SOC depends on various factors such as electrical parameters, process variations, environmental conditions, and the use cases during processor operation, it can still serve as a rough reference for SOC power consumption. Therefore, software measures such as frequency scaling and reducing loading can be evaluated using this tool as a general reference. The temperature and power consumption of SOC are positively correlated, the higher power consumption leads to faster temperature increases. Hence, the primary goal of software thermal management is to reduce SOC power consumption.

Table 3-1 uses an AM62A processor as an example, and the corresponding tool can be downloaded from here. This method is also applicable to other TI processors. The results of Table 3-1 are statistically analyzed under a junction temperature of 125°C and with all cores operating at maximum frequency.

Table 3-1. AM62A Power Consumption under Different Loading

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

Results in Table 3-2 are conducted under a junction temperature of 125°C and all cores operating at 80% loading.

Table 3-2. AM62A Power Consumption at Varying 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
DM Freq down	5084	1400MHz	3200MHz	1000MHz	800MHz	400MHz	52
All Freq down	4220	700MHz	1600MHz	500MHz	400MHz	400MHz	916

From the calculation results, the C7x core in the AM62A has the most significant impact on power consumption. Therefore, when designing the temperature control scheme of the system, after reaching the target temperature, prioritize reducing the frequency and loading of the C7x core.

3.2 Linux Temperature Protection Logic

From SDK 11.0 (or earlier), the Linux SDK began incorporating the Linux VTM driver. The Linux kernel VTM and the hardware VTM of the SOC are two distinct concepts. The kernel VTM framework uses device tree configurations (e.g., k3-am62-thermal.dtsi, defined by the kernel thermal binding document) to monitor sensor temperatures and take corresponding actions, such as reducing CPU frequency, shutting down, or restarting Linux. The sensor temperature used by Linux is obtained from the AM62x SOC hardware VTM module.

The definition and configuration in the kernel device tree k3-am62a-thermal.dtsi is merely an example. Customers can modify this according to project requirements, for example, to shut down or restart Linux. Using the default settings in the kernel device tree k3-am62a-thermal.dtsi, the kernel triggers a shutdown sequence when the temperature reaches 105°C. For Linux SDK 11.0, the Linux VTM driver also defines similar interrupt thresholds for hardware VTM, allowing up to three programmable temperature thresholds. Two of these thresholds are for values above the threshold, and one is for values below, enabling the VTM to alert the kernel to take action. For example, when the first threshold is exceeded, the kernel can be alerted to begin reducing the voltage and clock speed of the CPU, allowing the overall temperature of the SoC to stabilize. If the SoC temperature continues to rise, we can use the second threshold to take more aggressive actions. For example, we could issue a power-off command to completely shut down the device once the second threshold is exceeded. Threshold temperatures can be set in the kernel using the values defined in the device tree. This can be used to set a *critical* temperature at which the kernel shuts down the SoC. When the SoC overheats, it sends a *passive* alert to the kernel, and the MPU frequency can be reduced by registering the cpufreq driver as a cooling device. Software thermal protection thresholds can be modified by changing the code, such as adjusting the shutdown temperature from 105°C to 125°C, which switches the software protection from industrial-grade to automotive-grade temperature. The following code indicates a critical temperature of 55°C with a 5°C hysteresis control, entering a *passive* state to initiate cooling measures when reaching 55°C, and immediately shutting down at 125°C (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 core enters a *passive* state, the core can reduce the load on the corresponding core by killing low-priority processes. For critical tasks, reducing the operating frequency can lower power consumption. The Dynamic Frequency Selection (DFS) is an effective method for dynamically adjusting the CPU frequency to optimize performance. For more details, refer to the tutorial.

The DFS method currently allows for convenient adjustment of the frequency of A-core. The following section provides methods for changing the frequency of other cores. Firstly is to identify the clock source and PLL and divider associated with the core from the TRM (clock section). Modify the divider based on the PLL to ensure the final output clock meets the speed grade.

The following is an example of how to modify the C7x core clock of the AM62A. The PLL is identified as MAIN_PLL7 HSDIV0, which can be found in the TRM of the corresponding device. The changes made are as follows: the original 1GHz main clock frequency is divided to become 500MHz. Similar frequency adjustments for other cores can be made using this method. After the changes, the corresponding clock frequency can be verified using the k3conf command in Linux. Customers can also use the k3conf set clock $CLOCKID $FREQ command to modify the clock. The Device ID for different devices may need modification; for example, the C7x in the AM62A corresponds to ID 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,
```

Note that after modifying the above code, a clean build of the lib files in the SDK is required, and since the modified code runs on the DM (Device Management) core; perform a clean build of the firmware corresponding to the DM core.

3.3 Linux Disable Unused Cores

TI processors are composed of heterogeneous architectures with different cores, such as the TDA4VH, which includes 8 A72 cores, 8 R5F cores, and 4 DSP C7x cores. Some customers select the TDA4VH specifically for the 8 A72 and R5F cores, but do not utilize the 4 DSP C7x cores. Since TI provides open-source SDK for a given processor in a superset configuration, offering full-spec performance, and performs corresponding software modifications to remove unused cores to reduce unnecessary power consumption. The following example demonstrates the software modification of the four unused DSP C7x cores in TDA4VH in SDK 10.0 to remove them and reduce the actual operating temperature.

```
diff --git a/arch/arm64/boot/dts/ti/k3-j784s4-evm.dts b/arch/arm64/boot/dts/ti/k3-j784s4-evm.dts
index de256005f..dff4c4408 100644
--- a/arch/arm64/boot/dts/ti/k3-j784s4-evm.dts
+++ b/arch/arm64/boot/dts/ti/k3-j784s4-evm.dts
@@ -1310,28 +1310,28 @@
 };
 &c71_0 {
-       status = "okay";
+       status = "disabled";
        mboxes = <&mailbox0_cluster4 &mbox_c71_0>;
        memory-region = <&c71_0_dma_memory_region>,
                        <&c71_0_memory_region>;
 };
&c71_1 {
-       status = "okay";
+       status = "disabled";
        mboxes = <&mailbox0_cluster4 &mbox_c71_1>;
        memory-region = <&c71_1_dma_memory_region>,
                        <&c71_1_memory_region>;
 };
 &c71_2 {
-       status = "okay";
+       status = "disabled";
        mboxes = <&mailbox0_cluster5 &mbox_c71_2>;
        memory-region = <&c71_2_dma_memory_region>,
                        <&c71_2_memory_region>;
 };
 &c71_3 {
-       status = "okay";
+       status = "disabled";
        mboxes = <&mailbox0_cluster5 &mbox_c71_3>;
        memory-region = <&c71_3_dma_memory_region>,
                        <&c71_3_memory_region>;
 };
```

4 Summary

The TDA4x and AM6x, as the latest generation of TI processors, incorporate a powerful VTM temperature management module. This module helps TI's heterogeneous multi-core processors become efficient and powerful, enabling them to operate at their maximum capacity within a safe temperature range. The stronger the processing capability of the processor, the higher the heat generation. Therefore, customers must consider the thermal design of components during the overall hardware design phase of the board. During actual system operation, the system should always operate within the specified working temperature range of the components. The temperature protection measures mentioned in this text are supplementary measures for abnormal situations and should not be considered as regular thermal management solutions. The VTM only supports measuring on-chip temperature. For temperature management in other locations on the system board, such as DDR and eMMC, customers need to install temperature sensors or other sensors themselves. This application note introduces the working mechanism of the VTM module and the corresponding hardware and software temperature protection strategies currently enabled on TI SOC.

5 References

1. Texas Instruments, *AM62Ax Sitara™ Processors* , data sheet.
2. Texas Instruments, *TDA4VM Processors* , data sheet.
3. Texas Instruments, *AM62Ax Sitara Processors Technical Reference Manual* , technical reference manual.
4. *J721E DRA829/TDA4VM Processors Silicon Revision 2.0, 1.1 Technical Reference Manual*, technical reference manual
5. Texas Instruments, *(+) [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*, FAQs.
6. Texas Instruments, *(+) [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*, FAQs.
7. Texas Instruments, *(+) [FAQ] TDA4VM: How to read on-die temperatures on J7 family of SoCs using Linux*, FAQs.
8. Texas Instruments, *(+) AM625: Linux Thermal shut down*, forum.
9. Texas Instruments, *How to Boost Your CPU, GPU, and SoC Performance Through Thermal Accuracy*, FAQs.
10. Texas Instruments, *(+) [FAQ] SK-AM62: How do I measure power and temperature on the AM62A and the AM62X?* , FAQs.

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