Temperature Control

Application Note for 64-bit NuMicro® Family

Document Information

Abstract	This document presents an example implementation of MA35D1 chip temperature management.
Apply to	NuMicro® MA35D1 series.

Table of Contents

1 Overview

Modern system on chip (SOC) contains multiple CPU cores, the processor frequency is higher, the grid is dense and leaks more in the same area, so system management at higher temperatures needs to be considered.

When running in a high temperature environment and system running with heavy loading, MA35D1 chip temperature may becomes high. It is important to cool down the chip to prevent from system execution abnormal. Slow down CPU working frequency, system clock, PLL clock, and engine clock can help to cool down chip. Physical countermeasures, such as the use of fans, can help cool the IC more effectively.

This document presents an implementation example of temperature control. This example divides the MA35D1 temperature state into 4 temperature zones. Zone1 is the temperature range in which the chip normally operates, while Zone2 ~ Zone4 are the high temperature range of the chip, and different cooling strategies must be adopted.

MA35D1 is equipped with a thermal sensor, and the Linux thermal driver supports to read the current temperature value from it.

MA35D1 Linux BSP contains the proprietary ma35d1-misctrl driver for supporting chip cooling. This driver provide several ioctl methods to provide PMIC voltage control, CPU clock control, EPLL/VPLL clock control, and SYS-PLL control.

2 Temperature Zones and Control

2.1 Temperature Zones

This example implementation of MA35D1 temperature control divides the chip temperature state into four zones by temperature range, as shown in Figure 2-1.

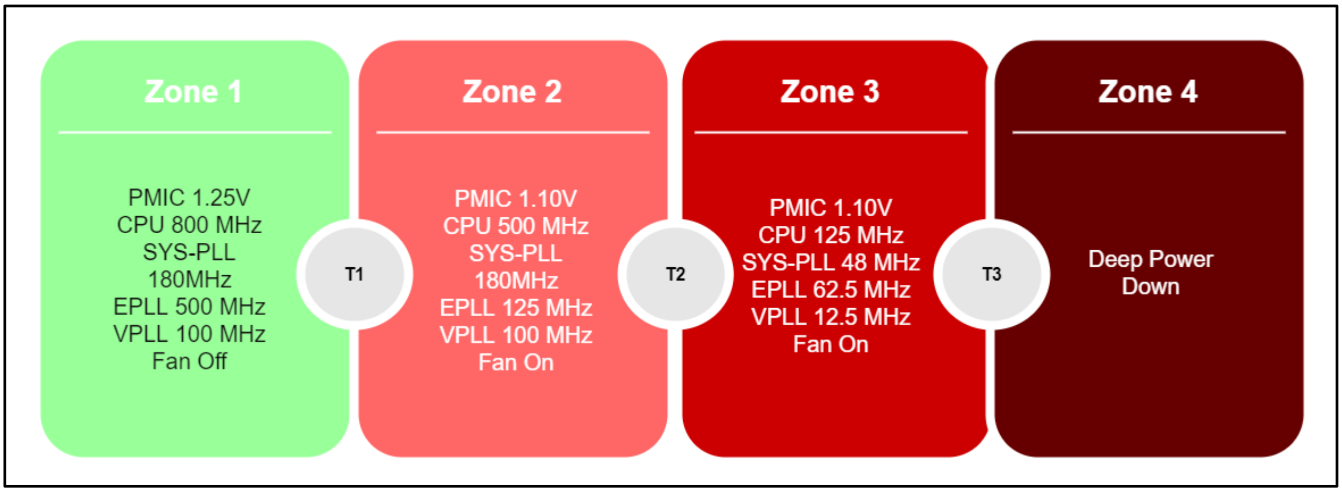

Figure 2-1 Temperature Zones

Zone 1 is the normal run zone. When the temperature rising over T1, the chip enters Zone 2 and this example will slow down CPU clock to 500 MHz, reduce PMIC to 1.10V, make EPLL divided by 2, and turn on the fan if presented. If the temperature keep rising over T2, the chip enters Zone 3 and this example will slow down CPU clock to 125 MHz, slow down SYS-PLL to 48 MHz, make both EPLL and VPLL divided by 8, and keep fan on. If the temperature keep rising over T3, the chip enters Zone 4 and this example will force system enter deep power down state and setup RTC to wakeup system after a period of time.

2.2 Temperature State Transition

The transition between temperature states are determined by the current temperature measured by MA35D1 built-in temperature sensor. This example defines four constant to indicate the transition conditions of temperature zone transition.

```
#define T_Z1_TO_Z2        115 /* Enter Z2 once larger than T_Z1_TO_Z2 */
#define T_Z2_TO_Z3        120 /* Enter Z3 once larger than T_Z2_TO_Z3 */
#define T_Z3_TO_Z4        125 /* Enter Z4 once larger than T_Z3_TO_Z4 */
#define T_DOWN_TO_Z1      105 /* Once enter Z2/Z3/Z4, keep cooling until lower than
                                 T_DOWN_TO_Z1 to back to Zone 1 */
```

In Figure 2-2, T_Z1_TO_Z2, T_Z2_TO_Z3, and T_Z3_TO_Z4 are referred to T1, T2, and T3 respectively. T_DOWN_TO_Z1 is referred to T0, and t is the temperature read from the temperature sensor. Once the chip temperature enters zone 2/3/4, it goes back to zone 1 only when the temperature is cooling down to be lower than T0. Note that T0 is lower than T1 generally, that intends to prevent frequent state transition.

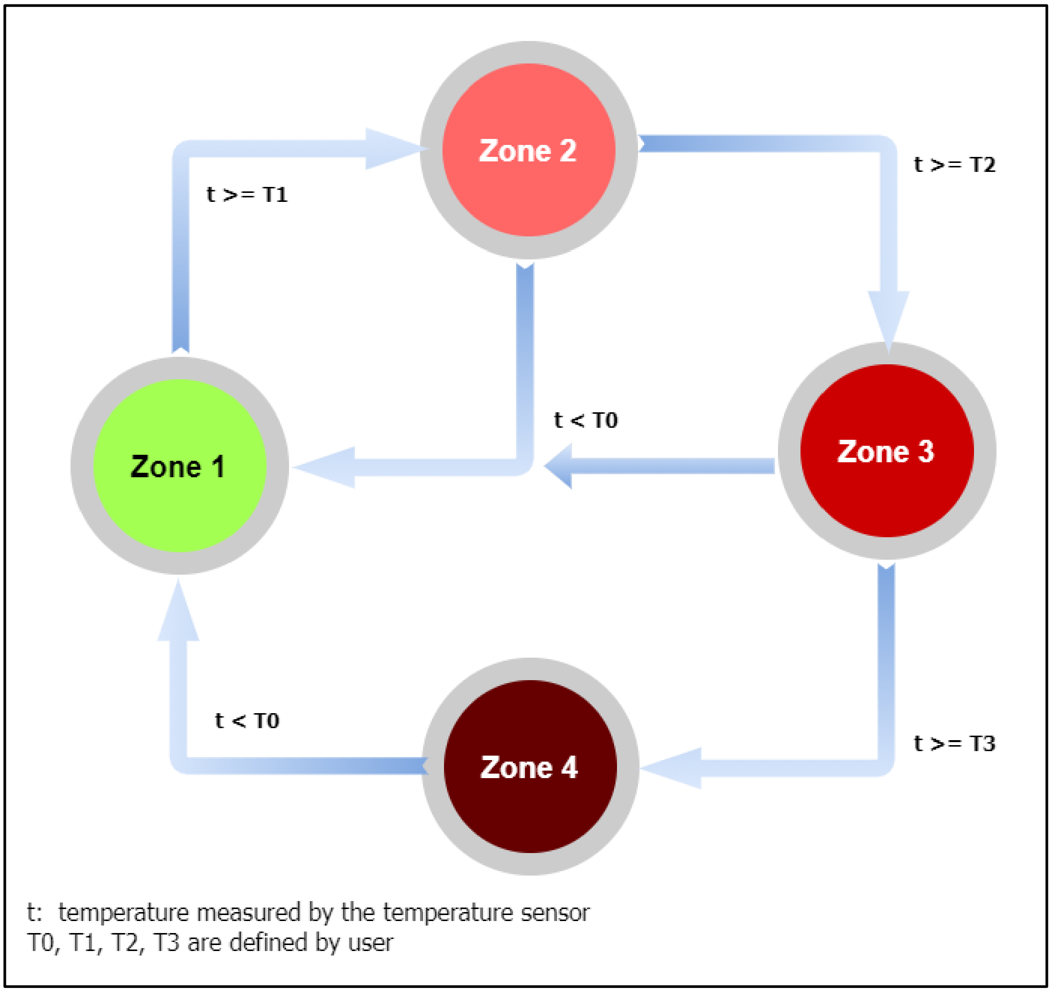

Figure 2-2 Temperature Zone Transition

3 Temperature Management Example

3.1 Required Drivers

This example program requires three drivers, temperature sensor driver, RTC driver, and miscellaneous control driver.

3.1.1 Thermal Sensor Driver

The MA35d1 thermal driver obtains temperature from the thermal sensor registers and transmit the temperature to the subsystem of the Linux kernel.

MA35D1 has only one thermal sensor to measure the processor temperature. So there's only 1 thermal zone and the sensor is assigned as "<&tsen>". "polling-delay-passive" is the maximum number of milliseconds to wait between polls when performing passive cooling and "polling-delay" is the maximum number of milliseconds to wait between polls when checking this thermal zone. The "polling-delay" should set with a minimum value of 400. "polling-delay-passive" can set to 0 or other value if preferred

```
tsen: tsen {
    compatible = "nuvoton,ma35d1-tsen";
    #thermal-sensor-cells = <0>;
    nuvoton,ma35d1-sys = <&sys>;
    status = "okay";
};
thermal-zones {
    cpu_thermal: cpu-thermal {
        polling-delay-passive = <0>;
        polling-delay = <400>;
        thermal-sensors = <&tsen>;
    };
};
```

3.1.2 RTC Driver

This example performs the system deep power down when the temperature state raised to zone 4. Before entering power down, this example configures the RTC to wake up system later. Check device tree to make sure RTC driver is enabled.

```
rtc: rtc@40410000 {
    compatible = "nuvoton,ma35d1-rtc";
    reg = <0x0 0x40410000 0x0 0x10000>;
    interrupts = <GIC_SPI 5 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&clk RTC_GATE>;
```

```
    status = "okay";
};
```

3.1.3 Miscellaneous Control Driver

The miscellaneous control driver provides a set of ioctl commands for configuring PMIC voltage, CPU clock, EPLL, VPLL, and SYS-PLL. Check device tree to make sure miscellaneous control driver is enabled.

```
misctrl {
    compatible    = "nuvoton,ma35d1-misctrl";
    status = "disabled";
};
```

3.2 The Example Code

The temperature control example can be download from github repository: https://github.com/OpenNuvoton/MA35D1_Linux_Applications. It can be found in the examples/temp_ctrl folder.

4 Conclusion

Users can easily obtain the current system core temperature through the temperature sensor driver and perform cooling operations provided by the miscellaneous driver. It is suggested to customized this example to add more countermeasures when core temperature is getting high, for example, force to close some heavy loading applications, sound warning, shutdown, etc.

Revision History

Date	Revision	Description
2022.11.01	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, “Insecure Usage”.

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer’s risk, and in the event that third parties lay claims to Nuvoton as a result of customer’s Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.