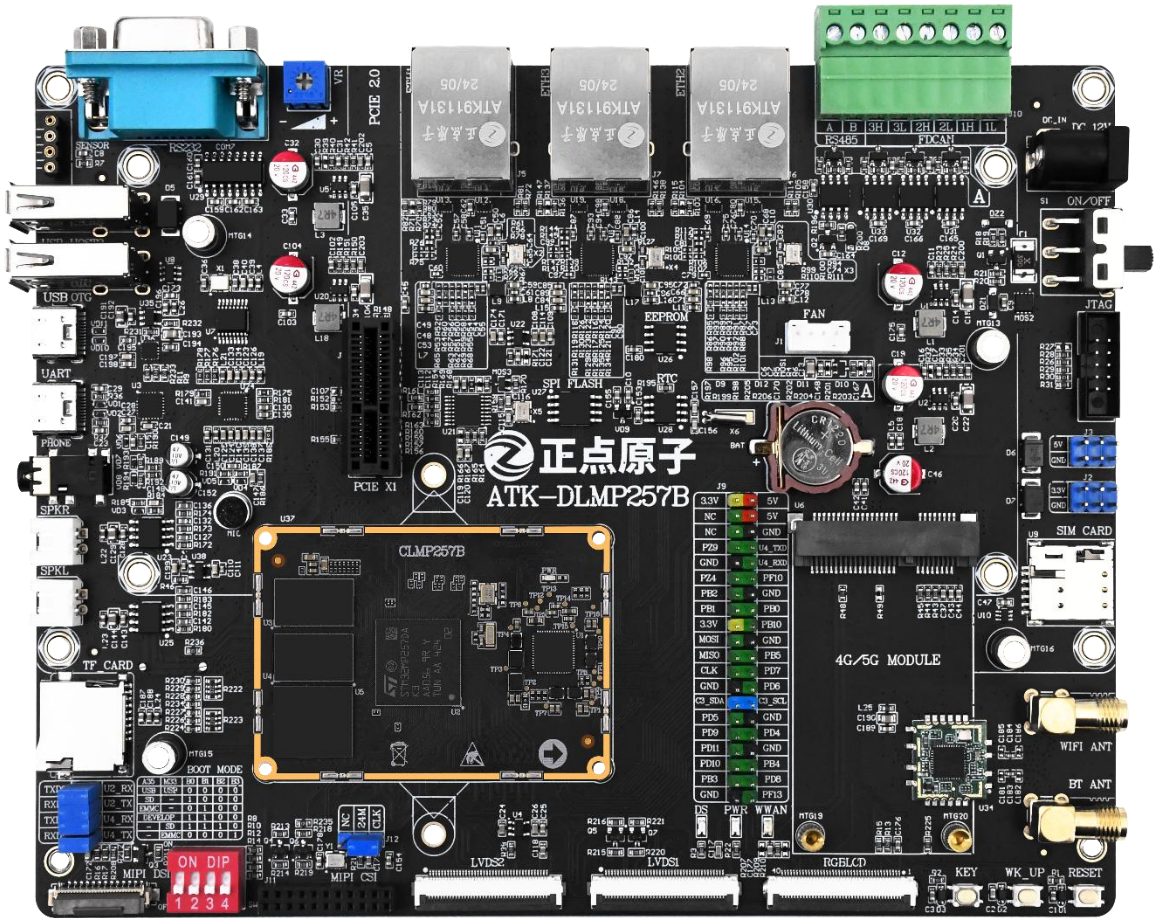
ATK-DLMP257B

Factory system peripheral reuse manual

V1.0



ALIENTEK

1. Shopping:

TMALL: https://zhengdianyuanzi.tmall.com

TAOBAO: https://openedv.taobao.com

2. Download

Address: http://www.openedv.com/docs/index.html

3. FAE

Website : www.alientek.com

Forum : http://www.openedv.com/forum.php

Videos : www.yuanzige.com

Fax : +86 - 20 - 36773971

Phone : +86 - 20 - 38271790

Disclaimer

The product specifications and instructions mentioned in this document are for reference only and subject to update without prior notice; Unless otherwise agreed, this document is intended as a product guide only, and none of the representations made herein constitutes a warranty of any kind. The copyright of this document belongs to Guangzhou Xingyi Electronic Technology Co., LTD. Without the written permission of the company, any unit or individual shall not be used for profit-making purposes in any way of dissemination.

In order to get the latest version of product information, please regularly visit the download center or contact the customer service of Taobao ALIENTEK flagship store. Thank you for your tolerance and support.

Revision History:

Version	Version Update Notes	Responsible person	Proofreading	Date
V1.0	release officially	ALIENTEK	ALIENTEK	2025.04.01

Catalogue

Introduction

This chapter introduces the methods to configure and reuse a variety of peripherals on the STM32MP257 platform, aiming to help developers make full use of the multifunction pins of the processor, improve the system scalability and hardware reuse rate.

First of all, by obtaining and compiling the factory source code, a complete development environment is established to ensure that the subsequent configuration can proceed smoothly. Then, how to configure and reuse peripherals in the device tree is introduced in detail, including the functional definition of pins, multiplexing Settings and device tree node configuration methods. Through the study of this chapter, developers can master a complete method to configure and reuse common peripherals on the STM32MP257 platform, which provides a solid foundation for the function expansion and hardware optimization of embedded systems.

Chapter 1. Factory source code acquisition and compilation

Before performing all the operations in this document, please obtain the factory source code in accordance with the contents of "\ Development board CD A- Basic information \10_user_manual \" [ALIENTEK] ATK-DLMP257B Factory System Source Code Use Guide V1.0"document, to ensure that the current virtual machine environment can successfully compile the factory source code.

Chapter 2. Multiplexing multiple GPIOs

For the STM32MP257 core board, most of the pins support multiplexing as GPIO functions. In STM32MP2, when an IO pin is used for GPIO functionality, there is no need to create a pinctrl node in the device tree.

In order to reuse the GPIO function of a pin, it is first necessary to verify whether the pin is configured as another peripheral function in the kernel or U-Boot's device tree. If it is, disable the relevant device tree node or comment out the corresponding pin reuse code. Once the pin is released, the high and low levels can be set using the system GPIO command. After the setup is completed, the level change can be tested in combination with the multimeter to confirm whether the pin is working properly.

For detailed operation tutorials, please refer to the development board data path: "Development board CD A- Basic Information \9_tutorials \" [ALIENTEK] ATK-DLMP257 Embedded Linux C Application Programming Guide "document" Chapter 16 GPIO application Programming ".

Chapter 3. Multiplexing UART/USART

3.1 Confirm that the pin belongs to the UART

The STM32MP257 supports up to 9 UART/USART channels and one LPUART channel. When configuring multiplexed UART, special attention must be paid to whether the pin multiplexing defined in the device tree conflicts with other peripherals such as I2C, SPI, etc. If you need to configure separate pins for different UART/USART, you should check the pin multiplexing mapping table of the target chip to confirm that the desired configuration is supported. In order to ensure that the target pin can be reused for UART/USART function, it is recommended to refer to the "ATK-CLMP257B Core Board Interface Data Sheet" for detailed query.

引脚座子	管脚号	核心板管脚名	出厂系统默认配置/开发板接口	GPIO编号	功能描述	管脚类型	电平域电压	AF0	AF1	AF2
	1	GND			接地					
	2	GND			接地					
	3	GND			接地					
	4	GND			接地					
	5	GND			接地					
	6									
	7	VDD_SDOUT	TF Card		核心板电源输出接口	电源	3.3V/1.8V			
	8									
	9	VREF+_1V8	CORE BOARD		核心板电源输出接口	电源	1.8V			
	10									
	11	1V8_OUT	CORE BOARD		核心板电源输出接口	电源	1.8V			
	12									
	13	VBAT	CORE BOARD		纽扣电池电源输入引脚	电源	3.3V			
	14									
	15	KEY0	KEY0	PM5	按键信号	输入/输出	3.3V	-	-	-
	16	OTG_PWR_CTRL	USB OTG	PZ5	USB OTG电源控制信号	输入/输出	3.3V	-	MCO1	LPTIM3_ETR
	17	USB_INT		PZ6	USB OTG中断信号	输入/输出	3.3V	DBTRGI	DBTRGO	-
	18	RTC_INT	RTC	PZ7	实时时钟的中断信号	输入/输出	3.3V	-	-	-
	19	1WIRE_DQ	TEMP&HUMI SENSOR	PZ8	低速设备串行通信信号	输入/输出	3.3V	-	-	LPTIM3_IN1
	20									
	21	GND			接地					
	22	ETH2_CLK125	10/100/1000M ENET2	PF8	千兆网口2_外部时钟信号线	输入/输出	1.8V	-	RTC_REFIN	-
	23	ETH2_MDIO		PC5	千兆网口2_MAC管理接口数据线	输入/输出	3.3V	-	-	SPDIFRX1_IN1
	24	ETH2_MDC		PC6	千兆网口2_MAC管理接口时钟线	输入/输出	3.3V	-	RTC_REFIN	SPDIFRX1_IN0
	25	ETH2_MDINT		PF5	千兆网口2_MAC管理接口中断线	输入/输出	3.3V	-	-	SPI6_SCK
	26	GND			接地					
	27	ETH2_TXD0		PC7	千兆网口2_发送端数据线0	输入/输出	1.8V	-	-	-
	28	ETH2_TXD1		PC8	千兆网口2_发送端数据线1	输入/输出	1.8V	-	LPTIM1_ETR	-
	29	ETH2_TXD2		PC9	千兆网口2_发送端数据线2	输入/输出	1.8V	-	MCO1	SPI3_MISO/ I2S3_SDI
	30	ETH2_TXD3		PC10	千兆网口2_发送端数据线3	输入/输出	1.8V	-	-	SPI3_MOSI/ I2S3_SDO
	31	ETH2_TX_CTL		PC4	千兆网口2_发送端时钟信号	输入/输出	1.8V	-	-	-
	32	ETH2_GTX_CLK		PF7	千兆网口2_发送端使能信号	输入/输出	1.8V	-	-	SPDIFRX1_IN1
	33	GND			接地					
	34	ETH2_RXD0		[illegible]	千兆网口2_接收端数据线0	输入/输出	1.8V	-	LPTIM1_IN1	-

Figure 1 ATK-CLMP257B core board interface data sheet

3.2 Configure the UART master node in the device tree

The device tree file is located at stm32mp251.dtsi, which needs to supplement the UART/USART master node content in stm32mp257d-atk-ddr-1GB.dts or stm32mp257d-atk-ddr-2GB.dts. The specific choice depends on the DDR size of the purchased core board.

Example device tree structure:

stm32mp251.dtsi

```
usart1: serial@40330000 {
        …
};


…


uart9: serial@402c0000 {
        …
};


…


lpuart1: serial@46030000 {
        …
```

```
};
```

3.3 Configuring UART nodes

When configuring the UART device tree node, you can refer to the device tree node of USART2 as an example for configuration. It is particularly important to note that in the default factory sources of the kernel, usart1 and usart2 are set as debug serial ports for serial1 and serial0, respectively. In order to ensure the stability of debugging functions, it is not recommended to modify the default configuration of these two serial ports. For other UART configuration methods, you can directly refer to the device tree of USART2 to modify.

stm32mp257d-atk-ddr-1GB.dts or stm32mp257d-atk-ddr-2GB.dts

```
…

aliases {
        …
        serial0 = &usart2;
        // By default, USART2 is defined as serial0 and is not recommended to be changed
        …
};

…

&usart2 {
        pinctrl-names = "default", "idle", "sleep";     // Three pin reuse states are defined
        pinctrl-0 = <&usart2_pins_a>;                   // Pin configuration in default mode
        pinctrl-1 = <&usart2_idle_pins_a>;              // Pin configuration in idle mode
        pinctrl-2 = <&usart2_sleep_pins_a>;             // Pin configuration in sleep mode
        /delete-property/dmas;                          // Remove the dmas attribute
        /delete-property/dma-names;                     // Remove the dma-names attribute
        status = "okay";                                // Enable USART2
};

…
```

Key configuration explained:

1. **serial0**

   Define USART2 as serial0 for the device /dev/ttySTM0.

```
root@ATK-DLMP257:~# ls /dev/ttySTM0
/dev/ttySTM0
root@ATK-DLMP257:~#
```

Figure 2 The serial port pair should be the device file

2. **pinctrl-names = "default", "idle", "sleep";**

   Three pin reuse states are defined: default, idle, and sleep mode.

3. **pinctrl-0、pinctrl-1、pinctrl-2**

   They correspond to the pin configuration in different modes. The pin configuration is referenced by <&usart2_pins_a>, <&usart2_idle_pins_a>, and <&usart2_sleep_pins_a>.

4. **/delete-property/dmas 和 /delete-property/dma-names**

   Remove the dmas and dma-names attributes to indicate that this configuration does not use DMA. Usually DMA is used to speed up data transfer, if not needed, these attributes can be removed to simplify the configuration.

5. **status = "okay"**

   USART2 is enabled to indicate that the peripheral is active and ready for use.

3.4 Modify the pin reuse configuration

Add the pinctrl description of the UART pin to the stm32mp25-pinctrl-atk.dtsi file and set its multiplexing function to AFx. The specific AFx value needs to be checked according to the STM32MP257D data sheet or "[ALIENTEK] ATK-CLMP257B Core board Interface Data Sheet" to confirm the correct reuse function number.

stm32mp25-pinctrl-atk.dtsi

```
usart2_pins_a: usart2-0 {
	pins1 {
		pinmux = <STM32_PINMUX('A', 4, AF6)>; /* USART2_TX */
		bias-disable;
		drive-push-pull;
		slew-rate = <0>;
	};
	pins2 {
		pinmux = <STM32_PINMUX('A', 8, AF8)>; /* USART2_RX */
		bias-pull-up;
	};
};

usart2_idle_pins_a: usart2-idle-0 {
	pins1 {
		pinmux = <STM32_PINMUX('A', 4, ANALOG)>; /* USART2_TX */
	};
	pins2 {
		pinmux = <STM32_PINMUX('A', 8, AF8)>; /* USART2_RX */
		bias-pull-up;
	};
};

usart2_sleep_pins_a: usart2-sleep-0 {
	pins {
		pinmux = <STM32_PINMUX('A', 4, ANALOG)>, /* USART2_TX */
			 <STM32_PINMUX('A', 8, ANALOG)>; /* USART2_RX */
```

```
        };
    };
```

If we need to reuse other pins as UART, just refer to the above writing of USART2 to modify. The number of the ANALOG mode can be kept the same as the example, except the AFx function number needs to be adjusted according to the definition of the specific pin.

3.5 UART function debugging

After the core board is started, enter the /dev directory of the file system, and you can see the device files corresponding to different serial ports, which are ttySTM0, ttySTM1 and ttySTM2. You can use the serial port function directly from these device files and use tools such as minicom to debug.

```
root@ATK-DLMP257:~# ls /dev/ttySTM*
/dev/ttySTM0  /dev/ttySTM1  /dev/ttySTM2
root@ATK-DLMP257:~# _
```

Figure 3 Serial debugging device file

Chapter 4. Multiplexing ADC

STM32MP257 supports ADC1, ADC2, and ADC3, providing a total of 23 channels. Although the ADC of STM32MP257 supports the acquisition of differential signals to improve accuracy, it is usually recommended to select an external ADC chip in order to obtain higher ADC data accuracy. If additional ADC acquisition pins using STM32MP257 are required, they can be configured as follows.

4.1 Confirm that the pin belongs to the ADC controller

First, confirm whether the pin used belongs to the ADC1, ADC2, or ADC3 controller. The STM32MP257 data sheet should be consulted, for example:

- PC5 pin can be used as:
  - ADC1_INP10: The 10th forward differential signal input channel of the ADC1 controller
  - ADC2_INP10: The 10th forward differential signal input channel of the ADC2 controller
  - ADC3_INP10: The 10th forward differential signal input channel of the ADC3 controller
- The PC3 pin can be used as:
  - ADC1_INP12: The 12th forward differential signal input channel of the ADC1 controller
  - Other features are similar

Pin number			Pin name (function after reset)	Pin type	I/O structure	Notes	Alternate functions	Additional functions
VFBGA361	VFBGA424	TFBGA436						
V9	AB14	AA12	PC0	I/O	TT	(1)	LPTIM1_CH1, SPI6_SCK, SAI3_MCLK_B, USART6_TX, DCMI_D0/PSSI_D0/DCMIPP_D0, ETH2_MII_RX_CLK/ ETH2_RMII_REF_CLK, ETH1_MII_TX_CLK, ETH1_RGMII_GTX_CLK, LCD_G7, EVENTOUT	-
V10	AG14	V11	PC1	I/O	TT_f	(1)	SPI3_MOSI/I2S3_SDO, USART2_TX, I2C7_SCL, ETH1_MII_TXD1/ETH1_RGMII_TXD1/ETH1_RMII_TXD1, EVENTOUT	-
T9	AE12	Y11	PC2	I/O	TT	(1)	SPI8_MOSI, LPTIM2_IN1, SAI4_MCLK_B, MDF1_SDI3, USART2_RTS/USART2_DE, ETH1_MII_RXD1/ ETH1_RGMII_RXD1/ETH1_RMII_RXD1, EVENTOUT	-
U4	AA9	W9	PC3	I/O	TT_a	(5)	LPTIM1_IN2, SPI3_NSS/I2S3_WS, SPI6_RDY, USART6_RTS/ USART6_DE, FDCAN2_TX, ETH2_MII_RX_DV/ ETH2_RGMII_RX_CTL/ETH2_RMII_CRS_DV, ETH1_MII_RX_ER, LCD_G6, DCMI_D3/PSSI_D3/DCMIPP_D3, EVENTOUT	ADC1_INP12, ADC1_INN10, ADC2_INP12, ADC2_INN10, ADC3_INP12, ADC3_INN10, TAMP_IN3
V3	AC9	V9	PC4	I/O	TT	(5)	SPI6_MISO, SAI3_FS_B, ETH2_MII_TX_EN/ ETH2_RGMII_TX_CTL/ETH2_RMII_TX_EN/ ETH1_RGMII_CLK125, LCD_R0, EVENTOUT	TAMP_IN1
T7	AD10	U9	PC5	I/O	TT_af	(5)	SPDIFRX1_IN1, MDF1_SDI1, TIM8_CH1N, I2C4_SDA, ETH2_MDIO, ETH1_MII_COL, FMC_A25, ETH1_PPS_OUT, LCD_DE, EVENTOUT	ADC1_INP10, ADC2_INP10, ADC3_INP10, TAMP_IN6

Figure 4 STM32MP257 data sheet

4.2 Configure the ADC master node in the device tree

The device tree file is located in stm32mp251.dtsi, which needs to supplement the ADC master node content in stm32mp257d-atk-ddr-1GB.dts or stm32mp257d-atk-ddr-2GB.dts, depending on the DDR size of the core board purchased.

Example device tree structure:

stm32mp251.dtsi

```
adc_12: adc@404e0000 {
        ...
```

```
        adc1: adc@0 {
                ...
        };
        adc2: adc@100 {
                ...
        };
};

adc_3: adc@404f0000 {
        ...
        adc3: adc@0 {
                ...
        };
};
```

4.3 Configure the ADC channel

Taking channel 15 of ADC1 (ADC1_INP15) as an example, the ADC node in the device tree is configured as follows:

```
stm32mp257d-atk-ddr-1GB.dts 或 stm32mp257d-atk-ddr-2GB.dts

&adc_12 {
        pinctrl-names = "default";
        pinctrl-0 = <&adc1_in15_pins_a>;
        vdda-supply = <&vdda_1v8>;
        vref-supply = <&vddref_1v8>;
        status = "okay";

        adc1: adc@0 {
                #address-cells = <1>;
                #size-cells = <0>;
                status = "okay";

                channel@15 {
                        reg = <15>;                             /* Channel 15 of ADC1 is used    */
                        st,min-sample-time-ns = <10000>;        /* The minimum sampling time is 10μs */
                };
        };
};
```

Key configuration explained:

1. **pinctrl-0 = <&adc1_in15_pins_a>;**
   Reference the pin configuration named adc1 in15 pins a.
2. **adc1: adc@0**
   Define the primary configuration node of ADC1, and @0 denotes the address.
3. **channel@15**

Configure channel 15 of ADC1:

- reg = <15>; Specifies that channel 15 is used.
- st,min-sample-time-ns = <10000>; The minimum sampling time was set to 10 microseconds.

4.4 Modify the pin reuse configuration

In the stm32mp25-pinctrl-atk.dtsi file, add the pinctrl description of the ADC pin and set its multiplexing function to "ANALOG".

stm32mp25-pinctrl-atk.dtsi

```
adc1_in15_pins_a: adc1-in15 {
	pins {
		pinmux = <STM32_PINMUX('B', 15, ANALOG)>;
	};
};
```

4.5 Update the device tree to support multiple ADC channels

When configuring multiple ADC channels, the master tree file should be updated to include the configuration of all relevant pins. For example, PC3 and PB15 are multiplexed as ADC1_INP12 and ADC1_INP15, PC6 is multiplexed as ADC2_INP6, and PC5 is multiplexed as ADC3_INP10.

The steps are as follows:.

1, modify the pin multiplexing configuration

- Check if there are other device tree nodes using the same pin.
- If a conflict is found, the associated device tree node must be annotated or disabled to avoid pin reuse conflicts.

stm32mp25-pinctrl-atk.dtsi

```
	adc1_in15_pins_a: adc1-in15 {
		pins {
			pinmux = <STM32_PINMUX('B', 15, ANALOG)>;
		};
	};

	adc1_in12_pins_a: adc1-in12 {
		pins {
			pinmux = <STM32_PINMUX('C', 3, ANALOG)>;
		};
	};

	adc2_in6_pins_a: adc2-in6 {
		pins {
			pinmux = <STM32_PINMUX('C', 6, ANALOG)>;
		};
	};
```

```
	adc3_in10_pins_a: adc3-in10 {
		pins {
			pinmux = <STM32_PINMUX('C', 5, ANALOG)>;
		};
	};
```

2、Update the kernel device tree

- Configure the channel Settings under the ADC master node to ensure that all required ADC channels have been configured correctly.
- Verify that the pin and register addresses for each ADC channel are set correctly.

stm32mp257d-atk-ddr-1GB.dts 或 stm32mp257d-atk-ddr-2GB.dts

```
&adc_12 {
	pinctrl-names = "default";
	pinctrl-0 = <&adc1_in12_pins_a>, <&adc1_in15_pins_a>, <&adc2_in6_pins_a>;
	vdda-supply = <&vdda_1v8>;
	vref-supply = <&vddref_1v8>;
	status = "okay";

	adc1: adc@0 {
		#address-cells = <1>;
		#size-cells = <0>;
		status = "okay";

		channel@12 {
			reg = <12>;
			st,min-sample-time-ns = <10000>;
		};

		channel@15 {
			reg = <15>;
			st,min-sample-time-ns = <10000>;
		};
	};

	adc2: adc@100 {
		#address-cells = <1>;
		#size-cells = <0>;
		status = "okay";

		channel@6 {
			reg = <6>;
			st,min-sample-time-ns = <10000>;
		};
```

```
        };
};

&adc_3 {
        pinctrl-names = "default";
        pinctrl-0 = <&adc3_in10_pins_a>;
        vdda-supply = <&vdda_1v8>;
        vref-supply = <&vddref_1v8>;
        status = "okay";

        adc3: adc@0 {
                #address-cells = <1>;
                #size-cells = <0>;
                status = "okay";

                channel@10 {
                        reg = <10>;
                        st,min-sample-time-ns = <10000>;
                };
        };
```

4.6 ADC function test

After the core board is started, enter the /sys/bus/iio/devices directory of the file system, and you can see the devices mounted under different ADC master controllers, namely iio:device0, iio:device1, and iio:device2. It should be noted that the ATK-DLMP257B backboard only supports the test of on-board ADC input (PB15 pin). If other ADC pins need to be tested, the corresponding hardware test scheme should be designed by itself.

```
cd /sys/bus/iio/devices
ls -l
```

```
root@ATK-DLMP257:/sys/bus/iio/devices# ls -l
total 0
lrwxrwxrwx 1 root root 0 Jan  1  2000 iio:device0 -> ../../../devices/platform/soc@0/42080000.rifsc/404e0000.adc/404e0000.adc:adc@0/iio:device0
lrwxrwxrwx 1 root root 0 Jan  1  2000 iio:device1 -> ../../../devices/platform/soc@0/42080000.rifsc/404e0000.adc/404e0000.adc:adc@100/iio:device1
lrwxrwxrwx 1 root root 0 Jan  1  2000 iio:device2 -> ../../../devices/platform/soc@0/42080000.rifsc/404f0000.adc/404f0000.adc:adc@0/iio:device2
root@ATK-DLMP257:/sys/bus/iio/devices#
```

Figure 5 List of ADC controller devices

It can be seen from Figure 4.6.1 that the corresponding ADC controller register addresses of each device. In the stm32mp251.dtsi device tree file, the register addresses of ADC1 to ADC3 master controllers are indicated, which is convenient to quickly locate and measure the device files under different ADC master controller channels.

ADC controller Controller	ADC controller Controller register address
ADC12	adc@404e0000
ADC3	adc@404f0000

Once the configuration is complete and restarted, the ADC data can be read using the following command:

```
cat /sys/bus/iio/devices/iio:device0/in_voltage12_raw
cat /sys/bus/iio/devices/iio:device0/in_voltage15_raw
cat /sys/bus/iio/devices/iio:device1/in_voltage6_raw
cat /sys/bus/iio/devices/iio:device2/in_voltage10_raw
```

```
root@ATK-DLMP257:/sys/bus/iio/devices# ls
iio:device0  iio:device1  iio:device2
root@ATK-DLMP257:/sys/bus/iio/devices# cat /sys/bus/iio/devices/iio:device0/in_voltage12_raw
1361
root@ATK-DLMP257:/sys/bus/iio/devices# cat /sys/bus/iio/devices/iio:device0/in_voltage15_raw
4071
root@ATK-DLMP257:/sys/bus/iio/devices# cat /sys/bus/iio/devices/iio:device1/in_voltage6_raw
819
root@ATK-DLMP257:/sys/bus/iio/devices# cat /sys/bus/iio/devices/iio:device2/in_voltage10_raw
4095
root@ATK-DLMP257:/sys/bus/iio/devices#
```

Figure 6 Read voltage acquisition analog

Or use the watch command to monitor all ADC channels in real time:

```
watch -n 1 "echo ADC1_INP12: \$(cat /sys/bus/iio/devices/iio:device0/in_voltage12_raw) && echo ADC1_INP15: \$(cat /sys/bus/iio/devices/iio:device0/in_voltage15_raw) && echo ADC2_INP6: \$(cat /sys/bus/iio/devices/iio:device1/in_voltage6_raw) && echo ADC3_INP10: \$(cat /sys/bus/iio/devices/iio:device2/in_voltage10_raw)"
```

```
root@ATK-DLMP257:/sys/bus/iio/devices# watch -n 1 "echo ADC1_INP12: \$(cat /sys/bus/iio/devices/iio:device0/in_voltage12_raw) && echo
 ADC1_INP15: \$(cat /sys/bus/iio/devices/iio:device0/in_voltage15_raw) && echo ADC2_INP6: \$(cat /sys/bus/iio/devices/iio:device1/in_
voltage6_raw) && echo ADC3_INP10: \$(cat /sys/bus/iio/devices/iio:device2/in_voltage10_raw)"
root@ATK-DLMP257:/sys/bus/iio/devices#
```

Figure 7 The watch instruction detects the ADC channel

```
Every 1.0s: echo ADC1_INP12: $(cat /sys/bus/iio/devices/iio:device0/in_voltage12_raw) && ec...  ATK-DLMP257: Fri Mar  3 16:44:10 2023

ADC1_INP12: 1361
ADC1_INP15: 4072
ADC2_INP6: 818
ADC3_INP10: 4095
```

Figure 8 ADC channel monitoring results show