StarFive 赛昉科技

JH7110 Thermal System Developing Guide

VisionFive 2

Version: 1.0

Date: 2023/08/04

Doc ID: JH7110-DGEN-19

Legal Statements

Important legal notice before reading this documentation.

PROPRIETARY NOTICE

Information in this document is provided "as is," with all faults. Contents may be periodically updated or revised due to product development. Shanghai StarFive Technology Co., Ltd. (hereinafter "StarFive") reserves the right to make changes without further notice to any products herein.

StarFive expressly disclaims all warranties, representations, and conditions of any kind, whether express or implied, including, but not limited to, the implied warranties or conditions of merchantability, fitness for a particular purpose, and non-infringement.

StarFive does not assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation indirect, incidental, special, exemplary, or consequential damages.

Contact Us

Address: Room 502, Building 2, No. 61 Shengxia Rd., China (Shanghai) Pilot Free Trade Zone, Shanghai, 201203, China

Website: http://www.starfivetech.com

Email:

- Sales: sales@starfivetech.com
- Support: support@starfivetech.com

Contents

List of Tables

List of Figures

Preface

About this guide and technical support information.

About this document

This document mainly provides the SDK developers with the programing basics and debugging know-how for the power management of the StarFive next generation SoC platform - JH7110.

Audience

This document mainly serves the thermal relevant driver developers. If you are developing other modules, place a request to your sales or support consultant for our complete documentation set on JH7110.

Revision History

Table 0-1 Revision History

Version	Released	Revision
1.0	2023/08/04	The First Official Release.

Notes and notices

The following notes and notices might appear in this guide:

- **Tip:**
  Suggests how to apply the information in a topic or step.
- **Note:**
  Explains a special case or expands on an important point.
- **Important:**
  Points out critical information concerning a topic or step.
- **CAUTION:**
  Indicates that an action or step can cause loss of data, security problems, or performance issues.
- **Warning:**
  Indicates that an action or step can result in physical harm or cause damage to hardware.

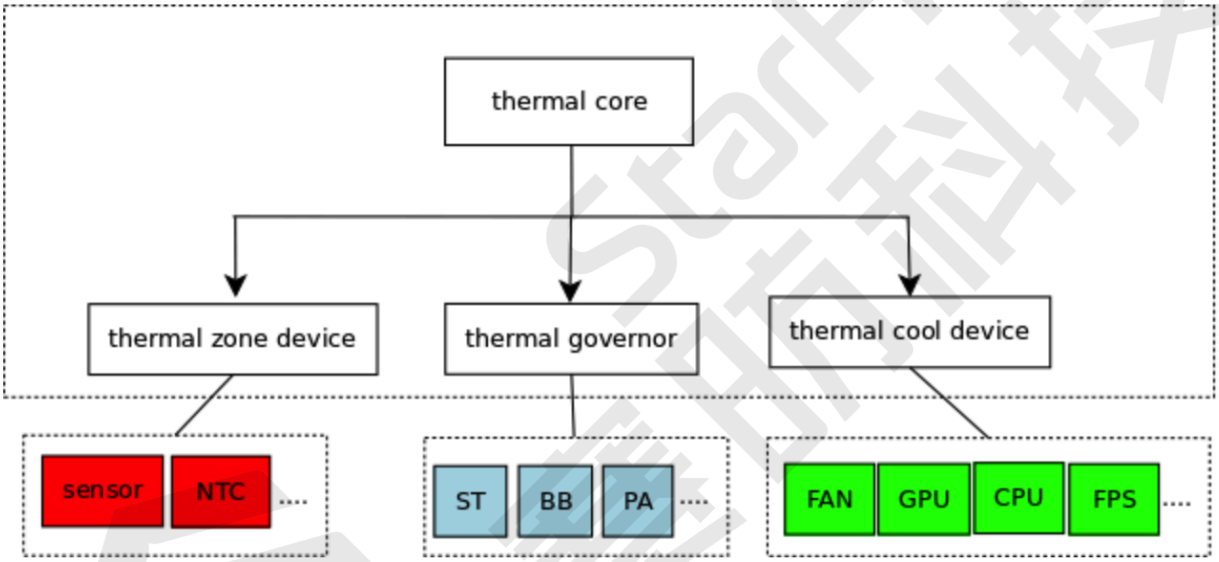
1. Introduction

The thermal system uses a thermal sensor to load the current temperature of the CPU and GPU. Then, based on the value, you can adjust the working frequency of the GPU and CPU, for example, control the maximum frequency. In the end, thermal control can avoid over-high temperatures for the entire JH7110 SoC platform.

1.1. Block Diagram

The following figure shows the block digram of the thermal system interface.

Figure 1-1 Block Diagram



1.2. Source Code Structure

The following code block shows the source code structure of the thermal control system.

```
kernel/
|-- drivers/thermal/thermal_core.c //thermal sensor core code
|-- drivers/thermal/cpu_cooling.c //thermal cpu cooling code
|-- drivers/thermal/gov_step_wise.c //step wise governor code
```

2. Configuration

This chapter includes the following parts:

2.1. Device Tree Configuration

The device tree stores all the configuration modules for a board.

The device tree file of the JH7110 EVB is in the following path:

`kernel/linux/arch/riscv/boot/dts/starfive/jh7110.dtsi.`

The following list provides more information on the device tree for the thermal system.

2.1.1. of thermal

By describing the relationships of the modules among thermal zone, thermal sensor, trip point, and cooling device in the DTS file following a standard format, the of-thermal module will automatically register the DTS information and maintain the logic relations to reduce code complexity.

The following code block provides a thermal node of JH7110.

```
thermal-zones {
            cpu-thermal {
                polling-delay-passive = <250>;
                polling-delay = <15000>;

                thermal-sensors = <&sfctemp>;

                trips {
                    cpu_alert0: cpu_alert0 {
                        /* milliCelsius */
                        temperature = <85000>;
                        hysteresis = <2000>;
                        type = "passive";
                    };

                    cpu_crit: cpu_crit {
                        /* milliCelsius */
                        temperature = <100000>;
                        hysteresis = <2000>;
                        type = "critical";
                    };
```

```
                };

                cooling-maps {
                    map0 {
                        trip = <&cpu_alert0>;
                        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>;
                    };
                };
            };
        };
```

2.1.2. Thermal driver

The following code block provides an example of the thermal driver.

```
sfctemp: tmon@120e0000  {
            compatible = "starfive,jh7110-temp";
            reg = <0x0 0x120e0000 0x0 0x10000>;
            interrupts = <81>;
            clocks = <&clkgen JH7110_TEMP_SENSOR_CLK_TEMP>,
                 <&clkgen JH7110_TEMP_SENSOR_CLK_APB>;
            clock-names = "sense", "bus";
            resets = <&rstgen RSTN_U0_TEMP_SENSOR_TEMP>,
                 <&rstgen RSTN_U0_TEMP_SENSOR_APB>;
            reset-names = "sense", "bus";
            #thermal-sensor-cells = <0>;
            status = "disabled";
        };
```

The following list provides explanations for the parameters included in the above code block.

- **compatible**: Compatibility information, used to associate the driver and its target device.
- **reg**: Register base address "`0x120e0000`" and range "`0x10000`".
- **interrupts**: Hardware interrupt ID.
- **clocks**: The clocks used by the thermal sensor module.
- **clock-names**: The names of the above clocks.
- **resets**: The reset signals used by the thermal sensor module.

- **reset-names**: The names of the above reset signals.
- **status**: The work status of the thermal sensor module. To enable the module, set this bit as "okay" or to disable the module, set this bit as "disabled".

2.1.3. Cooling device – CPU device

The following code block provides an example of the cooling device - CPU device.

```
cpu0: cpu@0 {
            compatible = "sifive,s7", "riscv";
            reg = <0>;
            d-cache-block-size = <64>;
            d-cache-sets = <64>;
            d-cache-size = <8192>;
            d-tlb-sets = <1>;
            d-tlb-size = <40>;
            device_type = "cpu";
            i-cache-block-size = <64>;
            i-cache-sets = <64>;
            i-cache-size = <16384>;
            i-tlb-sets = <1>;
            i-tlb-size = <40>;
            mmu-type = "riscv,sv39";
            next-level-cache = <&cachectrl>;
            riscv,isa = "rv64imac_zba_zbb";
            tlb-split;
            #cooling-cells = <2>;
            status = "disabled";

            cpu0intctrl: interrupt-controller {
                #interrupt-cells = <1>;
                compatible = "riscv,cpu-intc";
                interrupt-controller;
            };
        };
```

The following list provides explanations for the parameters included in the above code block.

- **compatible**: Compatibility information, used to associate the driver and its target device.
- **reg**: Register base address .
- **status**: The work status of the thermal sensor module. To enable the module, set this bit as "okay" or to disable the module, set this bit as "disabled".

2.1.4. Cooling device – CPU OP table

The following code block provides an example of the cooling device - CPU OP table.

```
cluster0_opp: opp-table-0 {
            compatible = "operating-points-v2";
            opp-shared;
            opp-375000000 {
                    opp-hz = /bits/ 64 <375000000>;
                    opp-microvolt = <800000>;
            };
            opp-500000000 {
                    opp-hz = /bits/ 64 <500000000>;
                    opp-microvolt = <800000>;
            };
            opp-750000000 {
                    opp-hz = /bits/ 64 <750000000>;
                    opp-microvolt = <800000>;
                    opp-suspend;
            };
            opp-1500000000 {
                    opp-hz = /bits/ 64 <1500000000>;
                    opp-microvolt = <1040000>;
            };

            /* CPU opp table for 1.25GHz */
            opp-312500000 {
                    opp-hz = /bits/ 64 <312500000>;
                    opp-microvolt = <800000>;
            };
            opp-417000000 {
                    opp-hz = /bits/ 64 <417000000>;
                    opp-microvolt = <800000>;
            };
            opp-625000000 {
                    opp-hz = /bits/ 64 <625000000>;
                    opp-microvolt = <800000>;
                    opp-suspend;
            };
            opp-1250000000 {
                    opp-hz = /bits/ 64 <1250000000>;
                    opp-microvolt = <1000000>;
            };
    };
```

The following list provides explanations for the parameters included in the above code block.

- **compatible**: Compatibility information, used to associate the driver and its target device.
- **opp-hz**: Frequency.
- **opp-microvolt**: The voltage corresponding to frequency.

2.2. Kernel Menu Configuration

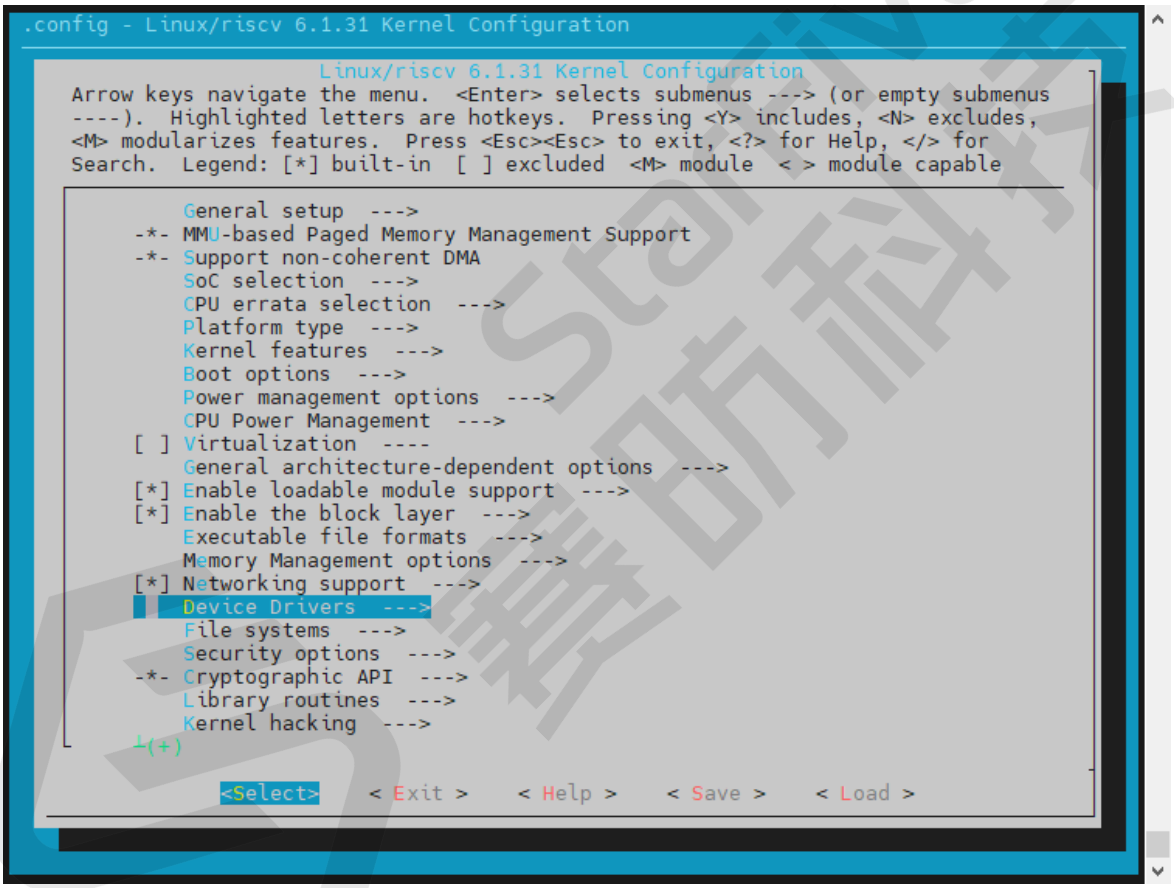
Follow the steps below to configure the kernel menu settings for the thermal sensor.

1. Under the root directory of `freelight-u-sdk`, type the following command to enter the kernel menu configuration GUI.

```
cd freelight-u-sdk
make linux-menuconfig
```

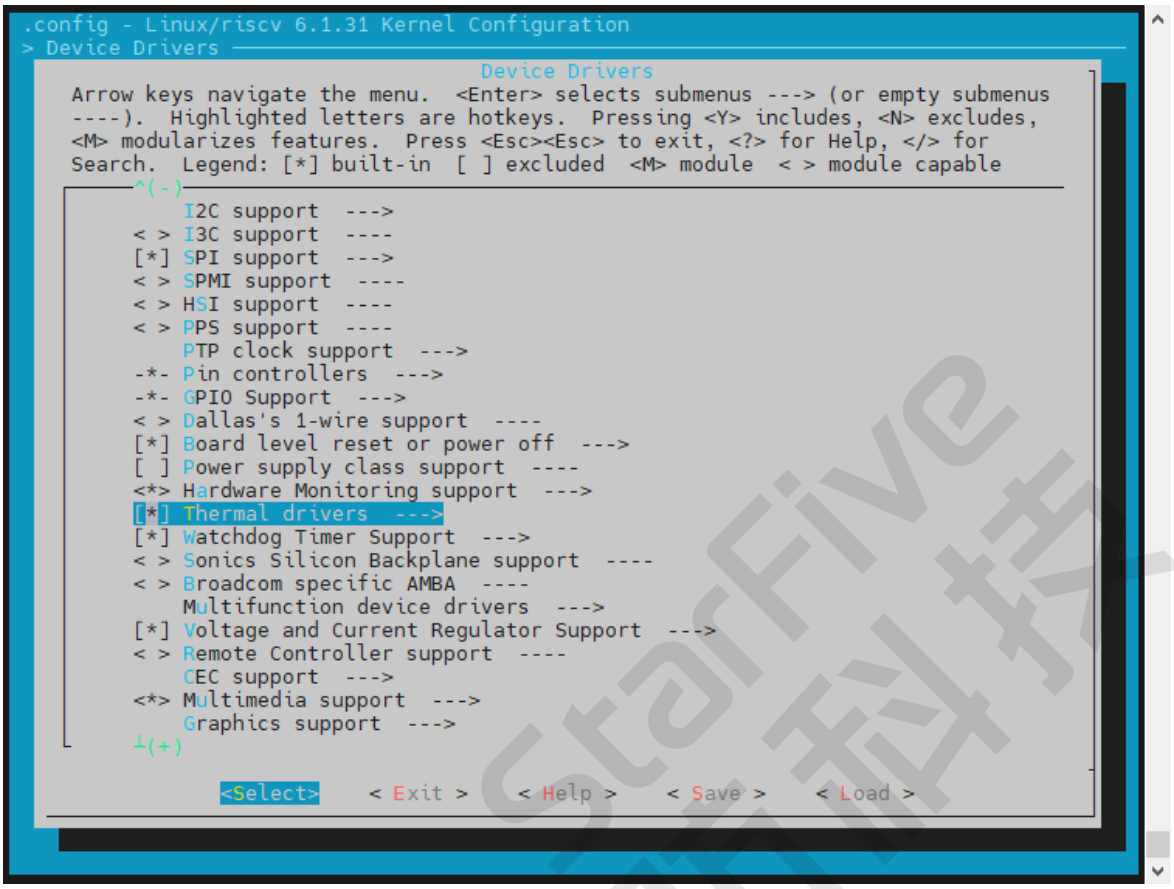
2. Enter the **Device Drivers** menu.

Figure 2-1 Device Drivers



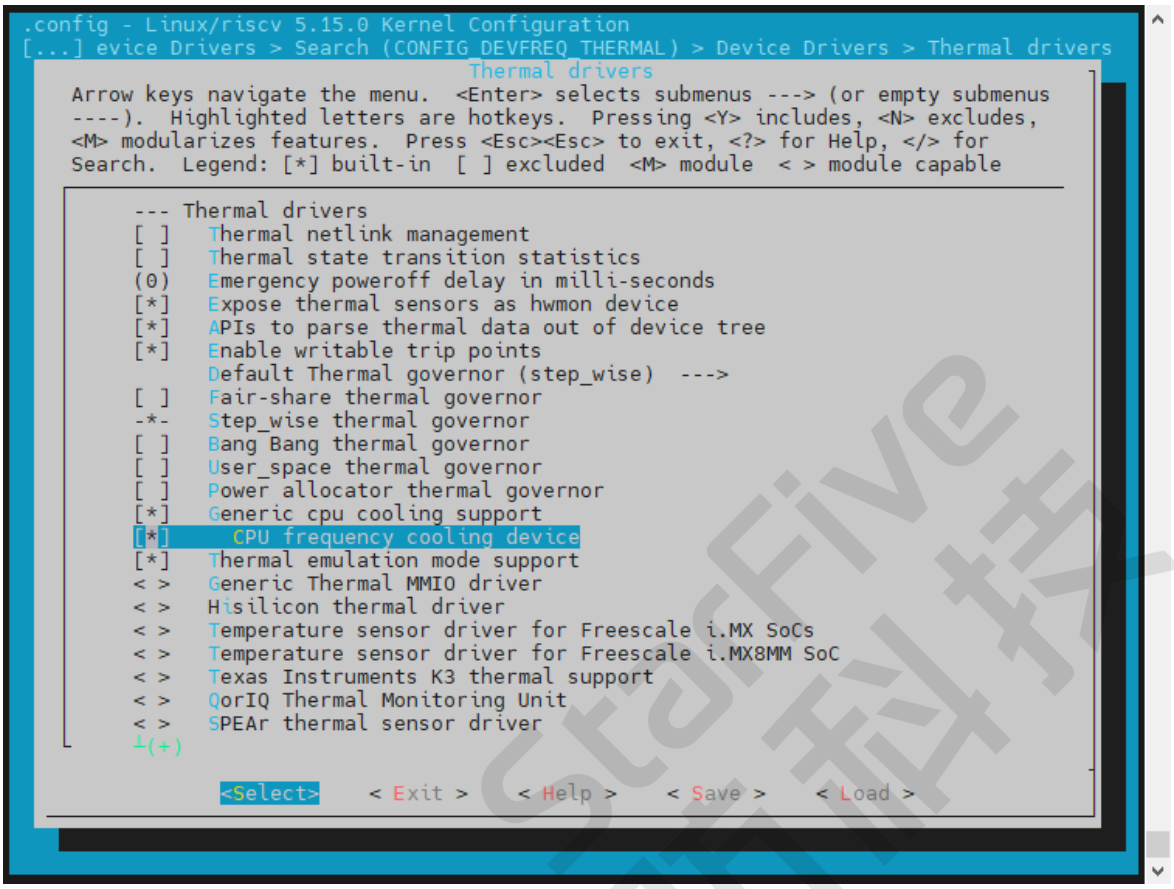
3. Enter the **Thermal Drivers** menu.

Figure 2-2 Thermal Drivers



4. In the **Thermal Drivers** menu, select the **CPU Frequency Cooling Device** option.

Figure 2-3 CPU Frequency Cooling Device



The following list provides explanations for the parameters included in the above menu.

- **Default Thermal governor (step_wise)**: Thermal governor select `step_wise` by default.
- **Expose thermal sensors as hwmon device**: Generic Thermal Management hwmon support.
- **APIs to parse thermal data out of device tree**: Generic Thermal Management device tree support.
- **Enable writable trip points**: Support modifying trip points in user space.
- **Thermal emulation mode support**: Support thermal emulation temperature function.
- **Generic cpu cooling support**: Open generic CPU cooling.
- **CPU frequency cooling device**: Supports generic CPU cooling mechanism through frequency reduction.

3. Use Example

This chapter includes the following parts:

3.1. Check Sensor Temperature

Follow the example below to check the sensor temperature of **thermal_zone0**.

- Check the type of **thermal_zone0**:

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

- Check the temperature of **thermal_zone0**:

```
#cat sys/class/thermal/thermal_zone0/temp 36000
```

Note:
The unit of temperature is mC, 36000 mC is 36 ℃。

3.2. Temperature Simulation

Temperature simulation test can be used to validate your temperature management policy. Follow the example below to use the thermal sensor to simulate temperature.

- Set the simulation temperature for **thermal_zone0**:

```
#echo 80000 > /sys/class/thermal/thermal_zone0/emul_temp
```

- Turn off the simulation temperature function of **thermal_zone0**:

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

3.3. Turning off Thermal Control

Follow the example below to turn off **thermal_zone0**.

- Turn off thermal control strategy:

```
#echo disabled > /sys/class/thermal/thermal_zone0/mode
```

- Unlock all the restrict of cooling device:

```
#echo 0 > /sys/class/thermal/thermal_zone0/cdev*/cur_state
```

3.4. Setting the Over-Heat Threshold to Avoid Automatic Shut-Down

You can avoid automatic shut-down by modifying the over-heat temperature threshold of the `cpu@crit` node to a greater value. Follow the example below:

```
cpu_trips: trips{
    cpu_crit: cpu_crit@0 {
    temperature = <110000>;
    type = "critical";
    hyste-resis = <0>;
};
```

When a PMIC is used, you may need to switch off the PMIC in advance. See the following example:

```
pmu0: pmu@0{
overtemp_shutdown = <1>;   //Overtemp protection, 0 means shutdown
overtemp_value = <145>;    //Overtemp protection temperature
```

3.5. Modifying the Target Temperature

You can modify target of temperature control on `trip-point@1` based on your temperature control policy.

```
cpu_trips: trips{
    cpu_threshold: trip-point@0 {
    temperature = <70000>;
    type = "passive";
    hysteresis = <0>;
};
    cpu_target: trip-point@1 {
    tem-perature = <80000>;
    type = "passive";
    hysteresis = <0>;
};
```