0 aic8800 sdk user guide

This document is a software development guide for the aic8800 series

chip SDK (applicable chips: AIC8800M/AIC8800A/AIC8800F/AIC8800MC/AIC8800FC/AIC8800M40B/AIC8800M80)

Contents:

Version Information:

Date	Version	Notes
2021-01-08	1.1.00	Initial version
2021-01-15	1.1.01	Add bin file burning
2021-01-16	1.1.02	Update test case introduction
2021-01-22	1.1.03	Update bin file burning
2021-03-01	1.1.04	Updated test case & wifi application introduction
2021-05-28	1.1.05	Added AIC8800A EVB & Problem Location section
2021-10-28	1.1.06	Update modules & add BT related introduction
2021-11-18	1.1.07	Add GPIO function selection
2021-11-26	1.1.08	Added VCORE09 configuration method
2021-12-07	1.1.09	Add chip system architecture and memory peripheral mapping
2021-12-14	1.1.10	Updated chip system architecture diagram
2022-02-09	1.1.11	Added PWRKEY timing diagram
2022-04-19	1.1.12	Modify the BT/BLE application layer related introduction
2023-02-06	1.1.13	Update source code compilation and WiFi application introduction
2023-03-14	1.2.00	Added AIC8800MC/FC series chip related content
2023-04-02	1.2.01	Update bin file burning
2023-07-27	1.2.02	Added content related to AIC8800M40B series chips
2023-10-17	1.2.03	Added content related to AIC8800M80 series chips

1 SDK Overview

1.1 Directory Structure

The sdk structure is as follows:

```
aic8800-sdk/
ÿÿÿ applications
ÿ ÿÿÿ blesmartconfigwifi
ÿ ÿÿÿ command_common
ÿ ÿÿÿ consolewifi
ÿ ÿÿÿ chiwifi
ÿ ÿÿÿ examplewifi
ÿ ÿÿÿ fhostifwifi
ÿ ÿÿÿ hostifwifi
ÿ ÿÿÿ http_otawifi
ÿ ÿÿÿ http_servwifi
ÿ ÿÿÿ micwifi
ÿ ÿÿÿ rawdatawifi
ÿ ÿÿÿ tls_examplewifi
ÿ ÿÿÿ uartwifi
ÿ ÿÿÿ whwifi
ÿÿÿ audio
ÿ ÿÿÿ app
ÿ ÿÿÿ insert
ÿ ÿÿÿ bt
ÿ ÿÿÿ common
ÿ ÿÿÿ config
ÿ ÿÿÿ device
ÿ ÿÿÿ lib
ÿ ÿÿÿ process
ÿ ÿÿÿ wifi
ÿÿÿ btdm
ÿ ÿÿÿ became
ÿ ÿÿÿ bt
ÿ ÿÿÿ config
ÿ ÿÿÿ lib
ÿÿÿ config
ÿ ÿÿÿ aic8800
ÿ ÿ ÿÿÿ target_btdm
ÿ ÿ ÿÿÿ target_test
ÿ ÿ ÿÿÿ target_wifi
ÿ ÿ ÿÿÿ target_wifi_fhostif
ÿ ÿÿÿ aic8800mc
ÿ ÿ ÿ ÿ ÿ      ÿÿÿ target_ble
ÿ ÿ ÿ ÿ        ÿÿÿ target_test
ÿÿÿ            ÿÿÿ target_wifi
               ÿÿÿ target_wifi_fhostif
aic8800m40
ÿ ÿÿÿ target_ble
ÿ ÿÿÿ target_test
ÿ ÿÿÿ target_wifi
ÿ ÿÿÿ target_ble_wifi_fhostif
ÿÿÿ freertos
ÿ ÿÿÿ config
ÿ ÿÿÿ FreeRTOS
```

ÿ ÿÿÿ rtos_al
ÿ rtos_ÿÿÿport
ÿÿÿ lwip
ÿ ÿÿÿ config
ÿ ÿÿÿ dhcps
ÿ ÿÿÿ lwip-STABLE-2_0_2_RELEASE_VER
net_al
ÿÿÿ modules
ÿ ÿÿÿ none
ÿ ÿÿÿ apps
ÿ ÿÿÿ atcmd
ÿ ÿÿÿ ble_task
ÿ ÿÿÿ bt_task
ÿ ÿÿÿ common
ÿ ÿÿÿ console
ÿ ÿÿÿ dbg
ÿ ÿÿÿ fatfs
ÿ ÿÿÿ gsensor
ÿ ÿÿÿ key
ÿ ÿÿÿ led
ÿ ÿÿÿ letter-shell-v3
ÿ ÿÿÿ light_sensor
ÿ ÿÿÿ m2d
ÿ ÿÿÿ order
ÿ ÿÿÿ rtos
ÿ ÿÿÿ screen
ÿ ÿÿÿ softwdg
ÿ ÿÿÿ xyzmodem
ÿÿÿ plf
ÿ ÿÿÿ aic8800
ÿ ÿÿÿ aic8800mc
ÿ ÿÿÿ aic8800m40
ÿÿÿ tests
ÿ ÿÿÿ config
ÿ ÿÿÿ src
ÿÿÿ tools
ÿÿÿ utils
ÿÿÿ wifi
ÿ ÿÿÿ config
ÿ ÿÿÿ even
ÿ ÿÿÿ hostif
ÿ ÿÿÿ lib
ÿ ÿÿÿ LinuxDriver
ÿ ÿÿÿ macif
ÿ ÿÿÿ rawdata
ÿ ÿÿÿ wlan
ÿÿÿ wpa_supplicant

1.2 Function Introduction

1) Platform related: plf/

Including CPU startup, system initialization, peripheral drivers, etc. Currently supported platforms include three categories: aic8800, aic8800mc and aic8800m40ÿ

Platform	ChipName	Notes
aic8800	AIC8800M AIC8800A AIC8800F	Built-in Flash, support USB/SDIO Built-in Flash, support SDIO External Flash, support USB
aic8800mc	AIC8800MC AIC8800FC	Built-in Flash, support USB/SDIO External Flash, support USB
aic8800m40	AIC8800M40B AIC8800M80	Built-in Flash, support USB/SDIO Built-in Flash, support USB/SDIO

2) Real-time system related: freertos/

The current freertos version is V10.2.1, in which the API in the rtos_al directory provides an easy-to-use interface for applications.

3) Software module related: modules/

Contains a series of relatively independent software modules to achieve specific functions.

4) Audio related: audio/

Realize audio encoding and decoding, recording and playback functions.

5) WiFi protocol stack related: lwip/, wifi/, wpa_supplicant/

Implements 802.11 protocol stack and TCP/IP network functions. The current lwip version is V2.0.2.

6) Compilation script related: config/, tools/

Depends on Python 2.7, and the recommended compiler is arm-none-eabi-gcc 9.2.1.

Taking the aic8800 platform as an example, the compilation scripts and library files for different application scenarios are listed in the directory config/aic8800/target_***/.

For the body compilation method, please refer to Section 2.3.

7) Test case related: tests/src/

Contains test cases for each peripheral or module to quickly verify specific functions.

8) WiFi application related: applications/

For application layer programs based on WiFi and TCP/IP, please refer to the use of the protocol stack API.

9) BTDM protocol stack related: btdm/

Implement basic HCI, L2CAP, RFCOMMÿSDPÿGAVDP(v1.3)ÿHFP(v1.7)ÿAVDTP(v1.3)/A2DP(v1.3)
ÿAVCTP(v1.4)/AVRCP(v1.4) ÿSPP(v1.2) ÿHID(v1.0);

ble profile: GATT , (only HOGP and BAS are currently open); and the ble network configuration protocol custom developed in conjunction with the SDK wifi protocol line smartconfig, custom profile audtransmit client/server, etc.

10) BT/BLE application related: modules/apps/

Application layer program based on BT and BLE protocol stack, which loads different combinations of single BT, single BLE and BR/BLE combo bt_task or ble_task works with apps.

2 Compilation environment configuration

2.1 Windows

1) Install Git/Python 2.7

Use git-bash as the command line tool and make sure python is added to PATH:

```
$ export PATH=/c/Python27:/c/Python27/Scripts:$PATH
```

2) Configure the compiler

Download gcc-arm 9.2.1 win32 version,

https://armkeil.blob.core.windows.net/developer/Files/downloads/gnu-rm/9-2019q4/gcc-arm-none-eabi-9-2019-q4-major-win32.zip , unzip to a non-Chinese path, such as: /d/software/gcc-arm-none-eabi-9_2_1, set the environment variables:

```
$ export PATH=/d/software/gcc-arm-none-eabi-9_2_1/bin:$PATH
$ export GNUARM_4_8_LIB=/d/software/gcc-arm-none-eabi-9_2_1/lib/gcc/arm-none-eabi/9.2.1
```

2.2 Linux

1) Install Python 2.7

Most Linux distributions come with Python. If not, please install it yourself according to the distribution you are using. I will not go into details here.

2) Configure the compiler

Download gcc-arm 9.2.1 for Linux

Download the package from https://armkeil.blob.core.windows.net/developer/Files/downloads/gnu-rm/9-2019q4/gcc-arm-none-eabi-9-2019-q4-major-x86_64-linux.tar.bz2 , unzip it to a specific directory, such as /opt/gcc_arm/gcc-arm-none-eabi-9_2_1, and set the environment variables:

```
$ export PATH=/opt/gcc_arm/gcc-arm-none-eabi-9_2_1/bin:$PATH
$ export GNUARM_4_8_LIB=/opt/gcc_arm/gcc-arm-none-eabi-9_2_1/lib/gcc/arm-none-eabi/9.2.1
```

2.3 Source code compilation

1) Compile script path

The config directory structure is as follows. Currently, according to the chip type, it is divided into three subfolders: aic8800, aic8800mc and aic8800m40.

The subfolders are divided according to the usage functions, and the compilation scripts that implement the same function type are classified into the same target folder.

For example, target_test. At the same time, under the target folder, different compilation scripts are divided according to specific subdivision functions.

When actually using it, compile the script of the corresponding function and generate a binary file in the SDK root directory build/path. The specific location is as follows:

aic8800-sdk/build/***-aic8800/host_wb.bin

```
config/
ÿÿÿ aic8800
ÿ ÿÿÿ target_test
ÿ ÿ ÿÿÿ config
ÿ ÿ ÿÿÿ lib
ÿ ÿ ÿÿÿ res
```

```
                ÿÿÿ tgt_cfg
                ÿÿÿ tgt_cfg_hw
                ÿÿÿ build_bootloader.sh
                ÿÿÿ build_test_case.sh
                ÿÿÿ build_test_case_dsp.sh
                ÿÿÿ [other-test-case.sh]
        ÿ ÿ ÿ ÿ ÿ ÿ ÿÿÿ target_wifi
        ÿÿÿ target_bt
        ÿÿÿ target_ble
        ÿÿÿ [other-target-module]
ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿÿÿ aic8800mc
ÿ ÿÿÿ target_test
ÿ ÿ ÿÿÿ config
ÿ ÿ ÿÿÿ lib
ÿ ÿ ÿÿÿ res
ÿ ÿ ÿÿÿ tgt_cfg
ÿ ÿ ÿÿÿ tgt_cfg_hw
ÿ ÿ ÿÿÿ build_test_case.sh
ÿ ÿ ÿÿÿ build_test_case_sleep.sh
ÿ ÿ ÿÿÿ [other-test-case.sh]
ÿ ÿÿÿ target_wifi
ÿ ÿÿÿ target_wifi_fhostif
ÿ ÿÿÿ [other-target-module]
ÿÿÿ aic8800m40
ÿ ÿÿÿ target_test
ÿ ÿ ÿÿÿ config
ÿ ÿ ÿÿÿ lib
ÿ ÿ ÿÿÿ res
ÿ ÿ ÿÿÿ tgt_cfg
ÿ ÿ ÿÿÿ tgt_cfg_hw
ÿ ÿ ÿÿÿ build_test_case.sh
ÿ ÿ ÿÿÿ [other-test-case.sh]
ÿ ÿÿÿ target_wifi
ÿ ÿÿÿ target_ble_wifi_fhostif
ÿ ÿÿÿ [other-target-module]
ÿÿÿ [some-tool-scripts]
```

2) Compile option settings

The SDK has many compilation options. In aic8800-sdk/config/scutils.py, the default values of various options have been set.

In scutils.py,

```
# Console
env['CONSOLE'] = ARGUMENTS.get('CONSOLE', 'on') means that the console is turned on by default.
```

In the compilation script (.sh file in the config directory structure), users can modify the value of the option. The script has a higher priority than scutils.py.

Priority.

If you add CONSOLE=off to the script, it will be turned off regardless of whether env['CONSOLE'] in scutils.py is 'on' or 'off'

consoleÿ

```
./build_test_case.sh CONSOLE=off -j4
```

3) Add user code

The SDK reserves the `user/` directory with the following structure:

```
user/
|-- config
        |-- includelist.txt
        |-- sourcelist.txt
        `-- sourcelist_lib.txt
| | | |-- lib
| |-- aic8800
| | `-- armgcc_4_8
| | `-- libuser.a
| |-- aic8800mc
| | | | | | |         `-- armgcc_4_8
`-- src                     `-- libuser.a
        `-- aic8800m40
                `-- armgcc_4_8
                        `-- libuser.a


        |-- priv
        | |-- demo_lib.c
                `-- demo_lib.h
        | `-- pub
                |-- demo_src.c
                `-- demo_src.h
```

It is recommended that users add their own code for the following reasons:

- The code remains independent, making it easier to port to the new version of SDK
- This directory supports compiling some source code into libraries for use by others.

Specific usage: The compilation option USER_CODE=src compiles all source files in the user/ directory according to the source code, and

The files specified in sourcelist_lib.txt are packaged into library files. The file generation address is: aic8800-sdk/build/xxx/armgcc_4_8/libuser.a , then you need to manually replace it to user/lib/aic8800xx/armgcc_4_8/libuser.a ; the compilation option USER_CODE=lib is automatically linked during compilation Library files under user/lib/ .

3 Software Modules

3.1 Module List

The directory modules/ contains software modules:

```
modules/
 ÿ ÿÿÿ none
 ÿ ÿÿÿ apps
 ÿ ÿÿÿ atcmd
 ÿ ÿÿÿ ble_task
 ÿ ÿÿÿ bt_task
 ÿ ÿÿÿ common
 ÿ ÿÿÿ console
 ÿ ÿÿÿ dbg
 ÿ ÿÿÿ fatfs
 ÿ ÿÿÿ gsensor
 ÿ ÿÿÿ key
 ÿ ÿÿÿ led
 ÿ ÿÿÿ letter-shell-v3
 ÿ ÿÿÿ light_sensor
 ÿ ÿÿÿ order
 ÿ ÿÿÿ rtos
 ÿ ÿÿÿ screen
 ÿ ÿÿÿ xyzmodem
```

3.2 Module Introduction

1) AT command atcmd

Implement several AT command operations.

2) Serial console, letter-shell

The command interaction platform based on the serial port can be used for verification and debugging of specific programs.

3) File system fatfs

The current fatfs version is R0.14.

4) Common part

It includes common co_task, co_event, co_timer based on rtos, as well as basic definitions such as list, pool, math, etc. that can be used.

5) Bluetooth apps

Application use cases for classic Bluetooth profile, eg: a2dp source/sink, hfp hf/ag, spp, avrcp, air rise
level; use cases in single BLE scenarios; and use cases for Bluetooth headsets/speakers such as TWS.

6) bt_task/ble_task

As a basic task drive different Bluetooth application use case scenarios.

7) any

When using Bluetooth scenarios, the application layer needs to save and restore memory processing before and after the system sleeps.

8) key

General key functions based on gpio.

9) led

General purpose LED functions based on PWM.

10) gsensor

Gsensor use cases based on hardware I2C or software simulated I2C interface.

11) light_sensor

Use cases of gpio-based light sensing components.

12) order

Flash upgrade software basic functions. Can be combined with WiFi or BT air upgrade cases.

13) screen

Basic LCD related driver cases.

14) xyzmodem

Basic example of xyzmodem file transfer protocol based on serial port.

4 Test case

4.1 Test Case List

The directory tests/ contains test cases for each peripheral or module.

```
tests/ ÿÿÿ
 src
        ÿÿÿ test_aes.c ÿÿÿ
        test_asio.c ÿÿÿ
        test_bootloader.c ÿÿÿ test_ce.c
        ÿÿÿ test_clksw.c
        ÿÿÿ test_dma.c ÿÿÿ
        test_dsp.c ÿÿÿ
        test_fft512.c ÿÿÿ
        test_fir.c ÿÿÿ test_flash.c
        ÿÿÿ test_gpadc.c ÿÿÿ
        test_gpio.c ÿÿÿ
        test_gpio_irq.c ÿÿÿ
        test_hci.c ÿÿÿ
        test_i2cm0.c ÿÿÿ test_mp3.c
        ÿÿÿ test_multadd.c
        ÿÿÿ test_psram.c ÿÿÿ
        test_pwm.c ÿÿÿ
        test_screen.c ÿÿÿ
        test_sdcard.c ÿÿÿ
        test_sleep.c ÿÿÿ
        test_spi_lcd_4wires.c ÿÿÿ
        test_spi0.c ÿÿÿ test_ticker.c
        ÿÿÿ test_trng.c ÿÿÿ
        test_uart.c ÿÿÿ test_upgrade.c ÿÿÿ
        test_usb_host.c ÿÿÿ
        test_usbh_video.c ÿÿÿ
        test_wdt.c
```

4.2 Test Case Introduction

It is mainly the test code implementation of peripheral drivers, including:

1) Basic peripherals: gpio, uart, spi, i2c master, timer (ticker), flash, wdt (GPIO function selection is as shown below)

AIC8800M/A/F iomux

Bank	PAD Name	Function Mode								
		Function 0	Function 1	Function 2	Function 3	Function 4	Function 5	Function 6	Function 8	Function 9
GPIOA (1.8V/3.3V)	GPIOA2	GPIOA2	uart0_rx	uart1_rx	tx_en_5g	pcm_din		i2s_dat_in_0		spi_di
	GPIOA3	GPIOA3	uart0_tx	uart1_tx	rx_en_5g	pcm_dout		i2s_dat_out_0		spi_do
	GPIOA4	GPIOA4	uart0_cts	uart1_cts	uart1_rx			codec_mclk		
	GPIOA5	GPIOA5	uart0_rts	uart1_rts	uart1_tx					
	GPIOA6	GPIOA6	i2cm_scl	uart2_rx	uart1_cts		pwm_a0			
	GPIOA7	GPIOA7	i2cm_sda	uart2_tx	uart1_rts		pwm_a1		i2s_bck_0	
	GPIOA8	uart0_rx	GPIOA8	uart2_cts			pwm_a2		i2s_lrck_0	
	GPIOA9	uart0_tx	GPIOA9	uart2_rts					i2s_dat_out_0	
	GPIOA10	GPIOA10	uart1_rx	sdio_s_data_1	spi_sck			sdhost_data_2	i2s_bck_1	uart2_rx
	GPIOA11	GPIOA11	uart1_tx	sdio_s_data_0	spi_csn			sdhost_data_3	i2s_lrck_1	uart2_tx
	GPIOA12	GPIOA12	uart1_cts	sdio_s_clk	spi_di	pwm_a0	pcm_fsync	sdhost_cmd	i2s_dat_out_1	uart2_cts
	GPIOA13	GPIOA13	uart1_rts	sdio_s_cmd	spi_do	pwm_a1	pcm_clk	sdhost_clk	i2s_dat_in_0	uart2_rts
	GPIOA14	GPIOA14	i2cm_scl	sdio_s_data_3		pwm_a2	pcm_din	sdhost_data_0	i2s_dat_in_1	
	GPIOA15	GPIOA15	i2cm_sda	sdio_s_data_2			pcm_dout	sdhost_data_1	codec_mclk	

Bank	PAD Name	Function Mode								
		ANA_Function 0	ANA_Function 1	ANA_Function 2	Function 0	Function 1	Function 2	Function 3	Function 4	
GPIOB (AIC8800M) (1.8v/3.3V/5V)	GPIOB0	led0	adc(0-1.1v)			GPIOB0(5V)	uart1_rx			
	GPIOB1		adc(0-1.1v)	TOUCH		GPIOB1	uart1_tx			
	GPIOB2	led1	adc(0-1.1v)		GPIOB2	uart1_rx				
	GPIOB3	led2	adc(0-1.1v)		GPIOB3	uart1_tx		pwm_b0		
	GPIOB4		adc(0-1.1v)	TOUCH	GPIOB4(AON)			pwm_b1	mclk_out	
	GPIOB5		adc(0-1.1v)	TOUCH	GPIOB5(AON)			pwm_b2		
	GPIOB6		adc(0-1.1v)	TOUCH	GPIOB6(AON)	uart0_rx				
	GPIOB7		adc(0-1.1v)	TOUCH	GPIOB7(AON)	uart0_tx				
	GPIOB8		adc(0-1.1v)	TOUCH	GPIOB8					
	GPIOB9				GPIOB9					
	GPIOB10				GPIOB10					
	GPIOB11				GPIOB11					
	GPIOB12				GPIOB12					
	GPIOB13				GPIOB13	pwm_b3				
	GPIOB14				GPIOB14	pwm_b4				
	GPIOB15				GPIOB15	pwm_b5				

Module	PAD Name	Analog Func0	Analog Func1	Function Mode			
				Function 0	Function 1	Function 2	Function 3
GPIOB (AIC8800A) (1.8v/3.3V)	GPIOB_0	adc(0-1.1v)	TOUCH		GPIOB_0		pwm_b0
	GPIOB_1	adc(0-1.1v)	TOUCH		GPIOB_1		pwm_b1
	GPIOB_2	adc(0-1.1v)	TOUCH	GPIOB_2	pwm_b0		
	GPIOB_3	adc(0-1.1v)	TOUCH	GPIOB_3	pwm_b1		
	GPIOB_4	adc(0-1.1v)		GPIOB_4	dmic_clk		
	GPIOB_6	HP_RP		GPIOB_6	dmic_dat_1		pwm_b0
	GPIOB_7	HP_RN		GPIOB_7	dmic_dat_2		pwm_b1
	GPIOB_8	HP_LN		GPIOB_8	GPIOB_0	pwm_b0	
	GPIOB_9	HP_LP		GPIOB_9	GPIOB_1	pwm_b1	
	GPIOB_11	MIC3_P		GPIOB_11	GPIOB_3	dmic_dat_0	dmic_clk
	GPIOB_13	MIC2_P		GPIOB_13	GPIOB_5	dmic_dat_1	dmic_clk
	GPIOB_15	MIC1_P		GPIOB_15	GPIOB_7	dmic_dat_2	dmic_clk

AIC8800MC/FC iomux

Module	PAD Name	Ext. Func	Function Mode																															
			Function 0	I/O	PULL	IDV	Function 1	I/O	PULL	IDV	Function 2	I/O	PULL	IDV	Function 3	I/O	PULL	IDV	Function 4	I/O	PULL	IDV	Function 5	I/O	PULL	IDV	Function 6	I/O	PULL	IDV
FLASH	FLS_CLK		spi_flash_clk	O	OFF																									
	FLS_CS		spi_flash_cs	O	UP																									
	FLS_SIO_0		spi_flash_sio_0	I/O	DN	0																								
	FLS_SIO_1		spi_flash_sio_1	I/O	DN	0																								
	FLS_SIO_2		spi_flash_sio_2	I/O	UP	1																								
	FLS_SIO_3		spi_flash_sio_3	I/O	UP	1																								
GPIOA	GPIOA_0		cpu_p_swclk	I	UP	1	gpioa_0	I/O	DN	0	i2cm_scl	I/O	UP	1	debug_clk	O	OFF		i2s_lrck_0	I/O	OFF	0	pcm_fsync	I/O	DN	0	spi_lcd_sck	I/O	DN	0
	GPIOA_1		cpu_p_swd	I/O	UP	1	gpioa_1	I/O	DN	0	i2cm_sda	I/O	UP	1	aon_uart_tx	O	OFF		i2s_bck_0	I/O	OFF	0	pcm_clk	I/O	DN	0	spi_lcd_csn_0	I/O	UP	1
	GPIOA_2		gpioa_2	I/O	DN	0	uart0_rx	I	UP	1	uart1_rx	I	UP	1	i2cm_scl	I/O	UP	1	i2s_dat_in_0	I	OFF	0	pcm_din	I	DN	0	spi_lcd_di	I/O	DN	0
	GPIOA_3		gpioa_3	I/O	DN	0	uart0_tx	O	OFF		uart1_tx	O	OFF		i2cm_sda	I/O	UP	1	i2s_dat_out_0	O	OFF		pcm_dout	O	OFF		spi_lcd_do	I/O	DN	0
	GPIOA_4		gpioa_4	I/O	DN	0	uart0_cts	I	UP	1	uart1_cts	I	UP	1	uart1_rx	I	UP	1	codec_mclk	I	OFF	0	pwm_0	O	OFF		spi_lcd_cd	O	OFF	
	GPIOA_5		gpioa_5	I/O	DN	0	uart0_rts	O	OFF		uart1_rts	O	OFF		uart1_tx	O	OFF		debug_clk	O	OFF		pwm_1	O	OFF		spi_lcd_fmark	I	DN	0
	GPIOA_6		gpioa_6	I/O	DN	0	i2cm_scl	I/O	UP	1	uart2_rx	I	UP	1	uart1_cts	I	UP	1	aon_pwm_0	O	OFF		i2s_bck_0	I/O	OFF	0	spi_lcd_csn_1	I/O	UP	1
	GPIOA_7		gpioa_7	I/O	DN	0	i2cm_sda	I/O	UP	1	uart2_tx	O	OFF		uart1_rts	O	OFF		aon_pwm_1	O	OFF		i2s_lrck_0	I/O	OFF	0	spi_lcd_csn_2	I/O	UP	1
	GPIOA_8		uart0_rx	I	UP	1	gpioa_8	I/O	DN	0	uart2_cts	I	UP	1																
	GPIOA_9		uart0_tx	O	OFF		gpioa_9	I/O	DN	0	uart2_rts	O	OFF						aon_pwm_2	O	OFF		i2s_dat_out_0	O	OFF		spi_lcd_csn_3	I/O	UP	1
	GPIOA_10	sdio_data_1	gpioa_10	I/O	DN	0	uart1_rx	I	UP	1	spi_lcd_sck	I/O	DN	0	psi_m_scl	O	DN	0	sdmmc_data_2	I/O	UP	1	i2s_bck_1	I/O	OFF	0	uart2_rx	I	UP	1
	GPIOA_11	sdio_data_0	gpioa_11	I/O	DN	0	uart1_tx	O	OFF		spi_lcd_csn_0	I/O	UP	1	psi_m_sda	I/O	DN	0	sdmmc_data_3	I/O	UP	1	i2s_lrck_1	I/O	OFF	0	uart2_tx	O	OFF	
	GPIOA_12	sdio_clk	gpioa_12	I/O	DN	0	uart1_cts	I	UP	1	spi_lcd_di	I/O	DN	0	pcm_fsync	I/O	DN	0	sdmmc_cmd	I/O	UP	1	i2s_dat_in_1	I	OFF	0	uart2_cts	I	UP	1
	GPIOA_13	sdio_cmd	gpioa_13	I/O	DN	0	uart1_rts	O	OFF		spi_lcd_do	I/O	DN	0	pcm_clk	I/O	DN	0	sdmmc_clk	O	UP	1	codec_mclk	I	OFF	0	uart2_rts	O	OFF	
	GPIOA_14	sdio_data_3	gpioa_14	I/O	DN	0	i2cm_scl	I/O	UP	1	spi_lcd_cd	O	OFF		pcm_din	I	DN	0	sdmmc_data_0	I/O	UP	1	i2s_dat_out_1	O	OFF		pwm_2	O	OFF	
	GPIOA_15	sdio_data_2	gpioa_15	I/O	DN	0	i2cm_sda	I/O	UP	1	spi_lcd_fmark	I	DN	0	pcm_dout	O	OFF		sdmmc_data_1	I/O	UP	1	i2s_dat_in_0	I	OFF	0	debug_clk	O	OFF	
GPIOB	GPIOB_0	host_wake_wl	gpioa_16	I/O	UP	0	pcm_fsync	I/O	DN	0	tports0	O	OFF		uart1_rx	I	UP	1	pwm_0	O	OFF		spi_lcd_csn_1	I/O	UP	1				
	GPIOB_1	wl_wake_host	gpioa_17	I/O	UP	0	pcm_clk	I/O	DN	0	tports1	O	OFF		uart1_tx	O	OFF		pwm_1	O	OFF		spi_lcd_csn_2	I/O	UP	1				
	GPIOB_2	bt_wake_host	gpioa_18	I/O	DN	0	pcm_din	I	DN	0	tports2	O	OFF		uart1_cts	I	UP	1	pwm_2	O	OFF		spi_lcd_csn_3	I/O	UP	1				
	GPIOB_3	host_wake_bt	gpioa_19	I/O	DN	0	pcm_dout	O	OFF		tports3	O	OFF		uart1_rts	O	OFF													
USB	USB_DP	GPIOA_20																												
	USB_DM	GPIOA_21																												

AIC8800M40B/M80 iomux

Module	PAD Name	Ext. Func	Function Mode												
			Function 0	Function 1	Function 2	Function 3	Function 4	Function 5	Function 6	Function 7	Function 8	Function 9	Function 10	Function 11	Function 12	Function 13
FLASH	FLS_CLK		spi_flash_clk													
	FLS_CS		spi_flash_cs													
	FLS_SIO_0		spi_flash_sio_0													
	FLS_SIO_1		spi_flash_sio_1													
	FLS_SIO_2		spi_flash_sio_2													
	FLS_SIO_3		spi_flash_sio_3													
GPIOA	GPIOA_0		cpu_p_swclk	gpioa_0	i2cm_scl	wf_ext_pa_ctrl_0	pcm_fsync	pta_ant_sw_0	i2s_lrck_0		spdif_in	spi_lcd_sck	pwm_0	pcm_dout	pcm_clk	bt_uart_cts
	GPIOA_1		cpu_p_swd	gpioa_1	i2cm_sda	wf_ext_pa_ctrl_1	pcm_clk	pta_ant_sw_1	i2s_bck_0		spdif_out	spi_lcd_csn_0	pwm_1	pcm_din	pcm_dout	bt_uart_rts
	GPIOA_2		gpioa_2	uart0_rx	uart1_rx	wf_ext_pa_ctrl_2	pcm_din	debug_clk	i2s_dat_in_0		pta_ant_sw_0	spi_lcd_di	pwm_2	pcm_fsync	pcm_din	bt_uart_rx
	GPIOA_3		gpioa_3	uart0_tx	uart1_tx	wf_ext_pa_ctrl_3	pcm_dout	aon_uart_tx	i2s_dat_out_0		pta_ant_sw_1	spi_lcd_do		pcm_clk	pcm_fsync	bt_uart_tx
	GPIOA_4		gpioa_4	uart0_cts	uart1_cts	uart1_rx	bt_uart_rx	aud_dac_sync	codec_mclk		i2s_bck_0	spi_lcd_cd				pcm_fsync
	GPIOA_5		gpioa_5	uart0_rts	uart1_rts	uart1_tx	bt_uart_tx	aon_uart_tx	debug_clk		i2s_lrck_0	spi_lcd_fmark				pcm_din
	GPIOA_6		gpioa_6	i2cm_scl	uart2_rx	uart1_cts	bt_uart_cts	psi_s_scl	bt_uart_rx			spi_lcd_csn_1				pcm_dout
	GPIOA_7		gpioa_7	i2cm_sda	uart2_tx	uart1_rts	bt_uart_rts	psi_s_sda	bt_uart_tx		aon_pwm_0	spi_lcd_csn_2				pcm_clk
	GPIOA_8		uart0_rx	gpioa_8	uart2_cts	spdif_in			bt_uart_cts							
	GPIOA_9		uart0_tx	gpioa_9	uart2_rts	spdif_out	aon_pwm_1		bt_uart_rts		i2s_dat_out_0	spi_lcd_csn_3				
	GPIOA_10	sdio_data_1	gpioa_10	uart1_rx	bt_uart_rx	spi_lcd_sck	bt_uart_cts		sdmmc_data_2		i2s_bck_1	uart2_rx	bt_uart_cts	bt_uart_rts		
	GPIOA_11	sdio_data_0	gpioa_11	uart1_tx	bt_uart_tx	spi_lcd_csn_0	bt_uart_rts	debug_clk	sdmmc_data_3		i2s_lrck_1	uart2_tx	bt_uart_rts	bt_uart_rx		
	GPIOA_12	sdio_clk	gpioa_12	uart1_cts	bt_uart_cts	spi_lcd_di	aon_pwm_2	pcm_fsync	sdmmc_cmd		i2s_dat_in_1	uart2_cts	bt_uart_rx	bt_uart_tx		
	GPIOA_13	sdio_cmd	gpioa_13	uart1_rts	bt_uart_rts	spi_lcd_do	pwm_0	pcm_clk	sdmmc_clk		codec_mclk	uart2_rts	bt_uart_tx	bt_uart_cts		
	GPIOA_14	sdio_data_3	gpioa_14	i2cm_scl	spdif_in	spi_lcd_cd	pwm_1	pcm_din	sdmmc_data_0		i2s_dat_out_1	pta_ant_sw_0	uart0_rx			
	GPIOA_15	sdio_data_2	gpioa_15	i2cm_sda	spdif_out	spi_lcd_fmark	pwm_2	pcm_dout	sdmmc_data_1		i2s_dat_in_0	pta_ant_sw_1	uart0_tx			
GPIOB	GPIOB_0		gpiob_0	pcm_fsync	i2cm_scl	spi_lcd_sck	aon_pwm_0	wf_ext_pa_ctrl_0								
	GPIOB_1		gpiob_1	pcm_clk	i2cm_sda	spi_lcd_csn_0	aon_pwm_1	wf_ext_pa_ctrl_1								
	GPIOB_2		gpiob_2	pcm_din	psi_m_scl	spi_lcd_di	aon_pwm_2	wf_ext_pa_ctrl_2								
	GPIOB_3		gpiob_3	pcm_dout	psi_m_sda	spi_lcd_do	spdif_in	wf_ext_pa_ctrl_3								
	GPIOB_4		gpiob_4	pwm_0	i2s_lrck_0	bt_uart_rx	spdif_out									
	GPIOB_5		gpiob_5	pwm_1	i2s_bck_0	bt_uart_tx										
	GPIOB_6		gpiob_6	pwm_2	i2s_dat_in_0	bt_uart_cts										
	GPIOB_7		gpiob_7	aon_pwm_0	i2s_dat_out_0	bt_uart_rts										
USB	USB_DP	gpioa_16														
	USB_DM	gpioa_17														

2. File system: sdcard, usb_host

3. Audio related: asio, mp3

4. Bluetooth related: hci

5. Hardware acceleration: aes, dma, fft512, trng

6. 8800M spi, i2c master Feature Description

I2C Feature	GPIOA I2C	GPIOB I2C
支持AMBA 2.0 APB总线	支持	支持
支持标准模式（100 Kb/s）和快速模式（400 Kb/s）协议	支持	支持
支持可编程主模式	不支持	不支持
支持7位和10位寻址模式	支持7bit	支持7bit
支持自动时钟延展	不支持	不支持
支持可编程时钟与数据时序	支持	支持
支持DMA	支持	不支持
支持通用呼叫地址	不支持	不支持
SPI Feature	**GPIOA SPI**	**GPIOB SPI**
支持AMBA 2.0 APB总线	支持	支持
支持标准3线或4线	支持	支持
支持时钟频率配置	支持	支持
支持读写时钟相位调整	支持	支持
支持DMA	支持	不支持
支持LCD spi 3线4线模式	支持	不支持

4.3 Compilation Instructions

The case can be specified through the compilation option: TEST=[CASE_NAME], for example, config/build_test_case.sh can be written as follows:

```
python ../tools/scons.py . PRODUCT=basic-rtos PLF=aic8800 BT=armgcc_4_8 PMIC_VER=lite
USE_LIB_DRV=on TEST=gpio_irq ÿÿ python ../tools/scons.py .

PRODUCT=basic-rtos PLF=aic8800 BT=armgcc_4_8 PMIC_VER=lite USE_LIB_DRV=on TEST=i2cm0
```

5 WiFi application

5.1 Application List

The directory applications/ contains the applications:

```
applications/ ÿÿÿ
 consolewifi ÿÿÿ
 examplewifi ÿÿÿ
 http_servwifi ÿÿÿ uartwifi
 ÿÿÿ fhostifwifi ÿÿÿ
 voicewifi
```

5.2 Application Introduction

1. examplewifi

   Implement basic softap startup and station association functions.

2. consolewifi

   A command console based on the console module, including some basic WiFi operation procedures. For each command interface, refer to the document WiFiConsole.pdf.

3. uartwifi

   AT command interaction interface based on atcmd module, suitable for WiFi transparent transmission module control. For each command interface, refer to the document WiFiATCommand.pdf.

4. fhostifwifi

   Based on the virtual network card driver netdrv, the system low power consumption function of the embedded master control and aic8800 series chips is realized. For development documents, refer to Fhostif_UserGuide.pdf, CustomMsg_UserGuide.pdf and FhostifOTA_UserGuide.pdf.

5. http_servwifi

   The http server implemented based on lwip, when used in conjunction with the file system, can access files in the U disk or TF card through the web.

6. voicewifi

   Wireless voice system based on lwip + audio.

6 btdm interface and application

6.1 Application List

The directory btdm/ contains the applications:

```
btdm/
 ÿÿÿ bt
 ÿ ÿÿÿ include
              ÿÿÿ aic_adp_api.h
              ÿÿÿ aic_adp_type.h
              ÿÿÿ aic_host_cfg.h
              ÿÿÿ bt_aon_sram.h
              ÿÿÿ bt_types_def.h
              ÿÿÿ co_errors.h
              ÿÿÿ co_types_def.h
              ÿÿÿ interface
                   ÿÿÿ aic_adp_a2dp.h
                   ÿÿÿ aic_adp_avrcp.h
                   ÿÿÿ aic_adp_dev.h
                   ÿÿÿ aic_adp_hfp.h
                   ÿÿÿ aic_adp_hid.h
                   ÿÿÿ aic_adp_hsp.h
                   ÿÿÿ aic_adp_mgr.h
                   ÿÿÿ aic_adp_spp.h
                   ÿÿÿ aic_adp_test.h
                   ÿÿÿ aic_adp_tws.h
                   ÿÿÿ aic_bt_msg.h
 ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿÿÿ became
 ÿ ÿÿÿ ble_adp
 ÿ ÿ ÿ        ÿÿÿ aic_ble_adp_api.h
 ÿÿÿ became_app
 ÿ ÿ ÿÿÿ app_audtransmitc
 ÿ ÿ ÿ ÿÿÿ app_audtransmitc.c
 ÿ ÿ ÿ ÿÿÿ app_audtransmitc.h
 ÿ ÿ ÿÿÿ app_audtransmits
 ÿ ÿ ÿ ÿÿÿ app_audtransmits.c
 ÿ ÿ ÿ ÿÿÿ app_audtransmits.h
 ÿ ÿ ÿÿÿ app_batt
 ÿ ÿ ÿ ÿÿÿ app_batt.c
 ÿ ÿ ÿ ÿÿÿ app_batt.h
 ÿ ÿ ÿÿÿ app_hid
 ÿ ÿ ÿ ÿÿÿ app_hid.c
 ÿ ÿ ÿ ÿÿÿ app_hid.h
 ÿ ÿ ÿÿÿ app_main
 ÿ ÿ ÿ ÿÿÿ app.c
 ÿ ÿ ÿ ÿÿÿ app.h
 ÿ ÿ ÿ ÿÿÿ app_task.c
 ÿ ÿ ÿ ÿÿÿ app_task.h
 ÿ ÿ ÿÿÿ app_sec
 ÿ ÿ ÿ ÿÿÿ app_sec.h
 ÿ ÿ ÿÿÿ app_smartconfig
 ÿ ÿ ÿÿÿ app_smartconfig.c
 ÿ ÿ ÿÿÿ app_smartconfig.h
 ÿ ÿÿÿ ble_dbg
 ÿ ÿ ÿÿÿ aicble_dbg.c
```

ÿ ÿÿÿ aicble_dbg.h
ÿÿÿ ble_profiles
ÿ ÿÿÿ audtransmit
ÿ ÿ ÿÿÿ audtransmitc
ÿ ÿ ÿ ÿÿÿ api
ÿ ÿ ÿ ÿ ÿÿÿ audtransmitc_task.h
ÿ ÿ ÿ ÿÿÿ sourcelist.txt
ÿ ÿ ÿ ÿÿÿ src
ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿÿÿ audtransmitc.c
ÿ ÿÿÿ audtransmits ÿÿÿ audtransmitc.h
ÿÿÿ audtransmitc_task.c

ÿÿÿÿÿÿÿÿÿ ÿÿÿ api
ÿÿÿÿÿÿÿÿÿ ÿ ÿÿÿ audtransmits_task.h
low ÿÿÿ sourcelist.txt
ÿÿÿ src
ÿÿÿ audtransmits.c
ÿÿÿ audtransmits.h
ÿÿÿ audtransmits_task.c

ÿÿÿÿÿÿÿ ÿÿÿ bass
ÿÿÿÿÿÿÿ ÿÿÿ api
hogp ÿÿÿ bass_task.h
ÿ ÿÿÿ src
ÿÿÿ bass.h

ÿ ÿ ÿÿÿ hogp_common.h
ÿ ÿ ÿÿÿ hogpd
ÿ ÿ ÿÿÿ api
ÿ ÿ ÿ ÿÿÿ hogpd_task.h
ÿ ÿ ÿÿÿ src
ÿ ÿ ÿ ÿÿÿ hogpd.h
ÿÿÿ prf
ÿ ÿ ÿÿÿ prf_user.c
ÿ ÿÿÿ smartconfig
ÿ ÿÿÿ smartconfigs
ÿ ÿÿÿ api
ÿ ÿ ÿÿÿ smartconfig_task.h
ÿ ÿÿÿ src
ÿÿÿ smartconfig.c
ÿÿÿ smartconfig.h
ÿÿÿ smartconfig_task.c
ÿ ÿ ÿ ÿÿÿ ble_stack
ÿÿÿ became_ip
ÿ ÿÿÿ aicble.h
ÿ ÿÿÿ rwapp_config.h
ÿ ÿÿÿ rwble_hl_config.h
ÿ ÿÿÿ rwip.h
ÿ ÿÿÿ rwip_config.h
ÿ ÿÿÿ rwip_task.h
ÿ ÿÿÿ rwprf_config.h
ÿÿÿ common
ÿ ÿÿÿ api
ÿ ÿÿÿ ble_utils.h
ÿ ÿÿÿ co_bt.h
ÿ ÿÿÿ co_bt_defines.h
ÿ ÿÿÿ co_error.h
ÿ ÿÿÿ co_hci.h

```
ÿÿÿ hl
ÿ ÿÿÿ api
ÿ ÿ ÿÿÿ that.h
ÿ ÿ ÿÿÿ gap.h
ÿ ÿ ÿÿÿ gapc_task.h
ÿ ÿ ÿÿÿ gapm_task.h
ÿ ÿ ÿÿÿ gattc_task.h
ÿ ÿ ÿÿÿ prf_types.h
ÿ ÿ ÿÿÿ rwble_hl_error.h
ÿ ÿÿÿ inc
ÿÿÿ attm.h
ÿÿÿ gapc.h
ÿÿÿ gapm.h
ÿÿÿ prf.h
ÿÿÿ prf_utils.h
ÿÿÿ smpc.h
ÿ ÿ ÿ ÿ ÿ ÿ ÿÿÿ the
ÿÿÿ api
ÿÿÿ ke_mem.h
ÿÿÿ to_msg.h
ÿÿÿ ke_task.h
ÿÿÿ ke_timer.h
ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿ ÿÿÿ config
ÿ ÿÿÿ includelist.txt
ÿ ÿÿÿ sourcelist.txt
ÿÿÿ lib
ÿÿÿ armgcc_4_8
ÿÿÿ libbleonly.a
ÿÿÿ libbt.a
ÿÿÿ libbtdm.a
ÿÿÿ libbttws.a
```

6.2 Protocol stack interface introduction

1) bt

bt/include: bt protocol stack basic interface and definition

bt/include/interface: bt protocol stack profile interface and definition

Please refer to modules/apps/src/bt for usage methods.

```
ÿÿÿ app_a2dp.c
ÿÿÿ app_a2dp_source.c
ÿÿÿ app_avrcp.c
ÿÿÿ app_bt.c
ÿÿÿ app_bt_queue.c
ÿÿÿ app_console.c
ÿÿÿ app_hfg.c
ÿÿÿ app_hfp.c
ÿÿÿ app_hid.c
ÿÿÿ app_hsp.c
ÿÿÿ app_ota_box.c
ÿÿÿ app_spp.c
ÿÿÿ app_test.c
ÿÿÿ app_tws.c
```

app_bt.cÿ

This is the main file of the classic Bluetooth application layer, which includes the registration of each profile, as well as basic interfaces such as message processing function registration and key message function registration.

The app_bt_init() function is the main entry point and is called after the task in bt_task.c is executed.

app_bt_queue.cÿ

It is the message queue of the BitTorrent application layer. If other tasks want to use the BitTorrent protocol stack API, they need to create a BitTorrent task to send messages and then execute it.

app_console.cÿ

Create a console interface corresponding to the BT protocol stack application layer.

The rest are profile applications.

2) became

ble/ble_stack: BLE protocol stack basic interface and definition

ble/ble_profiles: ble protocol stack profile interface and definition

ble/ble_dbg: ble protocol stack print level modification

ble/ble_app: ble protocol stack profile defines the interface and definition of app

Please refer to modules/apps/src/ble for usage methods.

```
ÿÿÿ app_ble_audtransmit.c ÿÿÿ
app_ble_console.c ÿÿÿ
app_ble_only.c ÿÿÿ
app_ble_queue.c ÿÿÿ
app_m2d.c
```

app_ble_only.c

This is the main file of the low-power Bluetooth application layer, which contains basic interfaces such as the registration of each profile and the registration of message processing functions.

app_ble_init() is the main entry point and is called after the task in ble_task.c is executed. It should be noted that when executing Bt/Ble Combo

When BLE is combined with classic Bluetooth in bt_task, app_ble_init() is called in bt_task.

app_ble_queue.c:

It is the message queue of the BLE application layer. If other tasks want to use the protocol stack API, they need to create a message to send in a single BLE or BT/BLE combo. ble_task or bt_task (combo) and then execute it.

app_ble_console.cÿ

Created for the console interface corresponding to the BLE protocol stack application layer.

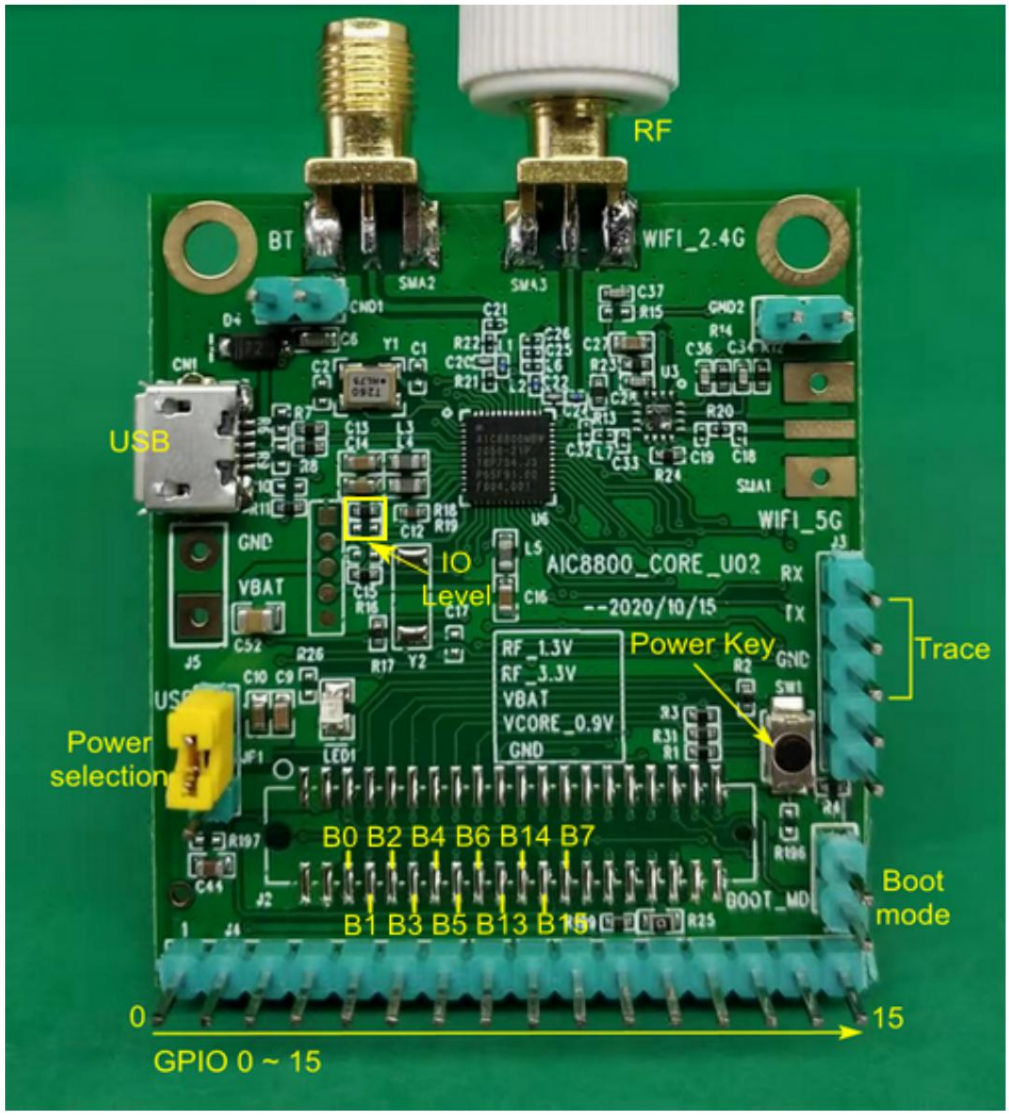
The rest are profile applications.

7 Using the demo board

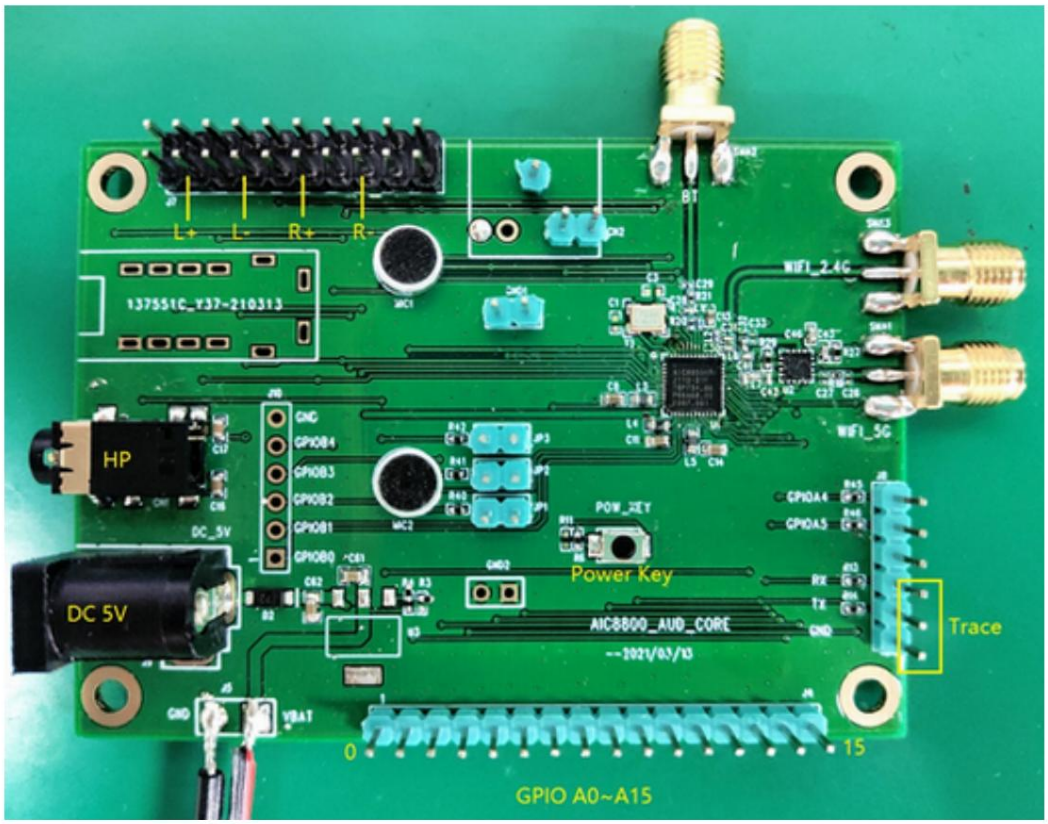
7.1 Board Appearance

The appearance of the demo board is shown in the figure:

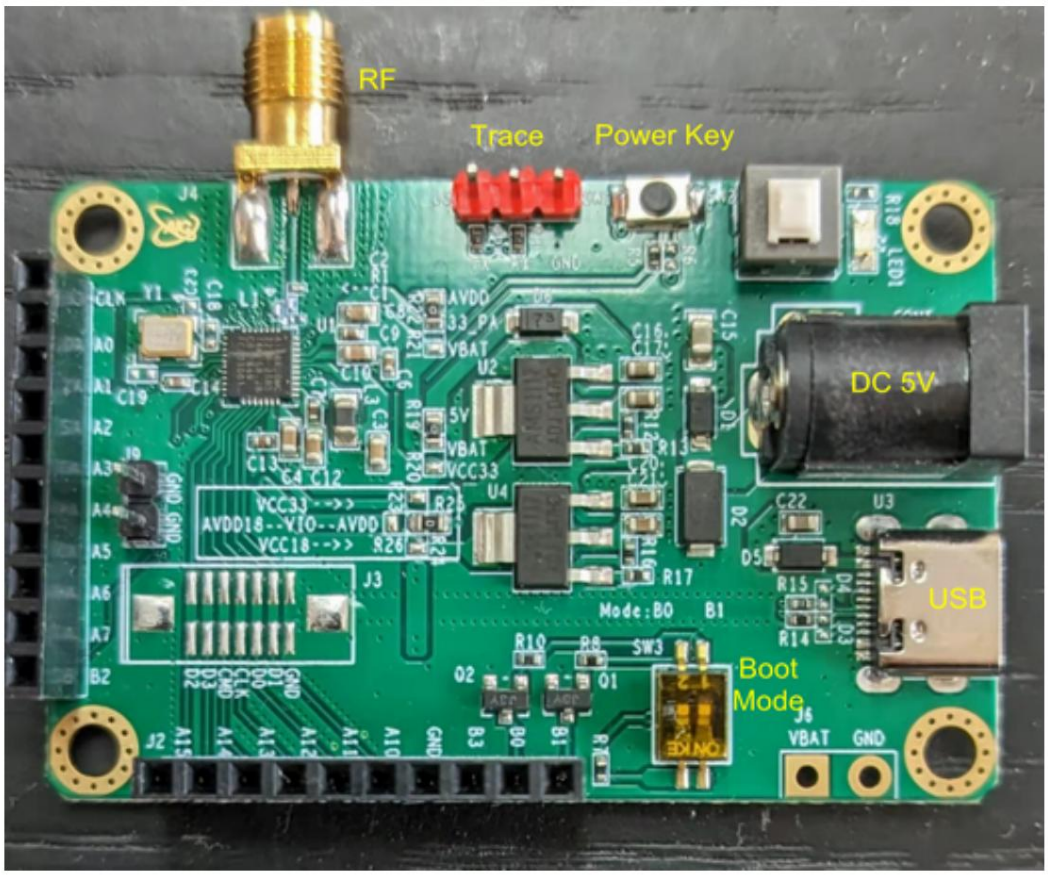
1) AIC8800M EVB



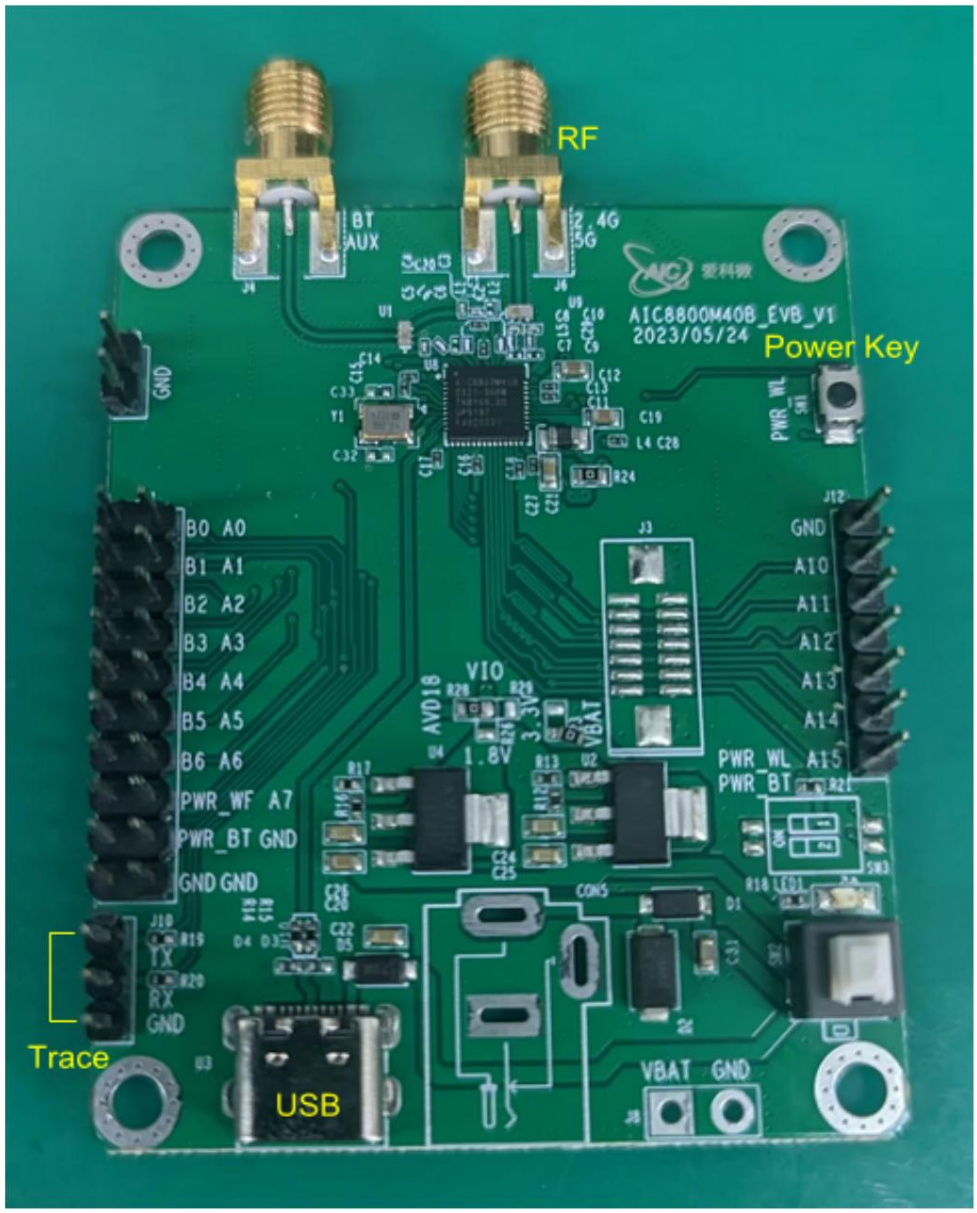
2) AIC8800A EVB



3) AIC8800MC EVB



3) AIC8800M40B EVB



7.2 Introduction to some interfaces

1) Power Key is the power on/off button. Its corresponding chip PWRKEY pin has an internal resistance of 200K ohms. When the voltage divider value V~PWRKEY~ >= VIO * 0.6 after being connected in series with the external pull-up resistor, the effective timing diagram is as follows:

Note: For designs that do not use buttons and the PWRKEY pin is always pulled high, the SDK should define USER_DEFINED_PWRKEY_EN as 0.

2) Trace serial port can be used to view log and burn bin files

Note: If you use a PC USB to serial port cable to connect the board's Trace serial port, make sure the serial port level matches the board's IO level.

The 3.3V serial port line may burn out the chip. Please refer to Article 5 for the specific IO level selection.

3) Boot mode jumper short circuit can force to enter download mode

Note: The Boot mode implementation methods of AIC8800M/AIC8800A/AIC8800MC are different, as follows:

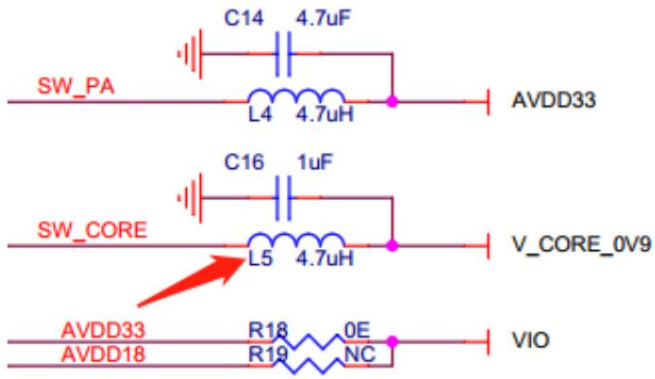
Chip Model	Boot mode implementation
AIC8800M	GPIOB6/B7 are both grounded
AIC8800A	HPRP/HPRN are both grounded
AIC8800MC	GPIOB0/B1 are both grounded
AIC8800M40B/M80	GPIOB0/B1 are both grounded

4) When AIC8800M EVB is powered by USB, select “USB” in the power selection jumper.

5) IO Level on AIC8800M EVB can select IO level (VIO) through R18, R19

VIO \ Resistor	R18	R19
3.3V	0 Ohm	NC
1.8V	NC	0 Ohm

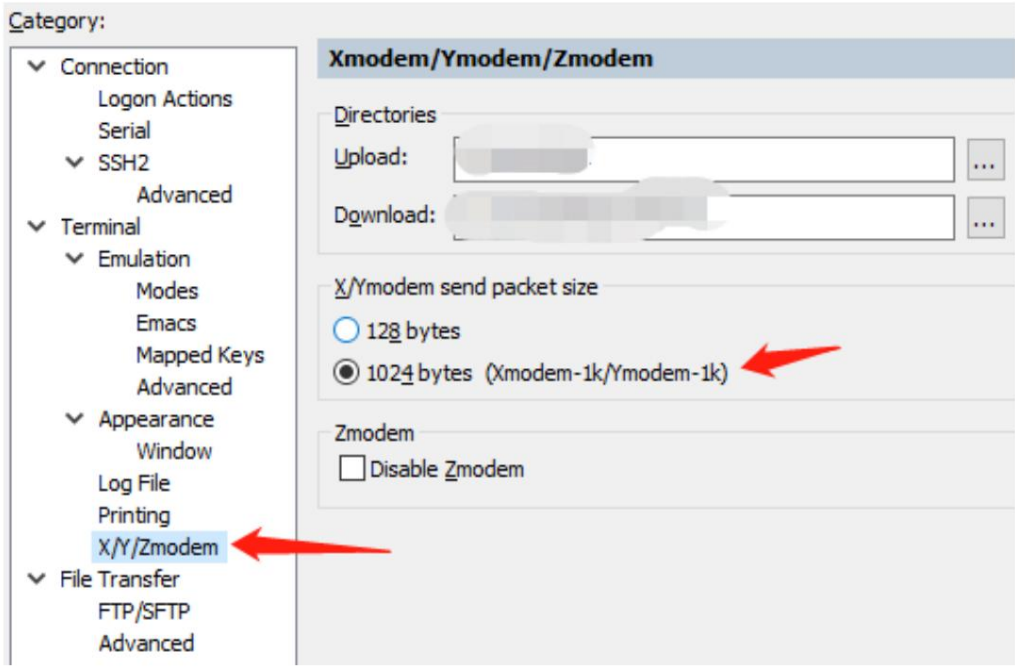
6) When the inductor L5 on the AIC8800M EVB is not soldered, the SDK compilation option should be set to: PMIC_VCORE=ldo



7.3 bin file burning

Release download mode

1) **Set the serial port software** to connect to the Trace serial port of the demo board through the serial port, open SecureCRT, set the baud rate to 921600, and set the Xmodem-1k mode. If you set it up with the serial port open, you need to reopen the serial port to make the software serial port configuration take effect.



2. **Enter the download mode** Boot Mode short circuit, power on the chip, press the Power Key

```
Bootrom [Jan 30 2021, c71e5d6]
Copyright (C) 2018-2020 AICSemi Ltd.

RstCause:0000,Boot:3d,1
Mcu mode

Boot >
Boot >
```

3. **Start file transfer,** type command **x ADDRESS LENGTH** (e.g. x 8000000 60000), the first parameter is the program start address (hexadecimal), the second parameter is the erase length (Byte, hexadecimal, usually the bin file length 4KB rounded up), SecureCRT Xmodem sends the bin file, and waits for the transfer to end. When burning, the second parameter can be determined according to the actual bin file size

```
Bootrom [Jan 30 2021, c71e5d6]
Copyright (C) 2018-2020 AICSemi Ltd.

RstCause:0000,Boot:3d,1
Mcu mode

Boot >
Boot >x 8000000 60000
Receiving XModem (921600 bps) data to 0x08000000
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

4. **Upload the file** as shown above. After the SecureCRT software starts the transfer, it will continue to print the C character. At this time, through the menu bar, Transfer->Send Xmodem, select the corresponding bin file to transfer

```
Boot >
Boot >x 8000000 60000
Receiving XModem (921600 bps) data to 0x08000000
CCCCCCCCCCCCCCCCCCCCCCCCCCC
Starting xmodem transfer.  Press Ctrl+C to cancel.
Transferring ble_wifi_fhostif.bin...
  100%     701 KB       21 KB/sec    00:00:32       0 Errors

xyzModem - CRC mode, 1(SOH)/701(STX)/0(CAN) packets, 3 retries
717952 (0x000AF480) bytes received

OK
```

5. **Execute the application** to remove the Boot Mode short circuit, re-power the chip, press the Power Key, and the program will automatically execute

debug download mode

1. **Set the serial port software** to the same download mode as **release**

2. **Enter the download mode** Boot Mode without shorting, power on the chip, and keep pressing Enter before pressing the Power Key. This operation needs to be done

   count=0 Completed before the timer ends

```
Bootrom [Jan 30 2021, c71e5d6]
Copyright (C) 2018-2020 AICSemi Ltd.

RstCause:0000,Boot:3d,0
Mcu mode
count=4
count=3
count=2

Boot >Boot abort

Boot >
Boot >
```

3. **Start file transfer** in the same way as **release download mode**

4. **Upload files** in the same way as **release download mode**

5. After **executing the application** SecureCRT Xmodem to send the bin file, you can directly type the command **g 8000000** to execute the burning

   bin file

```
Bootrom [Jan 30 2021, c71e5d6]
Copyright (C) 2018-2020 AICSemi Ltd.

RstCause:0000,Boot:3d,0
Mcu mode
count=4
count=3
count=2

Boot >Boot abort

Boot >
Boot >
Boot >
Boot >
Boot >
Boot >x 8000000 60000
Receiving XModem (921600 bps) data to 0x08000000
CCCCCCCCCCCCCCCCCCCCCCCCCCCCC
Starting xmodem transfer.  Press Ctrl+C to cancel.
Transferring ble_wifi_fhostif.bin...
  100%     701 KB      21 KB/sec    00:00:32       0 Errors

xyzModem - CRC mode, 1(SOH)/701(STX)/0(CAN) packets, 3 retries
717952 (0x000AF480) bytes received

OK
Boot >g 8000000
```

Note: If the chip is not factory-set, you will encounter "boot abort: -1" when starting. In this case, you need to execute the following command to power off the chip, and then repeat the above steps 2) to 5)

```
f 1 3 1 2 1
f 3
```

8 Problem Location

8.1 Pre-operation inspection

1) Chip power supply

- AIC8800M/A/F VBAT supports voltage 2.3~5V; VIO supports 1.8V or 3.3V, see Section 6.2 for configuration; VCORE09 supports DCDC or LDO mode, corresponding software configuration see Section 7.2 Item 6);
- AIC8800MC/FC VBAT supports voltage 2.3~5V; VIO supports 1.8V or 3.3V, VFLASH supports 1.8V or 3.3V configuration
- AIC8800M40B/M80 VBAT supports voltage 3.0~3.6V; VIO supports 1.8V or 3.3V, VFLASH supports 1.8V or 3.3V configuration

2) External Trace serial port

The trace serial port voltage level is VIO, and the external serial port voltage level should be consistent. If it is higher than 3.3V, there is a risk of damaging the chip.

3) WiFi/BT Antenna

Before testing the WiFi/BT function, make sure the antenna is connected.

8.2 FAQ

1) When the chip is used for the first time, "boot abort: -1" is encountered

See Section 7.3

2) GPIO location on the EVB board

See Section 7.1

3) What are the requirements for the starting address and data length of Flash related operations (erase/write/read)?

Flash operation needs to pay attention to: erase address and length require 4KB alignment, write address 256B alignment length no requirement, read address 4B alignment length no requirement

4) System restart interface

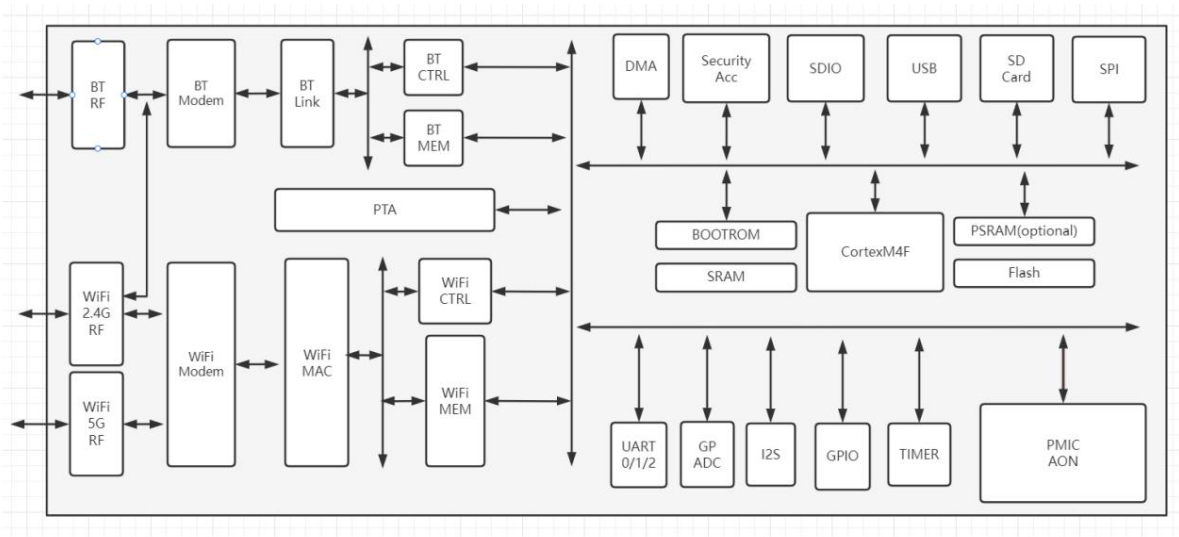
pmic_chip_reboot()

5) Differences between AIC8800M and AIC8800D

Chip Model	built-in Flash	Dependency on the master	Application Scenario	Development method
AIC8800M Yes		no	IoT Low power consumption	Develop FreeRTOS-based applications and run them on AIC8800M (virtual network card and other applications also require the development of host driver)
AIC8800D No		yes	Wireless network card high throughput	Develop Linux-based drivers and run them on the host

9 Appendix 1 Chip memory peripheral mapping

9.1 System Architecture



9.2 Memory and peripheral mapping

AIC8800M/A/F

Module	Base	Size
ROM	0x0000_0000	64KB
WIFI ROM		
BT ROM		
RAM0	0x0010_0000	384KB
RAM1	0x0016_0000	448KB
AON RAM	0x001E_0000	64KB
FLASH	0x0400_0000	32MB
PSRAM	0x0600_0000	32MB
CACHE0	0x0800_0000	64MB
WLAN	0x4030_0000	
UART0	0x4004_1000	
UART1	0x4004_2000	
UART2	0x4004_3000	
IPC	0x4004_4000	
ASIO	0x4005_0000	
DMA	0x4007_0000	
SDMMC	0x4008_2000	
PWM	0x4010_1000	
I2C_M	0x4010_2000	
WDG0	0x4010_3000	
WDG1	0x4010_4000	
SPI	0x4010_5000	
TIMER0	0x4010_6000	
HOUR1	0x4010_7000	
SPINLOCK	0x4010_8000	
TRNG	0x4010_9000	
WDG2	0x4010_A000	
WDG3	0x4010_C000	
BT	0x4060_0000	
USB	0x4020_0000	

Module	Base	Size
SDIO	0x4024_0000	
TIMER2	0x4050_1000	
WDG4	0x4050_2000	
IOMUX	0x4050_3000	
GPIO	0x4050_4000	
PWR_CTRL	0x4050_6000	

Module	Base	Size
ROM	0x0000_0000	64KB
WIFI ROM		
BT ROM		
RAM0	0x0010_0000	512KB
RAM1(CACHE)	0x0018_0000	32KB
BT_RAM	0x0018_8000	64KB
FLASH-DATA	0x0400_0000	
CACHE-DATA	0x0800_0000	
DMA	0x4002_0000	
UART0	0x4003_2000	
UART1	0x4003_3000	
UART2	0x4003_4000	
IPC	0x4003_5000	
SDMMC	0x4003_6000	
ASIO	0x4003_7000	
CPUSYSCTRL	0x4010_0000	
PWM	0x4010_1000	
I2C_M	0x4010_2000	
WDG0	0x4010_3000	
WDG1	0x4010_4000	
SPI	0x4010_5000	
TIMER0	0x4010_6000	
HOUR1	0x4010_7000	
TRNG	0x4010_8000	
ANA_REG0	0x4010_9000	
GPIO	0x4010_B000	
MSADC	0x4010_D000	
USB	0x4020_0000	
SDIO	0x4024_0000	
MMSYSCTRL	0x4024_1000	

Module	Base	Size
AONSYSCTRL	0x4050_0000	
TIMER2	0x4050_1000	
ANA_REG1	0x4050_2000	
IOMUX	0x4050_4000	
CM_RF_INTF	0x4050_5000	
PWR_CTRL	0x4050_6000	
AON_PWM	0x4050_7000	
WDG2	0x4050_8000	

Module	Base	Size
ROM	0x0000_0000	64KB
WIFI ROM		
BT ROM		
RAM0	0x0010_0000	256KB
RAM1	0x0014_0000	384KB
RAM2	0x001A_0000	256KB
RAM3(CACHE)	0x0018_0000	32KB
BT_RAM	0x0020_0000	64KB
FLASH-DATA	0x0400_0000	
CACHE-DATA	0x0800_0000	
DMA	0x4002_0000	
UART0	0x4003_2000	
UART1	0x4003_3000	
UART2	0x4003_4000	
IPC	0x4003_5000	
SDMMC	0x4003_6000	
ASIO	0x4003_7000	
CPUSYSCTRL	0x4010_0000	
PWM	0x4010_1000	
I2C_M	0x4010_2000	
WDG0	0x4010_3000	
WDG1	0x4010_4000	
SPI	0x4010_5000	
TIMER0	0x4010_6000	
HOUR1	0x4010_7000	
TRNG	0x4010_8000	
ANA_REG0	0x4010_9000	
GPIO	0x4010_B000	
MSADC	0x4010_D000	
USB	0x4020_0000	

Module	Base	Size
SDIO	0x4024_0000	
MMSYSCTRL	0x4024_1000	
AONSYSCTRL	0x4050_0000	
TIMER2	0x4050_1000	
ANA_REG1	0x4050_2000	
IOMUX	0x4050_4000	
GPIOB	0x4050_5000	
PWR_CTRL	0x4050_6000	
AON_PWM	0x4050_7000	
WDG2	0x4050_8000	
CM_RF_INTF	0x4050_9000	

10 Appendix 2 Chip Clock Settings

The content in this section is located in sysctrl_api.h.

10.1 Setting the clock

Configure the clock through the following interfaces:

```
void sysctrl_clock_cfg(int cfg);
```

The value range of the parameter cfg is as follows:

```
enum {
        CLK_CFG_D480S240P120F120 = 0,
        CLK_CFG_D480S240P120F80,
        CLK_CFG_D480S240P120F60,
        CLK_CFG_D480S240P60F120,
        CLK_CFG_D480S240P60F80,
        CLK_CFG_D480S240P60F60,
        CLK_CFG_D240S240P120F120,
        CLK_CFG_D240S240P120F60,
        CLK_CFG_D240S240P60F60,
        CLK_CFG_D240S120P60F120,
        CLK_CFG_D240S120P60F60,
        CLK_CFG_D240S120P30F60,
        CLK_CFG_D120S120P60F60,
        CLK_CFG_D120S120P30F60,
        CLK_CFG_D80S80P40F40,
        CLK_CFG_D52S52P26F26,
        CLK_CFG_MAX,
};
```

Taking CLK_CFG_D240S240P120F60 as an example, the meaning of each part is as follows:

Configuration	meaning
S240	Set the system clock (MCU main frequency) to 240M
P120	Configure the peripheral clock to 120M
F60	Configure the Flash clock to 60M

CLK_CFG_D240S240P120F60 is also the system default configuration.

10.2 Get clock settings

Get the clock settings through the following interface:

```
uint32_t sysctrl_clock_get(int sys_per);
```

The unit of the return value is Hz.

The value range of the parameter sys_per is as follows:

```
enum {
        SYS_FCLK = 0,
        SYS_HCLK,
        SYS_PCLK,
        PER_UART0,
        PER_UART1,
        PER_UART2,
        PER_PSRAM,
        PER_FLASH,
        PER_PWM,
};
```

10.3 Setting PLL

Set the PLL through the following interface:

```
void sysctrl_pll_cfg(int cfg);
```

The range of parameter cfg is:

```
enum {
        PLL_CFG_OFF = 0,
        PLL_CFG_320,
        PLL_CFG_480,
        PLL_CFG_320_480,
};
```

PLL is configured during initialization. In most cases, users do not need to configure it themselves. Self-configuration may cause unnecessary problems.

The configuration of PLL is related to whether there is BT, WIFI and clock configuration in the system. The following table shows the relationship:

BT WIFI		**Clock Configuration**	**PLL Configuration**
none	Yes/ No	CLK_CFG_D52S52P26F26	PLL_CFG_OFF*1ÿPLL_CFG_320ÿ PLL_CFG_480ÿPLL_CFG_320_480*2
none	Yes/ No	Configuration other than CLK_CFG_D52S52P26F26	PLL_CFG_480*1ÿPLL_CFG_320_480
Whether there is CLK_CFG_D52S52P26F26			PLL_CFG_320*1ÿPLL_CFG_320_480
Yes or No		Configuration other than CLK_CFG_D52S52P26F26	PLL_CFG_320_480
Yes CLK_CFG_D52S52P26F26			PLL_CFG_320*1ÿPLL_CFG_320_480
Yes		Configuration other than CLK_CFG_D52S52P26F26	PLL_CFG_320_480

*1: Configuration with the lowest power consumption.

*2: The configuration with the best compatibility.

11 Appendix 3 RF parameter settings

11.1 WiFi radio frequency parameters

For configuration of transmit power, TXOP, CCA and other parameters, refer to the document: WiFiApi

11.2 BT RF Parameters

[TBD]