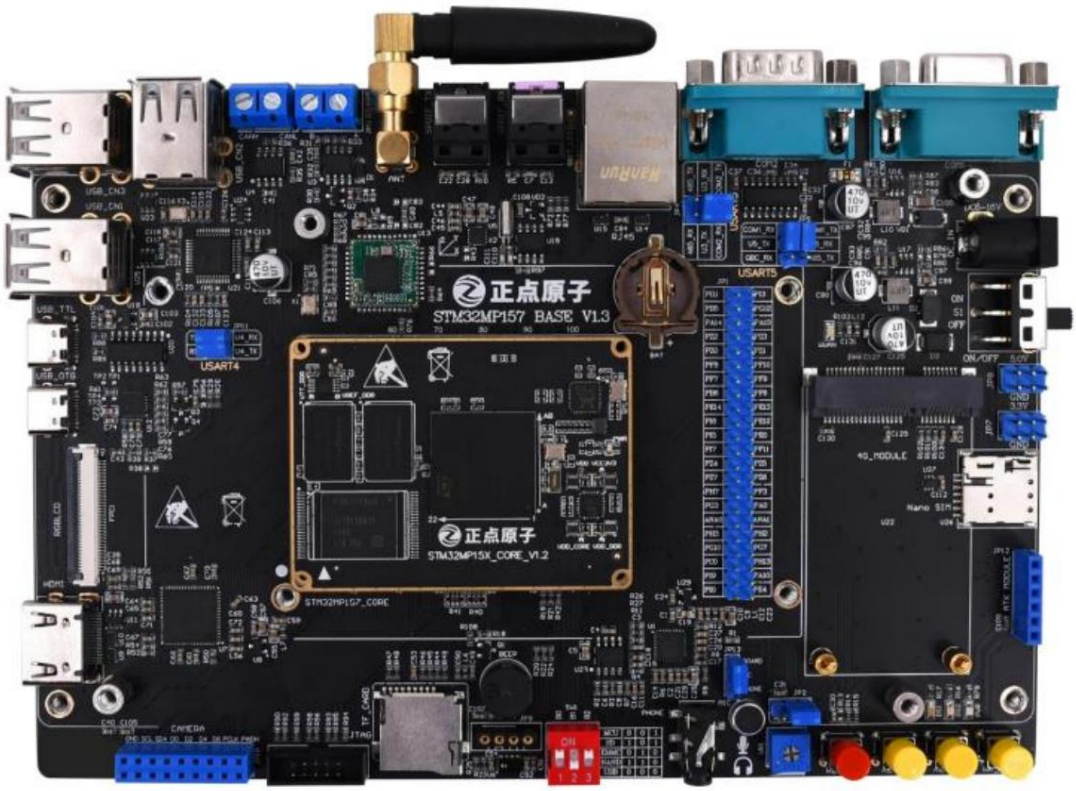
STM32MP157 factory system

peripheral reuse reference document V



正点原子

Original company name: Guangzhou Xingyi Electronic Technology Co., Ltd.

Yuanzige online teaching platform : www.yuanzige.com

Open Source Electronics Network/Forum : http://www.openedv.com/forum.php

Zhengdian Atom Taobao store : https://openedv.taobao.com

Official website of Alientek : www.alientek.com

Punctual Atom B Station video:

https://space.bilibili.com/394620890

Tel: 020-38271790 Fax: 020-36773971

Please follow the Zhengdian Atom public account, we will notify you when the information is released or updated.

Please download the Atom Brother APP and learn from thousands of lecture videos for free, faster and smoother.

扫码关注正点原子公众号

扫码下载“原子哥”APP

Document update notes

Version	Version update notes	Date of publication by responsible person		
V1.0	First draft:	Punctual Atom linux team	Punctual Atom linux team	2023.05.01

Table of contents

Preface

This document is written based on the needs of enterprise users to develop projects. It assumes that the user has already built a development environment and mastered Linux operations. This document is based on the Zhengdian Atom STM32MP157 core board and lists the modification methods of common peripherals. It only provides a reference for modification. Everyone's modification needs are different. Please choose peripherals to modify according to

your own plan and ability. Development environment:

Ubuntu18, vscode Source code master: [Zhengdian Atom] STM32MP157 development board (disk A) - basic information\01, program source code\01, Zhengdian Atomic Linux factory system source code

Hardware: Zhengdian Atom STM32MP157 core board, Zhengdian Atom STM32MP157 development board

Chapter 1 Factory System Source Code and Compilation Tools

1.1 Factory system source code download

Baidu Netdisk download:

Data disk development board data link: https://pan.baidu.com/s/1prWDeLnu2tTepPalQwOxJw Extraction code:
hn9h

Specific path: [On-time Atom] STM32MP157 development board (disk A) - basic information\01, program source code\01, on-time
Atomic Linux factory system source code

1.2 Compilation Tools

Reference documents: [Punctual Atom] STM32MP157 Development Board (Disk A) - Basic Information\10, User Manual\ [Punctual Atom]
STM32MP157 Quick Experience V1.6ÿ

Compilation tool: ostl cross-compilation tool chain

Installation environment: Ubuntu 18.04

Installation method: "ÿPunctual AtomÿSTM32MP157 Quick Experience V1.6" Section 6.2 Installation including compiling Qt application
Cross-compiler toolchain for

1.3 Factory source code compilation

Before you start modifying the source code, you must compile the factory source code at least once to ensure that there are no problems with the compilation environment and that the source code is complete.

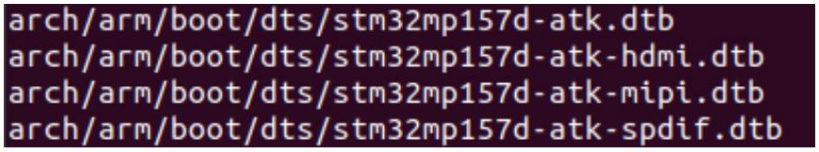
Compilation method:

Compile the factory uboot: "[Pure Atom] STM32MP157 Quick Experience V1.6" Section 6.3 Cross-Qt application
Compilation toolchain

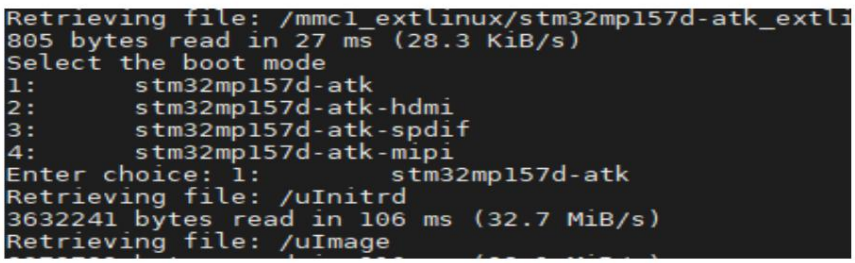
Compile the factory kernel: "ÿPunch AtomÿSTM32MP157 Quick Experience V1.6" Section 6.3 Cross-compilation of Qt applications
Translation tool chain

1.4 Device tree and uboot file description

The Zhengdian Atom Linux development board is compatible with multiple screens. The device tree is divided into multiple types according to the screen parameters, as shown in the following figure:



When starting the program, you need to manually select the corresponding device tree. If you do not select it, the device tree stm32mp157d-atk.dtb is selected by default.



Chapter 2 Factory System Peripheral Resources

2.1 Factory System Resource Table

•: Indicates that source code is provided

ÿ: Indicates providing source code and tutorial materials

ÿ: Indicates that the system and source code are provided, and the factory system can be used directly

Peripheral functions	Factory kernel Source code driver	Tutorial source code drive	Application Development Qt Development		Bare Metal Development
GPIO	ÿ	ÿ	ÿ	ÿ	ÿ
led	ÿ	ÿ	ÿ	ÿ	ÿ
KEY	ÿ	ÿ	ÿ	ÿ	ÿ
LCD		ÿ	ÿ	ÿ	ÿ
BackLight		ÿ	ÿ		ÿ
UART ÿ ÿ ÿ		ÿ	ÿ	ÿ	ÿ
I2C		ÿ	ÿ	ÿ	ÿ
SPI		ÿ	ÿ	ÿ	ÿ
USB	ÿ	ÿ			
FEC(NET)	ÿ	ÿ	ÿ	ÿ	
PWM	ÿ	ÿ	ÿ		ÿ
OV5640	ÿ	•	ÿ	ÿ	
OV2640	ÿ	• •	• •	•	
OV7725(without FIFO)	ÿ			•	
WM8960	ÿ	ÿ	ÿ	ÿ	
RTC	ÿ	ÿ			
WDOG	ÿ		ÿ		
CAN	ÿ	ÿ	ÿ	ÿ	
ADC	ÿ	ÿ	ÿ		
DHT11	ÿ				
DS18B20	ÿ				
RTL8189	ÿ	ÿ			
BEEP	ÿ	ÿ	ÿ	ÿ	ÿ
RS232	ÿ	ÿ			
RS485	ÿ	ÿ			
GPS	ÿ	ÿ			
ME3630	ÿ	ÿ			
EC20	ÿ	ÿ			
HDMI	ÿ	ÿ			ÿ
USB bluetooth ÿ				ÿ	

[Instructions for peripheral cutting]

The default factory system kernel is based on the STM32MP157 development board configuration. If the user needs certain peripheral functions, or needs to modify certain pins for other functions, peripheral tailoring is required. Peripheral tailoring generally involves

kernel tailoring. If some peripherals are used in uboot, uboot also needs to be tailored.

At the same time, the influence of the baseboard resistance and peripheral design should be considered in combination with the baseboard schematic diagram.

Chapter 3 Cutting CAMERA

3.1 Driver and pin definition

Taking the 157 development board as an example, the development board camera interface schematic diagram is as follows:

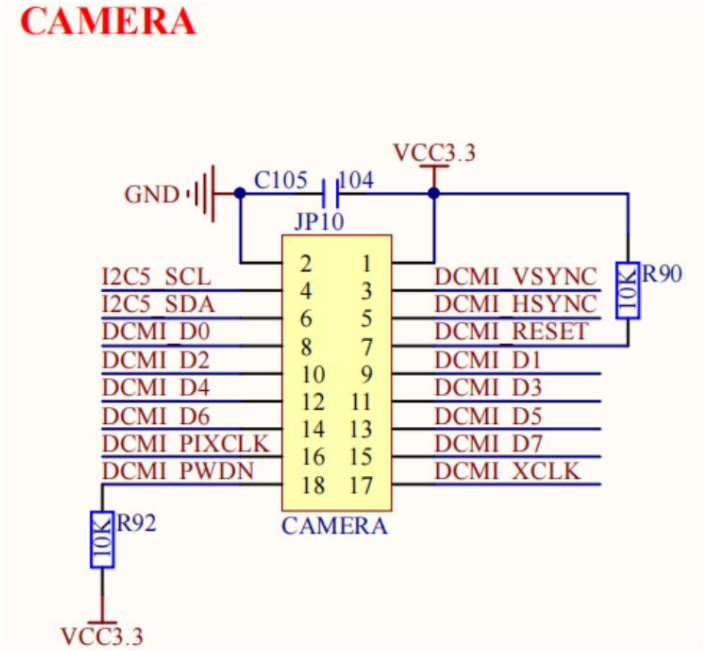


Figure 3.1-1 CAMERA interface schematic diagram

Note the pull-up and pull-down resistors on the schematic diagram, which will affect IO control.

Schematic Pin Name	GPIO	Reusable functions
DCMI_D0	PH9	TIM12_CH2, I2C3_SMBA, DCMI_D0, LCD_R3, EVENTOUT
DCMI_D1	PH10	TIM5_CH1, I2C4_SMBA, I2C1_SMBA, DCMI_D1, LCD_R4, EVENTOUT
DCMI_D2	PH11	TIM5_CH2, I2C4_SCL, I2C1_SCL, DCMI_D2, LCD_R5, EVENTOUT
DCMI_D3	PH12	HDP2, TIM5_CH3, I2C4_SDA, I2C1_SDA, DCMI_D3, LCD_R6, EVENTOUT
DCMI_D4	PH14	TIM8_CH2N, UART4_RX, FDCAN1_RX, DCMI_D4, LCD_G3, EVENTOUT
DCMI_D5	PI4	TIM8_BKIN, SAI2_MCLK_A, DCMI_D5, LCD_B4, EVENTOUT
DCMI_D7	PE6	TRACED2, TIM1_BKIN2, SAI1_D1, TIM15_CH2, SPI4_MOSI, SAI1_SD_A,SDMMC2_D0, SDMMC1_D2, SAI2_MCLK_B,FMC_A22,DCMI_D7,LCD_G1, EVENTOUT

DCMI_D6	PB8	HDP6,TIM16_CH1,TIM4_CH3,DFSDM1_CKIN7, I2C1_SCL,SDMMC1_CKIN,I2C4_SCL,SDMMC2_CKIN, UART4_RX,FDCAN1_RX, SDMMC2_D4, ETH1_GMII_TXD3,ETH1_MII_TXD3, ETH1_RGMII_TXD3,SDMMC1_D4, DCMI_D6, LCD_B6, EVENTOUT
I2C5_SCL	PA11	TIM1_CH4, I2C6_SCL, I2C5_SCL, SPI2_NSS/I2S2_WS, UART4_RX, USART1_CTS/USART1_NSS, FDCAN1_RX, LCD_R4, EVENTOUT
I2C5_SDA	PA12	TIM1_ETR, I2C6_SDA, I2C5_SDA, UART4_TX, USART1_RTS/USART1_DE, SAI2_FS_B, FDCAN1_TX, LCD_R5, EVENTOUT
DCMI_VSYNC	PB7	TIM17_CH1N, TIM4_CH2,I2C1_SDA, I2C4_SDA, USART1_RX, SDMMC2_D1,DFSDM1_CKIN5, FMC_NL, DCMI_VSYNC, EVENTOUT
DCMI_PIXCLK	PA6	TIM1_BKIN, TIM3_CH1, TIM8_BKIN, SAI4_CK2, SPI1_MISO/I2S1_SDI,SPI6_MISO, TIM13_CH1, MDIOS_MDC, SAI4_SCK_A,DCMI_PIXCLK, LCD_G2, EVENTOUT
DCMI_RESET	PE1	LPTIM1_IN2, I2S2_MCK, SAI3_SD_B, UART8_TX, FMC_NBL1, DCMI_D3, EVENTOUT
CSI_HSYNC	PH8	TIM5_ETR, I2C3_SDA, DCMI_HSYNC, LCD_R2, EVENTOUT
DCMI_PWDN	PE11	TIM1_CH2, DFSDM1_CKIN4, SPI4_NSS, USART6_CK, SAI2_SD_B, FMC_AD8/FMC_D8, DCMI_D4, LCD_G3, EVENTOUT

3.2 Kernel cropping CAMERA driver

3.2.1 Modify menuconfig configuration

Enter the factory kernel source directory and open the menuconfig configuration.

```
source /opt/st/stm32mp1/3.1-snapshot/environment-setup-cortexa7t2hf-neon-vfpv4-ostl-linux-gnueabi
make stm32mp1_atk_defconfig
make menuconfig Modify
```

the menuconfig configuration as follows:

```
Device Drivers --->
<M> Multimedia support --->
    I2C Encoders, decoders, sensors and other helper chips --->
      < > OmniVision ov5640 sensor support
```

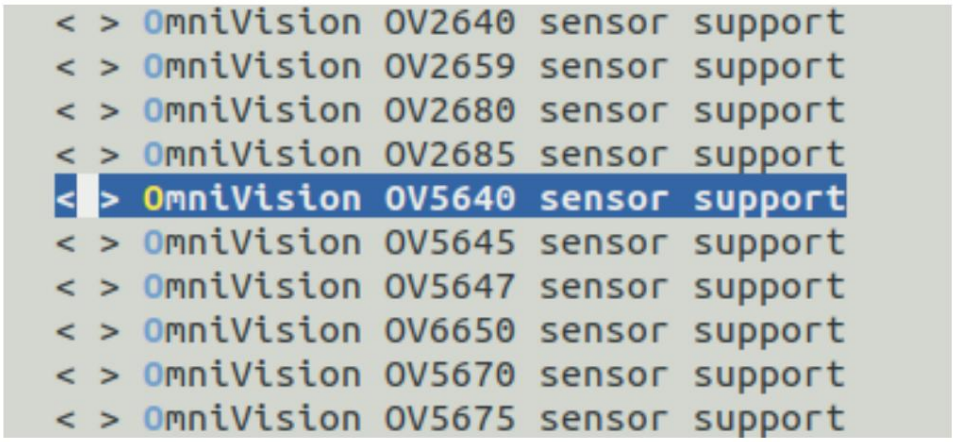


Figure 3.2-1 Cropping camera configuration

Press the ESC key to exit and save the menuconfig configuration

3.2.2 Modify the device tree file

Open ./arch/arm/boot/dts/stm32mp157d-atk.dtsi and search for keywords &dcmi

Comment out the &dcmi node

```
532  &dcmi {
533      status = "okay";
534      pinctrl-names = "default", "sleep";
535      pinctrl-0 = <&dcmi_pins_b>;
536      pinctrl-1 = <&dcmi_sleep_pins_b>;
537
538      port {
539          dcmi_0: endpoint {
540              remote-endpoint = <&ov5640_0>;
541              bus-width = <8>;
542              hsync-active = <0>;
543              vsync-active = <0>;
544              pclk-sample = <1>;
545              pclk-max-frequency = <77000000>;
546          };
547      };
548  };
```

Comment out the dcmi_pins_b and dcmi_sleep_pins_b pin configuration information:

```
dcmi_pins_b: dcmi-1 {
    pins {
        pinmux = <STM32_PINMUX('H', 8,  AF13)>,/* DCMI_HSYNC */
            <STM32_PINMUX('B', 7,  AF13)>,/* DCMI_VSYNC */
            <STM32_PINMUX('A', 6,  AF13)>,/* DCMI_PIXCLK */
            <STM32_PINMUX('H', 9,  AF13)>,/* DCMI_D0 */
            <STM32_PINMUX('H', 10, AF13)>,/* DCMI_D1 */
            <STM32_PINMUX('H', 11, AF13)>,/* DCMI_D2 */
            <STM32_PINMUX('H', 12, AF13)>,/* DCMI_D3 */
            <STM32_PINMUX('H', 14, AF13)>,/* DCMI_D4 */
            <STM32_PINMUX('I', 4,  AF13)>,/* DCMI_D5 */
            <STM32_PINMUX('B', 8,  AF13)>,/* DCMI_D6 */
            <STM32_PINMUX('E', 6,  AF13)>;/* DCMI_D7 */
        bias-disable;
    };
};
```

```
dcmi_sleep_pins_b: dcmi-sleep-1 {
    pins {
        pinmux = <STM32_PINMUX('H', 8,  ANALOG)>,/* DCMI_HSYNC */
            <STM32_PINMUX('B', 7,  ANALOG)>,/* DCMI_VSYNC */
            <STM32_PINMUX('A', 6,  ANALOG)>,/* DCMI_PIXCLK */
            <STM32_PINMUX('H', 9,  ANALOG)>,/* DCMI_D0 */
            <STM32_PINMUX('H', 10, ANALOG)>,/* DCMI_D1 */
            <STM32_PINMUX('H', 11, ANALOG)>,/* DCMI_D2 */
            <STM32_PINMUX('H', 12, ANALOG)>,/* DCMI_D3 */
            <STM32_PINMUX('H', 14, ANALOG)>,/* DCMI_D4 */
            <STM32_PINMUX('I', 4,  ANALOG)>,/* DCMI_D5 */
            <STM32_PINMUX('B', 8,  ANALOG)>,/* DCMI_D6 */
            <STM32_PINMUX('E', 6,  ANALOG)>;/* DCMI_D7 */
    };
};
```

Comment out ov5640 peripheral information

```
ov5640: camera@3c {
    compatible = "ovti,ov5640";
    reg = <0x3c>;
    clocks = <&clk_ext_camera>;
    clock-names = "xclk";
    DOVDD-supply = <&v2v8>;
    powerdown-gpios = <&gpioe 11 (GPIO_ACTIVE_HIGH | GPIO_PUSH_PULL)>;
    reset-gpios = <&gpioe 1 (GPIO_ACTIVE_LOW | GPIO_PUSH_PULL)>;
    rotation = <180>;
    status = "okay";

    port {
        ov5640_0: endpoint {
            remote-endpoint = <&dcmi_0>;
            bus-width = <8>;
            data-shift = <2>;
            hsync-active = <0>;
            vsync-active = <0>;
            pclk-sample = <1>;
            pclk-max-frequency = <77000000>;
        };
    };
};
```

Check whether there are any omissions in the modification and save the modified device tree file.

3.2.3 Compile kernel and device tree

At this point, the kernel and device tree clipping of camera peripherals is complete.

```
make uImage dtbs LOADADDR=0XC2000040 vmlinux -j16
```

Replace the compiled stm32mp157d-atk.dtb and uImage files on the development board and start it.

3.3 Command test GPIO output

Take the DCMI_D0 pin as an example. Start the development board, adjust the multimeter to the voltage position, connect the black probe to the ground, and the red probe to the DCMI_D0 (i.e. CAMERA's D0) interface.

Enter the development board system terminal and perform GPIO operations.

```
cd /sys/class/gpio/
echo 121 > export
echo low > gpio121/direction
echo high > gpio121/direction
```

Test that the DCMI_D0 pin can pull high/low output.

Schematic Pin Name	GPIO Name	GPIO Number	High/Low Level	Remark
DCMI_D0	PH9	121	3.3V/0V	
DCMI_D1	PH10	122	3.3V/0V	
DCMI_D2	PH11	123	3.3V/0V	
DCMI_D3	PH12	124	3.3V/0V	
DCMI_D4	PH14	126	3.3V/0V	
DCMI_D5	PI4	132	3.3V/0V	
DCMI_D6	PB8	twenty four	3.3V/0V	
DCMI_D7	PE6	70	3.3V/0V	
I2C5_SCL	PA11	11	3.3V/0V	
I2C5_SDA	PA12	12	3.3V/0V	
DCMI_VSYNC	PB7	23	3.3V/0V	
DCMI_PIXCLK	PA6	6	3.3V/0V	
DCMI_RESET	PE1	65	3.3V/0V	
DCMI_HSYNC	PH8	120	3.3V/0V	
DCMI_PWDN	PE11	75	3.3V/0V	

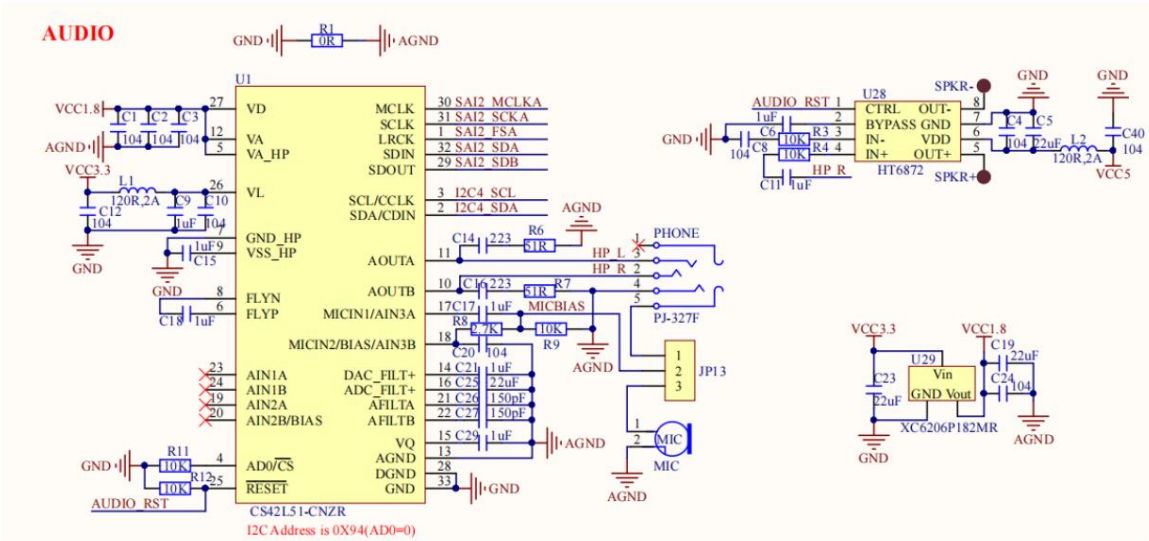
3.4 Program test GPIO output\input\interrupt

For the test program, refer to the GPIO application programming in "STM32MP1 Embedded Linux C Application Programming Guide Vx.x".

Chapter 4: Cutting Audio

4.1 Driver and pin definition

Taking the 157 development board as an example, the audio I2C audio interface is I2C4, and the relevant schematic diagram is as follows:



Alpha Schematic Pinout	GPIO	Reusable functions
SAI2_MCLKA	PE0	LPTIM1_ETR, TIM4_ETR, LPTIM2_ETR,SPI3_SCK/I2S3_CK, SAI4_MCLK_B, UART8_RX, SAI2_MCLK_A, FMC_NBL0, DCMI_D2, EVENTOUT
SAI2_SDB	PF11	SPI5_MOSI, SAI2_SD_B, DCMI_D12, LCD_G5, EVENTOUT
SAI2_SDA	PI6	TIM8_CH2, SAI2_SD_A, DCMI_D6, LCD_B6, EVENTOUT
SAI2_SCKA	PI5	TIM8_CH1, SAI2_SCK_A, DCMI_VSYNC, LCD_B5, EVENTOUT
SAI2_FSA	PI7	TIM8_CH3, SAI2_FS_A, DCMI_D7, LCD_B7, EVENTOUT
I2C4_SCL	PZ4	I2C6_SCL, I2C2_SCL, I2C5_SCL, I2C4_SCL,EVENTOUT
I2C4_SDA	PZ5	I2C6_SDA, I2C2_SDA, I2C5_SDA, I2C4_SDA, USART1_RTS/USART1_DE,EVENTOUT

AUDIO_RST	PZ7	I2C6_SDA, I2C2_SDA, USART1_TX, EVENTOUT

4.2 Kernel cutting AUDIO driver

4.2.1 Modify menuconfig configuration

Enter the factory kernel source directory and open the menuconfig configuration.

```
source /opt/st/stm32mp1/3.1-snapshot/environment-setup-cortexa7t2hf-neon-vfpv4-ostl-linux-gnueabi
make stm32mp1_atk_defconfig
make menuconfig Modify
```

the menuconfig configuration as follows:

```
Device Drivers
    -> Sound card support
        -> Advanced Linux Sound Architecture
            -> ALSA for SoC audio support
                -> < > ASoC Audio Graph sound card support
```

```
--- ALSA for SoC audio support
< >   AMD Audio Coprocessor support
<M>   SoC Audio for the Atmel System-on-Chip
< >     Support for Mikroe-PROTO board
< >   Synopsys I2S Device Driver
      SoC Audio for Freescale CPUs  --->
< >   Hisilicon I2S controller
[ ]   Audio support for Imagination Technologies designs
< >   ALSA BT SCO CVSD/MSBC Driver
[ ]   Sound Open Firmware Support
      STMicroelectronics STM32 SOC audio support  --->
< >   Audio support for the Xilinx I2S
< >   Audio support for the the Xilinx audio formatter
< >   Audio support for the the Xilinx SPDIF
< >   XTFPGA I2S master
< >   ZTE ZX TDM Driver Support
      CODEC drivers  --->
<M>   ASoC Simple sound card support
< >   ASoC Audio Graph sound card support
```

Figure 4.2-1 menuconfig configuration

Double-click the Esc key on the keyboard to exit and save the menuconfig configuration.

4.2.2 Modify the device tree file

Open the arch/arm/boot/dts/stm32mp157d-atk.dtsi device tree file and modify it. Search for the keyword sound. Comment out the sound node (you can delete the existing comments first).

```
/*  sound: sound {
        compatible = "audio-graph-card";
        label = "STM32MP1-DK";
        routing =
            "Playback" , "MCLK",
            "Capture" , "MCLK",
            "MICL" , "Mic Bias";
        dais = <&sai2a_port &sai2b_port &i2s2_port &spdifrx_port>;
        status = "okay";
    };
    sound: sound {
        compatible = "audio-graph-card";
        label = "STM32MP1-DK";

        widgets =
            "Microphone", "Mic Jack",
            "Line", "Line In",
            "Line", "Line Out",
            "Speaker", "Speaker",
            "Headphone", "Headphone Jack";

        routing =
            "Headphone Jack", "HP_L",
            "Headphone Jack", "HP_R",
            "Speaker", "SPK_LP",
            "Speaker", "SPK_LN",
            "Speaker", "SPK_RP",
            "Speaker", "SPK_RN",
            "LINPUT1", "Mic Jack",
            "LINPUT3", "Mic Jack",
            "RINPUT1", "Mic Jack",
            "RINPUT2", "Mic Jack";

        dais = <&sai2a_port &sai2b_port  &sai4a_port &spdifrx_port &i2s2_port
        status = "okay";
    }; */
```

Comment out the codec information (stm32mp157d-atk.dtsi) under the &i2c4 node in the kernel source code.

```
/* &i2c4 {…
};
*/
&i2c5 {
    pinctrl-names = "default", "sleep";
    pinctrl-0 = <&i2c5_pins_a>;
    pinctrl-1 = <&i2c5_pins_sleep_a>;
    i2c-scl-rising-time-ns = <100>;
    i2c-scl-falling-time-ns = <7>;
    status = "okay";
    /delete-property/dmas;
    /delete-property/dma-names;

    ap3216c@1e {
        compatible = "LiteOn,ap3216c";
        reg = <0x1e>;
    };
```

Open the arch/arm/boot/dts/stm32mp15-pinctrl.dtsi device tree file and modify it. Comment out the i2c4_pins_a and i2c4_pins_sleep_a pin multiplexing functions.

```
    i2c4_pins_a: i2c4-0 {
        pins {
    //      pinmux = <STM32_PINMUX('Z', 4, AF6)>, /* I2C4_SCL */
    //           <STM32_PINMUX('Z', 5, AF6)>; /* I2C4_SDA */
            bias-disable;
            drive-open-drain;
            slew-rate = <0>;
        };
    };

    i2c4_pins_sleep_a: i2c4-1 {
        pins {
    //      pinmux = <STM32_PINMUX('Z', 4, ANALOG)>, /* I2C4_SCL */
    //           <STM32_PINMUX('Z', 5, ANALOG)>; /* I2C4_SDA */
        };
    };
```

Comment out the &sound node in the stm32mp157d-atk.dts and stm32mp157d-atk-mipi.dts files

```
/* &sound {
    dais = <&sai2a_port &sai2b_port &spdifrx_port>;
}; */
```

Comment out the &sai2 node information (stm32mp157d-atk.dtsi).

```
&rng1 {
    status = "okay";
};
/*
&sai2 {···
}; */

&sai4 {
```

Comment out sai2a_pins_a, sai2a_sleep_pins_a, sai2b_pins_b, sai2b_sleep_pins_b pin multiplexing Function.(stm32mp15-pinctrl.dtsi)

```
    sai2a_pins_a: sai2a-0 {
        pins {
    //      pinmux = <STM32_PINMUX('I', 5, AF10)>, /* SAI2_SCK_A */
    //           <STM32_PINMUX('I', 6, AF10)>, /* SAI2_SD_A */
    //           <STM32_PINMUX('I', 7, AF10)>, /* SAI2_FS_A */
    //           <STM32_PINMUX('E', 0, AF10)>; /* SAI2_MCLK_A */
            slew-rate = <0>;
            drive-push-pull;
            bias-disable;
        };
    };

    sai2a_sleep_pins_a: sai2a-1 {
        pins {
    //      pinmux = <STM32_PINMUX('I', 5, ANALOG)>, /* SAI2_SCK_A */
    //           <STM32_PINMUX('I', 6, ANALOG)>, /* SAI2_SD_A */
    //           <STM32_PINMUX('I', 7, ANALOG)>, /* SAI2_FS_A */
    //           <STM32_PINMUX('E', 0, ANALOG)>; /* SAI2_MCLK_A */
        };
    };
```

```
802     sai2b_pins_b: sai2b-2 {
803         pins {
804         //  pinmux = <STM32_PINMUX('F', 11, AF10)>; /* SAI2_SD_B */
805             bias-disable;
806         };
807     };
808
809     sai2b_sleep_pins_b: sai2b-3 {
810         pins {
811         //  pinmux = <STM32_PINMUX('F', 11, ANALOG)>; /* SAI2_SD_B */
812         };
813     };
814
```

Check whether there are any omissions in the modification and save the modified device tree file.

4.2.3 Compile kernel and device tree

At this point, the kernel and device tree pruning of audio peripherals is complete.

```
make uImage dtbs LOADADDR=0XC2000040 vmlinux -j16
```

Replace the compiled stm32mp157d-atk.dtb and uImage files on the development board and start it.

4.3 IO Test

Enter the development board system terminal and perform GPIO operations. Take SAI2_SDA pin PI6 as an example, the corresponding IO number is 134.

```
cd /sys/class/gpio/ echo 134
> export echo low >
gpio134/direction echo high > gpio134/
direction
```

Schematic Pin Name	GPIO Name	GPIO number high/low level	
SAI2_MCLKA	PE0	64	3.3V/0V
SAI2_SDA	PI6	134	3.3V/0V
SAI2_SDB	PF11	91	3.3V/0V
SAI2_SCKA	PI5	133	3.3V/0V
SAI2_FSA	PI7	135	3.3V/0V
I2C4_SCL	PZ4		3.3V/0V
I2C4_SDA	PZ5		3.3V/0V
AUDIO_RST	PZ7		3.3V/0V

Note: PZ4, PZ5, PZ7 cannot be realized using the above operation method, and you need to write the test code yourself.

4.4 Program test GPIO output\input\interrupt

For the test program, refer to the GPIO application programming in "STM32MP1 Embedded Linux C Application Programming Guide Vx.x".

[Peripheral reuse description]

In the previous article, we explained the common peripheral cutting on Linux development board, including camera interface, dual network port, audio interface,

By cutting these peripherals, the JTAG interface can release a lot of pins. Users can reuse these pins into the functions they need according to the chip reference manual to meet project requirements.

This article is based on the pins released by the Linux development board to configure and implement common peripheral multiplexing. This document is for reference only.

The specific implementation effect also depends on whether the actual hardware is normal.

Chapter 5 Multiple GPIO Multiplexing

5.1 GPIO multiplexing modification

157 Most of the pins on the core board can be reused as GPIO functions. This part has specific modification instructions in the previous peripheral cutting section. To modify GPIO reuse, you must first determine whether the current pin has been used as other peripheral functions in the kernel, device tree, system, and hardware. If so, you need to cut and release it in turn. The released pins can generally be operated through the system's GPIO instructions. You can set high and low levels and use a multimeter to test the effect. If you need to manage GPIO more carefully, you can add the GPIO pin-related configuration to the device tree. If the GPIO level difference occurs during the startup or reset of the development board, such as setting a GPIO to a low level, and the GPIO is high or the level fluctuates during the reset process, you need to check whether there are configuration functions in the uboot and kernel that affect this GPIO, and whether the baseboard hardware circuit interferes with this GPIO.

5.2 Add GPIO to the device tree

This part can be combined with the 25th chapter of the pinctrl and gpio subsystem experiment of the [Zhengdian Atom] STM32MP1 Embedded Linux Driver Development Guide Vxx document, which contains specific tutorial instructions.

5.3 GPIO Test

After modifying the GPIO function configuration, you can perform GPIO testing.

The GPIO test in is not repeated here.

Here we will introduce how to view the usage of the development board's factory file system IO. You can execute the following command in the development board's factory file system:

```
cat /sys/kernel/debug/pinctrl/soc\:pin-controller@50002000/pinmux-pins
```

```
pin 68 (PE4): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 69 (PE5): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 70 (PE6): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 71 (PE7): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 72 (PE8): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 73 (PE9): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 74 (PE10): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 75 (PE11): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 76 (PE12): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 77 (PE13): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 78 (PE14): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 79 (PE15): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 80 (PF0): 48004000.sdmmc (GPIO UNCLAIMED) function analog group PF0
pin 81 (PF1): 48004000.sdmmc (GPIO UNCLAIMED) function analog group PF1
pin 82 (PF2): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 83 (PF3): (MUX UNCLAIMED) GPIOF:83
pin 84 (PF4): 48004000.sdmmc (GPIO UNCLAIMED) function analog group PF4
pin 85 (PF5): 48004000.sdmmc (GPIO UNCLAIMED) function analog group PF5
pin 86 (PF6): 40018000.serial (GPIO UNCLAIMED) function af7 group PF6
pin 87 (PF7): 40018000.serial (GPIO UNCLAIMED) function af7 group PF7
pin 88 (PF8): 40018000.serial (GPIO UNCLAIMED) function af7 group PF8
pin 89 (PF9): 40018000.serial (GPIO UNCLAIMED) function af7 group PF9
pin 90 (PF10): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 91 (PF11): (MUX UNCLAIMED) GPIOF:91
pin 92 (PF12): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 93 (PF13): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 94 (PF14): 40012000.i2c (GPIO UNCLAIMED) function af5 group PF14
pin 95 (PF15): 40012000.i2c (GPIO UNCLAIMED) function af5 group PF15
pin 96 (PG0): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 97 (PG1): (MUX UNCLAIMED) (GPIO UNCLAIMED)
pin 98 (PG2): 0-0022 GPIOG:98 function analog group PG2
pin 99 (PG3): (MUX UNCLAIMED) GPIOG:99
pin 100 (PG4): 5800a000.ethernet (GPIO UNCLAIMED) function af11 group PG4
```

example, (MUX UNCLAIMED) (GPIO UNCLAIMED) means that the current pin has no configuration function, such as pin 90 (PF10): (MUX UNCLAIMED) (GPIO UNCLAIMED), indicating that the PF10 pin can be used directly as a GPIO.

pin 83 (PF3): (MUX UNCLAIMED) GPIOF:83, indicating that this pin (PF3) has been used.

From the board schematic, we can see that PF3 has been used by LED1, which can be analyzed in conjunction with the device tree.

5.4 Program test GPIO output\input\interrupt

The test program refers to the GPIO application programming in "STM32MP1 Embedded Linux C Application Programming Guide Vx.x"

Chapter 6 8-channel UART multiplexing

The 157 development board is commonly used for industrial data acquisition. It supports up to 8 UARTs (4 USRTs + 4 USARTs) in hardware, with four embedded universal synchronous receiver transmitters (USART1, USART2, USART3 and USART6) and four universal asynchronous receiver transmitters (UART4, UART5, UART7 and UART8). See the table USARTx and UARTx functions for a summary.

USART modes/features(1)	USART1/2/3/6	UART4/5/7/8
Hardware flow control for modem	X	X
Continuous communication using DMA	X	X
Multiprocessor communication	X	X
Synchronous mode (master/slave)	X	-
Smartcard mode	X	-
Single-wire half-duplex communication	X	X
IrDA SIR ENDEC block	X	X
LIN mode	X	X
Dual clock domain and wakeup from low power mode	X	X
Receiver timeout interrupt	X	X
Modbus communication	X	X
Auto baud rate detection	X	X
Driver Enable	X	X
USART data length	7, 8 and 9 bits	

1. X = supported.

MP157DDA1 UART peripheral pins (refer to STM32MP157A&D data sheet)

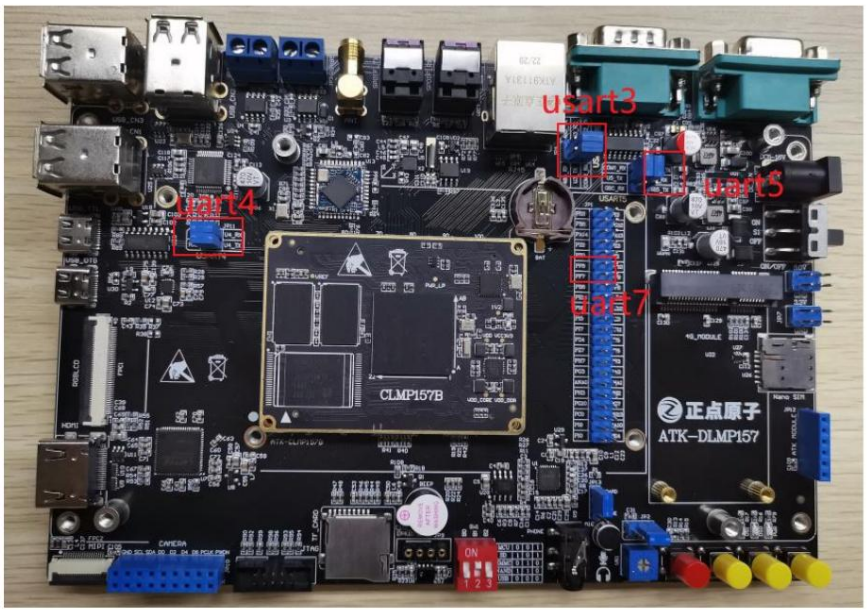
Port	AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7
	HDP/SYS/RTC	TIM1/2/16/17/ LPTIM1/SYS/ RTC	SAI1/4/I2C6/ TIM3/4/5/12/ HDP/SYS	SAI4/I2C2/ TIM8/ LPTIM2/3/4/5/ DFSDM1 /SDMMC1	SAI4/ I2C1/2/3/4/5/ USART1/ TIM15/LPTIM2/ DFSDM1/CEC	SPI1/I2S1/ SPI2/I2S2/ SPI3/I2S3/ SPI4/5/6/I2C1/ SDMMC1/3/ CEC	SPI3/I2S3/ SAI1/3/4/ I2C4/UART4/ DFSDM1	SPI2/I2S2/ SPI3/I2S3/ SPI6/ USART1/2/3/6/ UART7/ SDMMC2
PA2	-	TIM2_CH3	TIM5_CH3	LPTIM4_OUT	TIM15_CH1	-	-	USART2_TX
PA3	-	TIM2_CH4	TIM5_CH4	LPTIM5_OUT	TIM15_CH2	-	-	USART2_RX
PA9	-	TIM1_CH2	-	-	I2C3_SMBA	SPI2_SCK/ I2S2_CK	-	USART1_TX
PA10	-	TIM1_CH3	-	-	-	SPI3_NSS/ I2S3_WS	-	USART1_RX
PA11	-	TIM1_CH4	I2C6_SCL	-	I2C5_SCL	SPI2_NSS/ I2S2_WS	UART4_RX	USART1_CTS/ USART1_NSS
PA12	-	TIM1_ETR	I2C6_SDA	-	I2C5_SDA	-	UART4_TX	USART1_RTS/ USART1_DE

	PB2	TRACED4	RTC_OUT2	SAI1_D1	DFSDM1_CKIN1	USART1_RX	I2S_CKIN	SAI1_SD_A	SPI3_MOSI/I2S3_SDO
	PB6	-	TIM16_CH1N	TIM4_CH1	-	I2C1_SCL	CEC	I2C4_SCL	USART1_TX
	PB7	-	TIM17_CH1N	TIM4_CH2	-	I2C1_SDA	-	I2C4_SDA	USART1_RX
	PB11	-	TIM2_CH4	-	LPTIM2_ETR	I2C2_SDA	-	DFSDM1_CKIN7	USART3_RX
	PB14	-	TIM1_CH2N	TIM12_CH1	TIM8_CH2N	USART1_TX	SPI2_MISO/I2S2_SDI	DFSDM1_DATIN2	USART3_RTS/USART3_DE
	PB15	RTC_REFIN	TIM1_CH3N	TIM12_CH2	TIM8_CH3N	USART1_RX	SPI2_MOSI/I2S2_SDO	DFSDM1_CKIN2	-
	PC6	HDP1	-	TIM3_CH1	TIM8_CH1	DFSDM1_CKIN3	I2S2_MCK	-	USART6_TX
	PC7	HDP4	-	TIM3_CH2	TIM8_CH2	DFSDM1_DATIN3	-	I2S3_MCK	USART6_RX
	PC8	TRACED0	-	TIM3_CH3	TIM8_CH3	-	-	UART4_TX	USART6_CK
Port C	PC10	TRACED2	-	-	DFSDM1_CKIN5	-	-	SPI3_SCK/I2S3_CK	USART3_TX
	PC11	TRACED3	-	-	DFSDM1_DATIN5	-	-	SPI3_MISO/I2S3_SDI	USART3_RX
	PD5	-	-	-	-	-	-	-	USART2_TX
	PD6	-	TIM16_CH1N	SAI1_D1	DFSDM1_CKIN4	DFSDM1_DATIN1	SPI3_MOSI/I2S3_SDO	SAI1_SD_A	USART2_RX
	PD8	-	-	-	DFSDM1_CKIN3	-	-	SAI3_SCK_B	USART3_TX
	PD9	-	-	-	DFSDM1_DATIN3	-	-	SAI3_SD_B	USART3_RX
	PE7	-	TIM1_ETR	TIM3_ETR	DFSDM1_DATIN2	-	-	-	UART7_RX
	PE8	-	TIM1_CH1N	-	DFSDM1_CKIN2	-	-	-	UART7_TX
	PF4	-	-	-	-	-	-	-	USART2_RX
	PF5	-	-	-	-	-	-	-	USART2_TX
	PF6	-	TIM16_CH1	-	-	-	SPI5_NSS	SAI1_SD_B	UART7_RX
	PF7	-	TIM17_CH1	-	-	-	SPI5_SCK	SAI1_MCLK_B	UART7_TX
Port G	PG9	DBTRGO	-	-	-	-	-	-	USART6_RX
	PG10	TRACED10	-	-	-	-	-	-	-
	PG11	TRACED11	-	-	-	USART1_TX	-	UART4_TX	-
Port Z	PZ1	-	-	I2C6_SDA	I2C2_SDA	I2C5_SDA	SPI1_MISO/I2S1_SDI	I2C4_SDA	USART1_RX

Port		AF8	AF9	AF10	AF11	AF12	AF13	AF14	AF15
		SPI6/SAI2/ USART3/ UART4/5/8/ SDMMC1/2/ SPDIFRX	FDCAN1/2/ TIM13/14/ QUADSPI/ SDMMC2/3/ LCD/SPDIFRX	SAI2/4/ QUADSPI/ FMC/ SDMMC2/3/ OTG_FS/ OTG_HS	DFSDM1/ QUADSPI/ SDMMC1/ MDIOS/ETH1/ DSI	SAI4/UART5/ FMC/SDMMC1/ MDIOS	UART7/DCMI/ LCD/DSI/RNG	UART5/LCD	SYS
	PA0	UART4_TX	SDMMC2_CMD	SAI2_SD_B	ETH1_GMII_ CRS/ ETH1_MII_CRS	-	-	-	EVENTOUT
	PA1	UART4_RX	QUADSPI_ BK1_IO3	SAI2_MCLK_B	ETH1_GMII_RX _CLK/ ETH1_MII_RX_ CLK/ ETH1_RGMII_ RX_CLK/ ETH1_RMII_ REF_CLK	-	-	LCD_R2	EVENTOUT
	PA8	SDMMC2_ CKIN	SDMMC2_D4	OTG_FS_SOF/ OTG_HS_SOF	-	SAI4_SD_B	UART7_RX	LCD_R6	EVENTOUT
	PA13	UART4_TX	-	-	-	-	-	-	EVENTOUT
	PA15	UART4_RTS/ UART4_DE	SDMMC2_D5	SDMMC2_ CDIR	SDMMC1_D5	SAI4_FS_A	UART7_TX	LCD_R1	EVENTOUT
	PB2	UART4_RX	QUADSPI_CLK	-	-	-	-	-	EVENTOUT
	PB3	SPI6_SCK	SDMMC2_D2	-	-	SAI4_MCLK_A	UART7_RX	-	EVENTOUT
	PB4	SPI6_MISO	SDMMC2_D3	-	-	SAI4_SCK_A	UART7_TX	-	EVENTOUT
	PB5	SPI6_MOSI	FDCAN2_RX	SAI4_SD_A	ETH1_PPS_ OUT	UART5_RX	DCMI_D10	LCD_G7	EVENTOUT
	PB6	-	FDCAN2_TX	QUADSPI_BK1 _NCS	DFSDM1_ DATIN5	UART5_TX	DCMI_D5	-	EVENTOUT
Port B	PB8	UART4_RX	FDCAN1_RX	SDMMC2_D4	ETH1_GMII_ TXD3/ ETH1_MII_ TXD3/ ETH1_RGMII_ TXD3	SDMMC1_D4	DCMI_D6	LCD_B6	EVENTOUT
	PB9	UART4_TX	FDCAN1_TX	SDMMC2_D5	SDMMC1_CDI R	SDMMC1_D5	DCMI_D7	LCD_B7	EVENTOUT
Port B	PB12	USART3_RX	FDCAN2_RX	-	ETH1_GMII_ TXD0/ ETH1_MII_ TXD0/ ETH1_RGMII_ TXD0/ ETH1_RMII_ TXD0	-	-	UART5_RX	EVENTOUT
	PB13	-	FDCAN2_TX	-	ETH1_GMII_ TXD1/ ETH1_MII_ TXD1/ ETH1_RGMII_ TXD1/ ETH1_RMII_ TXD1	-	-	UART5_TX	EVENTOUT

Port C	PC10	UART4_TX	QUADSPI_BK1_IO1	SAI4_MCLK_B	-	SDMMC1_D2	DCMI_D8	LCD_R2	EVENTOUT
	PC11	UART4_RX	QUADSPI_BK2_NCS	SAI4_SCK_B	-	SDMMC1_D3	DCMI_D4	-	EVENTOUT
	PC12	UART5_TX	-	SAI4_SD_B	-	SDMMC1_CK	DCMI_D9	-	EVENTOUT
	PD0	UART4_RX	FDCAN1_RX	SDMMC3_CMD	DFSDM1_DATIN7	FMC_AD2/FMC_D2	-	-	EVENTOUT
	PD1	UART4_TX	FDCAN1_TX	SDMMC3_D0	DFSDM1_CKIN7	FMC_AD3/FMC_D3	-	-	EVENTOUT
	PD2	UART5_RX	-	-	-	SDMMC1_CMD	DCMI_D11	-	EVENTOUT
	PE0	UART8_RX	-	SAI2_MCLK_A	-	FMC_NBL0	DCMI_D2	-	EVENTOUT
	PE1	UART8_TX	-	-	-	FMC_NBL1	DCMI_D3	-	EVENTOUT
	PH13	UART4_TX	FDCAN1_TX	-	-	-	-	LCD_G2	EVENTOUT
	PH14	UART4_RX	FDCAN1_RX	-	-	-	DCMI_D4	LCD_G3	EVENTOUT
	PI9	UART4_RX	FDCAN1_RX	-	-	-	-	LCD_VSYNC	EVENTOUT
	PJ8	UART8_TX	-	-	-	-	-	LCD_G1	EVENTOUT
	PJ9	UART8_RX	-	-	-	-	-	LCD_G2	EVENTOUT

The development board has onboard serial ports 3 (ttySTM1), 4 (ttySTM0), 5 (ttySTM2), and 7 (ttySTM3). You can use these serial ports directly, and you can also add usart1, usart2, uart6, and uart8.



You can view it in the kernel source code path file arch/arm/boot/dts/stm32mp157d-atk.dts.

```
/ {
    model = "STMicroelectronics STM32MP157C-DK2 Discovery Board";
    compatible = "st,stm32mp157d-atk", "st,stm32mp157";

    aliases {
        ethernet0 = &ethernet0;
        serial0 = &uart4;
        serial1 = &usart3;
        serial2 = &uart5;
        serial3 = &uart7;
    };
```

6.1 UART1 (ttySTM4)

PB6	-	TIM16_CH1N	TIM4_CH1	-	I2C1_SCL	CEC	I2C4_SCL	USART1_TX
PB7	-	TIM17_CH1N	TIM4_CH2	-	I2C1_SDA	-	I2C4_SDA	USART1_RX

Because other pins that can be reused as uart1 have been reused as emmc or uart4, the basic peripherals of the core board, it is not recommended to cut them, so PB6 and PB7 are used to reuse as uart1. Here, only uart1 is configured instead of usart1, and the basic serial communication function is tested first.

Open arch/arm/boot/dts/stm32mp15-pinctrl.dtsi, search for 'B', 6, comment out the related pin multiplexing, the related nodes are cec_pins_b, cec_pins_sleep_b, qspi_bk1_pins_a, qspi_bk1_sleep_pins_a.

```
46      cec_pins_b: cec-1 {
47          pins {
48  //          pinmux = <STM32_PINMUX('B', 6, AF5)>;
49              bias-disable;
50              drive-open-drain;
51              slew-rate = <0>;
52          };
53      };
```

```
55      cec_pins_sleep_b: cec-sleep-1 {
56          pins {
57  //          pinmux = <STM32_PINMUX('B', 6, ANALOG)>; /* HDMI_CEC */
58          };
59      };
```

```
695     qspi_bk1_pins_a: qspi-bk1-0 {
696         pins1 {
697             pinmux = <STM32_PINMUX('F', 8, AF10)>, /* QSPI_BK1_IO0 */
698                  <STM32_PINMUX('F', 9, AF10)>, /* QSPI_BK1_IO1 */
699                  <STM32_PINMUX('F', 7, AF9)>, /* QSPI_BK1_IO2 */
700                  <STM32_PINMUX('F', 6, AF9)>; /* QSPI_BK1_IO3 */
701             bias-disable;
702             drive-push-pull;
703             slew-rate = <1>;
704         };
705         pins2 {
706 //          pinmux = <STM32_PINMUX('B', 6, AF10)>; /* QSPI_BK1_NCS */
707             bias-pull-up;
708             drive-push-pull;
709             slew-rate = <1>;
710         };
711     };
```

```
713     qspi_bk1_sleep_pins_a: qspi-bk1-sleep-0 {
714         pins {
715             pinmux = <STM32_PINMUX('F', 8, ANALOG)>, /* QSPI_BK1_IO0 */
716                  <STM32_PINMUX('F', 9, ANALOG)>, /* QSPI_BK1_IO1 */
717                  <STM32_PINMUX('F', 7, ANALOG)>, /* QSPI_BK1_IO2 */
718                  <STM32_PINMUX('F', 6, ANALOG)>; /* QSPI_BK1_IO3 */
719 //                <STM32_PINMUX('B', 6, ANALOG)>; /* QSPI_BK1_NCS */
720         };
721     };
```

Search for 'B', 7, and comment out the related pin multiplexing. The related nodes are dcmi_pins_a and dcmi_sleep_pins_a.

```
73      dcmi_pins_a: dcmi-0 {
74          pins {
75              pinmux = <STM32_PINMUX('H', 8,  AF13)>,/* DCMI_HSYNC */
76  //                <STM32_PINMUX('B', 7,  AF13)>,/* DCMI_VSYNC */
77                  <STM32_PINMUX('A', 6,  AF13)>,/* DCMI_PIXCLK */
78                  <STM32_PINMUX('H', 9,  AF13)>,/* DCMI_D0 */
79                  <STM32_PINMUX('H', 10, AF13)>,/* DCMI_D1 */
80                  <STM32_PINMUX('H', 11, AF13)>,/* DCMI_D2 */
81                  <STM32_PINMUX('H', 12, AF13)>,/* DCMI_D3 */
82                  <STM32_PINMUX('H', 14, AF13)>,/* DCMI_D4 */
83                  <STM32_PINMUX('I', 4,  AF13)>,/* DCMI_D5 */
84                  <STM32_PINMUX('B', 8,  AF13)>,/* DCMI_D6 */
85                  <STM32_PINMUX('E', 6,  AF13)>,/* DCMI_D7 */
86                  <STM32_PINMUX('I', 1,  AF13)>,/* DCMI_D8 */
87                  <STM32_PINMUX('H', 7,  AF13)>,/* DCMI_D9 */
88                  <STM32_PINMUX('I', 3,  AF13)>,/* DCMI_D10 */
89                  <STM32_PINMUX('H', 15, AF13)>;/* DCMI_D11 */
90              bias-disable;
91          };
92      };
```

```
94      dcmi_sleep_pins_a: dcmi-sleep-0 {
95          pins {
96              pinmux = <STM32_PINMUX('H', 8,  ANALOG)>,/* DCMI_HSYNC */
97  //                  <STM32_PINMUX('B', 7,  ANALOG)>,/* DCMI_VSYNC */
98                      <STM32_PINMUX('A', 6,  ANALOG)>,/* DCMI_PIXCLK */
99                      <STM32_PINMUX('H', 9,  ANALOG)>,/* DCMI_D0 */
100                     <STM32_PINMUX('H', 10, ANALOG)>,/* DCMI_D1 */
101                     <STM32_PINMUX('H', 11, ANALOG)>,/* DCMI_D2 */
102                     <STM32_PINMUX('H', 12, ANALOG)>,/* DCMI_D3 */
103                     <STM32_PINMUX('H', 14, ANALOG)>,/* DCMI_D4 */
104                     <STM32_PINMUX('I', 4,  ANALOG)>,/* DCMI_D5 */
105                     <STM32_PINMUX('B', 8,  ANALOG)>,/* DCMI_D6 */
106                     <STM32_PINMUX('E', 6,  ANALOG)>,/* DCMI_D7 */
107                     <STM32_PINMUX('I', 1,  ANALOG)>,/* DCMI_D8 */
108                     <STM32_PINMUX('H', 7,  ANALOG)>,/* DCMI_D9 */
109                     <STM32_PINMUX('I', 3,  ANALOG)>,/* DCMI_D10 */
110                     <STM32_PINMUX('H', 15, ANALOG)>;/* DCMI_D11 */
111         };
112     };
```

Open arch/arm/boot/dts/stm32mp157d-atk.dtsi and search for cec_pins_b, cec_pins_sleep_b, qspi_bk1_pins_a, qspi_bk1_sleep_pins_a, dcmi_pins_a, dcmi_sleep_pins_a node labels in turn, and set their node status to disabled. Only cec_pins_b and cec_pins_sleep_b are used here, and the &cec node is disabled.

```
511 &cec {
512     pinctrl-names = "default", "sleep";
513     pinctrl-0 = <&cec_pins_b>;
514     pinctrl-1 = <&cec_pins_sleep_b>;
515     status = "disabled";
516 };
```

Configure B6\B7 as USART1, open arch/arm/boot/dts/stm32mp157d-atk.dtsi, refer to usart3

Configuration method: add usart1 related node information under the &pinctrl node.

```
usart1_pins_a: uart1-0 {
    pins1 {
        pinmux = <STM32_PINMUX('B', 6, AF7)>; /* USART1_TX */
        bias-disable;
        drive-push-pull;
        slew-rate = <0>;
    };
    pins2 {
        pinmux = <STM32_PINMUX('B', 7, AF7)>; /* USART1_RX */
        bias-disable;
    };
};

usart1_idle_pins_a: uart1-idle-0 {
    pins1 {
        pinmux = <STM32_PINMUX('B', 6, ANALOG)>; /* USART1_TX */
    };
    pins2 {
        pinmux = <STM32_PINMUX('B', 7, AF7)>; /* USART1_RX */
        bias-disable;
    };
};

usart1_sleep_pins_a: uart1-sleep-0 {
    pins {
        pinmux = <STM32_PINMUX('B', 6, ANALOG)>, /* USART1_TX */
                 <STM32_PINMUX('B', 7, ANALOG)>; /* USART1_RX */
    };
};
```

Refer to &usart3 and add &usart1 node information under the root node

```
1072  &usart1 {
1073      pinctrl-names = "default", "sleep", "idle";
1074      pinctrl-0 = <&usart1_pins_a>;
1075      pinctrl-1 = <&usart1_sleep_pins_a>;
1076      pinctrl-2 = <&usart1_idle_pins_a>;
1077      /delete-property/dmas;
1078      /delete-property/dma-names;
1079      status = "okay";
1080  };
1081
```

Open alientek_kernel/arch/arm/boot/dts/stm32mp157d-atk.dts and add serial4 = &usart1;

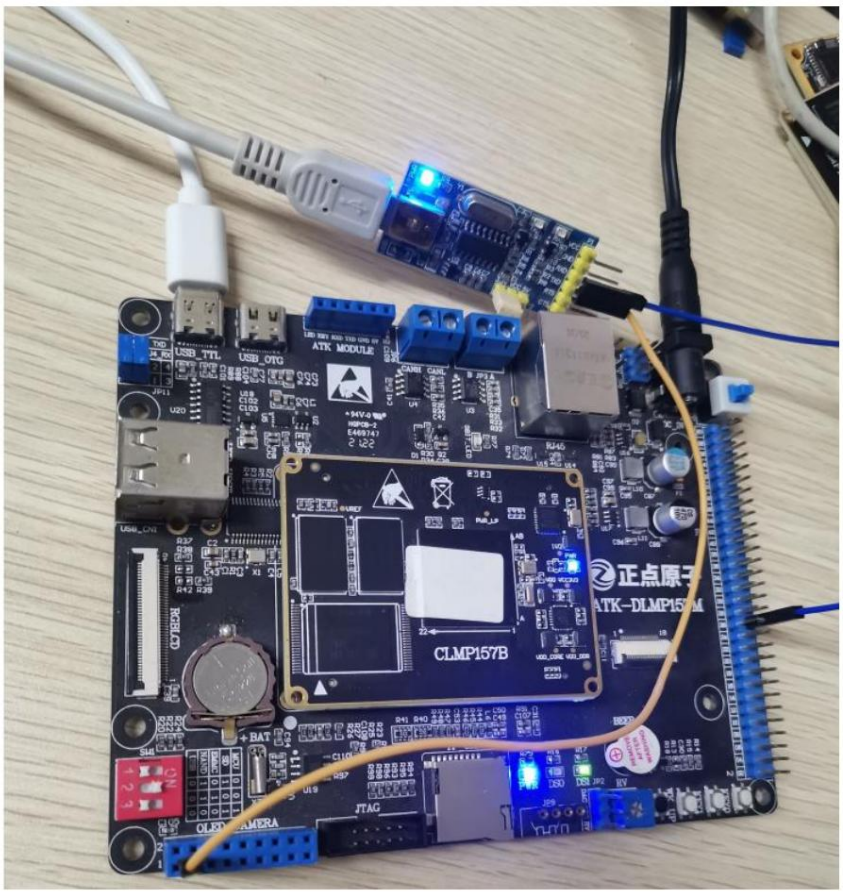
```
20      aliases {
21          ethernet0 = &ethernet0;
22          serial0 = &uart4;
23          serial1 = &usart3;
24          serial2 = &uart5;
25          serial3 = &uart7;
26          serial4 = &usart1;
27      };
```

Check whether there are any omissions in the modification and save the modified device tree file.

Save the modified file, compile the device tree, and use the compiled device tree to boot the factory system.

```
source /opt/st/stm32mp1/3.1-snapshot/environment-setup-cortexa7t2hf-neon-vfpv4-
ostl-linux-gnueabi make
stm32mp1_atk_defconfig
make dtbs
```

Hardware connection: You can connect the pin header of the mini board + the corresponding interface of the carema. The large board needs to be wired out for testing.



View serial port information

```
root@ATK-MP157:~# ls /dev/ttySTM*
/dev/ttySTM0  /dev/ttySTM1  /dev/ttySTM2  /dev/ttySTM3  /dev/ttySTM4
root@ATK-MP157:~#
```

Enter the minicon -s command on the development board system to configure it

```
+-----[configuration]------+
| Filenames and paths      |
| File transfer protocols  |
| Serial port setup        |
| Modem and dialing        |
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
| Exit from Minicom        |
+--------------------------+
```

```
+-----------------------------------------------------
| A -    Serial Device      : /dev/ttySTM4
| B - Lockfile Location     : /var/lock
| C -   Callin Program      :
| D -  Callout Program      :
| E -    Bps/Par/Bits       : 115200 8N1
| F - Hardware Flow Control : No
| G - Software Flow Control : No
|
|    Change which setting?
+-----------------------------------------------------
```

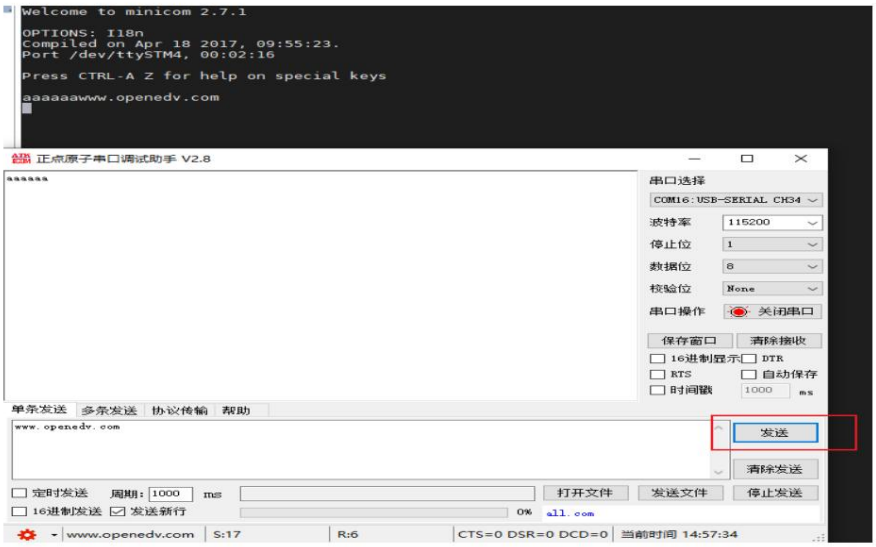
```
+-----[configuration]------+
| Filenames and paths      |
| File transfer protocols  |
| Serial port setup        |
| Modem and dialing        |
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
| Exit from Minicom        |
+--------------------------+
```

Enter CTRL+AZ command and press e to open the echo function

Use the serial port assistant to test the

development board to send data, as shown below

The serial port assistant sends data, as shown below



Enter the CTRL+AZ command and press q to exit the serial port test function.

6.2 UART2 (ttySTM5)

	PD5	-	-	-	-	-	-	-	USART2_TX
	PD6	-	TIM16_CH1N	SAI1_D1	DFSDM1_CKIN4	DFSDM1_DATIN1	SPI3_MOSI/I2S3_SDO	SAI1_SD_A	USART2_RX

Open arch/arm/boot/dts/stm32mp15-pinctrl.dtsi. The USART2 peripheral is configured by default, using PD5 and PD6.

```
1234    usart2_pins_a: usart2-0 {
1235        pins1 {
1236            pinmux = <STM32_PINMUX('D', 5, AF7)>, /* USART2_TX */
1237                 <STM32_PINMUX('D', 4, AF7)>; /* USART2_RTS */
1238            bias-disable;
1239            drive-push-pull;
1240            slew-rate = <3>;
1241        };
1242        pins2 {
1243            pinmux = <STM32_PINMUX('D', 6, AF7)>, /* USART2_RX */
1244                 <STM32_PINMUX('D', 3, AF7)>; /* USART2_CTS_NSS */
1245            bias-disable;
1246        };
1247    };
1248
1249    usart2_idle_pins_a: usart2-idle-0 {
1250        pins1 {
1251            pinmux = <STM32_PINMUX('D', 5, ANALOG)>, /* USART2_TX */
1252                 <STM32_PINMUX('D', 4, ANALOG)>, /* USART2_RTS */
1253                 <STM32_PINMUX('D', 3, ANALOG)>; /* USART2_CTS_NSS */
1254        };
1255        pins2 {
1256            pinmux = <STM32_PINMUX('D', 6, AF7)>; /* USART2_RX */
1257            bias-disable;
1258        };
1259    };
1260
1261    usart2_sleep_pins_a: usart2-sleep-0 {
1262        pins {
1263            pinmux = <STM32_PINMUX('D', 5, ANALOG)>, /* USART2_TX */
1264                 <STM32_PINMUX('D', 4, ANALOG)>, /* USART2_RTS */
1265                 <STM32_PINMUX('D', 6, ANALOG)>, /* USART2_RX */
1266                 <STM32_PINMUX('D', 3, ANALOG)>; /* USART2_CTS_NSS */
1267        };
1268    };
```

Continuing to look at the source code, you can see that the fmc_pins_a node also uses PD5, but this is not used in the factory system device tree.

No, it does not affect.

Open arch/arm/boot/dts/stm32mp157d-atk.dtsi, refer to &usart3, and add &usart2 node.

```
/* &usart2 begin */
&usart2 {
    pinctrl-names = "default", "sleep", "idle";
    pinctrl-0 = <&usart2_pins_a>;
    pinctrl-1 = <&usart2_sleep_pins_a>;
    pinctrl-2 = <&usart2_idle_pins_a>;
    /delete-property/dmas;
    /delete-property/dma-names;
    status = "okay";
};
/*  &usart2 end */
```

Open arch/arm/boot/dts/stm32mp157d-atk.dts, add serial5 = &usart2 under aliases;

```
aliases {
    ethernet0 = &ethernet0;
    serial0 = &uart4;
    serial1 = &usart3;
    serial2 = &uart5;
    serial3 = &uart7;
    serial4 = &usart1;
    serial5 = &usart2;
};
```

Check whether there are any omissions in the modification and save the modified device tree file.

Save the modified file, compile the device tree, and use the compiled device tree to boot the factory system.

```
source /opt/st/stm32mp1/3.1-snapshot/environment-setup-cortexa7t2hf-neon-vfpv4-
ostl-linux-gnueabi make
stm32mp1_atk_defconfig
make dtbs
```

View serial port devices

```
root@ATK-MP157:~# ls /dev/ttyS*
/dev/ttySTM0  /dev/ttySTM1  /dev/ttySTM2  /dev/ttySTM3  /dev/ttySTM4  /dev/ttySTM5
```

The hardware testing method can refer to the USART1 method.

6.3 USART6 (ttySTM6)

	PC6	HDP1	-	TIM3_CH1	TIM8_CH1	DFSDM1_CKIN3	I2S2_MCK	-	USART6_TX
	PC7	HDP4	-	TIM3_CH2	TIM8_CH2	DFSDM1_DATIN3	-	I2S3_MCK	USART6_RX

Open arch/arm/boot/dts/stm32mp15-pinctrl.dtsi ST does not configure usart6

by default. Later, we can refer to the configuration of usart3 under our own device tree &pinctrl node to add usart6 pin information

Continue searching for 'C', 6 and 'C', 7. Check the nodes of the ST default configuration for

these two pins and you can see that 'C', 6 is not used, and 'C', 7 is used in pwm3_pins_a, pwm3_sleep_pins_a, and sdmmc1_dir_pins_a, sdmmc1_dir_sleep_pins_a

Then search in our development board device tree stm32mp157d-atk.dtsi and stm32mp157d-atk.dts respectively I found several nodes and found that they were not used. If they were, I needed to disable their functions.

Add pin configuration

Open arch/arm/boot/dts/stm32mp157d-atk.dtsi. According to the schematic diagram, you can specify PC7 for the development board.

For a buzzer, you can comment out the beep configuration under the leds node.

```
102     leds {
103         compatible = "gpio-leds";
104
105         led1 {
106             label = "sys-led";
107             gpios = <&gpioi 0 GPIO_ACTIVE_LOW>;
108             linux,default-trigger = "heartbeat";
109             default-state = "on";
110             status = "okay";
111         };
112
113         led2 {
114             label = "user-led";
115             gpios = <&gpiof 3 GPIO_ACTIVE_LOW>;
116             linux,default-trigger = "none";
117             default-state = "on";
118             status = "okay";
119         };
120
121 //          beep {
122 //              label = "beep";
123 //              gpios = <&gpioc 7 GPIO_ACTIVE_LOW>;
124 //              default-state = "off";
125 //          };
126     };
```

Refer to the configuration method of usart3 and add the related node information of usart6 under the &pinctrl node.

```
422     usart6_pins_a: uart6-0 {
423         pins1 {
424             pinmux = <STM32_PINMUX('C', 6, AF7)>; /* USART6_TX */
425             bias-disable;
426             drive-push-pull;
427             slew-rate = <0>;
428         };
429         pins2 {
430             pinmux = <STM32_PINMUX('C', 7, AF7)>; /* USART6_RX */
431             bias-disable;
432         };
433     };
434
435     usart6_idle_pins_a: uart6-idle-0 {
436         pins1 {
437             pinmux = <STM32_PINMUX('C', 6, ANALOG)>; /* USART6_TX */
438         };
439         pins2 {
440             pinmux = <STM32_PINMUX('C', 7, AF7)>; /* USART6_RX */
441             bias-disable;
442         };
443     };
444
445     usart6_sleep_pins_a: uart6-sleep-0 {
446         pins {
447             pinmux = <STM32_PINMUX('C', 6, ANALOG)>, /* USART6_TX */
448                      <STM32_PINMUX('C', 7, ANALOG)>; /* USART6_RX */
449         };
450     };
```

Refer to &usart3 and add &usart6 node information under the root node

```
1143 &usart6 {
1144     pinctrl-names = "default", "sleep", "idle";
1145     pinctrl-0 = <&usart6_pins_a>;
1146     pinctrl-1 = <&usart6_sleep_pins_a>;
1147     pinctrl-2 = <&usart6_idle_pins_a>;
1148     /delete-property/dmas;
1149     /delete-property/dma-names;
1150     status = "okay";
1151 };
1152
```

Open arch/arm/boot/dts/stm32mp157d-atk.dts and add serial6 = &usart6;

```
20      aliases {
21          ethernet0 = &ethernet0;
22          serial0 = &uart4;
23          serial1 = &usart3;
24          serial2 = &uart5;
25          serial3 = &uart7;
26          serial4 = &usart1;
27          serial5 = &usart2;
28          serial6 = &uart6;
29      };
```

Check whether there are any omissions in the modification and save the modified device tree file.

Save the modified file, compile the device tree, and use the compiled device tree to boot the factory system.

```
source /opt/st/stm32mp1/3.1-snapshot/environment-setup-cortexa7t2hf-neon-vfpv4-
ostl-linux-gnueabi make
stm32mp1_atk_defconfig
make dtbs
```

View serial port devices

```
root@ATK-MP157:~# ls /dev/ttyS*
/dev/ttySTM0  /dev/ttySTM1  /dev/ttySTM2  /dev/ttySTM3  /dev/ttySTM4  /dev/ttySTM5  /dev/ttySTM6
root@ATK-MP157:~#
root@ATK-MP157:~#
```

The hardware testing method can refer to the USART1 method.

6.4 UART8 (ttySTM7)

	PE0	UART8_RX	-	SAI2_MCLK_A	-	FMC_NBL0	DCMI_D2	-	EVENTOUT
	PE1	UART8_TX	-	-	-	FMC_NBL1	DCMI_D3	-	EVENTOUT

When checking the

default configuration, open arch/arm/boot/dts/stm32mp15-pinctrl.dtsi.

ST does not configure uart8 by default. Later, we can refer to the configuration of uart4 under our own device tree & pinctrl node to add uart8

pin information. Continue to search for 'E', 0 and 'E', 1. Check the nodes of these

two pins in ST default configuration. You can see that 'E', 0 is used in sai2a_pins_a and sai2a_sleep_pins_a,

and 'E', 1 is not configured in stm32mp157d-atk.dtsi. Search sai2a_pins_a and sai2a_sleep_pins_a to see if they are used. Two nodes, if used, disable the related nodes

```
945  &sai2 {
946      clocks = <&rcc SAI2>, <&rcc PLL3_Q>, <&rcc PLL3_R>;
947      clock-names = "pclk", "x8k", "x11k";
948      pinctrl-names = "default", "sleep";
949      pinctrl-0 = <&sai2a_pins_a>, <&sai2b_pins_b>;
950      pinctrl-1 = <&sai2a_sleep_pins_a>, <&sai2b_sleep_pins_b>;
951      status = "disabled";
```

Add pin

configuration Open alientek_kernel/arch/arm/boot/dts/stm32mp157d-atk.dtsi, check where 'E', 0 and 'E', 1 are used (or gpioe 0 and gpioe 1) Add uart8 related node information under the &pinctrl node.

```
392     uart8_pins_a: uart8-1 {
393         pins1 {
394             pinmux = <STM32_PINMUX('E', 1, AF8)>; /* UART8_TX */
395             bias-disable;
396             drive-push-pull;
397             slew-rate = <0>;
398         };
399         pins2 {
400             pinmux = <STM32_PINMUX('E', 0, AF8)>;   /* UART8_RX */
401             bias-disable;
402         };
403     };
404
405     uart8_idle_pins_a: uart8-idle-1 {
406         pins1 {
407             pinmux = <STM32_PINMUX('E', 1, AF8)>; /* UART8_TX */
408         };
409         pins2 {
410             pinmux = <STM32_PINMUX('E', 0, AF8)>; /* UART7_RX */
411             bias-disable;
412         };
413     };
414
415     uart8_sleep_pins_a: uart8-sleep-1 {
416         pins {
417             pinmux = <STM32_PINMUX('E', 1, ANALOG)>, /* UART8_TX */
418                      <STM32_PINMUX('E', 0, ANALOG)>; /* UART8_RX */
419         };
420     };
```

Add &uart8 node information under the root node.

```
1195 &uart8 {
1196     pinctrl-names = "default", "sleep", "idle";
1197     pinctrl-0 = <&uart8_pins_a>;
1198     pinctrl-1 = <&uart8_sleep_pins_a>;
1199     pinctrl-2 = <&uart8_idle_pins_a>;
1200     /delete-property/dmas;
1201     /delete-property/dma-names;
1202     st,hw-flow-ctrl;
1203     status = "okay";
1204 };
```

Open stm32mp157d-atk.dts and add serial7 = &uart8 under aliases

```
20      aliases {
21          ethernet0 = &ethernet0;
22          serial0 = &uart4;
23          serial1 = &usart3;
24          serial2 = &uart5;
25          serial3 = &uart7;
26          serial4 = &usart1;
27          serial5 = &usart2;
28          serial6 = &usart6;
29          serial7 = &uart8;
30      };
```

Check whether there are any omissions in the modification and save the modified device tree file.

Save the modified file, compile the device tree, and use the compiled device tree to boot the factory system.

```
source /opt/st/stm32mp1/3.1-snapshot/environment-setup-cortexa7t2hf-neon-vfpv4-
ostl-linux-gnueabi make
stm32mp1_atk_defconfig
make dtbs
```

View serial port devices

```
root@ATK-MP157:~# ls /dev/ttySTM*
/dev/ttySTM0  /dev/ttySTM2  /dev/ttySTM4  /dev/ttySTM6
/dev/ttySTM1  /dev/ttySTM3  /dev/ttySTM5  /dev/ttySTM7
root@ATK-MP157:~#
```

The hardware testing method can refer to the USART1 method.

Chapter 7 2-way CAN multiplexing

The 157 development board can use up to two CAN FD channels. One CAN FD channel is configured by default on the 157 board and 157 MINI development board.

You can give us a reference, we only need one more CAN2.

There are multiple pins on the core board that can be reused as CAN2. For details, please refer to the STM32MP157A&D data sheet.pdf in the document.

Here we take the 157 development board as an example, and multiplex the U5_TX and U5_RX on the development board as the CAN2 function.

7.1 Modify the device tree file

Open the factory kernel source arch/arm/boot/dts/stm32mp157d-atk.dtsi file.

Add the node &m_can2, as shown below

```
&m_can2 {
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&m_can2_pins_a>;
        pinctrl-1 = <&m_can2_sleep_pins_a>;
        status = "okay";
};
```

Open the arch/arm/boot/dts/stm32mp15-pinctrl.dtsi file and add the m_can2_pins_a and

m_can2_sleep_pins_a pin configurations, as shown in the figure:

```
    m_can2_pins_a: m-can2-0 {
        pins1 {
            pinmux = <STM32_PINMUX('B', 13, AF9)>; /* CAN1_TX */
            slew-rate = <1>;
            drive-push-pull;
            bias-disable;
        };
        pins2 {
            pinmux = <STM32_PINMUX('B', 12, AF9)>; /* CAN1_RX */
            bias-disable;
        };
    };

    m_can2_sleep_pins_a: m_can2-sleep-0 {
        pins {
            pinmux = <STM32_PINMUX('B', 13, ANALOG)>, /* CAN1_TX */
                     <STM32_PINMUX('B', 12, ANALOG)>; /* CAN1_RX */
        };
    };
```

Search for the keywords 'B', 13 and 'B', 12, and comment out the related pins to prevent the IO from being occupied and causing the experiment to fail.

```
1334        pins2 {
1335        //  pinmux = <STM32_PINMUX('B', 12, AF8)>, /* USART3_RX */
1336        //       <STM32_PINMUX('B', 13, AF7)>; /* USART3_CTS_NSS */
1337            bias-disable;
1338        };
1339    };
1340
1341    usart3_idle_pins_b: usart3-idle-1 {
1342        pins1 {
1343            pinmux = <STM32_PINMUX('B', 10, ANALOG)>, /* USART3_TX */
1344                <STM32_PINMUX('G', 8, ANALOG)>; /* USART3_RTS */
1345        //       <STM32_PINMUX('B', 13, ANALOG)>; /* USART3_CTS_NSS */
1346        };
1347        pins2 {
1348        //  pinmux = <STM32_PINMUX('B', 12, AF8)>; /* USART3_RX */
1349            bias-disable;
1350        };
1351    };
```

记得补充双引号

```
usart3_sleep_pins_b: usart3-sleep-1 {
    pins {
        pinmux = <STM32_PINMUX('B', 10, ANALOG)>, /* USART3_TX */
            <STM32_PINMUX('G', 8, ANALOG)>; /* USART3_RTS */
        //   <STM32_PINMUX('B', 13, ANALOG)>, /* USART3_CTS_NSS */
        //   <STM32_PINMUX('B', 12, ANALOG)>; /* USART3_RX */
    };
};
```

双引号

```
114     dfsdm_clkout_pins_a: dfsdm-clkout-pins-0 {
115         pins {
116         //  pinmux = <STM32_PINMUX('B', 13, AF3)>; /* DFSDM_CKOUT */
117             bias-disable;
118             drive-push-pull;
119             slew-rate = <0>;
120         };
121     };
122
123     dfsdm_clkout_sleep_pins_a: dfsdm-clkout-sleep-pins-0 {
124         pins {
125         //  pinmux = <STM32_PINMUX('B', 13, ANALOG)>; /* DFSDM_CKOUT */
126         };
127     };
```

```
1298            pins2 {
1299            //  pinmux = <STM32_PINMUX('B', 12, AF8)>, /* USART3_RX */
1300            //         <STM32_PINMUX('I', 10, AF8)>; /* USART3_CTS_NSS */
1301                bias-disable;
1302            };
1303        };
1304
1305        usart3_idle_pins_a: usart3-idle-0 {
1306            pins1 {
1307                pinmux = <STM32_PINMUX('B', 10, ANALOG)>, /* USART3_TX */
1308                     <STM32_PINMUX('G', 8, ANALOG)>, /* USART3_RTS */
1309                     <STM32_PINMUX('I', 10, ANALOG)>; /* USART3_CTS_NSS */
1310            };
1311            pins2 {
1312            //  pinmux = <STM32_PINMUX('B', 12, AF8)>; /* USART3_RX */
1313                bias-disable;
```

```
1317        usart3_sleep_pins_a: usart3-sleep-0 {
1318            pins {
1319                pinmux = <STM32_PINMUX('B', 10, ANALOG)>, /* USART3_TX */
1320                     <STM32_PINMUX('G', 8, ANALOG)>, /* USART3_RTS */
1321                     <STM32_PINMUX('I', 10, ANALOG)>; /* USART3_CTS_NSS */
1322                //   <STM32_PINMUX('B', 12, ANALOG)>; /* USART3_RX */
1323            };
1324        };
```

双引号

Check to see if there are any pins that are not shielded.

Open the factory kernel source code arch/arm/boot/dts/stm32mp157d-atk.dtsi device tree file. Change the status of the node &uart5 to "disabled"

```
1071    &uart5 {
1072        pinctrl-names = "default", "sleep", "idle";
1073        pinctrl-0 = <&uart5_pins_a>;
1074        pinctrl-1 = <&uart5_sleep_pins_a>;
1075        pinctrl-2 = <&uart5_idle_pins_a>;
1076        /delete-property/dmas;
1077        /delete-property/dma-names;
1078        //status = "okay";
1079        status = "disabled";
```

Search for keywords 'B', 13 and 'B', 12, and comment out the related pins

```
422     uart5_pins_a: uart5-0 {
423         pins1 {
424         //      pinmux = <STM32_PINMUX('B', 13, AF14)>; /* UART5_TX */
425             bias-disable;
426             drive-push-pull;
427             slew-rate = <0>;
428         };
429         pins2 {
430         //      pinmux = <STM32_PINMUX('B', 12, AF14)>; /* UART5_RX */
431             bias-disable;
432         };
433     };
434
435     uart5_idle_pins_a: uart5-idle-0 {
436         pins1 {
437         //      pinmux = <STM32_PINMUX('B', 13, ANALOG)>; /* UART5_TX */
438         };
439         pins2 {
440         //      pinmux = <STM32_PINMUX('B', 12, AF14)>; /* UART5_RX */
441             bias-disable;
442         };
443     };
444
445     uart5_sleep_pins_a: uart5-sleep-0 {
446         pins {
447         //      pinmux = <STM32_PINMUX('B', 13, ANALOG)>, /* UART5_TX */
448         //          <STM32_PINMUX('B', 12, ANALOG)>; /* UART5_RX */
```

The CAN2 configuration is retained in the factory system kernel source code and does not need to be modified here.

```
545         pinctrl_flexcan2: flexcan2grp{
546             fsl,pins = <
547                 MX6UL_PAD_UART2_RTS_B__FLEXCAN2_RX  0x1b020
548                 MX6UL_PAD_UART2_CTS_B__FLEXCAN2_TX  0x1b020
549             >;
550         };
```

Search for pinctrl_flexcan2, find &flexcan2, and change status to okay, as follows:

```
286 &flexcan2 {
287     pinctrl-names = "default";
288     pinctrl-0 = <&pinctrl_flexcan2>;
289     xceiver-supply = <&reg_can_3v3>;
290     status = "okay";
291 };
```

Save the modified device tree file and compile the device tree.

```
source /opt/st/stm32mp1/3.1-snapshot/environment-setup-cortexa7t2hf-neon-vfpv4-ostl-linux-gnueabi
make uImage dtbs LOADADDR=0XC2000040 vmlinux -j16
```

Start the development board with the compiled device tree file.

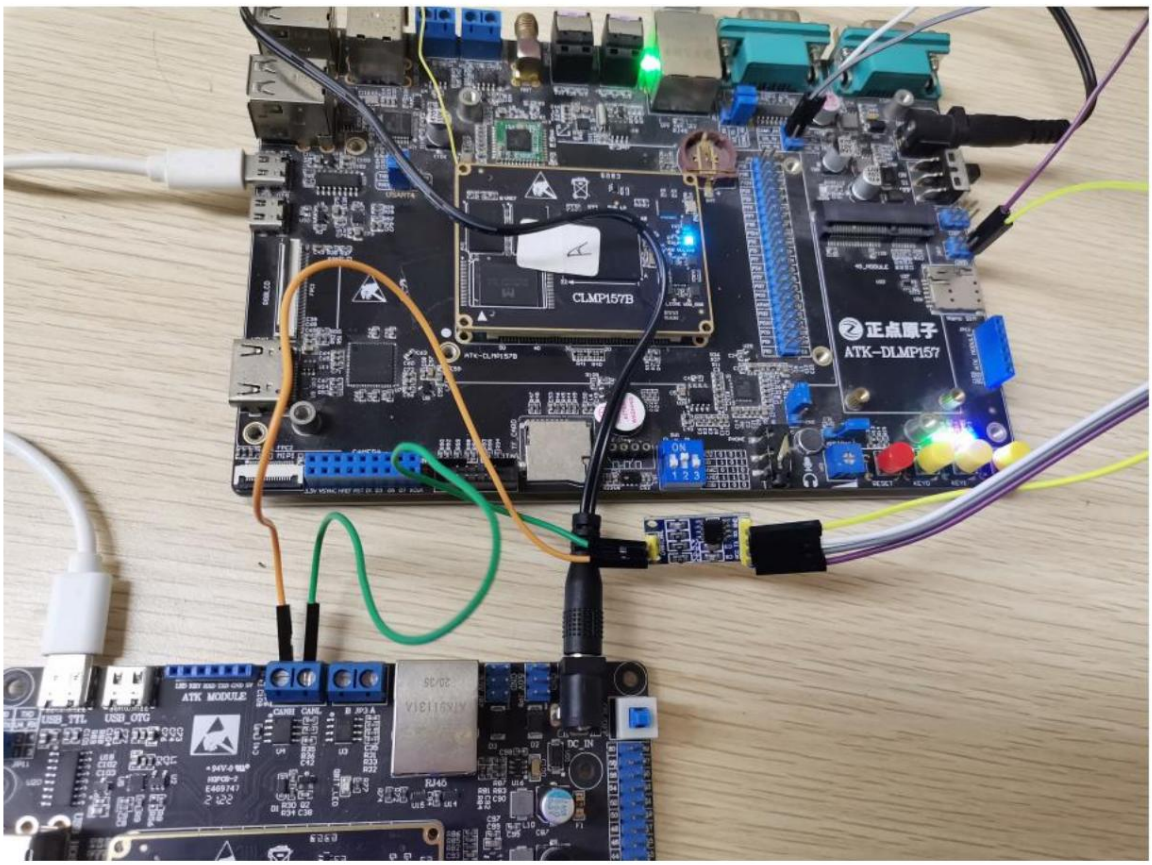
7.2 CAN Test

There is only one CAN chip on the development board, and a CAN chip module needs to be connected to U5_TX and U5_RX for testing.

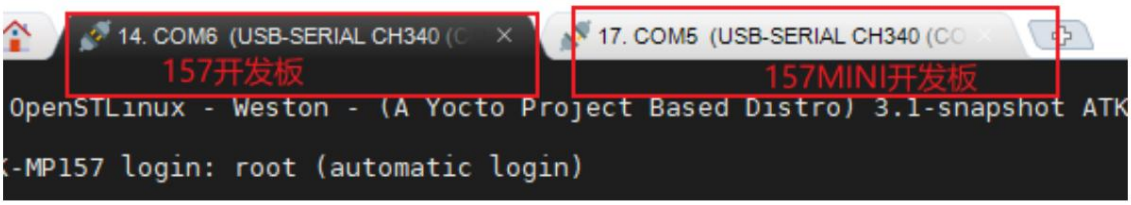
U5_TX is connected to the TX of the CAN module, and U5_RX is connected to the RX of the CAN module.

The CANH of the CAN module is connected to the CANH of the 157MINI board, and the CANL of the CAN module is connected to the CANL of the 157MINI board.

The CAN module is connected to 3.3V power supply and ground, as shown in the figure below.



Open two serial ports, one connected to the 157 development board and one connected to the 157MINI board. The author's is as follows:



Set the CAN device communication baud rate of can1 to 1000000, which means the maximum communication baud rate is 1MBit/s.

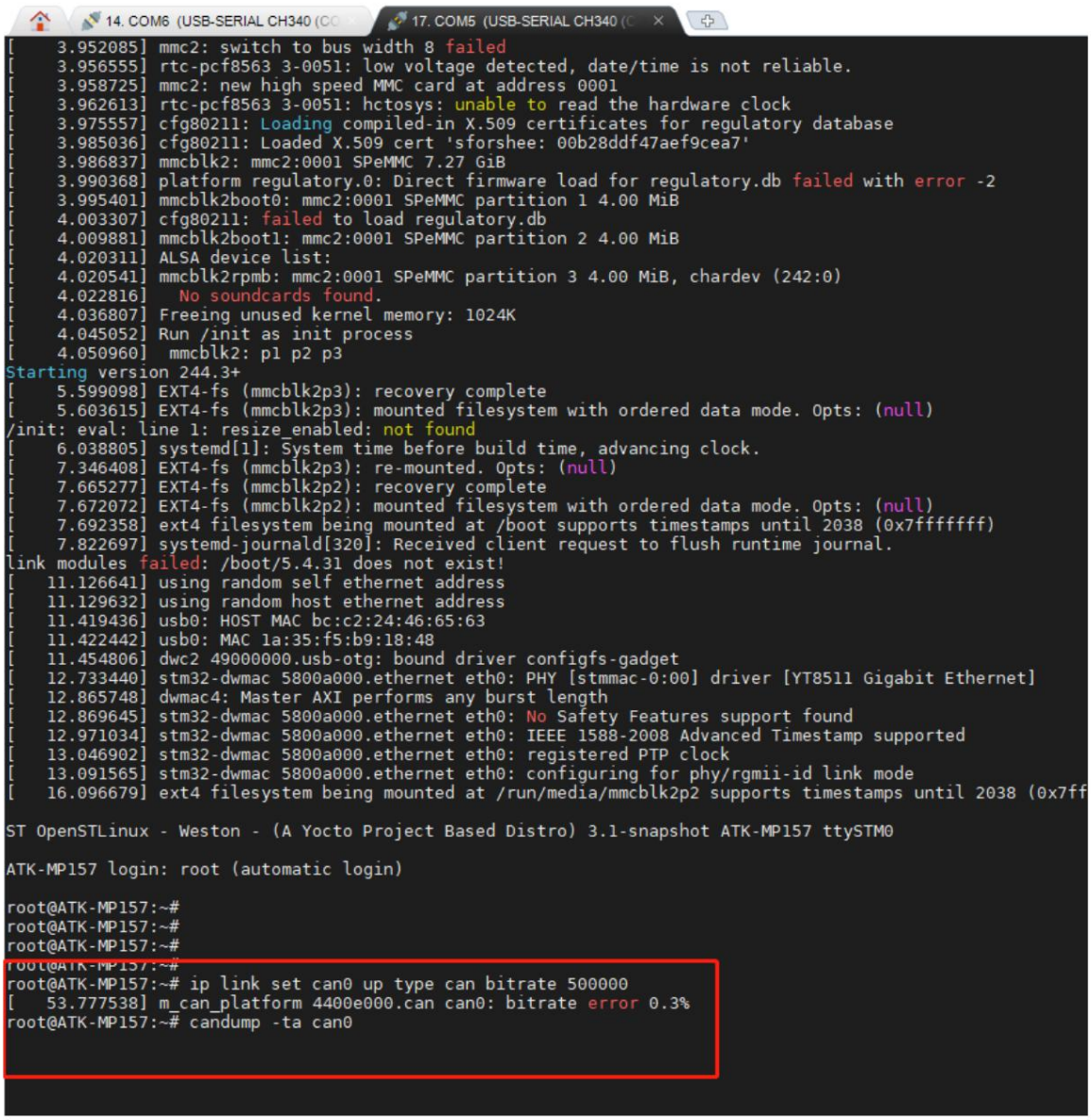
Set the test CAN1 device data bit rate to 50kBit/s. The 157MINI board also sets the same data bit rate.

```
ip link set can1 up type can bitrate 500000 //157MINI development board //157 Development Board
ip link set can0 up type can bitrate 500000 The 157MINI development board
```

uses the candump command to receive data from can1

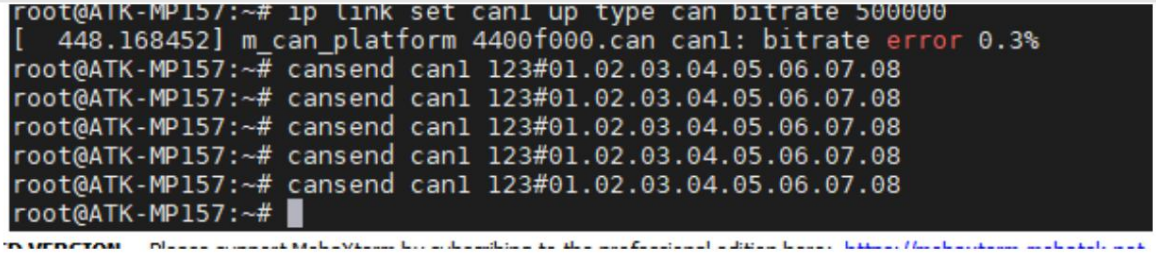
candump -ta can0 -ta: t development board

represents the printing time, a represents the opening of ASCII output. As shown in the figure below:



The 157 development board uses the cansend command to send data to the 157MINI board.

cansend can1 123#01.02.03.04.05.06.07.08



157MINI version acceptance information:

```
root@ATK-MP157:~#
root@ATK-MP157:~# ip link set can0 up type can bitrate 500000
[  504.072245] m_can_platform 4400e000.can can0: bitrate error 0.3%
root@ATK-MP157:~# candump -ta can0
 (1581091192.147824)  can0  123   [8]  01 02 03 04 05 06 07 08
 (1581091206.079547)  can0  123   [8]  01 02 03 04 05 06 07 08
 (1581091206.735518)  can0  123   [8]  01 02 03 04 05 06 07 08
 (1581091207.334840)  can0  123   [8]  01 02 03 04 05 06 07 08
 (1581091207.934949)  can0  123   [8]  01 02 03 04 05 06 07 08
```

The test method of CAN FD is similar. You can directly refer to the corresponding CAN test chapter of "ÿOn-point AtomÿSTM32MP157 Quick Experience" in the material for testing, and will not be repeated here.

Chapter 8 Multiple ADC Multiplexing

According to the reference manual and data sheet, the 157 core board can reuse up to 40 ADCs. The chip reference manual has the following descriptions for ADC Signal Description:

Note: The maximum absolute value of the ADC acquisition voltage is 3.3V. Please do not exceed 3.3V.

Core board pins	Pin Type	Function
PC3	I/O	ADC1_INP13, ADC1_INN12
PA3	I/O	ADC1_INP15
PC2	I/O	ADC1_INP12,ADC1_INN11
ANA0	A	ADC1_INP0,ADC1_INN1, ADC2_INP0,ADC2_INN1
PA0	I/O	ADC1_INP16
ANA1	A	ADC1_INP1,ADC2_INP1
PA1	I/O	ADC1_INP17,ADC1_INN16
PA5	I/O	ADC1_INP19,ADC1_INN18 ADC2_INP19,ADC2_INN18
PA4	I/O	ADC1_INP18,ADC2_INP18
PA2	I/O	ADC1_INP14
PC1	I/O	ADC1_INP11,ADC1_INN10 ADC2_INP11,ADC2_INN10
PB0	I/O	ADC1_INP9,ADC1_INN5 ADC2_INP9,ADC2_INN5
PB1	I/O	ADC1_INP5,ADC2_INP5
PF14	I/O	ADC2_INP6,ADC2_INN2
PF13	I/O	ADC2_INP2
PC5	I/O	ADC1_INP8,ADC1_INN4 ADC2_INP8,ADC2_INN4
PC4	I/O	ADC1_INP4,ADC2_INP4
PF12	I/O	ADC1_INP6, ADC1_INN2
PF11	I/O	ADC1_INP2
PA7	I/O	ADC1_INP7, ADC1_INN3, ADC2_INP7, ADC2_INN3
PA6	I/O	ADC1_INP3, ADC2_INP3
PC0	I/O	ADC1_INP10, ADC2_INP10

Note: ADC2_INP12-17 cannot be reused

Signal name	Source/destination
ADC2 V_{INP}[12]	V_{SENSE} (output voltage from internal temperature sensor)
ADC2 V_{INP}[13]	V_{REFINT} (output voltage from internal reference voltage)
ADC2 V_{INP}[15]	V_{BAT}/4 (external battery voltage supply voltage)
ADC2 V_{INP}[14]	V_{DDCORE} (core logic supply voltage)
ADC2 V_{INP}[16]	dac_out1
ADC2 V_{INP}[17]	dac_out2
adc_dat[15:0]	dfsdm_dat_adc[15:0]

This chapter takes PF13 of the 157MINI development board as an example and reuses ADC2_INP2. The configuration methods of other ADCs are similar.

8.1 ADC Modify the device tree and test

Open the factory kernel source code arch/arm/boot/dts/stm32mp157d-atk.dtsi device tree file. Add the node information of ADC2 under the &adc node, as shown below:

```
494  &adc {
495      /* ADC1 & ADC2 common resources */
496      pinctrl-names = "default";
497      pinctrl-0 = <&adc1_in6_pins_b>;
498      pinctrl-1 = <&adc2_in2_pins_b>;
499      vdd-supply = <&vdd>;
500      vdda-supply = <&vdd>;
501      vref-supply = <&vdd>;
502
503      status = "okay";
504
505      adc1: adc@0 {
506          /* private resources for ADC1 */
507          st,adc-channels = <19>;
508          st,min-sample-time-nsecs = <10000>;
509          status = "okay";
510      };
511
512      adc2: adc@100 {
513          /* private resources for ADC1 */
514          st,adc-channels = <2>;
515          st,min-sample-time-nsecs = <10000>;
516          status = "okay";
517      };
518  };
```

Add adc2_in2_pins_b node information under the &pinctrl node and add the ADC definition of PF13, as shown below:

```
303  &pinctrl {
304      dac_ch1_pins_a: dac-ch1 {
305          pins {
306              pinmux = <STM32_PINMUX('A', 4, ANALOG)>; /* configure 'PA4' as AI
307          };
308      };
309
310      adc1_in6_pins_b: adc1-in6 {
311          pins {
312              pinmux = <STM32_PINMUX('A', 5, ANALOG)>;
313          };
314      };
315
316      adc2_in2_pins_b: adc2-in2 {
317          pins {
318              pinmux = <STM32_PINMUX('F', 13, ANALOG)>;
319          };
320      };
321
```

Open the factory kernel source code arch/arm/boot/dts/stm32mp15-pinctrl.dtsi device tree file. Search for the keyword 'F', 13 to shield the related IO.

```
15    adc12_ain_pins_a: adc12-ain-0 {
16        pins {
17            pinmux = <STM32_PINMUX('C', 3, ANALOG)>, /* ADC1 in13 */
18                     <STM32_PINMUX('F', 12, ANALOG)>, /* ADC1 in6 */
19        //           <STM32_PINMUX('F', 13, ANALOG)>, /* ADC2 in2 */
20                     <STM32_PINMUX('F', 14, ANALOG)>; /* ADC2 in6 */
21        };
22    };
```

```
141   dfsdm_data3_pins_a: dfsdm-data3-pins-0 {
142       pins {
143       //  pinmux = <STM32_PINMUX('F', 13, AF6)>; /* DFSDM_DATA3 */
144       };
145   };
146
147   dfsdm_data3_sleep_pins_a: dfsdm-data3-sleep-pins-0 {
148       pins {
149       //  pinmux = <STM32_PINMUX('F', 13, ANALOG)>; /* DFSDM_DATA3 */
150       };
151   };
```

After the modification is completed, save the file and compile the device tree.

source /opt/st/stm32mp1/3.1-snapshot/environment-setup-cortexa7t2hf-neon-vfpv4-ostl-linux-gnueabi make

stm32mp1_atk_defconfig make uImage dtbs

LOADADDR=0XC2000040 vmlinux -j16 Start the development board with the generated device tree file and enter the development board file system to test ADC. The following figure shows the connection method of the 157MINI board.

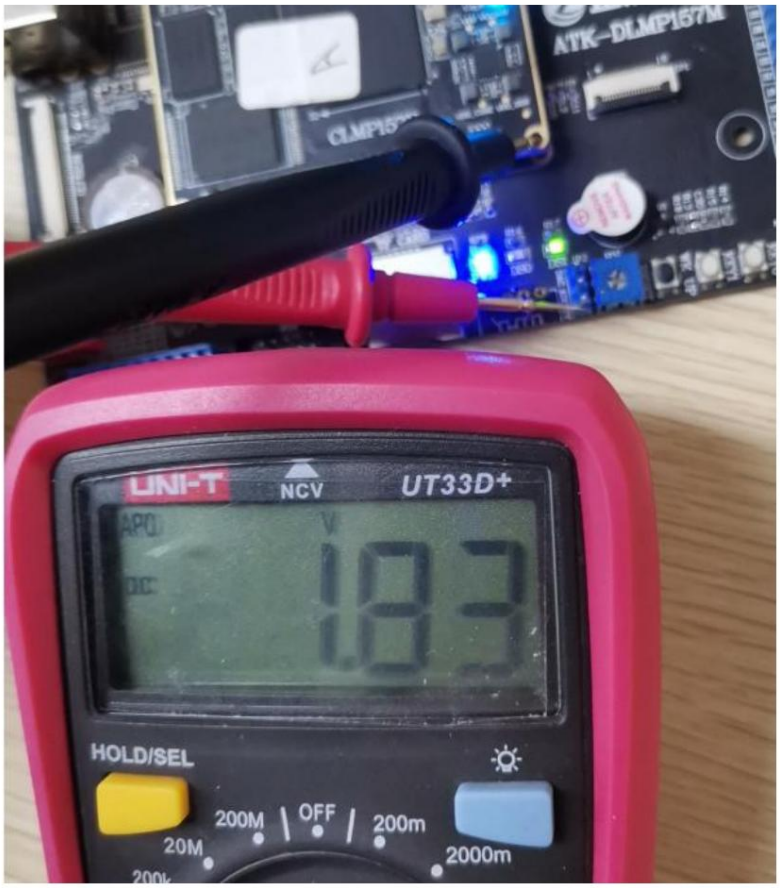
Execute the following command to obtain the ADC driver device value. cd /sys/bus/iio/devices/iio\:device1/ cat in_voltage2_raw cat in_voltage_scale //voltage1 is ADC1, voltage2 is ADC2

```
root@ATK-MP157:~# cd /sys/bus/iio/devices/iio\:device1
root@ATK-MP157:/sys/devices/platform/soc/48003000.adc/48003000.adc:adc@100/iio:device1# cat in_voltage2_raw
37449
root@ATK-MP157:/sys/devices/platform/soc/48003000.adc/48003000.adc:adc@100/iio:device1# cat in_voltage_scale
0.050354003
root@ATK-MP157:/sys/devices/platform/soc/48003000.adc/48003000.adc:adc@100/iio:device1#
```

Raw is the original value read out, scale is the conversion value (this value is related to the reference voltage), and the actual value needs to be obtained using the following formula.

real_value = (raw + offset) * scale //Offset defaults to 0 //

37449*0.050354003=1,885.707058347 Dividing by 1000 equals 1.88 (error is normal)



1

8.2 DAC Modify the device tree and test

According to the reference manual and data sheet, the 157 core board can reuse up to 2 DACs. The chip reference manual has the following descriptions for DAC Description of the signal.

Pin Number				Pin name (function after reset)	Pin type	I/O structure	Notes	Pin functions	
TFBGA257	LFBGA354	TFBGA361	LFBGA448					Alternate functions	Additional functions
1H1	P4	V3	U5	PA5	I/O	TT_ha	-	TIM2_CH1/TIM2_ETR, TIM8_CH1N, SAI4_CK1, SPI1_SCK/I2S1_CK, SPI6_SCK, SAI4_MCLK_A, LCD_R4, EVENTOUT	ADC1_INP19, ADC1_INN18, ADC2_INP19, ADC2_INN18, DAC_OUT2
1J1	R4	V4	V6	PA4	I/O	TT_a	-	HDP0, TIM5_ETR, SAI4_D2, SPI1_NSS/I2S1_WS, SPI3_NSS/I2S3_WS, USART2_CK, SPI6_NSS, SAI4_FS_A, DCMI_HSYNC, LCD_VSYNC, EVENTOUT	ADC1_INP18, ADC2_INP18, DAC_OUT1

Open the factory kernel source code arch/arm/boot/dts/stm32mp157d-atk.dtsi device tree file. Add the node information of ADC2 under the &dac node, as shown below:

```
542  &dac {
543      pinctrl-names = "default";
544      pinctrl-0 = <&dac_ch1_pins_a>;                  /* Use PA4 and PA5 pin a
545      pinctrl-0 =<&dac_ch2_pins_b>;
546      vref-supply = <&v3v3>;                          /* Example to use VREFBU
547      status = "okay";                                /* Enable the DAC block
548      dac1: dac@1 {
549          status = "okay";                            /* Enable DAC1 */
550      };
551
552      dac2: dac@2 {
553          status = "okay";                            /* Enable DAC2 */
554      };
555  };
```

Add adc2_in2_pins_b node information under the &pinctrl node and add the ADC definition of PF13, as shown below:

```
303  &pinctrl {
304      dac_ch1_pins_a: dac-ch1 {
305          pins {
306              pinmux = <STM32_PINMUX('A', 4, ANALOG)>; /* configure 'PA4' a
307          };
308      };
309
310      dac_ch2_pins_b: dac-ch2 {
311          pins {
312              pinmux = <STM32_PINMUX('A', 5, ANALOG)>; /* configure 'PA5' a
313          };
314      };
```

Search for keyword 'A', 5 to shield related IO.

```
316      adc1_in6_pins_b: adc1-in6 {
317          pins {
318      //          pinmux = <STM32_PINMUX('A', 5, ANALOG)>;
319          };
320      };
```

After the modification is completed, save the file and compile the device tree.

```
source /opt/st/stm32mp1/3.1-snapshot/environment-setup-cortexa7t2hf-neon-vfpv4-ostl-linux-gnueabi
```

```
make stm32mp1_atk_defconfig make uImage
dtbs LOADADDR=0XC2000040 vmlinux -j16 Use the generated device tree file to
```

start the development board and enter the development board file system to test the DAC.

The following figure shows the

connection method of the 157MINI board. Execute the

following command to obtain the ADC driver device value. cd /

```
sys/bus/iio/devices/iio\:device1/
cat in_voltage2_raw cat in_voltage_scale
```

Chapter 9 Multiple PWM Multiplexing

According to the chip manual and data sheet, the 157 core board can reuse up to 32 PWM channels.

Signal description: STM32MP157 has many PWMs, all of which are generated by timers:

TIM1/TIM8: These two are 16-bit advanced timers, mainly used for motor control. These two timers support PWM

Model, each timer supports 4-channel PWM signals.

TIM2/TIM3/TIM4/TIM5: These 4 are general-purpose timers, TIM3/TIM4 is a 16-bit timer, TIM2/TIM5 is a 16-bit timer.

It is a 32-bit timer. These 4 timers also support PWM output, and each timer supports 4-channel PWM signals.

TIM12/TIM13/TIM14: These three are 16-bit general-purpose timers. TIM12 supports 2 channels of PWM signal.

Each of the two timers, TIM13/TIM14, only supports 1 channel of PWM signal.

TIM15/TIM16/TIM17: These three are also 16-bit general-purpose timers. TIM15 supports 2-channel PWM

Signal, each timer of TIM16/TIM17 supports 1 channel PWM signal.

You can refer to the "STM32MP157A&D Data Sheet" for the pin multiplexing function to configure the PWM signal.

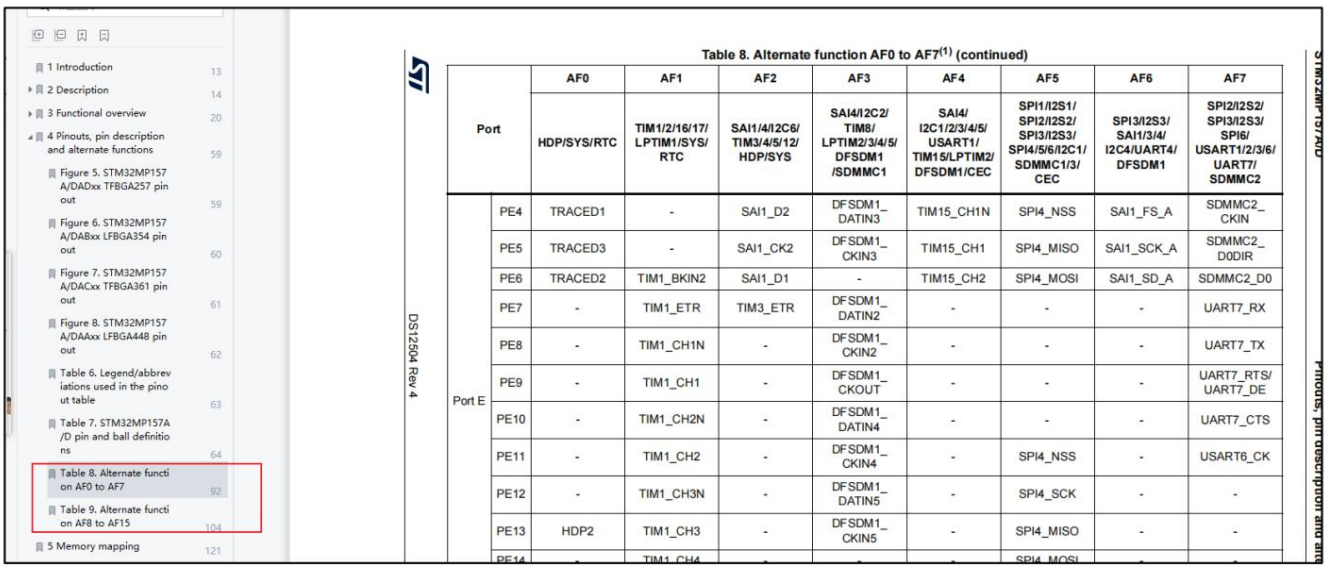


Table 8. Alternate function AF0 to AF7(1) (continued)

Port		AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7
		HDP/SYS/RTC	TIM1/2/16/17/ LPTIM1/SYS/ RTC	SAI1/4/I2C6/ TIM3/4/5/12/ HDP/SYS	SAI4/I2C2/ TIM8/ LPTIM2/3/4/5/ DFSDM1 /SDMMC1	SAI4/ I2C1/2/3/4/5/ USART1/ TIM15/LPTIM2/ DFSDM1/CEC	SPI1/I2S1/ SPI2/I2S2/ SPI3/I2S3/ SPI4/5/6/I2C1/ SDMMC1/3/ CEC	SPI3/I2S3/ SAI1/3/4/ I2C4/UART4/ DFSDM1	SPI2/I2S2/ SPI3/I2S3/ SPI6/ USART1/2/3/6/ UART7/ SDMMC2
Port E	PE4	TRACED1	-	SAI1_D2	DFSDM1_ DATIN3	TIM15_CH1N	SPI4_NSS	SAI1_FS_A	SDMMC2_ CKIN
	PE5	TRACED3	-	SAI1_CK2	DFSDM1_ CKIN3	TIM15_CH1	SPI4_MISO	SAI1_SCK_A	SDMMC2_ D0DIR
	PE6	TRACED2	TIM1_BKIN2	SAI1_D1	-	TIM15_CH2	SPI4_MOSI	SAI1_SD_A	SDMMC2_D0
	PE7	-	TIM1_ETR	TIM3_ETR	DFSDM1_ DATIN2	-	-	-	UART7_RX
	PE8	-	TIM1_CH1N	-	DFSDM1_ CKIN2	-	-	-	UART7_TX
	PE9	-	TIM1_CH1	-	DFSDM1_ CKOUT	-	-	-	UART7_RTS/ UART7_DE
	PE10	-	TIM1_CH2N	-	DFSDM1_ DATIN4	-	-	-	UART7_CTS
	PE11	-	TIM1_CH2	-	DFSDM1_ CKIN4	-	SPI4_NSS	-	USART6_CK
	PE12	-	TIM1_CH3N	-	DFSDM1_ DATIN5	-	SPI4_SCK	-	-
	PE13	HDP2	TIM1_CH3	-	DFSDM1_ CKIN5	-	SPI4_MISO	-	-
	PE14	-	TIM1_CH4	-	-	-	SPI4_MOSI	-	-

DS12504 Rev 4

STM32MP157A/D

Pinouts, pin description and alte

You will find that there are TIM1_CH1 and TIM1_CH1N. The difference between them is that they drive the upper and lower power tubes.

In other words, the two are complementary outputs and are both used to configure the synchronous PWM mode of TIM1.

9.1 TIM1 (4-way)

9.1.1 Modify the device tree file

PB13	-	TIM1_CH1N	-	DFSDM1_ CKOUT	LPTIM2_OUT	SPI2_SCK/ I2S2_CK	DFSDM1_ CKIN1	USART3_CTS/ USART3_NSS
PJ11	-	TIM1_CH2	-	TIM8_CH2N	-	SPI5_MISO	-	-
PA10	-	TIM1_CH3	-	-	-	SPI3_NSS/ I2S3_WS	-	USART1_RX
PA11	-	TIM1_CH4	I2C6_SCL	-	I2C5_SCL	SPI2_NSS/ I2S2_WS	UART4_RX	USART1_CTS/ USART1_NSS

channels directly here, of which TIM1_CH3 is already configured by the factory system. We only need to configure the other three road.

Open the factory system kernel source device tree file arch/arm/boot/dts/stm32mp157d-atk.dtsi, the factory system By default, one pwm is configured. We can refer to this to add other pwm configurations. Find the pwm1_pins_a and pwm1_sleep_pins_a node information as follows:

```
451     pwm1_pins_a: pwm1-0 {
452         pins {
453             pinmux = <STM32_PINMUX('A', 10, AF1)>; /* TIM1_CH3 */
454             bias-pull-down;
455             drive-push-pull;
456             slew-rate = <0>;
457         };
458     };
459
460     pwm1_sleep_pins_a: pwm1-sleep-0 {
461         pins {
462             pinmux = <STM32_PINMUX('A', 10, ANALOG)>; /* TIM1_CH3 */
463         };
464     };
```

Change it to the following:

```
451     pwm1_pins_a: pwm1-0 {
452         pins {
453             pinmux = <STM32_PINMUX('A', 10, AF1)>, /* TIM1_CH3 */
454                      <STM32_PINMUX('B', 13, AF1)>,   /* TIM1_CH1N */
455                      <STM32_PINMUX('J', 11, AF1)>,   /* TIM1_CH2 */
456                      <STM32_PINMUX('A', 11, AF1)>;   /* TIM1_CH4 */
457             bias-pull-down;
458             drive-push-pull;
459             slew-rate = <0>;
460         };
461     };
462
463     pwm1_sleep_pins_a: pwm1-sleep-0 {
464         pins {
465             pinmux = <STM32_PINMUX('A', 10, ANALOG)>, /* TIM1_CH3 */
466                      <STM32_PINMUX('B', 13, ANALOG)>,   /* TIM1_CH1N */
467                      <STM32_PINMUX('J', 11, ANALOG)>,   /* TIM1_CH2 */
468                      <STM32_PINMUX('A', 11, ANALOG)>;   /* TIM1_CH4 */
469         };
470     };
```

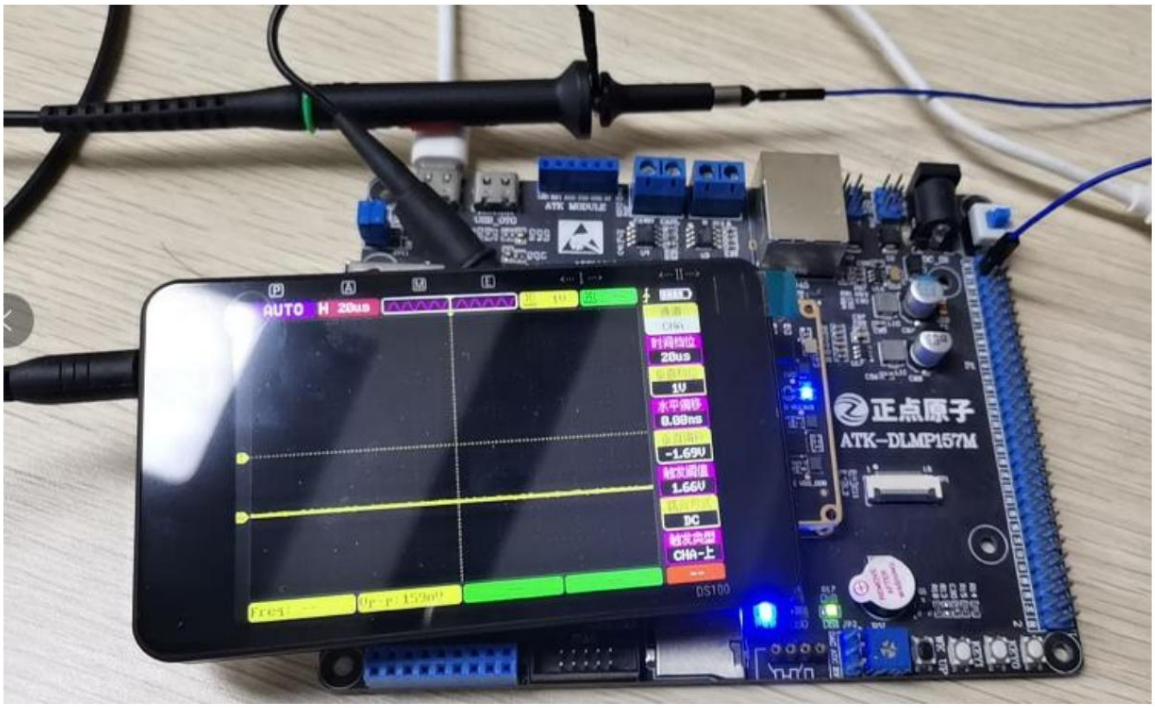
Find the pins PA10, PB13, PJ11, and PA11 in the file, and shield the pins used by other peripherals, as shown in the following figure:

```
422     uart5_pins_a: uart5-0 {
423         pins1 {
424     //          pinmux = <STM32_PINMUX('B', 13, AF14)>; /* UART5_TX */
425             bias-disable;
426             drive-push-pull;
427             slew-rate = <0>;
428         };
429         pins2 {
430             pinmux = <STM32_PINMUX('B', 12, AF14)>; /* UART5_RX */
431             bias-disable;
432         };
433     };
434
435     uart5_idle_pins_a: uart5-idle-0 {
436         pins1 {
437         //    pinmux = <STM32_PINMUX('B', 13, ANALOG)>; /* UART5_TX */
438         };
439         pins2 {
440             pinmux = <STM32_PINMUX('B', 12, AF14)>; /* UART5_RX */
441             bias-disable;
442         };
443     };
444
445     uart5_sleep_pins_a: uart5-sleep-0 {
446         pins {
447         //   pinmux = <STM32_PINMUX('B', 13, ANALOG)>, /* UART5_TX */
448         //          <STM32_PINMUX('B', 12, ANALOG)>; /* UART5_RX */
449        };
450   };
```

After the modification is completed, save the file and compile the device tree.

```
source /opt/st/stm32mp1/3.1-snapshot/environment-setup-cortexa7t2hf-neon-
vfpv4-ostl-linux-gnueabi make
stm32mp1_atk_defconfig make uImage dtbs
LOADADDR=0XC2000040 vmlinux -j16
```

Use the new device tree to copy to the board and replace the factory system's device tree, and use the new device tree to start the system. For the convenience of testing, the MINI 157 development board is used. Connect the pins to the oscilloscope and use the oscilloscope to view the PWM waveform. The board hardware wiring is shown in the figure below: (The mini version of the Zhengdian Atom DS100 handheld digital oscilloscope is used, and the Dupont line is connected to PB13, corresponding to pin 58 of the baseboard)



On the development board system, enter the directory /sys/class/pwm, as shown in the figure:

```
root@ATK-MP157:~# cd /sys/class/pwm/
root@ATK-MP157:/sys/class/pwm# ls
pwmchip0  pwmchip4  pwmchip8
root@ATK-MP157:/sys/class/pwm#
```

Note! There are three files in the figure, but we don't know which one is the file corresponding to TIM1. We can check one by one to see which one's corresponding address is consistent with the starting address of the TIM1 timer register. You can check the "STM32MP157 Reference Manual" to find out

0x44003400 - 0x44003FFF	3 KB	Reserved	-
0x44003000 - 0x440033FF	1 KB	USART6	*USART registers*
0x44001400 - 0x44002FFF	7 KB	Reserved	-
0x44001000 - 0x440013FF	1 KB	TIM 8	*TIM1/TIM8 registers*
0x44000400 - 0x44000FFF	3 KB	Reserved	-
0x44000000 - 0x440003FF	1 KB	TIM 1	*TIM1/TIM8 registers*

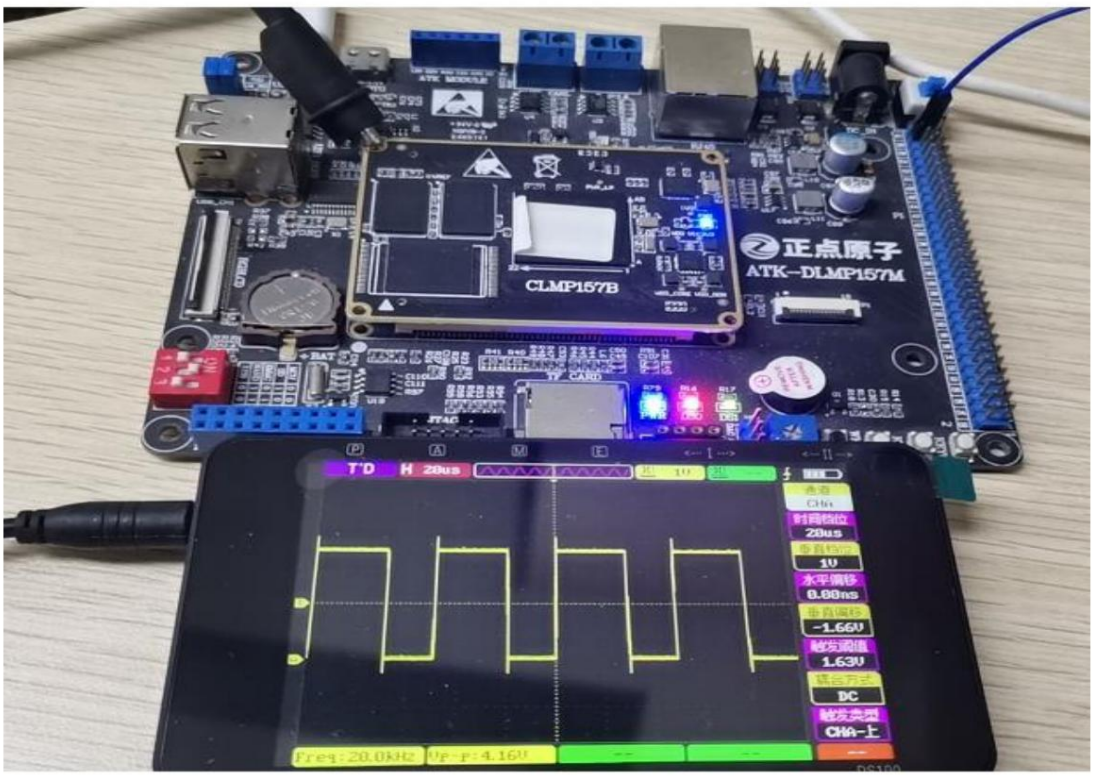
```
root@ATK-MP157:/sys/class/pwm# cd pwmchip8/
root@ATK-MP157:/sys/devices/platform/soc/44000000.timer/44000000.timer:pwm/pwm/pwmchip8# ls
device  export  npwm  power  pwm3  subsystem  uevent  unexport
root@ATK-MP157:/sys/devices/platform/soc/44000000.timer/44000000.timer:pwm/pwm/pwmchip8#
```

It can be known that the file corresponding to TIM1 is pwmchip8.

The following is an example of TIM1_CH1N: //Call

```
echo 0 > export echo 50000                    out the pwm0 subdirectory, which corresponds to CH1
> pwm0/period //Set the period value echo 20000 > pwm0/duty_cycle //Set the
duty cycle echo 1 > pwm0/enable //Enable channel 1
```

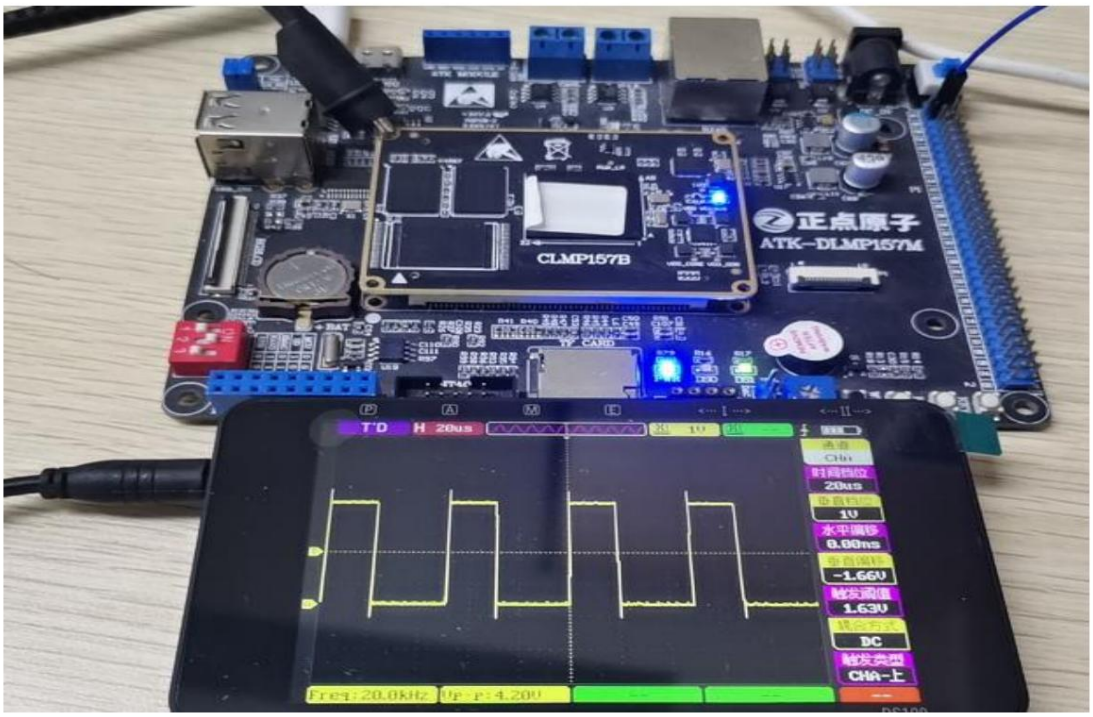
The experimental results are shown in the figure below:



It can be seen that the PWM frequency is 20KHz and the duty cycle is 60%. Why is it not 40% as set? Because it is TIM_CH1N, not TIM_CH1, so the experimental effect is correct. The above situation can be changed to 40% through the polarity inversion command. The command is as follows

```
echo "inversed" > pwm0/polarity The experimental effect is
```

shown in the figure below:



If you want to restore the original polarity, enter the following command

```
echo "normal" > pwm0/polarity
```

The other three channels are not shown one by one, they are the same as channel 1

above. Special attention should be paid to the fact that since a timer has 4 channels of PWM, and these 4 channels of PWM can only be set to the same period,

if you want multiple channels of PWM signals with different periods, you must use multiple different TIMs!!!

9.2 TIM2 (4-way)

9.2.1 Modifying the kernel and device tree

PA5	-	TIM2_CH1/ TIM2_ETR	-	TIM8_CH1N	SAI4_CK1	SPI1_SCK/I2S1 _CK	-	-
PB10	-	TIM2_CH3	-	LPTIM2_IN1	I2C2_SCL	SPI2_SCK/ I2S2_CK	DFSDM1_ DATIN7	USART3_TX
PA3	-	TIM2_CH4	TIM5_CH4	LPTIM5_OUT	TIM15_CH2	-	-	USART2_RX

 TIM2_CH2 has only PA1 and PB3 pins, but PA1 is used by the Gigabit chip and PB3 is used by EMMC, so CH2 cannot be cut out. Open the factory system

kernel source code device tree file arch/arm/boot/dts/stm32mp157d-atk.dtsi, and refer to the node &timers1 to configure it in the root node directory.

```
1052  &timers2 {
1053      status = "okay";
1054      /* spare all DMA channels since they are not needed for PWM output */
1055      /delete-property/dmas;
1056      /delete-property/dma-names;
1057      pwm2: pwm {
1058          pinctrl-0 = <&pwm2_pins_a>;
1059          pinctrl-1 = <&pwm2_sleep_pins_a>;
1060          pinctrl-names = "default", "sleep";
1061          #pwm-cells = <2>;
1062          status = "okay";
1063      };
1064  };
```

Add pwm2_pins_a and pwm2_sleep_pins_a nodes under the &pinctrl node and configure the pin information as follows:

```
472      pwm2_pins_a: pwm2-0 {
473          pins {
474              pinmux = <STM32_PINMUX('A', 5, AF1)>, /* TIM2_CH1 */
475                       <STM32_PINMUX('B', 10, AF1)>,   /* TIM2_CH3 */
476                       <STM32_PINMUX('A', 3, AF1)>;   /* TIM2_CH4 */
477              bias-pull-down;
478              drive-push-pull;
479              slew-rate = <0>;
480          };
481      };
482
483      pwm2_sleep_pins_a: pwm2-sleep-0 {
484          pins {
485              pinmux = <STM32_PINMUX('A', 5, ANALOG)>, /* TIM2_CH1 */
486                       <STM32_PINMUX('B', 10, ANALOG)>,   /* TIM2_CH3 */
487                       <STM32_PINMUX('A', 3, ANALOG)>;   /* TIM2_CH4 */
488          };
489      };
```

Remember: Check if the IO pins are occupied by other peripherals and need to be shielded. After the

modification is completed, save the file and compile the device tree.

```
source /opt/st/stm32mp1/3.1-snapshot/environment-setup-cortexa7t2hf-neon-vfpv4-ostl-linux-gnueabi make
stm32mp1_atk_defconfig make uImage
dtbs LOADADDR=0XC2000040 vmlinux -j16
```

Use the new device tree to copy to the board and replace the factory system device tree, and start the development board. The specific test method can be

Refer to the method of TIM1 in this section.

9.3 Cut AUDIO\JTAG to make PWM6\7\8

TIM3 (4-way)

PA6	-	TIM1_BKIN	TIM3_CH1	TIM8_BKIN	SAI4_CK2	SPI1_MISO/ I2S1_SDI	-	-
PC7	HDP4	-	TIM3_CH2	TIM8_CH2	DFSDM1_ DATIN3	-	I2S3_MCK	USART6_RX
PC8	TRACED0	-	TIM3_CH3	TIM8_CH3	-	-	UART4_TX	USART6_CK
PC9	TRACED1	-	TIM3_CH4	TIM8_CH4	I2C3_SDA	I2S_CKIN	-	-

Please refer to the TIM1 or TIM2 chapter for the operation method, so I will not repeat them here. The above 4 routes are OK after actual testing.

TIM4(4-way)

PB6	-	TIM16_CH1N	TIM4_CH1	-	I2C1_SCL	CEC	I2C4_SCL	USART1_TX
PD13	-	LPTIM1_OUT	TIM4_CH2	-	I2C4_SDA	I2C1_SDA	I2S3_MCK	-
PB8	HDP6	TIM16_CH1	TIM4_CH3	DFSDM1_ CKIN7	I2C1_SCL	SDMMC1_ CKIN	I2C4_SCL	SDMMC2_ CKIN
PB9	HDP7	TIM17_CH1	TIM4_CH4	DFSDM1_ DATIN7	I2C1_SDA	SPI2_NSS/ I2S2_WS	I2C4_SDA	SDMMC2_ CDIR

To use TIM4, the screen backlight must be disabled first, because the CH2 output of TIM4 is a PWM signal.

Open the arch/arm/bood/dts/stm32mp157d-atk.dtsi file and turn off the screen backlight signal.

```
129     panel_backlight: panel-backlight {
130         compatible = "pwm-backlight";
131         pwms = <&pwm4 1 5000000>;
132         brightness-levels = <0 4 8 16 32 64 128 255>;
133         power-supply = <&v3v3>;
134         default-brightness-level = <7>;
135         //status = "okay";
136         status = "disabled";
137     };
```

Please refer to the TIM1 or TIM2 chapter for the operation method, so I will not repeat them here. The above 4 paths are OK after actual testing.

TIM5(4-way)

PH10	-	-	TIM5_CH1	-	I2C4_SMBA	I2C1_SMBA	-	-
PH11	-	-	TIM5_CH2	-	I2C4_SCL	I2C1_SCL	-	-
PH12	HDP2	-	TIM5_CH3	-	I2C4_SDA	I2C1_SDA	-	-
PA3	-	TIM2_CH4	TIM5_CH4	LPTIM5_OUT	TIM15_CH2	-	-	USART2_RX

TIM8(4 channels)

PI5	-	-	-	TIM8_CH1	-	-	-	-
PI6	-	-	-	TIM8_CH2	-	-	-	-
PI7	-	-	-	TIM8_CH3	-	-	-	-

PI2	-	-	-	TIM8_CH4	-	SPI2_MISO/ I2S2_SDI	-	-

TIM12 (two channels)

PH6	-	-	TIM12_CH1	-	I2C2_SMBA	SPI5_SCK	-	-
PH9	-	-	TIM12_CH2	-	I2C3_SMBA	-	-	-

TIM13 (one channel)

PF8	-	TIM13_CH1	QUADSPI_ BK1_IO0	-	-	-	-	EVENTOUT

TIM14 (one channel)

PF9	-	TIM14_CH1	QUADSPI_ BK1_IO1	-	-	-	-	EVENTOUT

TIM15 (two channels)

PE5	TRACED3	-	SAI1_CK2	DFSDM1_ CKIN3	TIM15_CH1	SPI4_MISO	SAI1_SCK_A	SDMMC2_ D0DIR
PE6	TRACED2	TIM1_BKIN2	SAI1_D1	-	TIM15_CH2	SPI4_MOSI	SAI1_SD_A	SDMMC2_D0

TIM16 (one channel)

PF6	-	TIM16_CH1	-	-	-	SPI5_NSS	SAI1_SD_B	UART7_RX

TIM17 (one channel)

PF7	-	TIM17_CH1	-	-	-	SPI5_SCK	SAI1_MCLK_B	UART7_TX

Please refer to the TIM1 or TIM2 chapter for the operation method, and I will not repeat them here. The above is OK after actual testing.