SmartLink 软件
开发说明

珠海慧联科技有限公司

版本记录

版本号	日期	制/修订人	制/修订记录
V1.0	2019-03-28	luoyunfeng	初始版本
V1.1	2019-07-25	luoyunfeng	
V1.2	2019-08-30	wuxiaopeng	增加 PMU 和 PM 的使用说明
V1.3	2019-11-29	chengkan	增加音效软件 API 说明

目　录

1.编辑/编译/下载

1.1. 编辑

常用的编辑工具有 Source Insight、Sublime Text、Visual Studio Code，也可以使用其他编辑工具。

为更好地显示中文注释，Source Insight 请使用 4.0 或以后版本。

1.2. 编译

1.2.1. 编译工具

安装 Cygwin，安装选项上选择全部安装；

将 color-compile-master 文件夹下的 2 个文件复制到安装路径，例如：C:\cygwin\usr\local\bin。

双击“start.bat”,会打开 Cygwin Terminal 窗口，并切换到默认路径。或者打开安装好的 Cygwin Terminal，使用命令方式将当前路径切换到 code 文件夹，例如：cd d:/SLink_v1.00/code。

1.2.2. 工具链

请将工具链复制到 tools 文件夹下。

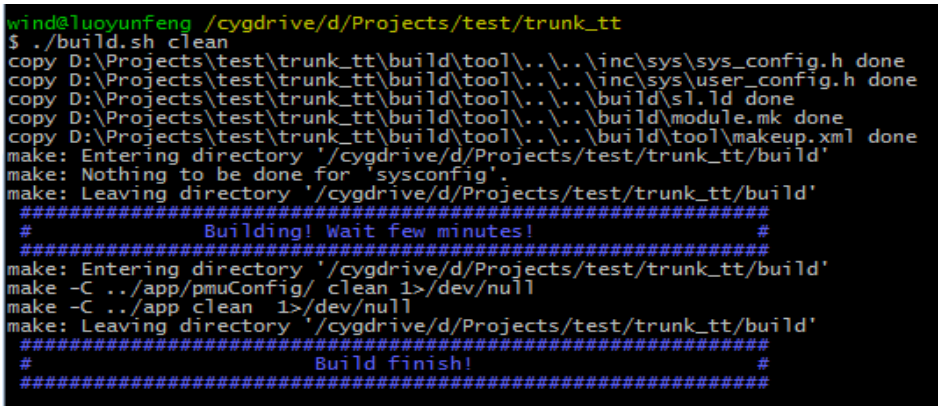
工具链有 2 个版本：

toolchain-7.3.1.tar.bz2，v1.30 以前版本的 SDK 上使用。

toolchain-7.3.1-e20.tar.bz2，v1.30 或以后版本的 SDK 上使用。

1.2.3. 操作指令

清除指令：./build.sh clean



编译指令：./build.sh app

首次编译时，会自动解压工具链，需要等待一段时间。

1.3. 下载

1.3.1. 下载文件

build/out/sl.up

1.3.2. 下载工具

tools/flash_loader/flash_loader.exe

1.3.3. 操作方法

在将 PB9 拉低（一般是按下第一个 AD 按键）时，USB 线连 PC，选择 sl.up 文件，点击盘符图标，开始下载，下载完成后，断开与 PC 的连接，上电启动。

2.文件结构

2.1. app

上层应用相关文件、代码。

2.1.1. configs

系统配置、用户配置的文件。

config.h：根据芯片型号/功能选择配置文件。

xx.ld：链接配置

xx_sys_config.h：系统配置文件

xx_user_config.h：用户配置文件

注意：用户请更改 app/configs/sl6800 路径下的 xx_user_config.h，在编译时会自动复制到 inc/sys 路径下，并替换文件 user_config.h。

建议所有.c 文件头中#include“user_config.h”

2.1.2. starts

硬件相关的模块接口。

2.1.3. moon

软件相关的模块接口。

2.1.4. task

各场景模式相关文件。

2.1.5. main.c

系统主入口。

2.2. build

编译相关文件、工具。

2.2.1. bin

提示音数据、EQ 数据。

2.2.2. out

编译输出文件。

2.2.3. tool

编译时用到的工具。

2.3. inc

应用层用到的头文件。

2.4. lib

底层编译生成的库文件。

2.5. tools

工具链和其他各工具。

2.6. start.bat

双击运行，会打开 Cygwin Terminal 窗口，并切换到默认路径。

3.代码空间

3.1. 概述

用户代码存储在 FLASH 中。

代码有 2 种运行方式：

（1）从 FLASH 加载到 SRAM 中运行，运行效率高。

（2）FLASH 直接运行，运行效率低。

所以，运行频率高、性能要求高的代码，如公共显示刷新、中断处理等，需要使用方式 1。运行频率不高、性能要求不高的代码，如初始化、切换等，可使用方式 2。

3.2. 分配方法

给代码函数分配空间的方法：

首先在函数前加 AT(段名)，如“AT(.main_seg)”，然后在.ld 文件中填写段名。

在.ld 文件中，各区域说明如下：

区域	说明
.system_sec	基础系统需要在 SRAM 运行的代码或常量，请不要修改
.sram0_sec	各模块需要在 SRAM 运行的代码或常量，用户代码请在此添加
.app_data_sec	有初值的变量
.stack_sec	栈
.app_bss_sec	无初值的变量
.user_heap	堆
.sram1_sec	音效相关且需要在 SRAM 运行的代码或常量
.sram2_sec	蓝牙相关的变量或常量
.bank_sec	在 FLASH 中运行的代码或常量，每个 bank 大小是 8Kbytes

3.3. 出错处理

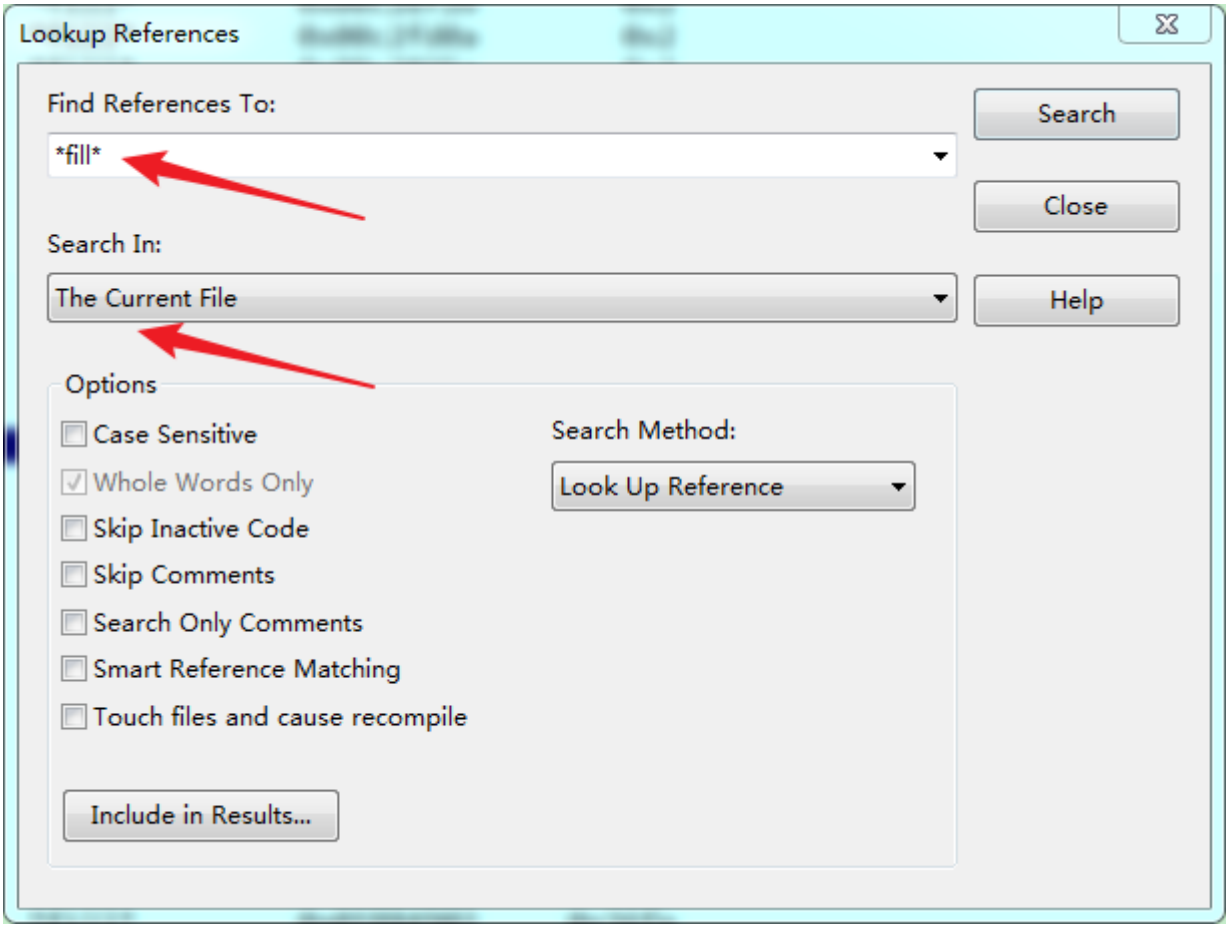
如果编译时出现错误提示“‘.bank_sec’will not fit in region‘BANK’”，如下图：

```
/cygdrive/d/Projects/test/trunk
$ ./build.sh app
copy D:\Projects\test\trunk\build\tool\..\..\inc\sys\sys_config.h done
copy D:\Projects\test\trunk\build\tool\..\..\inc\sys\user_config.h done
copy D:\Projects\test\trunk\build\tool\..\..\build\sl.ld done
copy D:\Projects\test\trunk\build\tool\..\..\build\module.mk done
make: Entering directory '/cygdrive/d/Projects/test/trunk/build'
make: Nothing to be done for 'sysconfig'.
make: Leaving directory '/cygdrive/d/Projects/test/trunk/build'
 ############################################################
 #              Building! Wait few minutes!                 #
 ############################################################
make: Entering directory '/cygdrive/d/Projects/test/trunk/build'
make -C ../app/pmu_config install 1>/dev/null
make -C ../app install 1>/dev/null
d:/projects/test/trunk/tools/toolchain/bin/../lib/gcc/arm-none-eabi/7.3.1/../../../../arm-none-eab
i/bin/ld.exe: ../build/out/sl.axf section `.bank_sec' will not fit in region `BANK'
d:/projects/test/trunk/tools/toolchain/bin/../lib/gcc/arm-none-eabi/7.3.1/../../../../arm-none-eab
i/bin/ld.exe: section .user_heap LMA [00816564,0081e563] overlaps section .bank_sec LMA [00016000,
00c9a407]
d:/projects/test/trunk/tools/toolchain/bin/../lib/gcc/arm-none-eabi/7.3.1/../../../../arm-none-eab
i/bin/ld.exe: region `BANK' overflowed by 12837896 bytes
collect2.exe: error: ld returned 1 exit status
make[1]: *** [../build/out/sl.axf] Error 1
Makefile:43: recipe for target 'app' failed
make: *** [app] Error 2
make: Leaving directory '/cygdrive/d/Projects/test/trunk/build'
 ############################################################
 #                        Build fail!                       #
 ############################################################
```

可以按以下方法处理：

（1）打开 app 文件夹下（不是 build/out 下）的 sl.map 文件。

（2）搜索内容“*fill*”，只选择在当前文件内。



（3）找到大小特别大（超出 0x00c00000）的那一行。

```
sl.map (D:\Projects\test\trunk\app) line 13023 :  *fill*          0x00c3377e        0x2
sl.map (D:\Projects\test\trunk\app) line 13231 :  *fill*          0x00c347c2        0x2
sl.map (D:\Projects\test\trunk\app) line 13270 :  *fill*          0x00c34ae4        0x4
sl.map (D:\Projects\test\trunk\app) line 13278 :  *fill*          0x00c35014        0x4
sl.map (D:\Projects\test\trunk\app) line 13294 :  *fill*          0x00c35538    0xc26ac8
sl.map (D:\Projects\test\trunk\app) line 13326 :  *fill*          0x0185c3a0      0x3c60
sl.map (D:\Projects\test\trunk\app) line 13337 :  *fill*          0x018600b6         0x2
sl.map (D:\Projects\test\trunk\app) line 13404 :  *fill*          0x01860efc      0x3104
sl.map (D:\Projects\test\trunk\app) line 13439 :  *fill*          0x018642e4      0x3d1c
```

（4）跳转到 sl.map 文件。

```
13286:  .text.nan       0x00c35168        0x10 d:/projects/test/trunk/tools/toolchain/bin/
13287:                  0x00c35168                nan
13288:  .text.__ieee754_log
13289:                  0x00c35178       0x3b4 d:/projects/test/trunk/tools/toolchain/bin/
13290:                  0x00c35178                __ieee754_log
13291:  .text.__errno   0x00c3552c         0xc d:/projects/test/trunk/tools/toolchain/bin/
13292:                  0x00c3552c                __errno
13293:                  0x0185c000                . = ALIGN ((BANK_ADDR + 0x16000))
13294:  *fill*          0x00c35538    0xc26ac8
13295:  *(.bank11_seg)
```

（5）可看出下一个是 bank11，那么确认是 bank10 空间溢出了，需要在.ld 文件中，将 bank10 的部分段名移出到其他 bank 内。

4.宏配置介绍

4.1. 系统相关宏配置

请更改在 app/configs 路径下的 xx_sys_config.h 文件，而不要更改 inc/sys 路径下的 sys_config.h 文件。

在 xx_sys_config.h 中的宏配置，没有提到的，建议用户不要更改。

宏	描述	默认值	说明
WATCHDOG_EN	看门狗使能	0	watchdog config
WATCHDOG_TIMEOUT	看门狗复位时间(单位：s)	5	watchdog config
DEBUG_LOG_EN	调试打印使能	0	debug config

4.2. 用户相关宏配置

请更改在 app/configs 路径下的 xx_user_config.h 文件，而不要更改 inc/sys 路径下的 user _config.h 文件。

宏	描述	默认值	说明
MODE_BT_EN	蓝牙模式使能	1	scene mode config
MODE_MUSIC_EN	本地音乐模式使能	1	scene mode config
MODE_RECORD_EN	录音模式使能	0	scene mode config
MODE_FM_EN	FM 收音模式使能	1	scene mode config
MODE_LINEIN_EN	音频输入模式使能	1	scene mode config
MODE_USBDEV_EN	USB 从机模式使能	1	scene mode config
MODE_SPDIF_EN	光纤输入模式使能	0	scene mode config
MODE_CLOCK_EN	时钟模式使能	1	scene mode config
MODE_POWEROFF_EN	软关机模式使能	1	scene mode config
MODE_CHARGE_EN	充电模式使能	1	scene mode config
MODE_DISKUPDATE_EN	设备升级固件模式使能	1	scene mode config

BT_BACKGROUND_EN	蓝牙后台功能	0	bt config
BT_PHONE_EN	蓝牙电话功能	1	bt config
BT_SPP_EN	蓝牙 SPP 功能	0	bt config
BT_BLE_EN	蓝牙 BLE 功能	0	bt config
BT_HID_EN	蓝牙 HID 功能	0	bt config
BT_ADDR_USE_RANDOM	蓝牙地址使用随机数	1	bt config
BT_FCC_TEST_EN	蓝牙 FCC 认证定频功能	0	bt config
BT_VOLUME_EN	音量与手机同步功能	1	bt config
RECONNECT_SCAN_EN	回连时可以被搜索和连接	1	bt config
BT_SIRI_EN	苹果设备的 siri 功能	0	bt config
BT_ADDR_DEFAULT	蓝牙地址默认值		bt config
BT_NAME_DEFAULT	蓝牙名字默认值		bt config
BT_TIMEOUT_RECONNECT_TIMES	超时/远距离断线后回连次数	20	bt config
BT_POWERON_RECONNECT_TIMES	开机时回连次数	3	bt config
MUSIC_NUM_STEP10	上下首时文件编号加减 10	0	music config
MUSIC_MUTE_FOR_FAST	快进/快退时静音	1	music config
MUSIC_DEVICE_SWITCH	U 盘或 SD 卡内循环播放	1	music config
MUSIC_METADATA	ID3 信息获取	0	music config
MUSIC_WAV	WAV 文件播放	1	music config
MUSIC_MP3	MP3 文件播放	1	music config
MUSIC_WMA	WMA 文件播放	1	music config
MUSIC_FLAC	FLAC 文件播放	1	music config
MUSIC_AAC	AAC 文件播放	1	music config
MUSIC_APE	APE 文件播放	1	music config
MUSIC_OGG	OGG 文件播放	1	music config
MUSIC_SBC	SBC 文件播放	0	music config
FM_CLK_SEL	FM 时钟源类型选择	2	fm config
FM_CLK_PIN_SEL	FM CLK 输出 PN 选择	5	fm config
FM_TWI_SEL	FM 使用的 TWI 模块选择	1	fm config
FM_TO_CODEC	FM 音频输入到 CODEC 模块使能	0	fm config
SD_D0_PIN_SEL	D0 PIN 选择	0	sd config

SD_CLK_PIN_SEL	CLK PIN 选择	0	sd config
SD_CMD_PIN_SEL	CMD PIN 选择	0	sd config
SD_CLKDIV	CLK 时钟选择	4	sd config
SD_DM_SHARE_EN	D0 与 USB_DM 共用 PB3	0	sd config
USB_EN	USB 功能（包括主机、从机模式）		usb config
USB_DETECT_EN	USB 检测		usb config
USB_DM_PIN_SEL	DM PIN 选择		usb config
USB_DP_PIN_SEL	DP PIN 选择		usb config
TWI1_EN	TWI1 模块	1	twi config
TWI1_SCL_PIN_SEL	TWI1_SCK PIN 选择	0	twi config
TWI1_SDA_PIN_SEL	TWI1_SDA PIN 选择	0	twi config
TWI1_SD_SHARE_EN	TWI1 与 SD 共用 PIN	1	twi config
TWI2_EN	TWI2 模块	0	twi config
TWI2_SCL_PIN_SEL	TWI2_SCK PIN 选择	3	twi config
TWI2_SDA_PIN_SEL	TWI2_SDA PIN 选择	3	twi config
TWI2_SD_SHARE_EN	TWI2 与 SD 共用 PB3/PB4	0	twi config
TWI2_DM_SHARE_EN	TWI2_SCL 与 USB_DM 共用 PB3	0	twi config
TWI2_DP_SHARE_EN	暂不支持	0	twi config
SPI0_EN	SPI0 模块	0	spi config
SPI0_CS_PIN_SEL	SPI0_CS PIN 选择	0	spi config
SPI0_CLK_PIN_SEL	SPI0_CLK PIN 选择	0	spi config
SPI0_MOSI_PIN_SEL	SPI0_MOSI PIN 选择	0	spi config
SPI0_MISO_PIN_SEL	SPI0_MISO PIN 选择	0	spi config
SPI1_EN	SPI1 模块	0	spi config
SPI1_MODE_SEL	SPI1 模式选择（硬件/软件）	1	spi config
SPI1_DIRECT_SEL	3 线/4 线模式选择	1	spi config
SPI1_CS_PIN_SEL	SPI1_CS PIN 选择	1	spi config
SPI1_CLK_PIN_SEL	SPI1_CLK PIN 选择	1	spi config
SPI1_MOSI_PIN_SEL	SPI1_MOSI PIN 选择	1	spi config
SPI1_MISO_PIN_SEL	SPI1_MISO PIN 选择	1	spi config
SPI_FLASH_TONE_SEL	提示音文件位置选择	0	spi config
SPI_FLASH_AUXTRACK_SEL	音效文件位置选择	1	spi config

TIMER0_EN	TIMER0 模块	0	timer config
TIMER0_PERIOD	TIMER0 周期(单位：us)	5000	timer config
TIMER1_EN	TIMER1 模块	0	timer config
TIMER1_PERIOD	TIMER1 周期(单位：us)	100000	timer config
TIMER2_EN	TIMER2 模块	0	timer config
TIMER2_PERIOD	TIMER2 周期(单位：us)	100000	timer config
TIMER3_EN	TIMER3 模块	0	timer config
TIMER3_PERIOD	TIMER3 周期(单位：us)	100000	timer config
PWM_EN	PWM 模块		pmw config
PWM_CH0_EN	PWM 模块通道 0 使能		pmw config
PWM_CH1_EN	PWM 模块通道 1 使能		pmw config
PWM_CH2_EN	PWM 模块通道 2 使能		pmw config
PWM_CH3_EN	PWM 模块通道 3 使能		pmw config
PWM_CH4_EN	PWM 模块通道 4 使能		pmw config
PWM_CH5_EN	PWM 模块通道 5 使能		pmw config
PWM_CH0_PIN_SEL	PWM 模块通道 0 PIN 选择		pmw config
PWM_CH1_PIN_SEL	PWM 模块通道 1 PIN 选择		pmw config
PWM_CH2_PIN_SEL	PWM 模块通道 2 PIN 选择		pmw config
PWM_CH3_PIN_SEL	PWM 模块通道 3 PIN 选择		pmw config
PWM_CH4_PIN_SEL	PWM 模块通道 4 PIN 选择		pmw config
PWM_CH5_PIN_SEL	PWM 模块通道 5 PIN 选择		pmw config
I2S0_EN	I2S0 模块	0	i2s config
I2S0_BCLK_PIN_SEL	I2S0 模块 BCLK PIN 选择	0	i2s config
I2S0_LRCK_PIN_SEL	I2S0 模块 LRCK PIN 选择	0	i2s config
I2S0_TX _SEL	I2S0 模块 TX 使能	1	i2s config
I2S0_TX_PIN_SEL	I2S0 模块 TX PIN 选择	0	i2s config
I2S0_RX_EN	I2S0 模块 RX 使能	0	i2s config
I2S0_RX_PIN_SEL	I2S0 模块 RX PIN 选择	2	i2s config
I2S0_MCLK_EN	I2S0 模块 MCLK 使能	0	i2s config
I2S0_MCLK_PIN_SEL	I2S0 模块 MCLK PIN 选择	2	i2s config
I2S0_MCLK_FACTOR	I2S0 模块 MCLK 相对采样率倍数	256	i2s config
I2S0_USE_WM8978	外部模块 WM8978 使能	0	i2s config

I2S0_USE_TAS5711	外部模块 TAS5711 使能	0	i2s config
SPDIF_EN	SPDIF 模块使能	0	spdif config
SPDIF_PIN_SEL	SPDIF PIN 选择	0	spdif config
UART0_EN	UART0 模块使能	1	uart config
UART0_BAUDRATE	UART0 模块波特率选择	3000000	uart config
UART0_SCLK_FREQ	UART0 模块时钟源选择	48000000	uart config
UART0_TX_PIN_SEL	UART0 模块 TX PIN 选择	3	uart config
UART0_RX_PIN_SEL	UART0 模块 RX PIN 选择	4	uart config
UART1_EN	UART1 模块使能	0	uart config
UART1_BAUDRATE	UART1 模块波特率选择	115200	uart config
UART1_HWFLOWCTL	UART1 模块流控使能		uart config
UART1_SCLK_FREQ	UART1 模块时钟源选择	48000000	uart config
UART1_CTS_PIN_SEL	UART1 模块 CTS PIN 选择	4	uart config
UART1_RTS_PIN_SEL	UART1 模块 RTS PIN 选择	4	uart config
UART1_TX_PIN_SEL	UART1 模块 TX PIN 选择	5	uart config
UART1_RX_PIN_SEL	UART1 模块 RX PIN 选择	5	uart config
UART2_EN	UART2 模块使能	0	uart config
UART2_BAUDRATE	UART2 模块波特率选择	115200	uart config
UART2_HWFLOWCTL	UART2 模块流控使能		uart config
UART2_SCLK_FREQ	UART2 模块时钟源选择	48000000	uart config
UART2_CTS_PIN_SEL	UART2 模块 CTS PIN 选择	0	uart config
UART2_RTS_PIN_SEL	UART2 模块 RTS PIN 选择	0	uart config
UART2_TX_PIN_SEL	UART2 模块 TX PIN 选择	0	uart config
UART2_RX_PIN_SEL	UART2 模块 RX PIN 选择	0	uart config
TONE_EN	提示音使能（总开关）	1	tone config
TONE_VOLUME	提示音音量	40	tone config
RTC_EN	RTC 模块使能	1	rtc config
ALARM0_EN	RTC 模块闹钟 0 使能	0	rtc config
ALARM1_EN	RTC 模块闹钟 1 使能	1	rtc config
KEY_AD_EN	AD 采样方式按键使能	1	key config

KEY_IO_EN	IO 扫描方式按键使能	0	key config
KEY_IR_EN	红外按键使能	1	key config
KEY_TOUCH_EN	触摸按键使能	0	key config
KEY_CODING_EN	编码开关旋钮使能	0	key config
KEY_UART_EN	UART 方式按键使能（调试用）	0	key config
KEY_PMU_EN	ONOFF 按键用作 P/P 键使能	1	key config
IRRX_PIN_SEL	红外接收模块 PIN 选择	1	irrx config
DISP_EN	显示功能使能	1	disp config
LED_DIGITAL_EN	数码式 LED 显示使能	1	disp config
LED_DIODE_EN	单个 LED 灯显示使能	0	disp config
LCD_SEG_EN	段码式 LCD 显示使能	0	disp config
MOTOR_EN	马达电机使能	0	motor config
PMU_POWER_ON_MODE	开机模式配置，ONOFF 或 SWITCH（烧录用）	0	pmu startup mode config
PMU_CORE_USE_DCDC	CORE 供电方式选择	0	pmu config
PMU_VBUS_POWERON_EN	使能 VBUS 插入开机	1	pmu config
PMU_SHORT_PRESS_POWERON_EN	使能短按开机	0	pmu config
PMU_ULTRA_SHORT_PRESS_POWERON_EN	使能超短按开机	0	pmu config
PMU_POWEROFF_WHEN_BAT_FIRST_IN	使能第一次插入电池时关机	0	pmu config
PMU_ULTRA_LONG_PRESS_TIME	设置超长按时长	8000	pmu config
PMU_LONG_PRESS_TIME	设置长按时长	3000	pmu config
PMU_CHARGE_CURRENT	设置充电电流	200	pmu config
PMU_VBUSIN_RESTART	使能 VBUS 插入后重启	0	pmu config
PMU_VCC_RTC_ALWAYS_ON	设置 RTC 是否常在电	1	pmu config
PMU_VCC_IO_VOLT	设置 VCC-IO 电压	3300	pmu config
PMU_AVCC_VOLT	设置 AVCC 电压	2700	pmu config

BAT_EXIST	使用内部 PMU 管理电池	1	battery config
BAT_VOLT_DET	使能内部 PMU 电池电池电压检测	1	battery config
BAT_QUANTITY_STEP_BIG	电池电量百分比选择 0：10 级，1：5 级	1	battery config
BAT_HW_LB_SHUT_EN	使能硬件低电关机	1	battery config
BAT_HW_LB_SHUT_VOLT	设置硬件低电关机电压	2700	battery config
BAT_HW_OK_VOLT	设置硬件允许开机电压	2700	battery config
BAT_SW_LB_WRN_VOLT	设置软件低电提醒电压	3300	battery config
BAT_SW_LB_SHUT_VOLT	设置软件低电关机电压	3100	battery config
BAT_SW_LB_WARNING_PERIOD	设置软件低电提醒周期	120000	battery config
SLEEP_EN	睡眠使能	1	power/sleep config
SLEEP_DELAY	睡眠前延时（单位：s）	10	power/sleep config
POWER_OFF_DELAY	关机前延时（单位：ms）	0	power/sleep config
POWER_OFF_FIRST_EN	vbus 首次上电后进入关机	0	power/sleep config
DVFS_KARAOKE_FREQ	KARAOKE 场景下 CPU 频率	192e6	cpu freq config
DVFS_EFFECTS_FREQ	音效场景下 CPU 频率	192e6	cpu freq config
DVFS_TONEMIXER_FREQ	提示音乐场景下 CPU 频率	64e6	cpu freq config
DVFS_A2DPAUDIO_FREQ	蓝牙音乐场景下 CPU 频率	64e6	cpu freq config
DVFS_SCOAUDIO_FREQ	蓝牙通话场景下 CPU 频率	64e6	cpu freq config
DVFS_SCOAENC_FREQ	蓝牙通话+降噪场景下 CPU 频率	64e6	cpu freq config
DVFS_USBAUDIO_FREQ	USB 声卡播放场景下 CPU 频率	192e6	cpu freq config
DVFS_LINEINAUDIO_FREQ	LINEIN 播放场景下 CPU 频率	64e6	cpu freq config
DVFS_LOCAL_ENCODE_WAV_FREQ	本地编码（WAV）场景下 CPU 频率	64e6	cpu freq config
DVFS_LOCAL_ENCODE_MP3_FREQ	本地编码（MP3）场景下 CPU 频率	192e6	cpu freq config
DVFS_LOCAL_DECODE_WAV_FREQ	本地解码（WAV）场景下 CPU 频率	64e6	cpu freq config
DVFS_LOCAL_DECODE_MP3_F	本地解码（其他）场景下 CPU 频率	192e6	cpu freq config

REQ			
DVFS_DEFAULT_FREQ	默认 CPU 频率	64e6	cpu freq config

4.3. 音效相关宏配置

下表为 xx_user_config.h 和 app_afx_config.h 中的可调配置

宏	描述	作用通路	使用限制
EQ_REALTIME	Eq Controller 实时调试	N/A	此二者一起开或者一起关，如果单独不开 EQ_ALL_MODE_SUPPORT，调试将只能在 PC 模式下运行
EQ_ALL_MODE_SUPPORT	Eq Controller 全模式调试	N/A	
AUDIO_AFX_SCO_TSPS_EN	通话模式变调使能	通话	
AUDIO_PLBK_CPUX_ASYNC	PLBK 一路算法是否采用 CPU0/1 异步并行处理。牺牲(proc samples)*2*4 bytes RAM，换取 CPU0/CPU1 算法的异步并行处理，提升 CPUX 协同效率	播放	AUDIO_KARAOKE_MODE 关闭，且 AUDIO_PLAYBACK_VSS 使能
AUDIO_PLAYBACK_EQ	Eq 使能	播放	
AUDIO_PLAYBACK_DRC	Drc 使能	播放	
AUDIO_PLAYBACK_TSPS	变调使能	播放	
AUDIO_PLAYBACK_VSS	虚拟环绕使能	播放	
AUDIO_AFX_VSS_DEFAULT_SWITCH	虚拟环绕默认开关，0 关 1 开		
AUDIO_AFX_VSS_DEFAULT_HP_MODE	虚拟环绕默认打开是否是耳机模式 0 否 1 是		
AUDIO_AFX_VSS_SWITCH_FADEINOUT	虚拟环绕开关是否淡入淡出 0 否 1 是		
AUDIO_AFX_VSS_EQ_ENABLE	虚拟环绕内部 EQ 使能		
AUDIO_PLAYBACK_DR	延时/反向使能	播放	AUDIO_AFX_PLBK_POST_PROC_EN 使能
AUDIO_PLAYBACK_TBVC	等响使能	播放	
AUDIO_AFX_TBVC_VOL_SYNC_PRIRY	等响同步外部音量频率，x=(0-7)，代表每 2^x 帧同步一次	播放	AUDIO_PLAYBACK_TBVC 使能
AUDIO_AFX_PLBK_POST_PROC_EN	音频后处理使能	播放	后处理相当于多了一份 14 段双通道 EQ 和 DRC，当 VSS 打开时，后处理中的 EQ,DRC 要谨慎使用，EQ 尽量用少点段。
AUDIO_AFX_HLF_EN	后处理输出 1、2 专业高低通使能	播放	输出 1、2 的 EQ 点数降为 10，AUDIO_AFX_PLBK_POST_PROC_EN 使能
AUDIO_KARAOKE_MODE	卡拉 OK 录音总开关	MIC	
AUDIO_MIC_REC_CPUX_ASYNC	MIC 一路算法是否采用 CPU0/1 异步并行处理。牺牲(proc samples)*4 bytes RAM，换取 CPU0/CPU1 算法的异步并行处理，提升 CPUX 协同效率	MIC	AUDIO_KARAOKE_MODE 使能，且 AUDIO_MIC_REVERB 或者 AUDIO_MIC_TSPS 使能。
AUDIO_MIC_EQ	Eq 使能	MIC	AUDIO_KARAOKE_MODE 使能
AUDIO_MIC_DRC	Drc 使能	MIC	AUDIO_KARAOKE_MODE 使能
AUDIO_MIC_ECHO	Echo 使能	MIC	AUDIO_KARAOKE_MODE 使能

AUDIO_AFX_ECHO_MAX_DLY	Echo 最大回声设置	MIC	如果 AUDIO_AFX_ECHO_MAX_DLY 超过 200，建议打开 AUDIO_AFX_ECHO_CUSTDOWN
AUDIO_AFX_ECHO_CUSTDOWN	Echo 低资源模式	MIC	
AUDIO_AFX_POST_ECHO_EQ_EN	Echo 后置 EQ 使能	MIC	AUDIO_MIC_ECHO 使能
AUDIO_MIC_REVERB	混响使能	MIC	AUDIO_KARAOKE_MODE 使能
AUDIO_AFX_POST_REVB_EQ_EN	混响后置 EQ 使能	MIC	AUDIO_MIC_REVERB 使能
AUDIO_MIC_TSPS	变调使能	MIC	AUDIO_KARAOKE_MODE 使能
AUDIO_AFX_REC_FORMANT_EN	变调共振峰子功能使能	MIC	AUDIO_MIC_TSPS 使能
AUDIO_AFX_REC_AUTOTUNE_EN	变调电音使能	MIC	AUDIO_MIC_TSPS 使能
AUDIO_KARAOKE_USE_FS	移频算法抗啸叫	MIC	AUDIO_KARAOKE_MODE 使能
AUDIO_AFX_FREQ_SHIFT_IMPROVE	FS 高阶希尔伯特滤波器使能，减轻低频衰减	MIC	AUDIO_KARAOKE_USE_FS 使能，高阶 CPU 使用有所提高
AUDIO_AFX_FREQ_SHIFT_RUN_CPU1	FS 算法是否在 CPU1 中运行，否的话，分摊算力到 CPU0 运行	MIC	AUDIO_KARAOKE_USE_FS 使能 AUDIO_MIC_REC_CPUX_ASYNC 打开
AUDIO_KARAOKE_USE_NHS	动态陷波抗啸叫	MIC	AUDIO_KARAOKE_MODE 使能
AUDIO_AFX_NHS_RUN_CPU1	动态限波算法是否在 CPU1 中运行，否的话，分摊算力到 CPU0 运行		AUDIO_KARAOKE_USE_NHS 使能 AUDIO_MIC_REC_CPUX_ASYNC 打开
AUDIO_PR_MIX_SPECTRUM	KARAOKE 混音之后的频谱	MIX	KARAOKE 混音频谱优先使能 两者互斥
AUDIO_PLAYBACK_SPECTRUM	单音乐频谱	播放	
BETTER_VOCAL_CUT	高阶消原音使能	MIX	AUDIO_KARAOKE_MODE 使能
AUDIO_KARAOKE_USE_DUCKER	Karaoke 闪避功能使能	MIX	AUDIO_KARAOKE_MODE 使能
AUDIO_KARAOKE_START_WITH_REVERB	Karaoke 开机启动默认混响	MIC	AUDIO_KARAOKE_MODE，AUDIO_MIC_REVERB 使能
AUDIO_KARAOKE_START_WITH_ECHO	Karaoke 开机启动默认回声	MIC	AUDIO_KARAOKE_MODE，AUDIO_MIC_ECHO 使能
EQ_LOAD_EN	使能加载固化 EQ 功能	播放/录音	

5.系统调试快速入门

5.1. 确认芯片型号

目前有以下型号的芯片：

序号	芯片型号	封装	core 供电方式
1	HS100	QFN32	DC/DC
2	WS100	QFN48	LDO
3	WS101	QFN32	LDO
4	WS102	QFN48	LDO
5	CA100	QFN48	LDO
6	CA101	QFN32	LDO
7	CA102	QFN48	LDO

5.2. 选择配置文件

根据芯片型号选择配置文件，配置文件与芯片型号的对应关系见下表，在 app\configs\config.h 文件中更改宏定义 SYS_BOARD。

序号	配置文件	
1	BOARD_SL6800	普通功能
2	BOARD_SL6800_ KARAOKE	带 karaoke 功能

5.3. 选择供电/开机方式

在文件 xx_user_config.h 里 battery config 一栏，宏定义 BAT_EXIST，表示是否使用带充电方式的电池供电。

宏 BAT_EXIST	供电方式	开机方式
1	带充电方式的电池供电	按 ONOFF 键开机
0	其他方式供电	电源接通就开机

5.4. 配置打印输出

5.4.1. 开/关配置

在文件 xx_sys_config.h 里 debug config 一栏，宏定义 DEBUG_LOG_EN 配置为 1。

在文件 xx_user_config.h 里 uart config 一栏，宏定义 UART0_EN 配置为 1。

5.4.2. IO 配置

宏定义 UART0_TX_PIN_SEL 配置打印输出 IO。

5.4.3. 波特率配置

宏定义 UART0_BaudRate 配置合适的波特率，最高支持 3Mbps，串口工具上使用相同波特率。

5.4.4. 打印等级

在各.c 文件头定义打印等级，如#define LOG_LEV 4。

使用各打印函数，如 logi，见文件 log.h。

5.5. 注意

请不要更改 inc/sys 路径下的 sys_config.h 和 user_config.h 文件，因为改了也不起作用。

6.音频控制软件接口

6.1. 通用接口

通用接口在任何模式下都可以调用：

函数	作用
int volume_down();	减播放音量
int volume_up();	加播放音量
int volume_set(uint8_t vol);	设置播放音量; vol: 0 ~ 100
void audio_service_set_record_volume(uint8_t volume);	设置录音(MIC)音量; volume: 0 ~ 100
uint8_t audio_service_get_record_volume(void);	获取当前录音(MIC)音量(0~100)
void audio_service_set_playback_volume(uint8_t volume);	设置音乐播放音量; volume：0 ~ 100
uint8_t audio_service_get_playback_volume(void);	获取当前音乐播放音量(0~100)
void audio_service_set_karaoke_mixout_volume(uint8_t volume);	设置卡拉 OK 混音输出音量; volume: 0 ~ 100
uint8_t audio_service_get_karaoke_mixout_volume(void);	获取当前卡拉 OK 混音输出音量(0~100)
void audio_scene_save(void);	保存播放场景
void audio_scene_recovery(void);	恢复之前的播放场景

6.2. SD 卡/U 盘播放

SD 卡/U 盘播放接口只允许在 MUSIC MODE 下被调用：

函数	作用
int play_file(int index, bool reset_pause_sta);	播放第 index 首歌曲
void play_switch_file(bool next_file);	播放上/下一曲
void play_switch_folder(bool next_folder);	播放上/下一文件夹
int play_or_pause();	播放或者暂停。如果当前正在播放，调用此函数会暂停播放；如果当前暂停播放，调用此函数会继续播放。
void fast_skip(bool forward);	快进快退(forward: 0-快退；1-快进)
int fast_end(void);	结束快退（快进）
int set_repeat();	切换重复播放模式。调用此函数会切换文件重复播放模式：全部循环->单曲循环->随机播放->文件夹循环
int get_play_time(void);	获取当前播放时间(进度)，播放时间存放在 np.ptime 结构体中。

	返回值：返回负值表示失败，返回 0 表示成功
int set_play_time(void);	设置播放时间。 返回值：返回负值表示失败，返回 0 表示成功
void play_info_write(uint32_t dev_num, uint32_t fnum, uint32_t ptime);	保持播放信息： dev_num:表示当前设备是 U 盘还是 SD 卡 fnum:表示当前文件编号 ptime:表示当前播放时间（进度），单位：毫秒
void play_num_read(uint32_t dev_num);	从 flash 读取上次播放的文件编号，存放到 np.file_num 中
void play_time_read(uint32_t dev_num);	从 flash 读取上次播放的时间(进度)，存放到 np.ptime.ms

6.3. 提示音播放接口

函数	作用
void audio_service_set_tone_volume(uint8_t volume);	设置提示音音量; volume: 0 ~ 100
void audio_service_play_sin_tone(uint32_t tone_type);	播放单频提示音。低电量提示音。最大最小音量提示音通过此接口播放。 tone_type: SIN_TONE_VOLUME_LIMIT:最大(最小)音量提示音 SIN_TONE_BAT_LOW:低电量提示音 SIN_TONE_APP：用户自定义的单频提示音
int play_tone(uint8_t tone_num, bool block);	播放 flash 里面的提示音。 tone_num：提示音编号 block： 0 为非阻塞方式，即调完此函数时提示音由音频服务在后台播放；
int play_common_tone(uint8_t tone_num, uint32_t trigger_event);	插播提示音。
uint32_t get_common_tone_trigger_event(void);	获取触发 common tone 的事件
bool common_tone_is_playing(void);	判断 common tone 是否正在播放
void common_tone_status_clear(void);	清除 common tone 状态
int play_tone_chain(int (*set_next_tone_name)(source_info_t *));	播放提示音链(连续播放多个提示音）

6.4. 蓝牙播放接口

蓝牙播放接口由 bt_audio_server.h 给出，API 列表如下：

函数	作用
void bt_audio_server_create(void);	创建一个 bt_audio 实例
void bt_audio_server_destroy(void);	销毁一个 bt_audio 实例
uint8_t bt_audio_sta_get(void);	获取 bt_audio 当前的状态(返回状态，参考@enum bt_audio_status_e)
bt_audio_a2dp_start();	开始播放蓝牙音乐
bt_audio_a2dp_stop();	停止播放蓝牙音乐
bt_audio_sco_start();	开始通话
bt_audio_sco_stop();	停止通话
bt_audio_set_volume(vol);	通知 bt_audio 当前的音量
bt_audio_a2dp_skip_time(skip_ms);	跳过多少毫秒的音乐数据不播放

6.5. Linein 播放接口

Linein 播放接口由 linein_audio.h 给出，API 列表如下：

函数	作用
void create_linein_audio(void);	创建一个 linein audio 实例
void destroy_linein_audio(void);	销毁一个 linein audio 实例
linein_audio_mute(mute);	mute/unmute linein 播放
linein_audio_is_muted();	获取 linein audio muted 状态
linein_audio_set_volume(vol);	设置 linein audio 音量(音量范围：0 ~ 100)
linein_audio_get_volume();	获取 linein audio 音量(音量范围：0 ~ 100)
void linein_audio_source_ctrl_set_mute(bool mute);	Mute 或者 Unmute Linein
void app_linein_audio_source_process(audio_stream_info_t *info);	linein 播放预留处理接口，此接口会被 audio 库调用。如果需要，请实现此处理函数

6.6. FM 播放接口

和 linein 播放接口一致

6.7. USB Audio 播放/录音接口

USB Audio 播放录音接口由 usbaudio.h 给出，API 列表如下：

函数	作用
void create_usbaudio(usbaudio_init_t *params);	创建一个 usbaudio 实例
void destroy_usbaudio(void);	销毁一个 usbaudio 实例

usbaudio_playback_set_volume(vol);	设置 USB audio 播放音量(范围:0~100)
usbaudio_playback_get_volume();	获取 USB audio 播放音量(范围:0~100)
usbaudio_playback_set_mute(mute);	mute/unmute USB audio 播放
usbaudio_playback_get_mute();	获取 USB audio 播放 mute 状态
usbaudio_record_set_volume(vol);	设置 USB audio 录音音量(范围：0~100)
usbaudio_record_get_volume();	获取 USB audio 录音音量(范围：0~100)

usbaudio_record_set_type(capture_type)	配置 USB audio 录音的位置
	Example:
	//录音乐数据
	usbaudio_record_set_type(AUDIO_CAPTURE_TYPE_MUSIC_AFX_OUTPUT);
	//录 MIC 算法前数据
	usbaudio_record_set_type(AUDIO_CAPTURE_TYPE_MIC_AFX_INPUT);
	//录 MIC 算法后数据
	usbaudio_record_set_type(AUDIO_CAPTURE_TYPE_MIC_AFX_OUTPUT);
	//录 MIC+Music 数据（USB IN 数据除外）
	usbaudio_record_set_type(AUDIO_CAPTURE_TYPE_MUSIC_MIC_MIX);
	//录所有数据（包含 USB IN 数据）
	usbaudio_record_set_type(AUDIO_CAPTURE_TYPE_ALL);

usbaudio_record_set_mute(mute)	mute/unmute USB audio 录音
usbaudio_record_get_mute()	获取 USB audio 录音 mute 状态
usbaudio_playback_write_data(buf, samples)	write pcm data to the usbaudio
usbaudio_record_read_data(buf, samples)	read pcm data from the usbaudio
void app_usb_audio_source_process(audio_stream_info_t *info);	USB audio 播放预留处理接口，此接口会被 audio 库调用。如果需要，请实现此处理函数
void app_usb_audio_sink_process(audio_stream_info_t *info);	USB audio 录音预留处理接口，此接口会被 audio 库调用。如果需要，请实现此处理函数

6.8. Spdif 播放接口

Spdif 播放录音接口由 spdifaudio.h 给出，API 列表如下：

函数	作用
void create_spdifaudio(bool mute);	创建一个 spdifaudio 实例

void destroy_spdifaudio(void);	销毁一个 spdifaudio 实例
spdifaudio_mute(mute);	mute/unmute spdifaudio
spdifaudio_is_muted();	获取 spdifaudio mute 状态

6.9. SD 卡/U 盘/flash 录音接口

函数	作用
void start_record(void);	开始录音
void stop_record(void);	停止录音
int start_play_record_file(void);	开始播放录音文件
void stop_play_record_file(void);	停止播放录音文件
int32_t select_record_device(uint32_t dev_num);	选择录音设备
void get_record_time(void);	获取录音时间
void delete_record_file_by_num(void);	删除当前录音文件
int play_next_record_file(void);	播放下一个录音文件
int play_prev_record_file(void);	播放上一个录音文件
int play_or_pause_record(void);	录音文件播放/暂停
void linein_record_init(uint32_t rec_func);	Linein(fm) 模式录音初始化
void linein_record_deinit(void);	linein (fm) 模式录音退出
void linein_record_update_disp_time(void);	更新 linein(fm) 模式录音(或者录音播放)的显示时间
void linein_record_proc_sd_event(uint8_t mode, uint32_t event);	Linein(fm) 模式录音处理 SD 卡事件
void linein_record_proc_udisk_event(uint8_t mode, uint32_t event);	Linein(fm) 模式录音处理 U 盘事件
void linein_record_play_rec_file(uint32_t mode_id);	Linein(fm) 模式录音文件播放
void linein_stop_play_rec_file(uint8_t mode);	Linein(fm) 模式停止播放录音文件

6.10. 卡拉 OK 方案相关接口

6.10.1. AuxTrack 接口

AuxTrack 接口用于播放音乐时播放另一路辅助音频数据，它从外置或内置 flash 中读出预存 SBC 数据，解码后和音乐以及 MIC 做混音后输出。目前此功能仅在卡拉 OK 模式下可用。

函数	作用
audio_service_cmd(AS_CMD_PLAY_AUXTRACK, num)	播放编号为 num 的 AuxTrack
audio_service_cmd(AS_CMD_SET_AUXTRACK_VOLUME,vol)	设置 AuxTrack 音量（vol: 0~100）
uint32_t sta; audio_service_cmd(AS_CMD_GET_AUXTRACK_STATUS,	获取播放状态： sta 为 0： 没有正在播放。

(uint32_t)&sta)	sta 为 1：正在播放。

7.输入/输出（IO）

7.1. 概述

7.1.1. 基本情况

可供用户使用的 IO 有以下这些，不同封装上有差异：

PA：PA4 ~ PA5，共 2 个。

PB：PB0 ~ PB13，共 14 个。

PC：PC3 ~ PC4，共 2 个。

PD：PD0 ~ PD1，共 2 个。

DIO：DIO0 ~ DIO1，共 2 个。

7.1.2. 特别 IO 说明

其中以下 IO 需要特别注意：

PA4/5：

在上电瞬间存在短时间的输出高。

不支持外部中断功能。

PB9：

芯片进入升级模式前需要将这个 IO 拉低。

在开机上电后的 50ms 时间范围内，存在 2K 欧的内部上拉，之后上拉关闭。

PC3/4：

工作在 1.8V 电源域，不能直接用 3.3V 的信号输入。如需跟外部 3.3V 输入信号对接，需要通过电阻做分压设计，保证输入信号不超过 1.8V。

PC3/4 上电默认功能是外部低频晶振功能，在上电瞬间存在短时间的输出状态不确定，不建议用作状态指示灯。

PC3/4 建议用作基本输入输出控制（状态灯除外）。

PC3/4 在软关机（VBAT 引脚保持有电）中，不会像其他 GPIO 那样自动处于低电平状态，需要手动配置 GPIO 的功能和状态，例如输出 0。

PD1：

为部分功放的 mute 操作作特别设计，内部设置有常开的 100K 下拉电阻，不能关闭。

DIO0/1：

不支持外部中断功能。

DIO0/1 使用专门的驱动接口，见代码中 dio.c 和 dio.h 文件。

7.2. 功能

各 IO 的功能，见下表：

Name	IN	OUT	IRQ	FUNCTION MULTI								
PA4	INPUT	OUTPUT	/	/	/	/	/	/	/	/	/	/
PA5	INPUT	OUTPUT	/	/	/	/	/	/	/	/	/	/
PB0	INPUT	OUTPUT	PB_EINT0	SDC0_D0	TWI1_SCK	LED_D5	SPI0_CLK	UART2_TX	/	/	/	/
PB1	INPUT	OUTPUT	PB_EINT1	SDC0_CLK	FM_CLK	UART0_TX	SPI0_MOSI	/	/	/	/	/
PB2	INPUT	OUTPUT	PB_EINT2	SDC0_CMD	TWI1_SDA	I2S0_D_1	SPI0_MISO	UART2_RX	LED_D6	/	/	/
PB3	INPUT	OUTPUT	PB_EINT3	USB0-DM	UART0_TX	SDC0_D0	SDC0_D1	UART1_RTS	TWI2_SCK	UART2_TX	/	/
PB4	INPUT	OUTPUT	PB_EINT4	USB0-DP	UART0_RX	SDC0_CMD	SDC0_D2	UART1_CTS	TWI2_SDA	UART2_RX	/	/
PB5	INPUT	OUTPUT	PB_EINT5	MIC_CLK	PWM0	I2S0_MCLK	SPI1_CS	SDC0_D3	LED_D0	/	/	/
PB6	INPUT	OUTPUT	PB_EINT6	MIC_DATA0	PWM1	SDC0_CLK	I2S0_LRCK	SPI1_CLK	UART0_TX	TWI2_SCK	LED_D1	/
PB7	INPUT	OUTPUT	PB_EINT7	MIC_DATA1	SPI0_CS1	I2S0_BCLK	SPI1_MOSI	TWI2_SDA	PWM2	LED_D2	/	/
PB8	INPUT	OUTPUT	PB_EINT8	FM_CLK	UART0_TX	I2S0_D_0	SPI1_MISO	KEYADC5	PWM3	LED_D3	C_Touch_1	/
PB9	INPUT	OUTPUT	PB_EINT9	KEYADC0	C_Touch_0	UART0_TX	PWM4	/	/	/	/	/
PB10	INPUT	OUTPUT	PB_EINT10	KEYADC1	I2S0_D_0	SPI1_MISO	UART0_RX	I2S0_D_1	UART1_TX	PWM3	LED_D4	/
PB11	INPUT	OUTPUT	PB_EINT11	KEYADC2	UART0_TX	I2S0_MCLK	UART1_RX	PWM4	LED_D5	/	/	/
PB12	INPUT	OUTPUT	PB_EINT12	KEYADC3	TWI2_SCK	UART0_RX	I2S1_BCLK	UART1_RTS	UART2_RTS	PWM0	LED_D6	/
PB13	INPUT	OUTPUT	PB_EINT13	KEYADC4	TWI2_SDA	UART0_TX	I2S1_LRCK	UART1_CTS	UART2_CTS	PWM1	LED_D7	FM_CLK
PC3	INPUT	OUTPUT	PC_EINT3	X32KIN	IR_Rx	/	/	/	/	/	/	/
PC4	INPUT	OUTPUT	PC_EINT4	X32KOUT	FM_CLK	/	/	/	/	/	/	/
PD0	INPUT	OUTPUT	PD_EINT0	TWI2_SDA	PWM5	TWI1_SCK	UART1_TX	UART0_TX	SPDIF_IN_A	FM_CLK	MIC_CLK	LED_D8
PD1	INPUT	OUTPUT	PD_EINT1	TWI2_SCK	PWM4	TWI1_SDA	UART1_RX	UART0_RX	IR_Rx	SPDIF_IN_B	MIC_DATA0	/
DIO0	INPUT	OUTPUT	/	LED_D0	/	/	/	/	/	/	/	/
DIO1	INPUT	OUTPUT	/	LED_D1	/	/	/	/	/	/	/	/

7.3. 驱动能力

有 4 档可配置：level 0(450ohm)、level 1(140ohm)、level 2(80ohm)、level 3(60ohm)，默认配置为 level 1。

7.4. 上下拉电阻

各 IO 上下拉电阻有 2K ohm、100 K ohm，可以单个、多个、全部使能。

7.5. 中断

PB0 ~ PB13、PC3 ~ PC4、PD0 ~ PD1 有中断的功能，可选择不同的中断触发模式：Positive Edge、Negative Edge、High Level、Low Level、Double Edge(Positive/Negative)。

7.6. 软件配置

接口函数如下：

函数	作用
void pin_config(uint32_t cfg)	初始化
void pin_set_func(uint32_t pin, uint32_t func)	功能设置
uint32_t pin_get_func(uint32_t pin)	功能获取

void pin_set_value(uint32_t pin, uint32_t value)	输出值设置
uint32_t pin_get_value(uint32_t pin)	输入状态获取
void pin_set_driving(uint32_t pin, uint32_t driving)	驱动能力设置
void pin_set_pull(uint32_t pin, uint32_t pull)	上下拉电阻设置
void pin_irq_config(uint32_t pin, uint32_t irq_cfg)	中断设置
void pin_irq_clk_config(uint32_t pin, uint32_t irqclk_cfg)	中断设置

8.按键 ADC（KEYADC）

8.1. 概述

最多支持 6 个通道输入。

分辨率 12-bit，每次采样存在误差，其中高 6 bit 有效，低 6 bit 可忽略。

最大采样频率 500Hz。

支持中断方式、查询方式，在采用查询方式时，硬件上需要确保不出现最高的 VCCIO 电压，可使用弱下拉（例如 200K）的电阻。

4 种工作模式可选。

8.2. 工作模式

8.2.1. Normal Mode

采集得到一个数据，更新到数据寄存器中，同时置起数据中断标志，按此方式重复采样，直到停止 ADC。

8.2.2. Continue Mode

每隔 8*(N+1)个周期，采集得到一个数据，更新到数据寄存器中，同时置起数据中断标志，按此方式重复采样，直到停止 ADC。其中 N 对应 continue time set。

8.2.3. Single Mode

采集得到一个数据，更新到数据寄存器中，同时置起数据中断标志，此后 ADC 停止采样。

8.2.4. Knob Mode

采集得到一个数据，当这个数值大于初始配置的阈值时，更新到数据寄存器中，同时置起数据中断标志，按此方式重复采样，直到停止 ADC。

8.3. 中断

有 3 种中断源可配置。

8.3.1. KEYUP

按键抬起时产生中断。

8.3.2. KEYDOWN

按键按下时产生中断。

8.3.3. KEYDATA

按键持续按下，每采集得到一个数据时产生中断。

8.4. 软件配置

接口函数如下：

函数	作用
uint32_t kadc_data_get(uint32_t ch)	获取当前通道的 ad 值

ADC 采样值低 6 位可忽略，但软件上不要使用右移 6 位的方式。

9.LED 显示控制器（LEDC）

9.1. 概述

支持任意真值组合的 9 脚及以下 LED 显示屏，最多 72 个 LED 灯。

支持刷新周期可配置。

支持输入检测与 LED 显示同时使用，有中断方式可选。

支持硬件自动 gating 可配置。

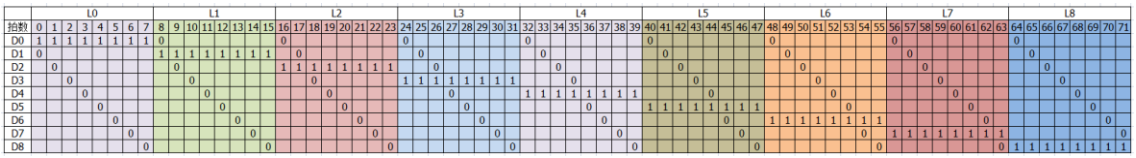
每操作一次开关 LED，维持 LEDC_CLK 数可配置。

支持单线 LED 显示屏。

9.2. 软件配置

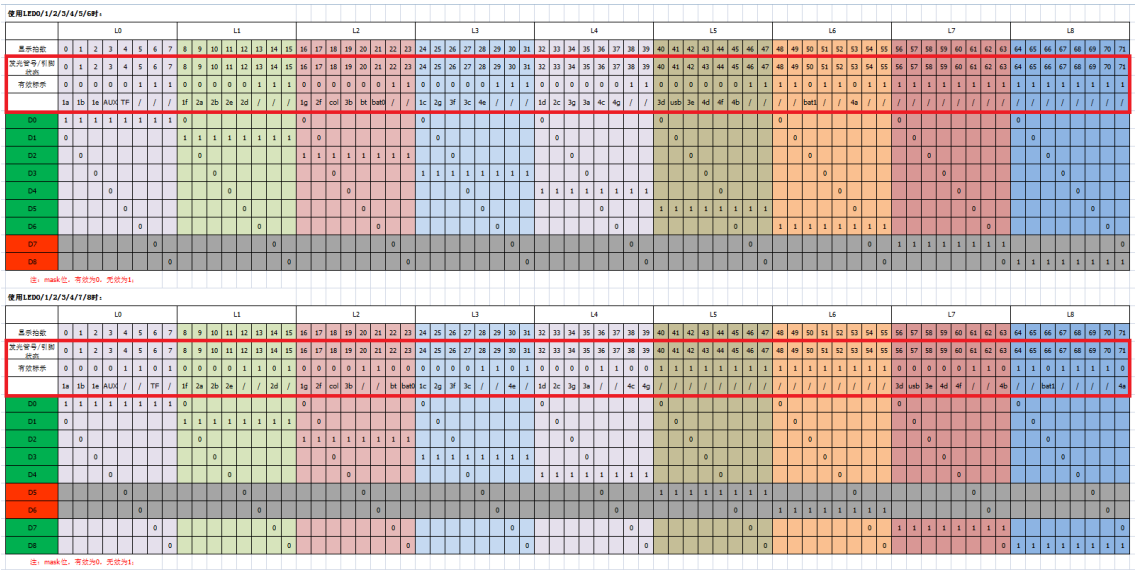
9.2.1. 显示真值表

扫描拍数如下图所示：



同一个 LED 显示屏，在使用不同 IO 实现时，软件中真值表 led_truth_tbl[]不同。

下图是在分别使用 LED0/1/2/3/4/5/6 和 LED0/1/2/3/4/7/8 时的情况：



9.2.2. 输入检测

变量	作用
ledc_param.det_en	选择 IO 并使能
ledc_param.int_en	选择上升沿/下降沿

10. 定时器（TIMER）

10.1. 概述

有 TIMER0/1/2/3 共 4 组模块。

位宽 32 位。

时钟源可以选择 LOSC 或 24M 晶振。

11. 脉宽调制（PWM）

11.1. 概述

支持 6 组 pwm 通道。

支持 single pulse mode 和 continue cycle mode。

11.2. 软件配置

时钟分频系数，有 120/180/240/360/480/600/720/840/12000/24000/36000/48000/72000。

entire cycles：如果 pwm_config 函数中参数是 N，那么 entire cycles = N+1。

active cycles：如果 pwm_config 函数中参数是 N，那么 active cycles = N+1。

12. 串行通信（UART）

12.1. 概述

有 UART0/1/2 共 3 组模块。

UART0 用于调试打印、测试，持波特率最高到 3Mbps。

UART1/2 用于其他数据传输，支持波特率最高到 3Mbps，支持 DMA 模式，支持 4 线模式（TX/RX/RTS/CTS）。

12.2. 软件配置

接口函数如下：

函数	作用
void uart_tx(uint32_t uart_id, uint32_t data)	数据发送
void uart_register_rx_callback(uint8_t uart_id, void (*rx_callback)(uint32_t rx_data))	数据接收函数注册

13. 两线串行接口（TWI）

13.1. 概述

TWI 接口完全兼容 I2C 总线。

有 TWI1/2 共 2 组模块。

支持主机模式。

支持速率 93 Kbps ~ 400Kbps。

13.2. 软件配置

接口函数如下：

函数	作用
int twi_write(uint32_t id, uint32_t addr, const uint8_t *data, uint32_t bytes)	写
int twi_read(uint32_t id, uint32_t addr, uint8_t *data, uint32_t bytes)	读

14. 串行外设接口（SPI）

14.1. 概述

有 SPI0/1 共 2 组模块。

SPI0 用于操作内置 spi nor flash 或外置 spi nor flash，支持 DMA 方式，仅支持 4 线模式（CS/CLK/MOSI/MISO）。

SPI1 用于操作外置 spi nor flash 或其他 spi 接口的设备，支持 4 线模式（CS/CLK/MOSI/MISO）或 3 线模式（CS/CLK/MOSI），但 3 线模式的最高工作频率比 4 线模式低些。

支持使用 GPIO 软件模拟 SPI 协议。

14.2. 软件配置

接口函数如下：

函数	作用
void spi_flash_erase(uint32_t flash_addr, uint32_t wait_busy)	擦外置 spi nor flash
void spi_flash_write(uint8_t *ram_addr, uint32_t flash_addr, uint32_t bytes, uint32_t wait_busy)	写外置 spi nor flash
void spi_flash_read(uint8_t *ram_addr, uint32_t flash_addr, uint32_t bytes)	读外置 spi nor flash
void spi_write(uint32_t data)	SPI1 写
uint8_t spi_read(void)	SPI1 读

15. 红外接收（IRRX）

15.1. 概述

支持 NEC 协议。

支持引导码、 电位’0’/’1’等时间可配置。

15.2. 软件配置

16. 内置音频总线（I2S）

16.1. 概述

支持主机模式。

支持 I2S/PCM/Left-Justified/Right-Justified/TDM(Time Division Multiplexing)模式。

支持采样率范围：8K~384K。

支持 4/6 线模式：4 线:TX 和 RX 用相同的 BCLK 和 LRCK，6 线模式，TX 和 RX 用不同的 BCLK 和 LRCK。

支持通道数：最大支持 16 通道，注：通道数*采样率*slot 宽度<=24.576M/22.5792M

支持采样精度：16~32bit

支持 slot 宽度：16/32

16.2. 软件配置

I2s.c 根据 user_config 做初始化配置。之后由 audio 库对 I2S 模块进行管理，APP 不要对其进行任何操作。

17. 数字音频接口（SPDIF）

17.1. 概述

支持从模式

采样率支持 22.05K~192K

17.2. 软件配置

spdif.c 根据 user_config 做初始化配置。之后由 audio 库对 spdif 模块进行管理，APP 不要对其进行任何操作。

18. Audio Codec

18.1. 概述

支持 3 路 ADC， 2 路 DAC。

18.2. 软件配置

18.2.1. 基本配置接口

函数	作用
void codec_init(void)	初始化 codec 模块 如果使用 Mute GPIO，在此函数初始化 Mute GPIO
void snd_spk_mute(void)	Mute 功放，播放完成后此函数被调用
void snd_spk_unmute(void)	Unmute 功放，播放开始前此函数被调用
void set_mic_adc_volume(uint8_t volume)	配置 MIC 音量，volume 范围：0~100
void auss_micbias_enable(bool enable)	Enable/Disable MIC bias

18.2.2. 其他 Codec 配置接口

函数	作用
mixer_set_ctl(SND_CODEC, MICBIAS, val)	mic bias: 0 ~ 3, 0-2.0v; 1-2.34v; 2-2.67v; 3-2.9v
mixer_set_ctl(SND_CODEC, LINEOUTL_EN, val) mixer_set_ctl(SND_CODEC, LINEOUTR_EN, val)	lineinL /lineinR 使能： val: 0:Disable；1：Enable
mixer_set_ctl(SND_CODEC, LINEOUTL_GAIN, val) mixer_set_ctl(SND_CODEC, LINEOUTR_GAIN, val)	lineinL /lineinR 模拟增益： val：0 ~ 7
mixer_set_ctl(SND_CODEC, ADC0_HPF_EN, 1);	使能 ADC0 高通滤波器
mixer_set_ctl(SND_CODEC, ADC1_HPF_EN, 1);	使能 ADC1 高通滤波器
mixer_set_ctl(SND_CODEC, ADC2_HPF_EN, 1);	使能 ADC2 高通滤波器
mixer_set_ctl(SND_CODEC, ADC0_DITHER_EN, 1);	使能 ADC0 dither
mixer_set_ctl(SND_CODEC, ADC1_DITHER_EN, 1);	使能 ADC1 dither
mixer_set_ctl(SND_CODEC, ADC2_DITHER_EN, 1);	使能 ADC2 dither

mixer_set_ctl(SND_CODEC, OUTPUT_POWER_BOOST, 1);	使能输出功率增强

mixer_set_ctl(SND_CODEC, AMIC0_SINGLE_END, 1); //单端
mixer_set_ctl(SND_CODEC, AMIC0_SINGLE_END, 0); //差分
	AMIC0 单端或者差分接入

mixer_set_ctl(SND_CODEC, AMIC1_SINGLE_END, 1); //单端
mixer_set_ctl(SND_CODEC, AMIC1_SINGLE_END, 0); //差分
	AMIC1 单端或者差分接入

mixer_set_ctl(SND_CODEC, AMIC2_SINGLE_END, 1); //单端
mixer_set_ctl(SND_CODEC, AMIC2_SINGLE_END, 0); //差分
	AMIC2 单端或者差分接入

mixer_set_ctl(SND_CODEC, LINEIN0_SINGLE_END, 1); //单端
mixer_set_ctl(SND_CODEC, LINEIN0_SINGLE_END, 0); //差分
	Linein0 单端或者差分接入

mixer_set_ctl(SND_CODEC, LINEIN1_SINGLE_END, 1); //单端
mixer_set_ctl(SND_CODEC, LINEIN1_SINGLE_END, 0); //差分
	Linein1 单端或者差分接入

mixer_set_ctl(SND_CODEC, AMIC0_GAIN, 3);//设置增益为 3	AMIC0 模拟增益（gain:0 ~ 14） 0:-4.5dB; 1:-3dB; 2:-1.5dB; 3:0dB; 4:1.5dB; 5:3dB; 6:4.5dB; 7:6dB; 8:24dB; 9:27dB; 10:30dB; 11:33dB; 12:36dB; 13:39dB; 14:42dB;
mixer_set_ctl(SND_CODEC, AMIC1_GAIN, 3);//设置增益为 3	AMIC1 模拟增益（gain:0 ~ 14） 0:-4.5dB; 1:-3dB; 2:-1.5dB; 3:0dB; 4:1.5dB; 5:3dB; 6:4.5dB; 7:6dB; 8:24dB; 9:27dB; 10:30dB; 11:33dB; 12:36dB; 13:39dB; 14:42dB;
mixer_set_ctl(SND_CODEC, AMIC2_GAIN, 3);//设置增益为 3	AMIC2 模拟增益（gain:0 ~ 14） 0:-4.5dB; 1:-3dB; 2:-1.5dB; 3:0dB; 4:1.5dB; 5:3dB; 6:4.5dB; 7:6dB; 8:24dB; 9:27dB; 10:30dB; 11:33dB; 12:36dB; 13:39dB; 14:42dB;
mixer_set_ctl(SND_CODEC, LINEIN0_GAIN, 3); //设置增益为 3	Linein0 模拟增益（gain:0 ~ 7） 0:-4.5dB; 1:-3dB; 2:-1.5dB; 3:0dB; 4:1.5dB; 5:3dB; 6:4.5dB; 7:6dB;
mixer_set_ctl(SND_CODEC, LINEIN1_GAIN,	Linein1 模拟增益（gain:0 ~ 7）

3); //设置增益为 3	0:-4.5dB; 1:-3dB; 2:-1.5dB; 3:0dB; 4:1.5dB; 5:3dB; 6:4.5dB; 7:6dB;

更多 audio codec 相关函数接口，请参考 audio_service.h, audiopath_conf.h 文件中说明。

19. PMU

19.1. user config

19.1.1. pmu startup mode config

19.1.1.1. PMU_POWER_ON_MODE

开机模式配置：

0: 按键模式，长按开机，如果硬件方案是按键开机/板上硬开关/板上软开关应该选择为按键模式。

1: 拨动开关模式，拨到高电平开机。

详细参考慧联蓝牙 IC 上电开机方式说明文档。

只能通过烧录器修改 PMU OTP，一旦烧码了就无法修改了。

19.1.2. pmu config

19.1.2.1. PMU_CORE_USE_DCDC

设置 core 由谁供电。

如果是 HSxxx 系列的就用 DCDC 供电。

如果是 WSxxx 和 CAxxx 系列的就用 LDO 供电。

如果配置错误，会导致启动失败。

19.1.2.2. PMU_VBUS_POWERON_EN

使能 vbus 插入即开机。

19.1.2.3. PMU_SHORT_PRESS_POWERON_EN

使能短按开机。

19.1.2.4. PMU_ULTRA_SHORT_PRESS_POWERON_EN

使能超短按开机功能。

19.1.2.5. PMU_POWEROFF_WHEN_BAT_FIRST_IN

使能电池插入时关机，电池插入时 PMU 默认会开机，但是很多产品在产品线上一焊上电池就会开机，如果不使能该功能就会导致电池被耗完。

像耳机这类产品建议使能该功能。

19.1.2.6. PMU_ULTRA_LONG_PRESS_TIME

超长按时间设置，单位为 ms，如果设置为 8000ms，那么按下 8s 后就会超长按复位。

时间设置的范围只有 6s/8s/10s/12s。

19.1.2.7. PMU_LONG_PRESS_TIME

长按时间设置，单位为 ms，如果设置为 3000ms，那么按下 3s 就会触发长按中断，如果是在关机时按下 3s 就会开机。

时间设置的范围只有 1s/2s/3s/4s。

19.1.2.8. PMU_CHARGE_CURRENT

充电电流设置，单位为 mA，设置范围为 25mA-450mA, 步进为 25mA。

注意这个电流是电池的尽流入和系统当时耗电的总和。

一般设置充电电流不要超过电池容量的 80%，不然可能会影响电池的使用寿命。

例如一个容量为 100mAh 的电池，充电电流最好不要超过 80mA。

19.1.2.9. PMU_VBUSIN_RESTART

vbus 插入就重启，因为有些产品需要在关机下才能进行充电，所以需要插入 usb 线马上重启，进入关机充电模式，常用于耳机中。

19.1.2.10. PMU_VCC_RTC_ALWAYS_ON

如果需要 rtc 时钟在关机时能够继续走，则需要打开 vcc-rtc 常在电功能。

如果你的产品没有 rtc 功能的需求，可以关闭，减小关机功耗。

19.1.2.11. PMU_VCC_IO_VOLT

设置 vcc-io 电压，默认为 3300mV，单位 mV，电压范围为 2700~3400mV，步进 100mV，该电压会影响外设 gpio 电压，一般不修改。

19.1.2.12. PMU_AVCC_VOLT

设置 avcc 电压，默认为 3300mV，单位 mV，电压范围为 2700~3400mV，步进 100mV，该电压会影响音频最大音量输出能力。

如果你在乎电池电压低于 3300mV 时，音频质量，可以把 avcc 设置为低于 3300mV，可以设置为 2700mV。

19.1.3. battery config

19.1.3.1. BAT_EXIST

如果使用了芯片内部 PMU 管理电池，则使能电池功能。

如果使用外部充电芯片或者升压芯片，但是想用电池管理的代码，则使能电池功能。

19.1.3.2. BAT_VOLT_DET

如果使用了芯片内部 PMU 管理电池，把电池的电压接到芯片的 vbat 管脚，则使能电池电压检测。

如果是使用了外部充电芯片，但是利用芯片的通用 ADC 检测电池电压，则无需使能该功能。

19.1.3.3. BAT_QUANTITY_STEP_BIG

PMU 驱动支持两种步进的电量显示，一个 5 档，一个为 10 档。由于电池电压 ADC 精度只有 100mV，所以无法精确支持到 10 档，建议使用 5 档的电量显示，已经可以满足大部分产品。

如果使用 5 档电量显示，则使能该功能。

19.1.3.4. BAT_HW_LB_SHUT_EN

使能电池硬件低电关机，一旦电池电压低于等于硬件低电关机的电压，硬件则会自动关机。

如果需要保存信息到 flash，则不希望触发硬件低电关机，应该在硬件低电关机之前进行软件低电关机，并保存信息到 flash。

19.1.3.5. BAT_HW_LB_SHUT_VOLT

设置电池硬件关机电压，单位为 mV，默认为 2700mV，设置范围只有 2700/2800/2900/3100。

注意：设置的电压只能在四个电压里面选择，而且硬件低电关机电压要低于软件低电关机电压。

19.1.3.6. BAT_HW_OK_VOLT

设置允许芯片开机的电压，如果低于该电压则无法开机。单位为 mV，默认值 2900mV，只能在 2900/3000/3100/3300 四个电压中选择。

注意：由于该配置属于 PMU OTP，所以只能通过 PMU 烧码修改，软件设置无效，设置后重启会被 PMU OTP 覆盖掉，等于软件设置是无效的。

19.1.3.7. BAT_SW_LB_WRN_VOLT

3300 //2900~4400mV, step 100mV, low power warning voltage;

设置软件电池低电提醒电压，单位为 mV，电压范围为 2900~4400mV，步进 100mV。

19.1.3.8. BAT_SW_LB_SHUT_VOLT

设置软件低电关机电压，单位为 mV，默认为 3100mV，电压只能在 2900/3000/3100/3300mV 四个电压中选择。

19.1.3.9. BAT_SW_LB_WARNING_PERIOD

设置软件低电提醒周期，单位为 ms，步进 1000ms，默认为 2min，一旦低电就会提醒，过了 2min 再次提醒。

19.1.4. pmu key config

19.1.4.1. KEY_PMU_EN

pmu 电源键作为其他功能按键使用，例如作为暂停和播放键使用。

19.1.4.2. KEY_PMU_MULTFUC_EN

pmu 电源键多功能按键，支持单击，双击和长按。

19.2. user interfaces

19.2.1. bool pmu_is_enter_charge_mode(void)

判断是否要进入充电模式，开机过程中使用。

19.2.2. void pmu_set_core_voltage(uint32_t val)

设置 cpu core 电压，用于调频调压。

19.2.3. bool pmu_is_use_hsw(void)

判断 pmu 使用 onoff（按键方案）还是 hsw（硬开关方案）。

19.2.4. void pmu_sel_onoff_hsw(enum pmu_onoff_hsw_e sel)

设置 pmu 为 onoff（按键方案）或 hsw（硬开关方案）。

19.2.5. void pmu_hsw_reset_enable(bool enable)

使能和除能硬开关拉低超过 4s 复位系统的功能。

19.2.6. bool pmu_bat_is_charging(void)

判断电池是否在充电。

19.2.7. bool pmu_bat_is_exist(void)

判断电池是否存在。

19.2.8. bool pmu_bat_is_full(void)

判断电池是否为满电。

19.2.9. bool pmu_bat_is_low(void)

判断电池是否为低电。

19.2.10. void pmu_charge_enable(bool enable)

使能和除能电池充电功能。

19.2.11. uint8_t pmu_get_bat_quantity_percent(void)

获取电池电量百分比。返回值为 0~100。

19.3. PMU event

PMU_EVENT_BAT_LOW	电池低电
PMU_EVENT_BAT_FULL	电池满电
PMU_EVENT_BAT_TO	电池充电超时
PMU_EVENT_POWERKEY_US	电源键超短按
PMU_EVENT_POWERKEY_S	电源键短按
PMU_EVENT_POWERKEY_L	电源键长按
PMU_EVENT_INT	PMU 中断
PMU_EVENT_CHARGE_IN	vbus 插入
PMU_EVENT_CHARGE_OUT	vbus 拔出
PMU_EVENT_HSW_OFF,	硬开关拉低
PMU_EVENT_HSW_ON	硬开关拉高
PMU_EVENT_VBUSUV	vbus 欠压

PMU_EVENT_BAT_UNLOW	电池非低电
PMU_EVENT_BAT_LOW_WRN	电池低电提醒

20. Power management

20.1. Sleep

20.1.1. SLEEP_EN

使能 sleep 功能，如果使能后蓝牙保存连接时就会自动进入 sniff mode，然后芯片会动态的进入和退出 sleep。如果没有蓝牙的场景，例如本地音乐，暂停播放一段时间之后会自动进入 sleep 状态，系统的功耗非常低，最低达到 400uA。

20.1.2. SLEEP_DELAY

满足 sleep 所有条件之后等 10s 才进入 sleep，例如本地音乐播放时，暂停 10s 才会开始进入 sleep。

单位为 s，默认值是 10s。

20.1.3. TIME_BEFORE_WFI

OS 使能了 tickless 功能，就是系统会在 idle 任务里面去统计未来的所有 timer 包括任务的 resume 等的最近时间是多少，如果所有 timer 的最快到来时间大于 TIME_BEFORE_WFI 则让 cpu 进入 wfi，等到时间到了之后才唤醒 cpu。

只有系统尽量少的使用 timer，或者 timer 的周期更长，才能有更多的机会进入 wfi 状态，才能把 wfi 的阈值设置得更低，cpu 才能更加节省功耗。

20.1.4. BT_CON_AUTO_SLEEP_EN

bt 连接时自动睡眠，当 bt 进入到 sniff mode 时会进入睡眠，芯片处于 sleep 状态，因为 bt 需要保持连接，所以需要隔一段时间起来通信，所以这个时候 sleep 是动态过程。

20.1.5. BT_CON_AUTO_SLEEP_WAIT

bt 连接时等待多久之后手机还没主动发出进入 sniff mode，则我们设备端主动发送请求进入 sniff mode，防止长时间处于功耗比较大的状态。

单位是 ms，一般 IPhone 比 Android 慢请求，IPhone 不会超过 30s，所以这里只需要设置成 40s 就足够了。

20.1.6. BT_DIS_AUTO_SLEEP_EN

bt 没有连接上时主动进入睡眠，但是为了保证随时能够连上设备，需要隔一段时间起来通信，所以这个时候 sleep 是动态过程。

20.1.7. BT_DIS_AUTO_SLEEP_MAX

bt 没有连接上时主动进入睡眠的最大时间，单位为 slot，625us，默认设置为 800slot。

20.2. Poweroff

20.2.1. POWER_OFF_DELAY

poweroff 之前延时多少 ms 再关机，一般配置为 0。

20.2.2. POWER_OFF_FIRST_EN

当使用 vbus 供电，第一次启动关机。

20.2.3. AUTO_POWER_OFF

蓝牙断开时，超过多长时间自动关机。

20.2.4. AUTO_POWER_OFF_PERIOD

蓝牙断开时，超过多长时间自动关机。单位为 s，默认为 300s，5min，即断开蓝牙 5 分钟后关机。一般用于蓝牙耳机上。

20.3. Dvfs

dvfs:是 dynamically voltage frequency scaling 的缩写，即动态调频调压。

我们是基于各个场景进行动态调频调压，各个场景的频率如下：

设置项	场景	频率
DVFS_KARAOKE_FREQ	karaoke 模式	192e6
DVFS_EFFECTS_FREQ	开启音频音效	192e6

DVFS_TONEMIXER_FREQ	提示音和音乐混音	64e6
DVFS_A2DPAUDIO_FREQ	蓝牙音乐	64e6
DVFS_SCOAUDIO_FREQ	蓝牙通话	64e6
DVFS_SCOAENC_FREQ	蓝牙通话并使能降噪	64e6
DVFS_USBAUDIO_FREQ	usb 音乐	192e6
DVFS_LINEINAUDIO_FREQ	linein 音乐	64e6
DVFS_LOCAL_ENCODE_WAV_FREQ	本地 wav 格式录音	64e6
DVFS_LOCAL_ENCODE_MP3_FREQ	本地 mp3 格式录音	192e6
DVFS_LOCAL_DECODE_WAV_FREQ	本地 wav 格式音乐播放	64e6
DVFS_LOCAL_DECODE_MP3_FREQ	本地 mp3 格式音乐播放	192e6
DVFS_DEFAULT_FREQ	默认频率	64e6

21. BLE

21.1. 概述

支持基于 GATT 的 BLE 应用，目前版本只支持 GATT server，后面的说明前提也是”Soc 芯片/平台”作为 GATT server

21.2. 软件配置

sl6800_user_config.h

```
#define BT_BLE_EN                       1       //BLE 功能
#define BLE_ADDR_USE_RANDOM     1       //BLE 设备地址使用随机值功能
#define BLE_ADDR_DEFAULT    {0xFA,0xFB,0xFC,0xFD,0xFE,0xFF}     //BLE 设备默认地址
#define BLE_NAME_DEFAULT    "SL-BLE"                            //BLE 设备默认名字
```

21.3. 数据结构

```
typedef struct ble_profile {
    uint8_t own_addr[6];
    uint8_t own_addr_type;
    uint8_t *adv_data;
    uint8_t adv_data_len;
    uint8_t *scan_rsp;
    uint8_t scan_rsp_len;
    uint8_t *att_db;
    uint16_t start_handle;
    uint16_t end_handle;
    profile_read_callback_t read_callback;
    profile_write_callback_t write_callback;
} ble_profile_t;
```

own_addr：ble 地址，如果 own_addr_type 为 PUBLIC_ADDRESS，则 BLE 地址是 BLE_ADDR_DEFAULT，如果 own_addr_type 为 RANDOM_ADDRESS，则是烧写后第一次启动随机生成的地址

adv_data：广播数据

adv_data_len：广播数据长度

scan_rsp：扫描响应数据

scan_rsp_len：扫描响应数据长度

att_db：ATT 数据库

start_handle：ATT 起始 handle

end_handle：ATT 结束 handle，和 start_handle 一起限定了 handle 的范围

read_callback：读取 ATT 的回调，一般用于拷贝 ATT 值到指定 buffer，最终通过协议栈发送出去

write_callback：写 ATT 的回调，用于接收 client 端写某个 ATT 的值

21.4. demo

demo 应用：ble_app.c

对于大多数应用，只需要关心 3 个部分：

1. 广播数据和扫描响应数据，这里面又主要关心设备名称

```
/*
Reference BT Spec 4.2 Vol 3, Part C, Chapter 11
adv_data and scan_rsp format:
|Byte0 |Byte1~N|Byte(N+1)~Byte(length)|
|length|AD type |                AD data                |
*/
uint8_t simple_adv_data[31] =
{
    0x02, AD_TYPE_FLAGS, FLAG_LE_GENERAL_DISCOVER_MODE |
FLAG_BR_EDR_NOT_SUPPORTED,
    0x00, AD_TYPE_COMPLETE_LOCAL_NAME,
};

uint8_t simple_scan_rsp[31] =
{
    0x00, AD_TYPE_COMPLETE_LOCAL_NAME,
};
```

真正的设置名称的地方在 ble_app_init()：

```
uint8_t name_len;
ble_name_set(&simple_adv_data[5], &name_len);
simple_adv_data[3] = 1 + name_len;

ble_name_set(&simple_scan_rsp[2], &name_len);
simple_scan_rsp[0] = 1 + name_len;
```

2. GATT server 数据库，请参考 simple_att_db，每行代表一个 attribute，组织方式如下

Byte0~1	Byte2~3	Byte4~5	Byte6 ~ Byte7	Byte8~ Byte(size -1)
size	flag	handle	UUID	data

或者：

Byte0~1	Byte2~3	Byte4~5	Byte6 ~ Byte21	Byte22~ Byte(size -1)
size	flag	handle	UUID	data

size: Attribute 大小，包含 size 在内的所有元素，单位：Byte
flag: Attribute Property Flags，请参考 ATT_PROPERTY_开头的宏定义，比如

```
#define ATT_PROPERTY_BROADCAST                0x01
#define ATT_PROPERTY_READ                     0x02
#define ATT_PROPERTY_WRITE_WITHOUT_RESPONSE 0x04
```

handle: Attribute handle
UUID: Attribute uuid，可以是官方定义的，也可以是自定义的
data: Attribute data，具体含义需要遵循 GATT 规范

3. 数据收发接口

ble_app_data_receive
ble_app_data_send

send 可以采用 notify 方式或者 indicate 方式，分别对应的接口

ble_profile_data_send_notify

ble_profile_data_send_indicate

22. BT

22.1. 软件配置

Sl6800_user_config.h

```
#define BT_BACKGROUND_EN                0       //蓝牙后台（全模式支持蓝牙）功能
#define BT_PHONE_EN                     1       //蓝牙通话功能
#define BT_SPP_EN                       0       //SPP 功能

#define BT_HID_EN                       0       //HID 功能
#define BT_ADDR_USE_RANDOM              1       //蓝牙设备地址使用随机值功能
#define BT_FCC_TEST_                    0       //FCC 测试功能
#define BT_VOLUME_EN                    1       //音量与手机同步功能
#define BT_RECONNECT_SCAN_EN            1       //回连时可被搜索到功能
#define BT_SIRI_EN                      0       //苹果 siri 或安卓语音助手功能

#define BT_ADDR_DEFAULT     {0xF0,0xF1,0xF2,0xF3,0xF4,0xF5}     //蓝牙设备默认地址
#define BT_NAME_DEFAULT     "SL0000"                //蓝牙设备默认名字

#define BT_RFPOWER_FORCE_EN                     0       //RF 发送功率固定
#define BT_RFPOWER_FORCE_VAL                    16      //RF 发送功率固定的档位，范围：
0~18，功率值：(-34+2*BT_RFPOWER_FORCE_VAL) dbm
1
#define BT_MAXPOWER_SET_EN                      0       //RF 发送最大功率限制（在
BT_RFPOWER_FORCE_EN 是 0 时有效）
#define BT_MAXPOWER_SET_VAL                     16      //RF 发送最大功率的档位，范围：
0~18，功率值：(-34+2*BT_MAXPOWER_SET_VAL) dbm
```

22.2. API 函数

uint8_t bt_connect(uint8_t times, uint8_t scan_en)
功能：Soc 小机端主动连接。
参数：times：尝试连接的次数，每次大概进行 5 秒；
scan_en：主动连接的过程中仍可以被手机发现并进行连接。

uint8_t bt_disconnect_direct(void)
功能：Soc 小机端主动断开连接。

23. 工具

23.1. EQ 工具

慧联科技 EQ 软件专为慧联科技开发的音频类产品调节 EQ 参数所用，支持 14 路 EQ 参数以及多种效果器的调节，支持导入导出以及颜色配置等。

23.1.1. 调节 EQ

1、 调节 EQ 有四种个方法：

2、 在 EQ 曲线插件上面选中所要调节的参数，按住鼠标左键，然后拖到 EQ 按钮即可调节，如图 21-1-1 中箭头 1 指示。

3、 在 EQ 曲线插件上面选中相应的按钮后，滚动鼠标滚轮，可以微调频率值，单位默认为 0.1HZ，如果按住 shift 键再滚动鼠标滚轮，则微调 Q 值，单位默认为 0.1，如果按住 ctrl 键再滚动鼠标滚轮，则变为微调增益，单位默认为 0.1。

4、 在 EQ 参数区直接输入相应的值 即可，参数会实时生效，如图 21-1-1 中箭头 2 所示

5、 将鼠标移动到相应的参数值区域，滚动鼠标滚轮，可以粗调相应的参数值，频率的粗调单位为默认 10HZ，Q 值和增益默认为 0.1

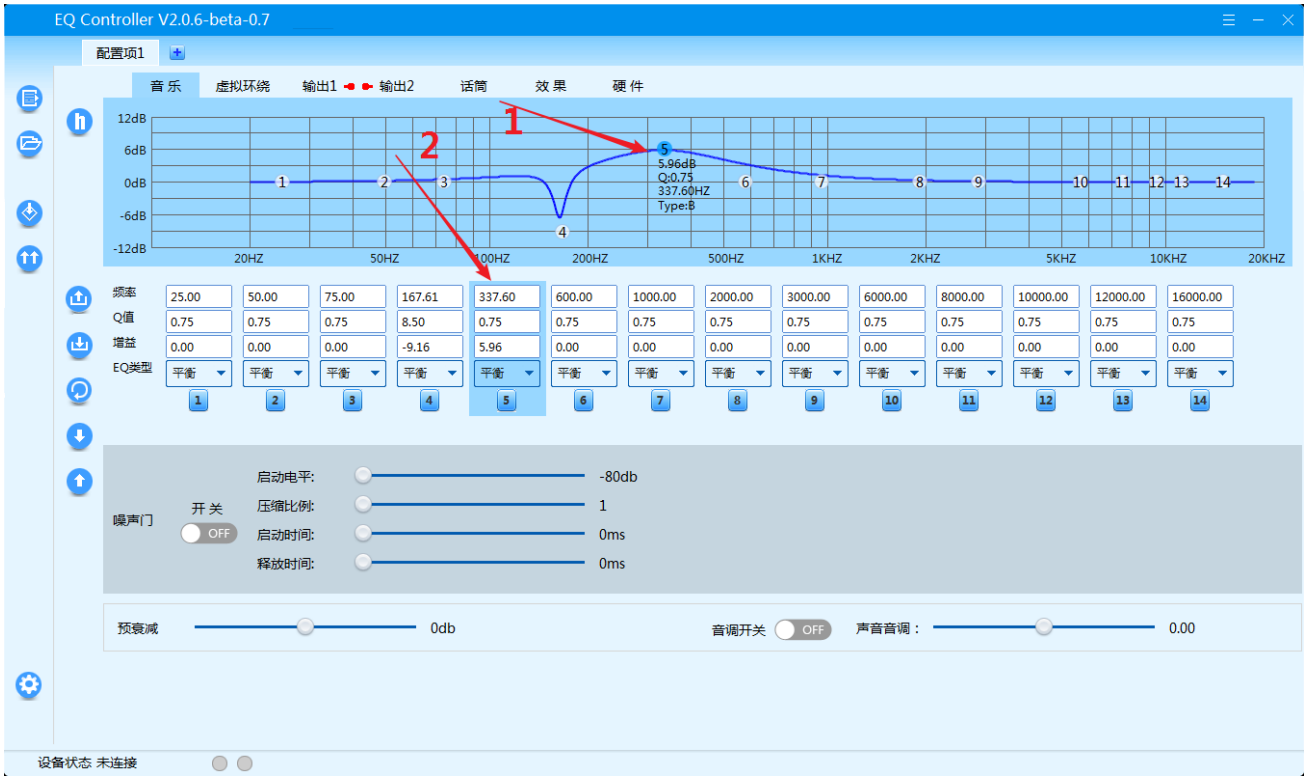


图 21-1-1

23.1.2. EQ 类型

EQ 类型目前共有六种，分别为低棚架，平衡，高棚架、高通带通、低通，可在 EQ 类型下拉菜单中选择，默认为“平衡”，如图 21-1-2 所示

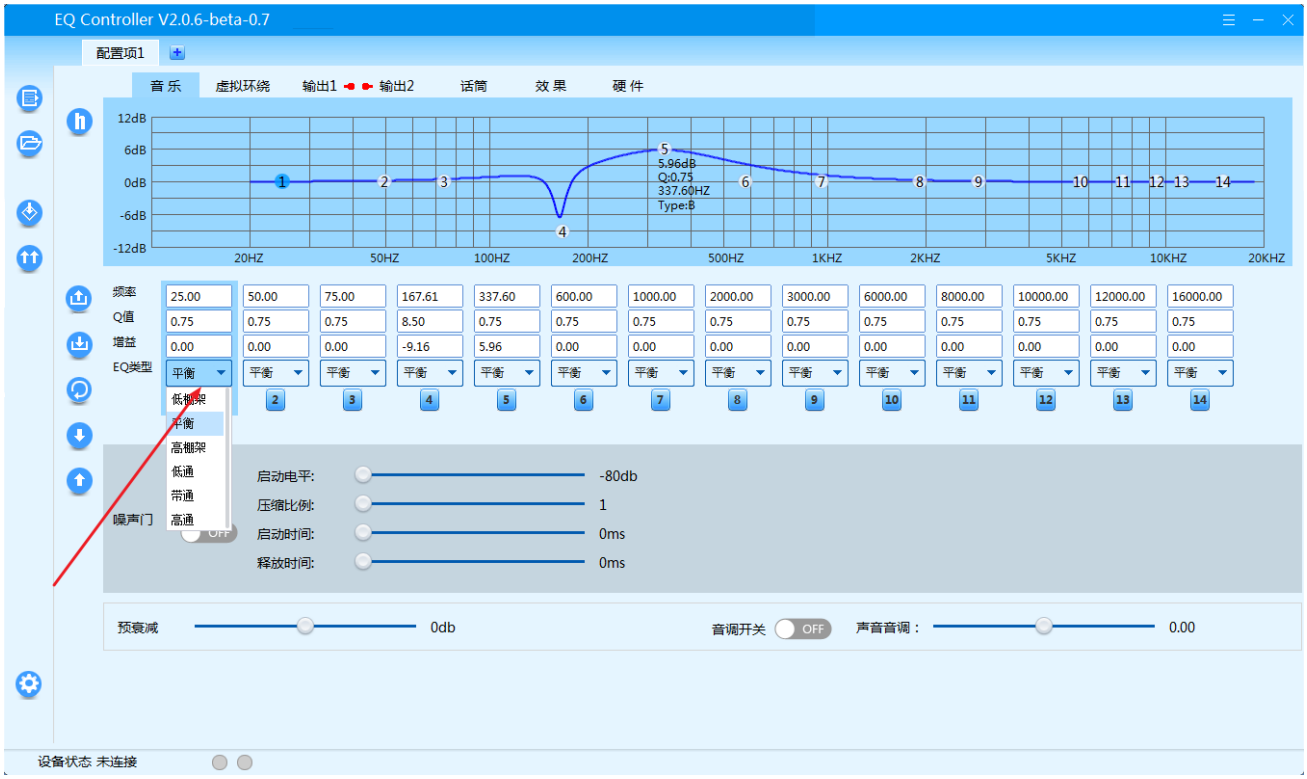


图 21-1-2

23.1.3. EQ 使能

每一路 EQ 均有各自的使能按钮，如图 21-1-3 所示，可以相应的禁止或者打开该路 EQ 调节功能。

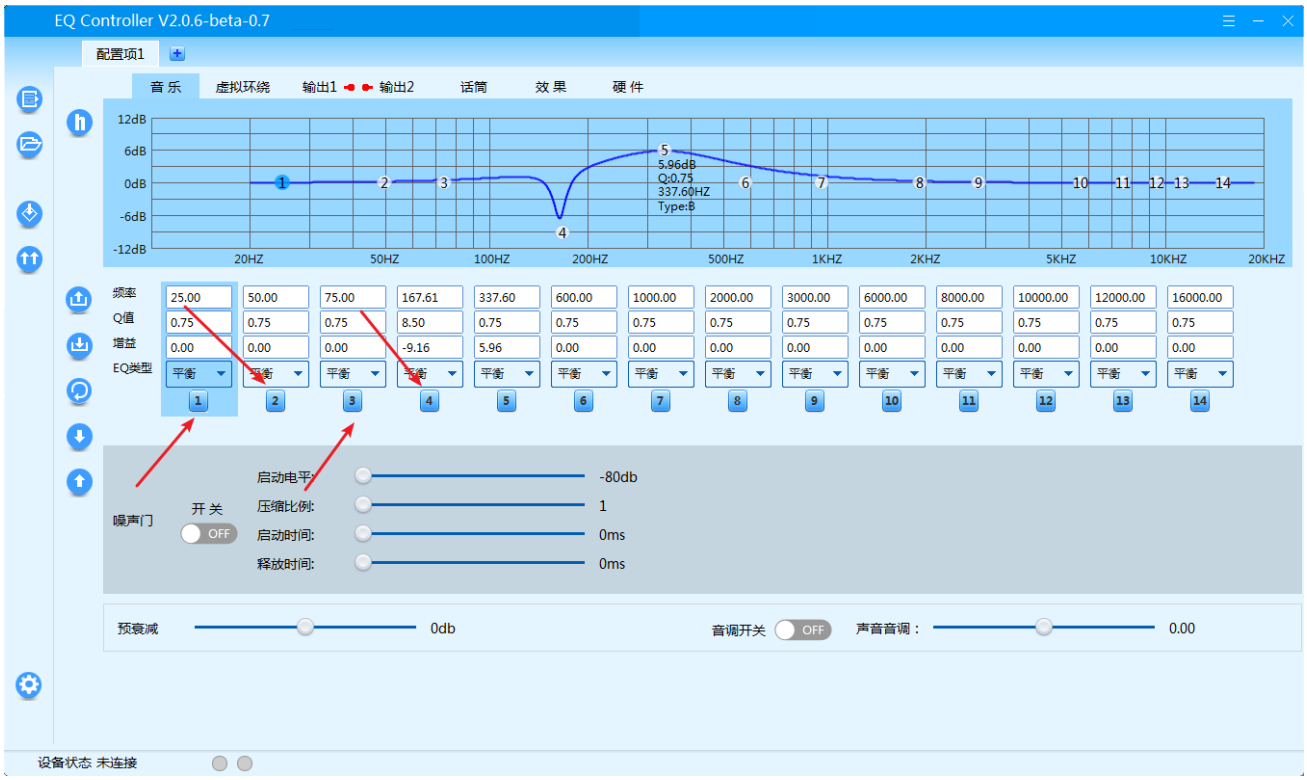


图 21-1-3

23.1.4. 调节其它效果参数

拖动相应设置项的滑动按钮或者选择相应的设置项即可，还可以输入一个值，如图 21-1-4 所示。

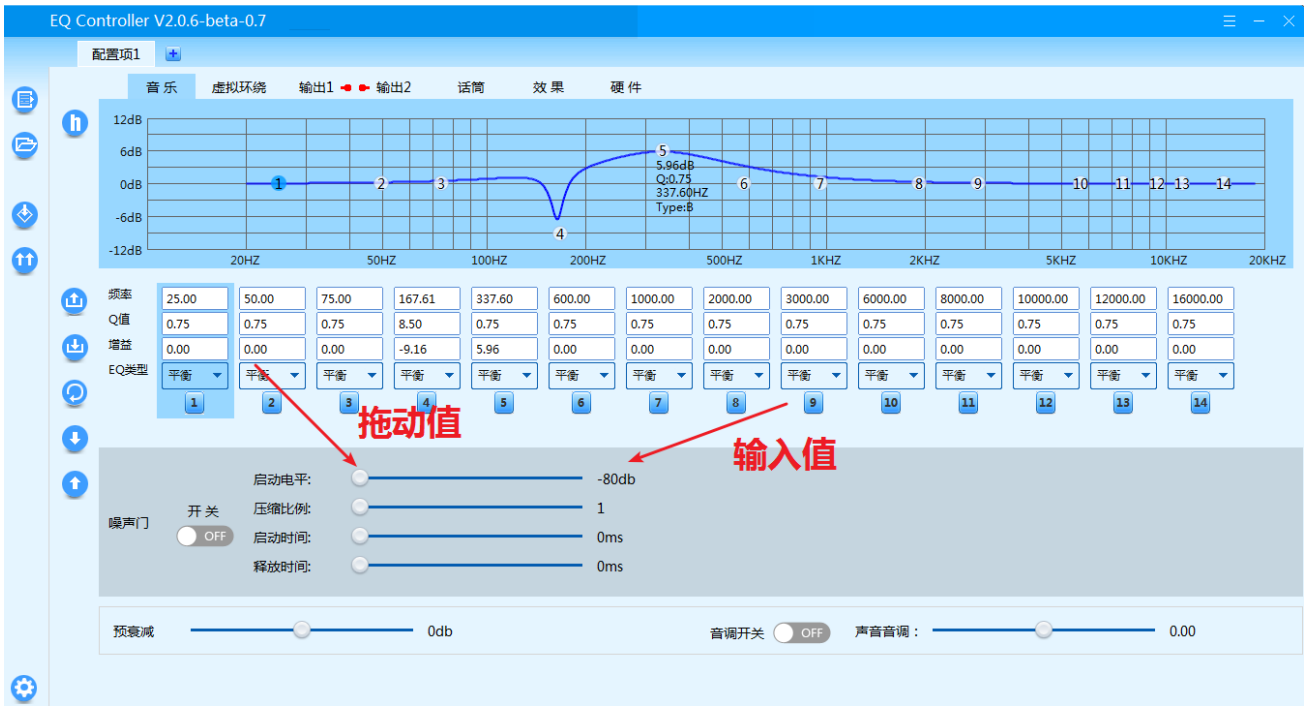


图 21-1-4

23.1.5. 设备连接

EQ 软件目前支持 USB 连接，设备不在线时如图 21-1-5-1 所示，设备状态显示“未连接”

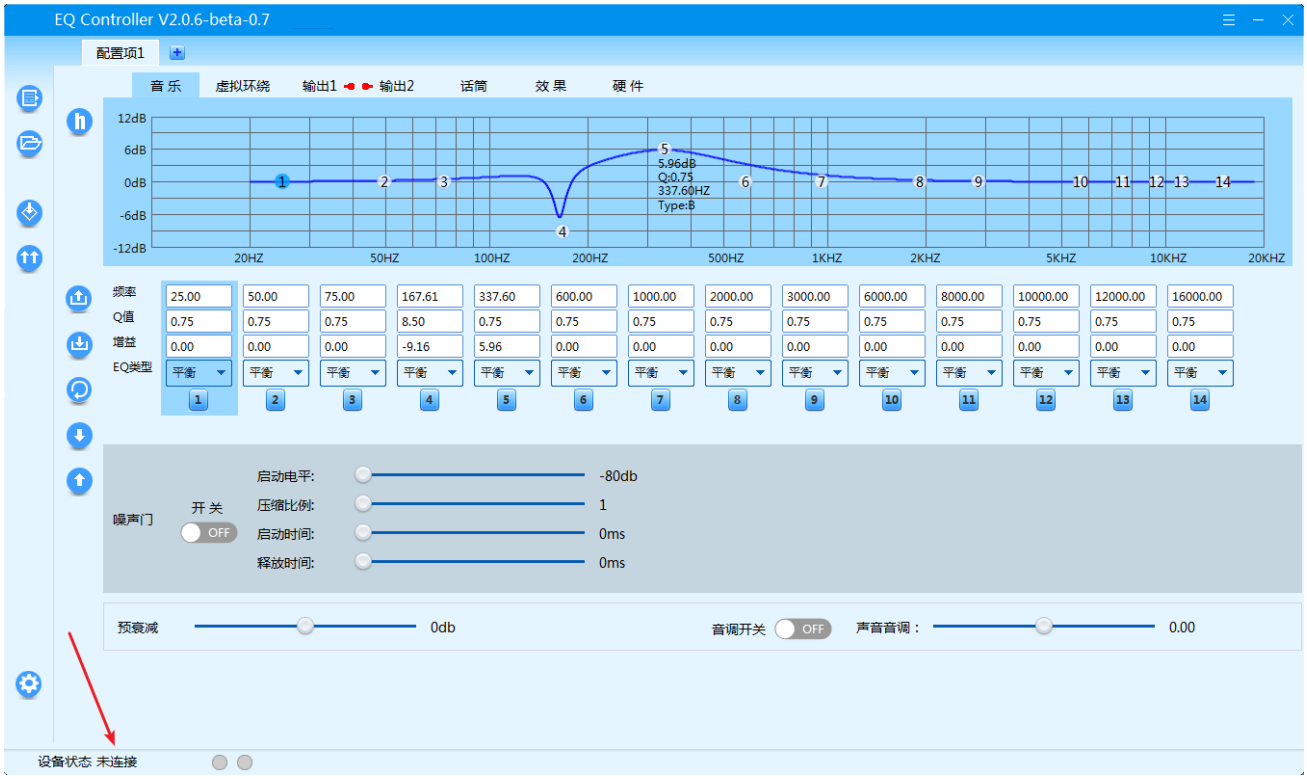


图 21-1-5-1

当在线时设备状态显示“已连接”，并且绿色的指示灯交替闪烁，如图 21-1-5-2 所示

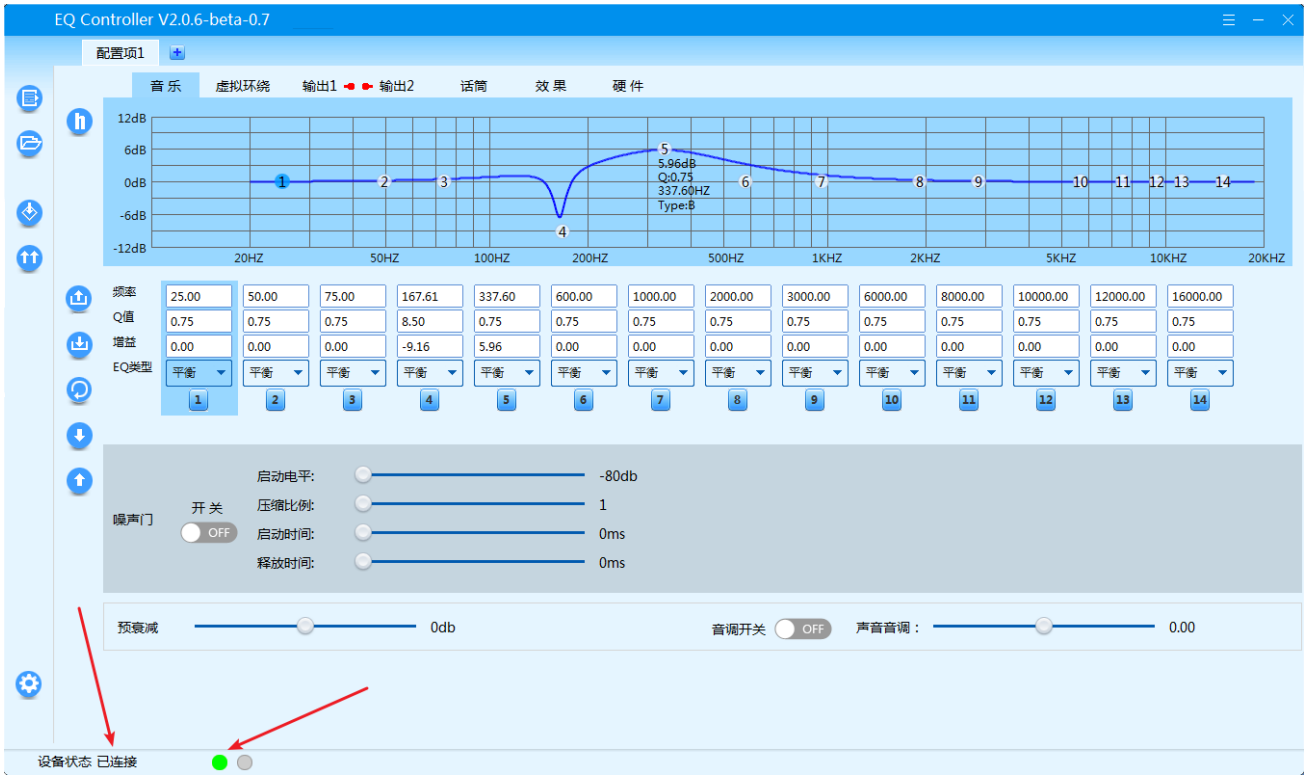


图 21-1-5-2

23.1.6. 错误隐患

当设置的 EQ 参数导致 EQ 曲线出现削峰时，EQ 插件会出现红色半透明区域，表明大致出现削峰的范围，需引起注意，如图 21-1-6 所示

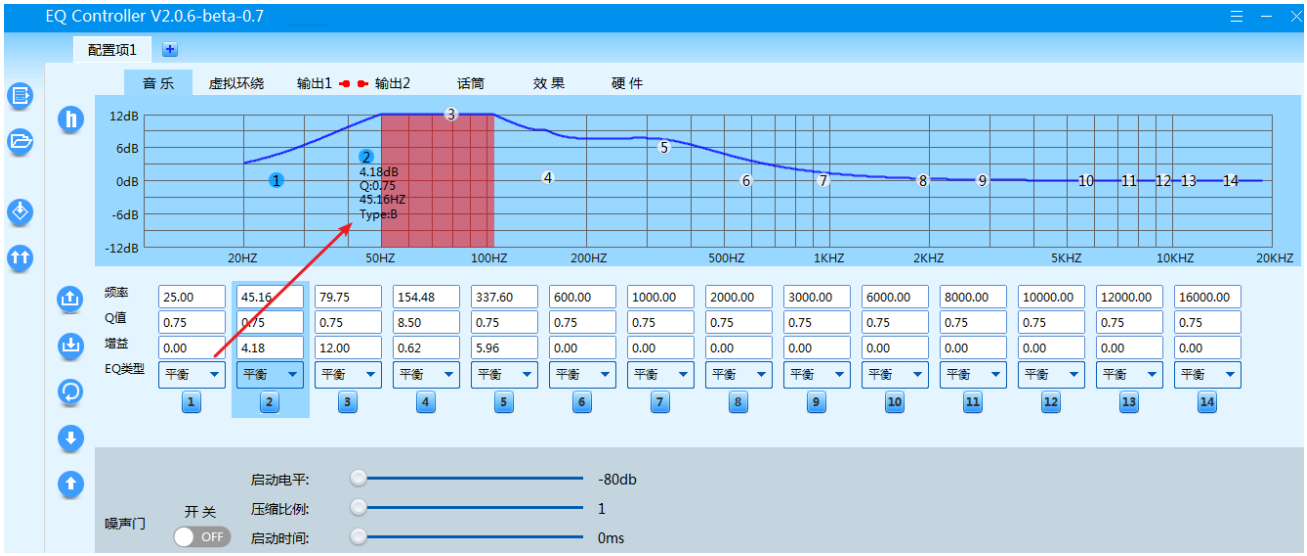


图 21-1-6

23.1.7. 导出

如图 21-1-7 所示，点击导出按钮后，输入名称即可导出当前配置，生成 eq2 后缀文件，以备将来使用

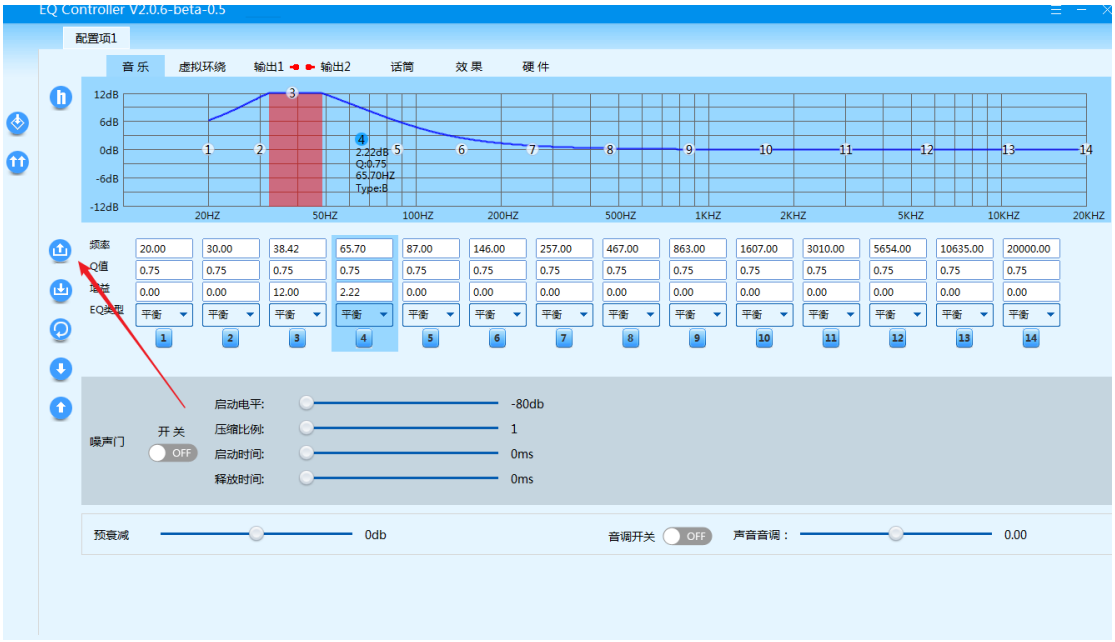


图 21-1-7

23.1.8. 导入

导入之前保存的数据，刷新当前配置界面，如图 21-1-8 所示

图 21-1-8

23.1.9. 重置

如图 21-1-9 所示，点击重置按钮后，EQ 软件会将当前配置的所有 EQ 参数重置为默认值

图 21-1-9

23.1.10. 同步 PC 设置

将 PC 的配置刷新到设备中，如图 21-1-10 所示

图 21-1-10

23.1.11. 提取设备 EQ

将设备中运行的 EQ 参数刷新到界面中

图 21-1-11

23.1.12. 多配置 EQ 的新建与删除

新建配置。如图 21-1-12-1 所示，点击“+”按钮，弹出命名窗口如图 21-1-12-2 所示，输入名字后，点击确定即可

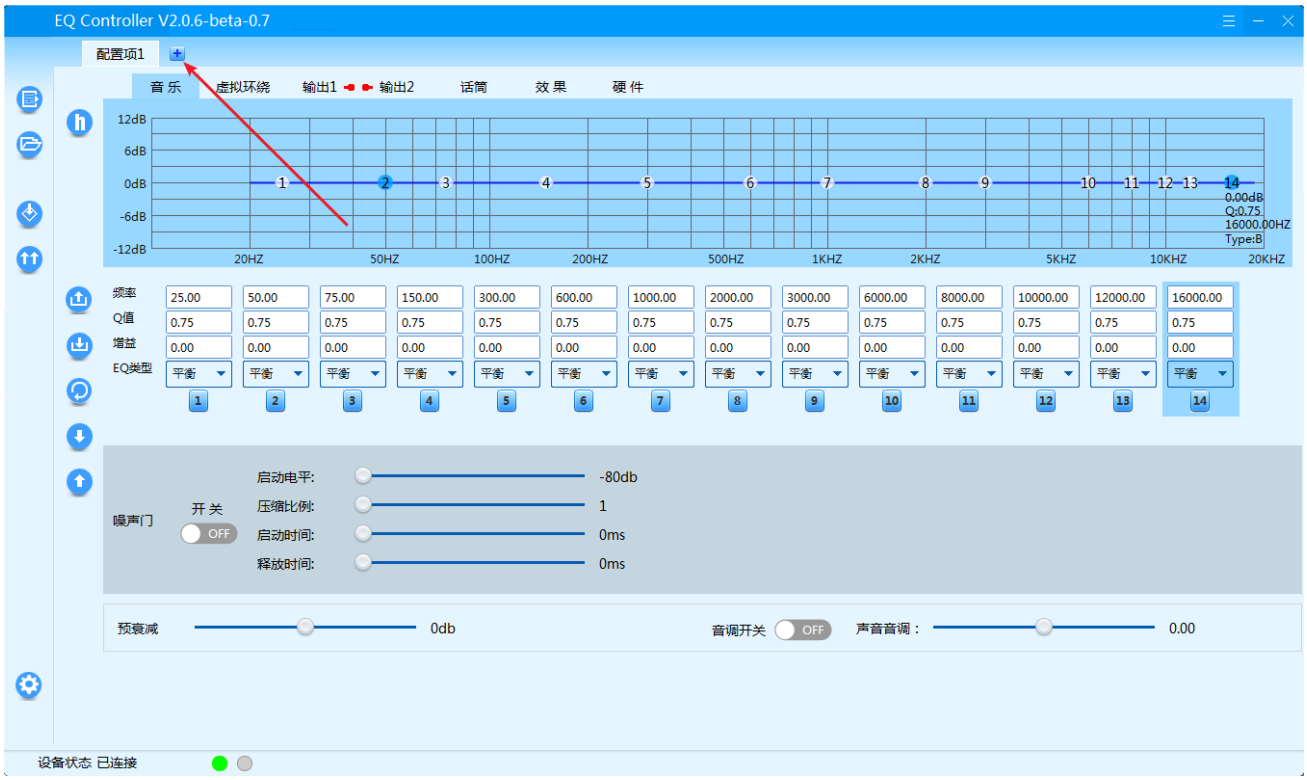


图 21-1-12-1

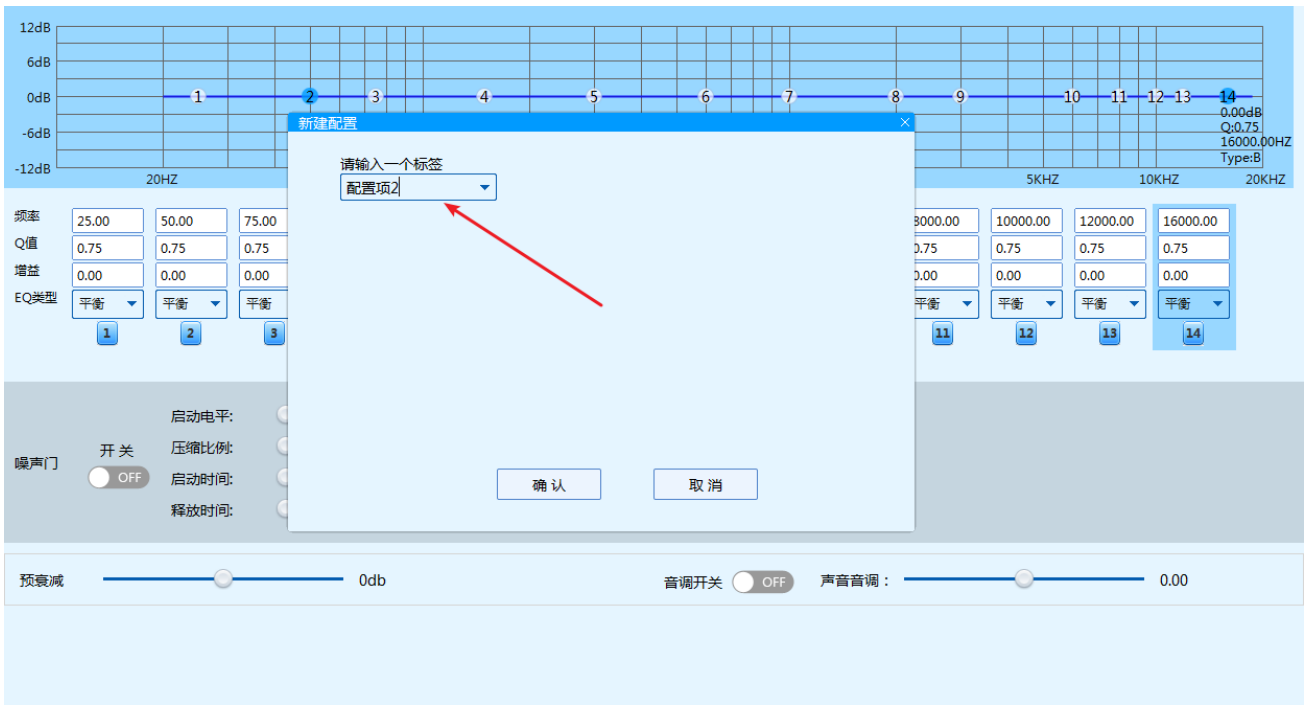


图 21-1-12-2

删除配置。右键点击页面名称，弹出菜单，点击“删除”按钮即可删除，如图 21-1-12-3

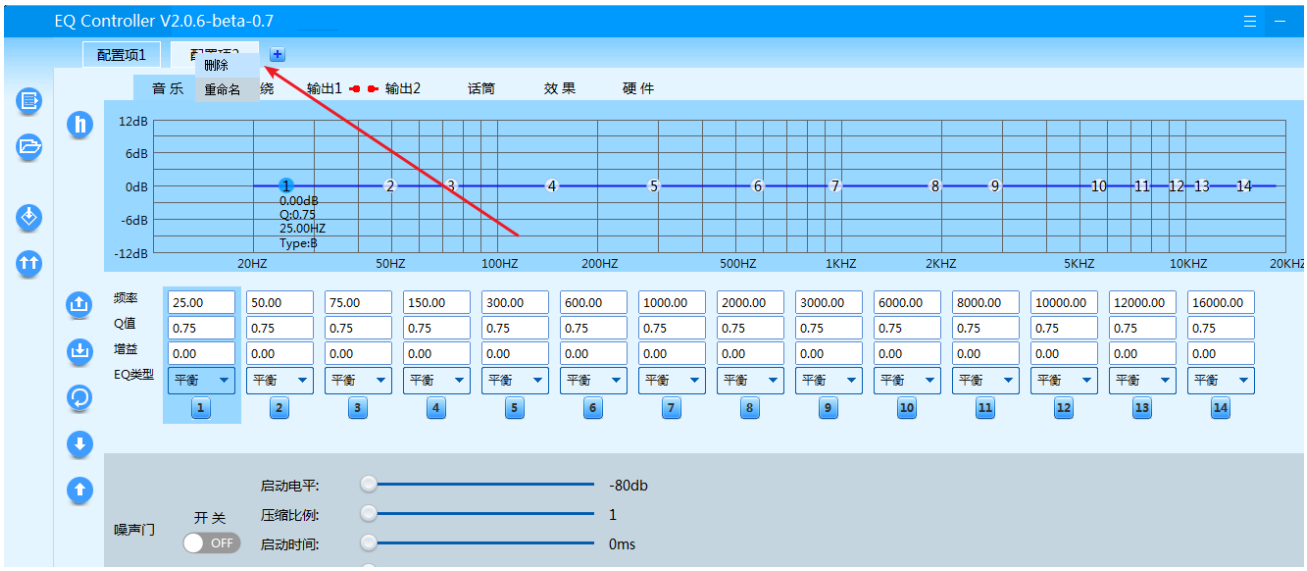


图 21-1-12-3

23.1.13. 合并配置项保存

将多个配置项的所有数据合并成一个 eq2x 的配置文件，如图 21-1-13

图 21-1-13

23.1.14. 导入多配置数据

导入多配置会导致当前页面以及配置被删除，以便重新配置界面

图 21-1-14

23.1.15. 合并选项固化到设备

将当前的 EQ 参数永久保存到设备中，如图 21-1-15 所示

图 21-1-15

23.1.16. 提取固化的 EQ 数据

将设备中的数据提取到 PC 界面，如下图 21-1-16 所示

图 21-1-16

23.1.17. 设置

如下图 21-1-17，点击设置按钮，弹出设置窗口，可以修改相关的配置，设置修改后，点击退出时保存。

图 21-1-17

23.2. Flash_loader 工具

23.2.1. 更新内置 nor flash

1、按下硬件 PB9 按钮，保持 PB9 低电平，同时上电

2、PC 软件识别出设备接入，出现盘符，如图 21-2-1 所示

3、选择.up 后缀的文件

4、点击盘符按钮即可更新数据（这里的 J 盘符也可能是其它名字）

图 21-2-1

23.2.2. 更新外部 FLASH

1、设备上电

2、打开 PC 软件，选中“外部 FLASH”复选框，如图 3-1 所示

3、填入 CS/CLK/MOSI/MISO 对应的 pin 脚，如果使用三线模式，MOSI 和 MISO 填写一样的 pin 脚

4、点击盘符更新数据

图 3-1

23.2.3. 辅助功能

“自动下载文件”：勾选后，如果文件有效，设备接入后就会自动更新数据

“下载结束后复位硬件”：下载结束后烧录控制程序会启动 watchdog 以及 PMU 复位芯

23.3. 加密软件

加密软件总有两个，对应一级加密和二级加密

eclGenerator.exe

这个工具用于生成一对一级授权文件。选择原厂授权的 cert 文件，并输入一个 1-0xFFFF(32bit)的 16 进制数据后,点击“生成”按钮就即可生成后缀为 ecl 和 dcl 的一对文件，ecl 用于芯片的密钥或者二级授权文件，dcl 用于芯片的解密密钥，芯片发布前需要烧录 dcl 文件

efuseGenerator.exe

这个工具用于生成一对二级授权文件。选择一级授权文件 ecl，并输入一个 1-0xFFF(24bit)的 16 进制数据后，点击“生成”按钮就即可生成后缀为 efs 和 dfs 的一对文件，efs 用于芯片的密钥或者二级授权文件，dfs 用于芯片的解密密钥，如果代码使用 efs 加密，那么需要烧录 dfs 文件，程序才能正常运行

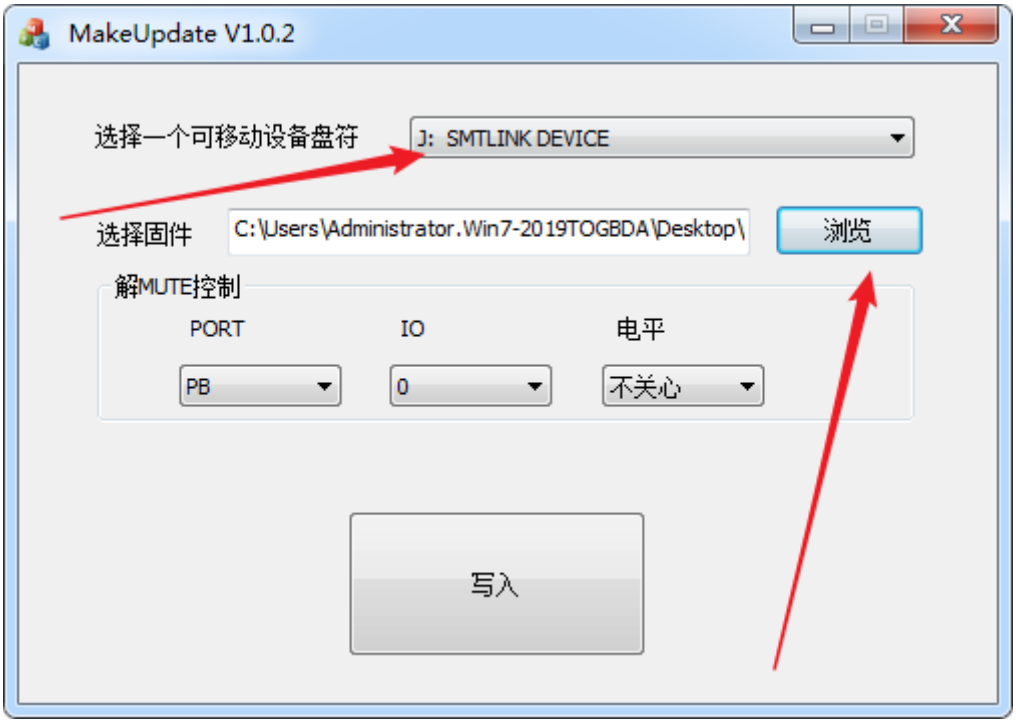
23.4. 升级软件

固件的升级有四种软件，其中一种是使用 flash_loader 以及烧录器，这里不在叙述，另外两种对应两个场景

makeupdate.exe

当芯片已经上板但是没有程序，并且板子 USB 不可用，SD 接口可用时，按照下面操作流程进行升级：

1、插入一个带 SD 卡的读卡器，程序会识别出其可移动盘符并显示，选择对应的盘符，如下图所示



2、如果有外部 Mute 控制，可以在“解 MUTE 控制”当中进行相应设置，以便可以接听相关提示

3、选择所要升级的固件，然后点击写入即可

4、将 SD 卡在断电的情况下插入待升级设备中

5、将 PB9 拉低(按下第一个 AD 按键)，上电，持续 1 秒钟后，松开按键即进入升级模式即可

makeupd.exe

这个软件用于芯片已经存在固件，需要进行代码升级的情况，那么您需要按以下流程进行：

1、点击“浏览”按钮选择一个要升级的固件

2、点击“生成”按钮，生成 sl_update.upd2 或者 sl_update.upd 文件,生成的文件后缀由固件启动格式决定，V1.4 版本之前的 SDK 开发的代码会生成 upd 格式，后面的版本会生成 upd2 格式

3、将 sl_update.upd 或者 sl_update.upd2 拷贝到一个 U 盘或者 SD 卡中

4、将 U 盘或者 SD 卡插入设备，程序会进入升级模式，并播放中文提示音“正在升级”，注意，如果要使用升级功能，那么您需要在代码中将宏“MODE_DISKUPDATE_EN”设置为 1

5、期间不要断开电源，升级成功后会复位

6、如果无法升级失败，会自动转入音乐模式

7、如果 upd2 后缀的文件升级失败，还可以使用 loader 方式进行升级，你需要将 sl_update_root.bin 以及 sl_update.upd2 一起拷贝到 U 盘或者 SD 卡中，插入设备即可，升级前会提示“正在升级”，升级结束后提示“升级成功”并对设备进行复位

8、升级过一次后设备会记录升级的时间戳，相同的时间戳不能再次升级

23.5. 提示音合成软件 maketone

要使用自定义的提示音，您需要安装下面流程进行操作：

1、将音频文件（只支持.mp3 和.sbc 后缀名）复制到 maketone 文件夹内。

2、运行 maketone.exe，会生成新的 tone.bin 和 tone.h 文件。

3、将 tone.bin 文件复制到 build/bin 文件夹内，将 tone.h 文件复制到 app/moon/audio 文件夹内。

4、重新编译代码即可。

音频文件的相关参数建议如下：

格式： mp3

码率模式：CBR

码率： 24kb/s

采样率： 16kHz

声道: 1 声道

特别注意：

出于节省资源的目的，通话过程的 call.mp3 和报号 tone(numx.mp3)会共用一套解码器，所以 call.mp3 和 numx.mp3 需要用同一个编码器用相同规格编码。

24. 音效软件 API

24.1. 基本 API 说明

24.1.1. on_x_eq_setting_for_pageid

EQ_SETTING_STU on_x_eq_setting_for_pageid(PageClass page_id, EQ_SUBBAND band, uint32_t mask, eq_band_setting* setting, bool set)

描述	设置/获取指定页面指定段的 EQ 参数	
返回值		EQ_SETTING_STU
参数一	page_id，	要调节 EQ 的 LOCATION PAGE，只支持 PAGE_MUSIC, PAGE_MIC
参数二	band	要调节的段点，见 EQ_SUBBAND
参数三	mask	本接口支持灵活设置，可设置的项见 EQ_SUBBAND_PARAS，以下为常见的设置方式 全部设置 mask = EQ_SB_ALL_PAPAS_MASK 单设置（以设置频点为例） mask = COMMON_PARA_MASK(EQ_FC) 除单设置（以除了设置频点，其他都设为例） mask = COMMON_PARAS_MASK_EXCLUDE(EQ_FC)
参数四	setting	设置结构体指针，见 eq_band_setting
参数五	set	0 代表获取，1 代表设置

Example

```
#include “eq_process.h”
    eq_band_setting ebs = {0};
    //获取
    on_x_eq_setting_for_pageid(PAGE_MUSIC, EQ_BAND_1, EQ_SB_ALL_PAPAS_MASK, &ebs, 0);
    //设置
    on_x_eq_setting_for_pageid(PAGE_MUSIC, EQ_BAND_1, EQ_SB_ALL_PAPAS_MASK, &ebs, 1);
```

24.1.2. on_x_eq_preamp_for_pageid

EQ_SETTING_STU on_x_eq_preamp_for_pageid(PageClass page_id, float* preamp, bool set)

描述	设置/读取指定页面的 EQ 预增益	
返回值		EQ_SETTING_STU
参数一	page_id，	要调节 EQ 的 LOCATION PAGE，只支持 PAGE_MUSIC, PAGE_MIC
参数二	preamp	Float 型预增益指针，
参数三	set	0 代表获取，1 代表设置

Example

```
#include “eq_process.h”
    float preamp = 0.0f;
    //获取
    on_x_eq_preamp_for_pageid(PAGE_MUSIC, &preamp , 0);
    //设置
    on_x_eq_preamp_for_pageid(PAGE_MUSIC, &preamp , 1);
```

24.1.3. on_x_dr_switch_for_pageid

EQ_SETTING_STU on_x_dr_switch_for_pageid(PageClass page_id, bool* on_off, bool set);

描述	获取/设置延时反向的开关状态	
返回值		EQ_SETTING_STU
参数一	page_id，	要调节的 LOCATION PAGE，只支持 PAGE_OUTPUT1，PAGE_OUTPUT2
参数二	on_off	开关的指针
参数三	set	0 代表获取，1 代表设置

Example

```
#include “eq_process.h”
    bool on_off = 0.0f;
    //获取
    on_x_dr_switch_for_pageid(PAGE_OUTPUT1, &on_off, 0);
    //设置
    on_x_dr_switch_for_pageid(PAGE_OUTPUT1, &on_off, 1);
```

24.1.4. on_x_dr_setting_for_pageid

EQ_SETTING_STU on_x_dr_setting_for_pageid(PageClass page_id, dly_setting* setting, bool set);

描述	获取/设置延时参数	
返回值		EQ_SETTING_STU
参数一	page_id，	要调节的 LOCATION PAGE，只支持 PAGE_OUTPUT1，PAGE_OUTPUT2
参数二	setting	延时反向结构体指针，类型 dly_setting
参数三	set	0 代表获取，1 代表设置

Example

```
#include “eq_process.h”
    dly_setting setting = {0};
    //获取
    on_x_dr_setting_for_pageid(PAGE_OUTPUT1, &setting , 0);
    //设置
    setting.rev_factor = …;
    setting.rtdly = …;
    on_x_dr_setting_for_pageid(PAGE_OUTPUT1, &setting , 1);
```

24.1.5. on_x_drc_makeup_gain_for_pageid

EQ_SETTING_STU on_x_drc_makeup_gain_for_pageid(PageClass page_id, float* makeup, bool set)

描述	获取/设置 DRC 的补偿增益	
返回值		EQ_SETTING_STU
参数一	page_id，	要调节的 LOCATION PAGE，不支持 PAGE_MUSIC，PAGE_EFFECT
参数二	makeup	Drc 补偿增益指针
参数三	set	0 代表获取，1 代表设置

Example

```
#include “eq_process.h”
    float makeup = 0.0f;
    //获取
    on_x_drc_makeup_gain_for_pageid(PAGE_MIC, &makeup , 0);
```

```
//设置
makeup = …;
on_x_drc_makeup_gain_for_pageid(PAGE_MIC, &makeup , 1);
```

24.1.6. on_x_drc_subfunc_for_pageid

EQ_SETTING_STU on_x_drc_subfunc_for_pageid(PageClass page_id, uint32_t mask, DRC_SUBFUNC drc_sb, drc_subfunc_setting* setting, bool set)

描述	获取/设置 DRC 子模块的参数	
返回值		EQ_SETTING_STU
参数一	page_id，	要调节的 LOCATION PAGE，不支持 PAGE_EFFECT
参数二	mask	本接口支持灵活设置，可设置的项见 DRC_SUBFUNC_PARAS，以下为常见的设置方式 全部设置 mask = DRC_SUBFUNC_ALL_MASK 单设置（以设置启动电平为例） mask = COMMON_PARA_MASK(DRC_SFPARA_THD) 除单设置（以除了设置启动电平，其他都设为例） mask = COMMON_PARAS_MASK_EXCLUDE(DRC_SFPARA_THD)
参数三	drc_sb	设置子模块，类型见 DRC_SUBFUNC 定义
参数四	setting	Drc 设置参数结构体指针
参数五	set	0 代表获取，1 代表设置

Example

```
#include “eq_process.h”
    drc_subfunc_setting dss = {0};
    //获取
    on_x_drc_subfunc_for_pageid(PAGE_MIC,  DRC_SUBFUNC_ALL_MASK ,  DRC_LIMITER，&dss , 0);
    //设置
    dss.x_thd = …;
    dss.x_ratio = …;
    dss.x_att = …;
    dss.x_rel = …;
    dss.enable = …;
    on_x_drc_subfunc_for_pageid(PAGE_MIC,  DRC_SUBFUNC_ALL_MASK ,  DRC_LIMITER，&dss , 1);
```

24.1.7. on_x_tsps_for_pageid

EQ_SETTING_STU on_x_tsps_for_pageid(PageClass page_id, uint32_t mask, tsps_setting* setting, bool set)

描述	获取/设置变调的参数	
返回值		EQ_SETTING_STU
参数一	page_id，	要调节的 LOCATION PAGE，支持 PAGE_MIC，PAGE_MUSIC
参数二	mask	本接口支持灵活设置，可设置的项见 TSPS_PARAS，以下为常见的设置方式 全部设置 mask = TSPS_ALL_PAPAS_MASK 单设置（以设置声调为例） mask = COMMON_PARA_MASK(TSPS_PS) 除单设置（以除了设置启动电平，其他都设为例） mask = COMMON_PARAS_MASK_EXCLUDE(TSPS_PS)
参数三	setting	变调设置参数结构体指针
参数四	set	0 代表获取，1 代表设置

Example

```
#include “eq_process.h”
    tsps_setting ts = {0};
    //获取
    on_x_tsps_for_pageid(PAGE_MIC, TSPS_ALL_PAPAS_MASK, &ts , 0);
    //设置
    ts .alpha_ts = 1.0f;
    ts .alpha_ps = …;
    ts .enable = …;
    on_x_tsps_for_pageid(PAGE_MIC, TSPS_ALL_PAPAS_MASK, &ts , 1);
```

24.1.8. on_x_atune_for_pageid

EQ_SETTING_STU on_x_atune_for_pageid(PageClass page_id, uint32_t mask, atune_setting* setting, bool set)

描述	获取/设置电音的参数	
返回值		EQ_SETTING_STU
参数一	page_id，	要调节的 LOCATION PAGE，支持 PAGE_MIC
参数二	mask	本接口支持灵活设置，可设置的项见 ATUNE_PARAS，以下为常见的设置方式 全部设置 mask = ATUNE_ALL_PAPAS_MASK 单设置（以设置基调为例） mask = COMMON_PARA_MASK(ATUNE_NOTE) 除单设置（以除了设置基调，其他都设为例） mask = COMMON_PARAS_MASK_EXCLUDE(ATUNE_NOTE)
参数三	setting	电音设置参数结构体指针
参数四	set	0 代表获取，1 代表设置

Example

```
#include “eq_process.h”
    atune_setting as = {0};
    //获取
    on_x_atune_for_pageid(PAGE_MIC, ATUNE_ALL_PAPAS_MASK, &as , 0);
    //设置
    as.note = …;
    as.oct = …;
    as.scale = …;
    as.speed = …;
    as.enable = …;
    on_x_atune_for_pageid(PAGE_MIC, ATUNE_ALL_PAPAS_MASK, &as , 1);
```

24.1.9. on_x_formant_for_pageid

EQ_SETTING_STU on_x_formant_for_pageid(PageClass page_id, uint32_t mask, fc_setting* setting, bool set)

描述	获取/设置共振峰的参数	
返回值		EQ_SETTING_STU
参数一	page_id，	要调节的 LOCATION PAGE，支持 PAGE_MIC
参数二	mask	本接口支持灵活设置，可设置的项见 FC_PARAS，以下为常见的设置方式 全部设置 mask = FC_ALL_PAPAS_MASK 单设置（以设置共振系数为例）

		mask = COMMON_PARA_MASK(FC_FC) 除单设置（以除了设置共振系数，其他都设为例） mask = COMMON_PARAS_MASK_EXCLUDE(FC_FC)
参数三	setting	共振峰设置参数结构体指针
参数四	set	0 代表获取，1 代表设置

Example

```
#include “eq_process.h”
    fc_setting fs = {0};
    //获取
    on_x_formant_for_pageid(PAGE_MIC, FC_ALL_PAPAS_MASK, &fs , 0);
    //设置
    fs.alpha_fc = …;
    fs.enable = …;
    on_x_formant_for_pageid(PAGE_MIC, FC_ALL_PAPAS_MASK, &fs , 1);
```

24.1.10. on_x_output_flt_type_setting_for_pageid

EQ_SETTING_STU on_x_output_flt_type_setting_for_pageid(PageClass page_id, hl_type hl, hl_filter_setting* setting, bool set)

描述	获取/设置输出 12 高低通滤波器的参数	
返回值		EQ_SETTING_STU
参数一	page_id，	要调节的 LOCATION PAGE，支持 PAGE_OUTPUT1, PAGE_OUTPUT2
参数二	hl	滤波器类型，分高低通，具体见 hl_type
参数三	setting	高低通滤波器设置参数结构体指针
参数四	set	0 代表获取，1 代表设置

Example

```
#include “eq_process.h”
    hl_filter_setting hlfs = {0};
    //获取
    on_x_output_flt_type_setting_for_pageid(PAGE_OUTPUT1, LOW_PASS, &hlfs , 0);
    //设置
    hlfs.sub_type = …;
    hlfs.fc = …;
    on_x_output_flt_type_setting_for_pageid(PAGE_OUTPUT1, LOW_PASS, &hlfs , 1);
```

24.1.11. on_x_echo_for_pageid

EQ_SETTING_STU on_x_echo_for_pageid(PageClass page_id, uint32_t mask, echo_setting* setting, bool set)

描述	获取/设置回声参数	
返回值		EQ_SETTING_STU
参数一	page_id，	要调节的 LOCATION PAGE，支持 PAGE_EFFECT
参数二	mask	本接口支持灵活设置，可设置的项见 ECHO_PARAS，以下为常见的设置方式 全部设置 mask = ECHO_ALL_PAPAS_MASK 单设置（以设置湿声音量为例） mask = COMMON_PARA_MASK(ECHO_GAIN_WET) 除单设置（以除了设置湿声音量，其他都设为例） mask = COMMON_PARAS_MASK_EXCLUDE(ECHO_GAIN_WET)
参数三	setting	回声设置参数结构体指针

参数四	set	0 代表获取，1 代表设置

Example

```
#include “eq_process.h”
    echo_setting es = {0};
    //获取
    on_x_echo_for_pageid(PAGE_EFFECT, ECHO_ALL_PAPAS_MASK, &es , 0);
    //设置
    …;
    on_x_echo_for_pageid(PAGE_EFFECT, ECHO_ALL_PAPAS_MASK, &es , 1);
```

24.1.12. on_x_rev_for_pageid

EQ_SETTING_STU on_x_rev_for_pageid(PageClass page_id, uint32_t mask, rev_setting* setting, bool set)

描述	获取/设置混响参数	
返回值		EQ_SETTING_STU
参数一	page_id，	要调节的 LOCATION PAGE，支持 PAGE_EFFECT
参数二	mask	本接口支持灵活设置，可设置的项见 REVB_PARAS，以下为常见的设置方式 全部设置 mask = REVERB_ALL_PAPAS_MASK 单设置（以设置湿声音量为例） mask = COMMON_PARA_MASK(REVB_PARA_GAIN_WET) 除单设置（以除了设置湿声音量，其他都设为例） mask = COMMON_PARAS_MASK_EXCLUDE(REVB_PARA_GAIN_WET)
参数三	setting	混响设置参数结构体指针
参数四	set	0 代表获取，1 代表设置

Example

```
#include “eq_process.h”
    rev_setting rs = {0};
    //获取
    on_x_rev_for_pageid(PAGE_EFFECT, REVERB_ALL_PAPAS_MASK, &rs , 0);
    //设置
    …;
    on_x_rev_for_pageid(PAGE_EFFECT, REVERB_ALL_PAPAS_MASK, &rs , 1);
```

24.1.13. on_x_vss_pro_for_pageid

EQ_SETTING_STU on_x_vss_pro_for_pageid(PageClass page_id, vss_mode* mode, vss_pro_setting* setting, bool set)

描述	获取/设置 3D 环绕和状态参数	
参数一	page_id，	要调节的 LOCATION PAGE，支持 PAGE_VSS
参数二	mode	3D 模式状态指针。在获取行为下，通过预设 mode，可以获取对应模式的参数，并且返回实际 running 的模式。模式有两种，耳机模式和扬声器模式
参数三	setting	3D 具体模式的参数结构体指针
参数四	set	0 代表获取，1 代表设置

Example

```
#include “eq_process.h”
    vss_mode mode = VSS_MODE_HEADPHONE;
    vss_pro_setting vps = {0};
    //获取耳机模式参数
```

```
on_x_vss_pro_for_pageid(PAGE_VSS, &mode , &vps , 0);
//强制设成扬声器模式，并设置相应参数
mode = VSS_MODE_SPEAKER;
vps.sound_field_width.spk_span = …;
…;
on_x_vss_pro_for_pageid(PAGE_VSS, &mode , &vps , 1);
```

24.1.14. on_x_vss_switch_for_pageid

EQ_SETTING_STU on_x_vss_switch_for_pageid(PageClass page_id, bool* on_off, bool set)

描述	获取/设置 3D 环绕总开关状态	
参数一	page_id，	要调节的 LOCATION PAGE，支持 PAGE_VSS
参数二	on_off	3D 总开关状态指针
参数三	set	0 代表获取，1 代表设置

Example

```
#include “eq_process.h”
    bool on_off = 0;
    //获取
    on_x_vss_switch_for_pageid(PAGE_VSS, &on_off , 0);
    //设置
    …;
    on_x_vss_switch_for_pageid(PAGE_VSS, &on_off , 1);
```

24.2. 其他集成高级接口

函数	描述	编译要求
void on_set_bypass_profile()	直通模式设置	
void clear_all_holder()	取消所有保持节点	
void on_set_default_reverb_profile(bool holdRevb, bool onlyRevb)	默认混响效果设置	AUDIO_MIC_REVERB(1)
void on_set_rec_reverb_ms_switch(bool stereo)	设置目前混响效果的单双通道模式，true 双，false 单	
void on_set_reverb_gwet(uint8_t g)	0 <= g <= 100, means 0% -> 100% 设置当前混响效果的湿声大小，0% - 100%	
void on_set_reverb_time(uint8_t time)	0 <= time <= 100, means 0% -> 100% 设置当前混响效果的长度大小，0% - 100%	
void on_set_default_echo_profile (bool holdEcho, bool cleanOtherAfx)	设置默认的回声模式 holdEcho ->是否保持回声 cleanOtherAfx -> 是否清除其他效果	AUDIO_MIC_ECHO(1)
void on_set_echo_delay(uint8_t dly)	0 <= dly <= 100, means 0% -> 100% 设置目前回声效果的延时大小，0% - 100%	
void on_set_echo_gwet(uint8_t g)	0 <= g <= 100, means 0% -> 100% 设置目前回声效果的湿声大小，0% - 100%	
void on_set_female_profile(bool cleanOtherAfx)	设置女声模式 cleanOtherAfx -> 是否清除其他效果	AUDIO_MIC_TSPS(1) AUDIO_AFX_REC_FORMANT_EN(1)
void on_set_male_profile(bool cleanOtherAfx)	设置男声模式 cleanOtherAfx -> 是否清除其他效果	AUDIO_MIC_TSPS(1)
void on_set_robot_profile(bool cleanOtherAfx)	设置机器人模式 cleanOtherAfx -> 是否清除其他效果	AUDIO_MIC_TSPS(1) AUDIO_MIC_ECHO(1)

void on_set_kid_profile(bool cleanOtherAfx)	设置童声模式 cleanOtherAfx -> 是否清除其他效果	AUDIO_MIC_TSPS(1) AUDIO_MIC_ECHO(1)
void on_set_voc_change_profile(bool cleanOtherAfx)	设置变声模式 cleanOtherAfx -> 是否清除其他效果	AUDIO_MIC_TSPS(1)
void on_set_radio_profile(bool cleanOtherAfx)	设置喇叭音模式 cleanOtherAfx -> 是否清除其他效果	AUDIO_MIC_EQ(1)
void on_set_treb_bass_gain(int8_t gain, uint8_t stream, eq_type_t eq_band)	设置 MIC/播放流的高低音 gain -> 增益， (-12,12）db stream -> 音源， 0(MIC), 1(播放) eq_band -> 频带， 0(低频), 1(中频), 2(高频)	AUDIO_MIC_EQ(1) AUDIO_PLAYBACK_EQ(1)
void on_set_eq_focus_preamp(int8_t db)	设置当前 EQ 配置的预增益 db : -12 - 12	
void on_set_autotune_profile(ATUNE_CMD cmd, uint8_t val);	设置电音模式 cmd : ATUNE_SET_SWITCH val : 0(off), 1(on, donot clear other effects), 2(on, and clear other effects) cmd : ATUNE_SET_KEY val : 0(KEY_C), 1(KEY_Db), 2(KEY_D), 3(KEY_Eb), 4(KEY_E), 5(KEY_F) 6(KEY_Gb), 7(KEY_G), 8(KEY_Ab), 9(KEY_A), 10(KEY_Bb), 11(KEY_B) cmd : ATUNE_SET_SCALE val : 0(CHROMATIC), 1(MAJOR), 2(MINOR)	AUDIO_MIC_TSPS(1) AUDIO_AFX_REC_AUTOTUNE_EN(1)
void on_set_natural_noise_gate(EFFECT_SWTICH sw, bool hold)	设置噪声门 sw : 0(off), 1(on, donot clear other effects), 2(on, and clear other effects) hold : 0(do not hold), 1(hold)	AUDIO_MIC_DRC(1)
void on_set_default_vss_spk_profile(EFFECT_SWTICH sw)	设置 3D 效果扬声器模式 sw : 0(off), 1(on, donot clear other effects), 2(on, and clear other effects)	AUDIO_PLAYBACK_VSS(1) AUDIO_AFX_VSS_EQ_ENABLE(1)
void on_set_default_vss_hp_profile(EFFECT_SWTICH sw)	设置 3D 效果耳机模式 sw : 0(off), 1(on, donot clear other effects), 2(on, and clear other effects)	同上
void on_set_rock_mic_eq_profile(EFFECT_SWTICH sw)	设置 MIC 的喊麦模式 sw : 0(off), 1(on, donot clear other effects), 2(on, and clear other effects)	AUDIO_MIC_ECHO(1) AUDIO_AFX_POST_ECHO_EQ_EN(1) AUDIO_MIC_REVERB(1) AUDIO_AFX_POST_REVB_EQ_EN(1)
void on_set_sonic_boom_eq_profile(EFFECT_SWTICH sw)	设置 MIC 的爆音模式 sw : 0(off), 1(on, donot clear other effects), 2(on, and clear other effects)	同上
void on_set_tbvc_switch(bool on);	设置等响的动态开关 on : 0(off), 1(on)	AUDIO_PLAYBACK_TBVC(1)