**SmartLink软件**

**开发说明**

**珠海慧联科技有限公司**

**未经许可，禁止外传**

**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 7](#_Toc26019580)

[1.1. 编辑 7](#_Toc26019581)

[1.2. 编译 7](#_Toc26019582)

[1.2.1. 编译工具 7](#_Toc26019583)

[1.2.2. 工具链 8](#_Toc26019584)

[1.2.3. 操作指令 8](#_Toc26019585)

[1.3. 下载 8](#_Toc26019586)

[1.3.1. 下载文件 8](#_Toc26019587)

[1.3.2. 下载工具 8](#_Toc26019588)

[1.3.3. 操作方法 9](#_Toc26019589)

[2. 文件结构 10](#_Toc26019590)

[2.1. app 10](#_Toc26019591)

[2.1.1. configs 10](#_Toc26019592)

[2.1.2. starts 10](#_Toc26019593)

[2.1.3. moon 10](#_Toc26019594)

[2.1.4. task 10](#_Toc26019595)

[2.1.5. main.c 10](#_Toc26019596)

[2.2. build 10](#_Toc26019597)

[2.2.1. bin 10](#_Toc26019598)

[2.2.2. out 10](#_Toc26019599)

[2.2.3. tool 10](#_Toc26019600)

[2.3. inc 11](#_Toc26019601)

[2.4. lib 11](#_Toc26019602)

[2.5. tools 11](#_Toc26019603)

[2.6. start.bat 11](#_Toc26019604)

[3. 代码空间 12](#_Toc26019605)

[3.1. 概述 12](#_Toc26019606)

[3.2. 分配方法 12](#_Toc26019607)

[3.3. 出错处理 12](#_Toc26019608)

[4. 宏配置介绍 15](#_Toc26019609)

[4.1. 系统相关宏配置 15](#_Toc26019610)

[4.2. 用户相关宏配置 15](#_Toc26019611)

[4.3. 音效相关宏配置 22](#_Toc26019612)

[5. 系统调试快速入门 24](#_Toc26019613)

[5.1. 确认芯片型号 24](#_Toc26019614)

[5.2. 选择配置文件 24](#_Toc26019615)

[5.3. 选择供电/开机方式 24](#_Toc26019616)

[5.4. 配置打印输出 24](#_Toc26019617)

[5.4.1. 开/关配置 24](#_Toc26019618)

[5.4.2. IO配置 24](#_Toc26019619)

[5.4.3. 波特率配置 25](#_Toc26019620)

[5.4.4. 打印等级 25](#_Toc26019621)

[5.5. 注意 25](#_Toc26019622)

[6. 音频控制软件接口 26](#_Toc26019623)

[6.1. 通用接口 26](#_Toc26019624)

[6.2. SD卡/U盘播放 26](#_Toc26019625)

[6.3. 提示音播放接口 27](#_Toc26019626)

[6.4. 蓝牙播放接口 28](#_Toc26019627)

[6.5. Linein播放接口 28](#_Toc26019628)

[6.6. FM播放接口 28](#_Toc26019629)

[6.7. USB Audio播放/录音接口 28](#_Toc26019630)

[6.8. Spdif播放接口 29](#_Toc26019631)

[6.9. SD卡/U盘/flash录音接口 30](#_Toc26019632)

[6.10. 卡拉OK方案相关接口 30](#_Toc26019633)

[6.10.1. AuxTrack接口 30](#_Toc26019634)

[7. 输入/输出（IO） 32](#_Toc26019635)

[7.1. 概述 32](#_Toc26019636)

[7.1.1. 基本情况 32](#_Toc26019637)

[7.1.2. 特别IO说明 32](#_Toc26019638)

[7.2. 功能 32](#_Toc26019639)

[7.3. 驱动能力 33](#_Toc26019640)

[7.4. 上下拉电阻 33](#_Toc26019641)

[7.5. 中断 33](#_Toc26019642)

[7.6. 软件配置 33](#_Toc26019643)

[8. 按键ADC（KEYADC） 35](#_Toc26019644)

[8.1. 概述 35](#_Toc26019645)

[8.2. 工作模式 35](#_Toc26019646)

[8.2.1. Normal Mode 35](#_Toc26019647)

[8.2.2. Continue Mode 35](#_Toc26019648)

[8.2.3. Single Mode 35](#_Toc26019649)

[8.2.4. Knob Mode 35](#_Toc26019650)

[8.3. 中断 35](#_Toc26019651)

[8.3.1. KEYUP 35](#_Toc26019652)

[8.3.2. KEYDOWN 35](#_Toc26019653)

[8.3.3. KEYDATA 35](#_Toc26019654)

[8.4. 软件配置 36](#_Toc26019655)

[9. LED显示控制器（LEDC） 37](#_Toc26019656)

[9.1. 概述 37](#_Toc26019657)

[9.2. 软件配置 37](#_Toc26019658)

[9.2.1. 显示真值表 37](#_Toc26019659)

[9.2.2. 输入检测 37](#_Toc26019660)

[10. 定时器（TIMER） 38](#_Toc26019661)

[10.1. 概述 38](#_Toc26019662)

[11. 脉宽调制（PWM） 39](#_Toc26019663)

[11.1. 概述 39](#_Toc26019664)

[11.2. 软件配置 39](#_Toc26019665)

[12. 串行通信（UART） 40](#_Toc26019666)

[12.1. 概述 40](#_Toc26019667)

[12.2. 软件配置 40](#_Toc26019668)

[13. 两线串行接口（TWI） 41](#_Toc26019669)

[13.1. 概述 41](#_Toc26019670)

[13.2. 软件配置 41](#_Toc26019671)

[14. 串行外设接口（SPI） 42](#_Toc26019672)

[14.1. 概述 42](#_Toc26019673)

[14.2. 软件配置 42](#_Toc26019674)

[15. 红外接收（IRRX） 43](#_Toc26019675)

[15.1. 概述 43](#_Toc26019676)

[15.2. 软件配置 43](#_Toc26019677)

[16. 内置音频总线（I2S） 44](#_Toc26019678)

[16.1. 概述 44](#_Toc26019679)

[16.2. 软件配置 44](#_Toc26019680)

[17. 数字音频接口（SPDIF） 45](#_Toc26019681)

[17.1. 概述 45](#_Toc26019682)

[17.2. 软件配置 45](#_Toc26019683)

[18. Audio Codec 46](#_Toc26019684)

[18.1. 概述 46](#_Toc26019685)

[18.2. 软件配置 46](#_Toc26019686)

[18.2.1. 基本配置接口 46](#_Toc26019687)

[18.2.2. 其他Codec配置接口 46](#_Toc26019688)

[19. PMU 49](#_Toc26019689)

[19.1. user config 49](#_Toc26019690)

[19.1.1. pmu startup mode config 49](#_Toc26019691)

[19.1.2. pmu config 49](#_Toc26019692)

[19.1.3. battery config 51](#_Toc26019693)

[19.1.4. pmu key config 53](#_Toc26019694)

[19.2. user interfaces 53](#_Toc26019695)

[19.2.1. bool pmu\_is\_enter\_charge\_mode(void) 53](#_Toc26019696)

[19.2.2. void pmu\_set\_core\_voltage(uint32\_t val) 53](#_Toc26019697)

[19.2.3. bool pmu\_is\_use\_hsw(void) 53](#_Toc26019698)

[19.2.4. void pmu\_sel\_onoff\_hsw(enum pmu\_onoff\_hsw\_e sel) 53](#_Toc26019699)

[19.2.5. void pmu\_hsw\_reset\_enable(bool enable) 53](#_Toc26019700)

[19.2.6. bool pmu\_bat\_is\_charging(void) 53](#_Toc26019701)

[19.2.7. bool pmu\_bat\_is\_exist(void) 54](#_Toc26019702)

[19.2.8. bool pmu\_bat\_is\_full(void) 54](#_Toc26019703)

[19.2.9. bool pmu\_bat\_is\_low(void) 54](#_Toc26019704)

[19.2.10. void pmu\_charge\_enable(bool enable) 54](#_Toc26019705)

[19.2.11. uint8\_t pmu\_get\_bat\_quantity\_percent(void) 54](#_Toc26019706)

[19.3. PMU event 54](#_Toc26019707)

[20. Power management 56](#_Toc26019708)

[20.1. Sleep 56](#_Toc26019709)

[20.1.1. SLEEP\_EN 56](#_Toc26019710)

[20.1.2. SLEEP\_DELAY 56](#_Toc26019711)

[20.1.3. TIME\_BEFORE\_WFI 56](#_Toc26019712)

[20.1.4. BT\_CON\_AUTO\_SLEEP\_EN 56](#_Toc26019713)

[20.1.5. BT\_CON\_AUTO\_SLEEP\_WAIT 56](#_Toc26019714)

[20.1.6. BT\_DIS\_AUTO\_SLEEP\_EN 57](#_Toc26019715)

[20.1.7. BT\_DIS\_AUTO\_SLEEP\_MAX 57](#_Toc26019716)

[20.2. Poweroff 57](#_Toc26019717)

[20.2.1. POWER\_OFF\_DELAY 57](#_Toc26019718)

[20.2.2. POWER\_OFF\_FIRST\_EN 57](#_Toc26019719)

[20.2.3. AUTO\_POWER\_OFF 57](#_Toc26019720)

[20.2.4. AUTO\_POWER\_OFF\_PERIOD 57](#_Toc26019721)

[20.3. Dvfs 57](#_Toc26019722)

[21. BLE 59](#_Toc26019723)

[21.1. 概述 59](#_Toc26019724)

[21.2. 软件配置 59](#_Toc26019725)

[21.3. 数据结构 59](#_Toc26019726)

[21.4. demo 59](#_Toc26019727)

[22. BT 62](#_Toc26019728)

[22.1. 软件配置 62](#_Toc26019729)

[22.2. API函数 62](#_Toc26019730)

[23. 工具 63](#_Toc26019731)

[23.1. EQ工具 63](#_Toc26019732)

[23.1.1. 调节EQ 63](#_Toc26019733)

[23.1.2. EQ类型 64](#_Toc26019734)

[23.1.3. EQ使能 64](#_Toc26019735)

[23.1.4. 调节其它效果参数 65](#_Toc26019736)

[23.1.5. 设备连接 66](#_Toc26019737)

[23.1.6. 错误隐患 67](#_Toc26019738)

[23.1.7. 导出 68](#_Toc26019739)

[23.1.8. 导入 68](#_Toc26019740)

[23.1.9. 重置 69](#_Toc26019741)

[23.1.10. 同步PC设置 69](#_Toc26019742)

[23.1.11. 提取设备EQ 70](#_Toc26019743)

[23.1.12. 多配置EQ的新建与删除 71](#_Toc26019744)

[23.1.13. 合并配置项保存 72](#_Toc26019745)

[23.1.14. 导入多配置数据 72](#_Toc26019746)

[23.1.15. 合并选项固化到设备 73](#_Toc26019747)

[23.1.16. 提取固化的EQ数据 74](#_Toc26019748)

[23.1.17. 设置 74](#_Toc26019749)

[23.2. Flash\_loader工具 75](#_Toc26019750)

[23.2.1. 更新内置 nor flash 75](#_Toc26019751)

[23.2.2. 更新外部 FLASH 76](#_Toc26019752)

[23.2.3. 辅助功能 77](#_Toc26019753)

[23.3. 加密软件 77](#_Toc26019754)

[23.4. 升级软件 78](#_Toc26019755)

[23.5. 提示音合成软件maketone 79](#_Toc26019756)

[24. 音效软件API 81](#_Toc26019757)

[24.1. 基本API说明 81](#_Toc26019758)

[24.1.1. on\_x\_eq\_setting\_for\_pageid 81](#_Toc26019759)

[24.1.2. on\_x\_eq\_preamp\_for\_pageid 81](#_Toc26019760)

[24.1.3. on\_x\_dr\_switch\_for\_pageid 82](#_Toc26019761)

[24.1.4. on\_x\_dr\_setting\_for\_pageid 82](#_Toc26019762)

[24.1.5. on\_x\_drc\_makeup\_gain\_for\_pageid 82](#_Toc26019763)

[24.1.6. on\_x\_drc\_subfunc\_for\_pageid 83](#_Toc26019764)

[24.1.7. on\_x\_tsps\_for\_pageid 83](#_Toc26019765)

[24.1.8. on\_x\_atune\_for\_pageid 84](#_Toc26019766)

[24.1.9. on\_x\_formant\_for\_pageid 84](#_Toc26019767)

[24.1.10. on\_x\_output\_flt\_type\_setting\_for\_pageid 85](#_Toc26019768)

[24.1.11. on\_x\_echo\_for\_pageid 85](#_Toc26019769)

[24.1.12. on\_x\_rev\_for\_pageid 86](#_Toc26019770)

[24.1.13. on\_x\_vss\_pro\_for\_pageid 86](#_Toc26019771)

[24.1.14. on\_x\_vss\_switch\_for\_pageid 87](#_Toc26019772)

[24.2. 其他集成高级接口 87](#_Toc26019773)

# 编辑/编译/下载

## 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。

## 编译

### 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。

### 工具链

请将工具链复制到tools文件夹下。

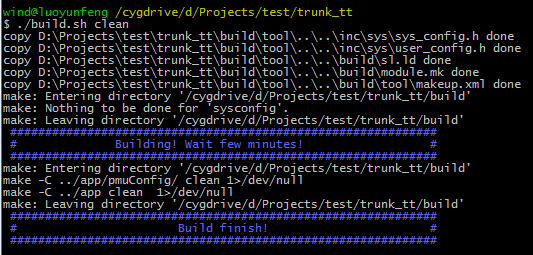
工具链有2个版本：

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

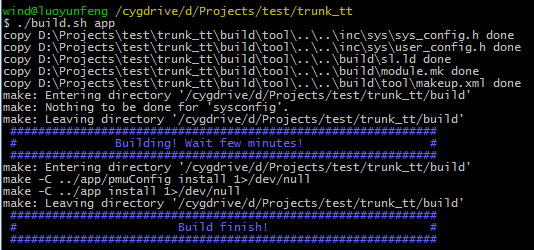
toolchain-7.3.1-e20.tar.bz2，v1.30或以后版本的SDK上使用。

### 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。

## 下载

### 下载文件

build/out/sl.up

### 下载工具

tools/flash\_loader/flash\_loader.exe

### 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。



# 文件结构

## app

上层应用相关文件、代码。

### 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”

### starts

硬件相关的模块接口。

### moon

软件相关的模块接口。

### task

各场景模式相关文件。

### main.c

系统主入口。

## build

编译相关文件、工具。

### bin

提示音数据、EQ数据。

### out

编译输出文件。

### tool

编译时用到的工具。

## inc

应用层用到的头文件。

## lib

底层编译生成的库文件。

## tools

工具链和其他各工具。

## start.bat

双击运行，会打开Cygwin Terminal窗口，并切换到默认路径。

# 代码空间

## 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。

## 分配方法

给代码函数分配空间的方法：

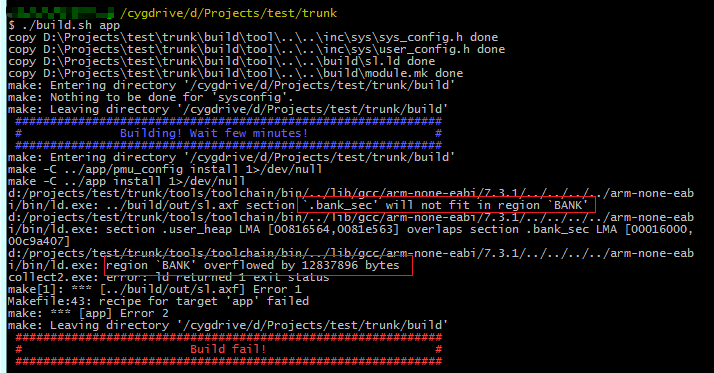
首先在函数前加AT(段名)，如“AT(.main\_seg)”，然后在.ld文件中填写段名。

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 |

## 出错处理

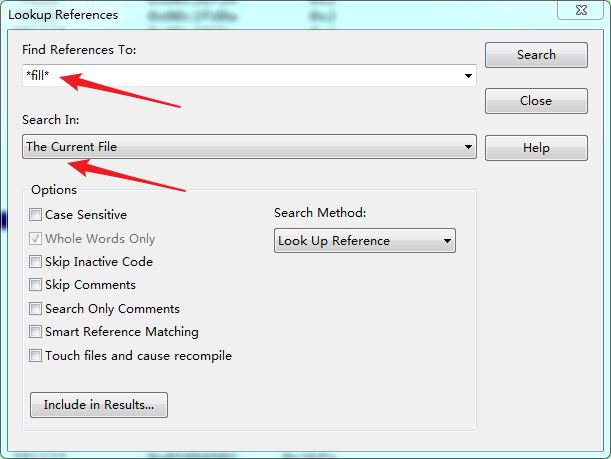
如果编译时出现错误提示“‘.bank\_sec’will not fit in region‘BANK’”，如下图：



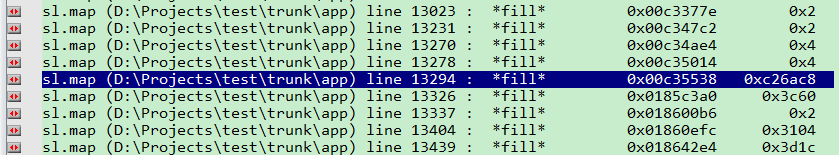
可以按以下方法处理：

（1）打开app文件夹下（不是build/out下）的sl.map文件。

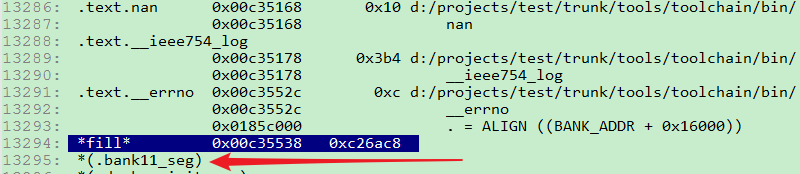
（2）搜索内容“\*fill\*”，只选择在当前文件内。



（3）找到大小特别大（超出0x00c00000）的那一行。



（4）跳转到sl.map文件。



（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。

# 宏配置介绍

## 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 |

## 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\_FREQ | 本地解码（其他）场景下CPU频率 | 192e6 | cpu freq config |
| DVFS\_DEFAULT\_FREQ | 默认CPU频率 | 64e6 | cpu freq config |

## 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 |  |

# 系统调试快速入门

## 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 |
|  |  |  |  |

## 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 |
|  |  |  |

## 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 |

## 配置打印输出

### 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。

### IO配置

宏定义UART0\_TX\_PIN\_SEL配置打印输出IO。

### 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。

### 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。

## 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。

# 音频控制软件接口

## 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 |

## 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 |

## 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） |

## 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 |

## 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 |

## FM播放接口

和linein播放接口一致

## 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 |

## 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 |

## 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 |

## 卡拉OK方案相关接口

### 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, (uint32\_t)&sta) | 获取播放状态：  sta为0： 没有正在播放。  sta为1：正在播放。 |

# 输入/输出（IO）

## 概述

### 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。

### 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。

## 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 | / | / | / | / | / | / | / | / |

## 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。

## 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。

## 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)。

## 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 |

# 按键ADC（KEYADC）

## 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。

## 工作模式

### 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。

### 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。

### 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。

### 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。

## 中断

有3种中断源可配置。

### KEYUP

按键抬起时产生中断。

### KEYDOWN

按键按下时产生中断。

### KEYDATA

按键持续按下，每采集得到一个数据时产生中断。

## 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。

# LED显示控制器（LEDC）

## 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。

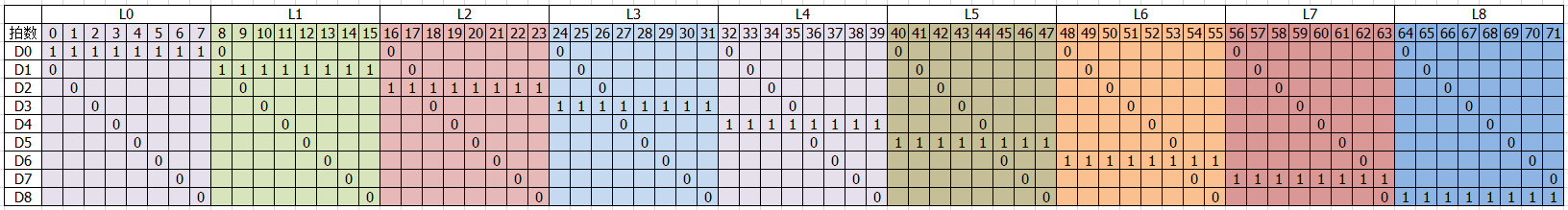
每操作一次开关LED，维持LEDC\_CLK数可配置。

支持单线LED显示屏。

## 软件配置

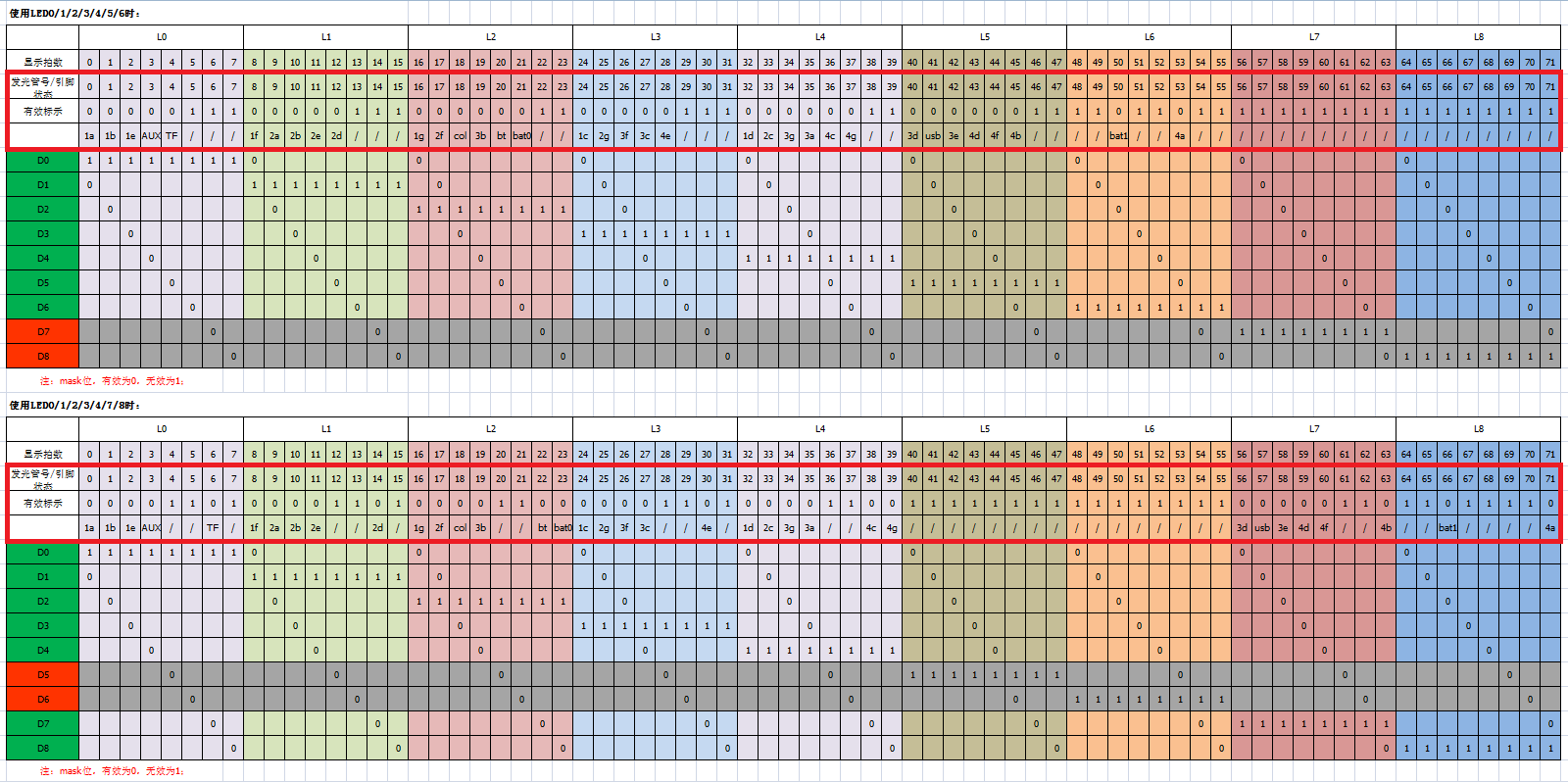
### 显示真值表

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：



### 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 |

# 定时器（TIMER）

## 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。

# 脉宽调制（PWM）

## 概述

支持6组pwm通道。

支持single pulse mode和continue cycle mode。

## 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。

# 串行通信（UART）

## 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）。

## 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 |

# 两线串行接口（TWI）

## 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。

## 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 |

# 串行外设接口（SPI）

## 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。

## 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 |

# 红外接收（IRRX）

## 概述

支持NEC协议。

支持引导码、 电位’0’/’1’等时间可配置。

## 软件配置

# 内置音频总线（I2S）

## 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

## 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。**

# 数字音频接口（SPDIF）

## 概述

支持从模式

采样率支持22.05K~192K

## 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。**

# Audio Codec

## 概述

支持3路ADC， 2路DAC。

## 软件配置

### 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 |

### 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, 3); //设置增益为3 | Linein1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; |

**更多audio codec相关函数接口，请参考audio\_service.h, audiopath\_conf.h文件中说明。**

# PMU

## user config

### pmu startup mode config

#### 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。

### pmu config

#### 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。

#### PMU\_VBUS\_POWERON\_EN

使能vbus插入即开机。

#### PMU\_SHORT\_PRESS\_POWERON\_EN

使能短按开机。

#### PMU\_ULTRA\_SHORT\_PRESS\_POWERON\_EN

使能超短按开机功能。

#### 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。

#### 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。

#### 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。

#### 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。

#### 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。

#### 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。

#### 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。

#### 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。

### battery config

#### 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。

#### 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。

#### 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。

#### 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。

#### 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。

#### 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。

#### 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。

#### 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。

#### 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。

### pmu key config

#### 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。

#### KEY\_PMU\_MULTFUC\_EN

pmu电源键多功能按键，支持单击，双击和长按。

## user interfaces

### 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。

### void pmu\_set\_core\_voltage(uint32\_t val)

设置cpu core电压，用于调频调压。

### bool pmu\_is\_use\_hsw(void)

判断pmu使用onoff（按键方案）还是hsw（硬开关方案）。

### 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）。

### 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。

### bool pmu\_bat\_is\_charging(void)

判断电池是否在充电。

### bool pmu\_bat\_is\_exist(void)

判断电池是否存在。

### bool pmu\_bat\_is\_full(void)

判断电池是否为满电。

### bool pmu\_bat\_is\_low(void)

判断电池是否为低电。

### void pmu\_charge\_enable(bool enable)

使能和除能电池充电功能。

### uint8\_t pmu\_get\_bat\_quantity\_percent(void)

获取电池电量百分比。返回值为0~100。

## 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 |

# Power management

## Sleep

### 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。

### 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。

### 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。

### 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。

### 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。

### 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。

### 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。

## Poweroff

### POWER\_OFF\_DELAY

poweroff之前延时多少ms再关机，一般配置为0。

### POWER\_OFF\_FIRST\_EN

当使用vbus供电，第一次启动关机。

### AUTO\_POWER\_OFF

蓝牙断开时，超过多长时间自动关机。

### 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。

## 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 |

# BLE

## 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

## 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

## 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

## 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;

1. 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

1. 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

# BT

## 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

## 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。

# 工具

## 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。

### 调节EQ

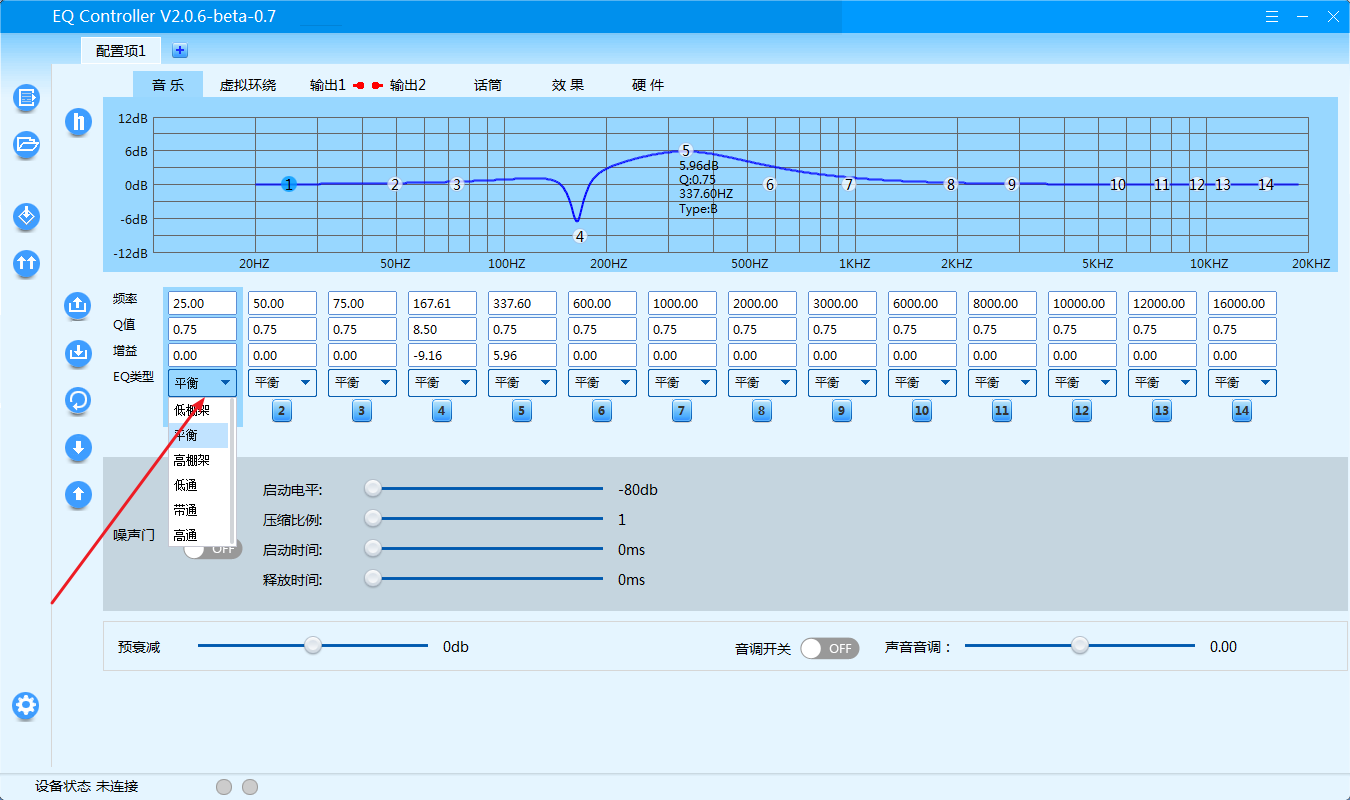
1. 调节EQ有四种个方法：
2. 在EQ曲线插件上面选中所要调节的参数，按住鼠标左键，然后拖到EQ按钮即可调节，如图21-1-1中箭头1指示。
3. 在EQ曲线插件上面选中相应的按钮后，滚动鼠标滚轮，可以微调频率值，单位默认为0.1HZ，如果按住shift键再滚动鼠标滚轮，则微调Q值，单位默认为0.1，如果按住ctrl键再滚动鼠标滚轮，则变为微调增益，单位默认为0.1。
4. 在EQ参数区直接输入相应的值 即可，参数会实时生效，如图21-1-1中箭头2所示
5. 将鼠标移动到相应的参数值区域，滚动鼠标滚轮，可以粗调相应的参数值，频率的粗调单位为默认10HZ，Q值和增益默认为0.1



**图21-1-1**

### EQ类型

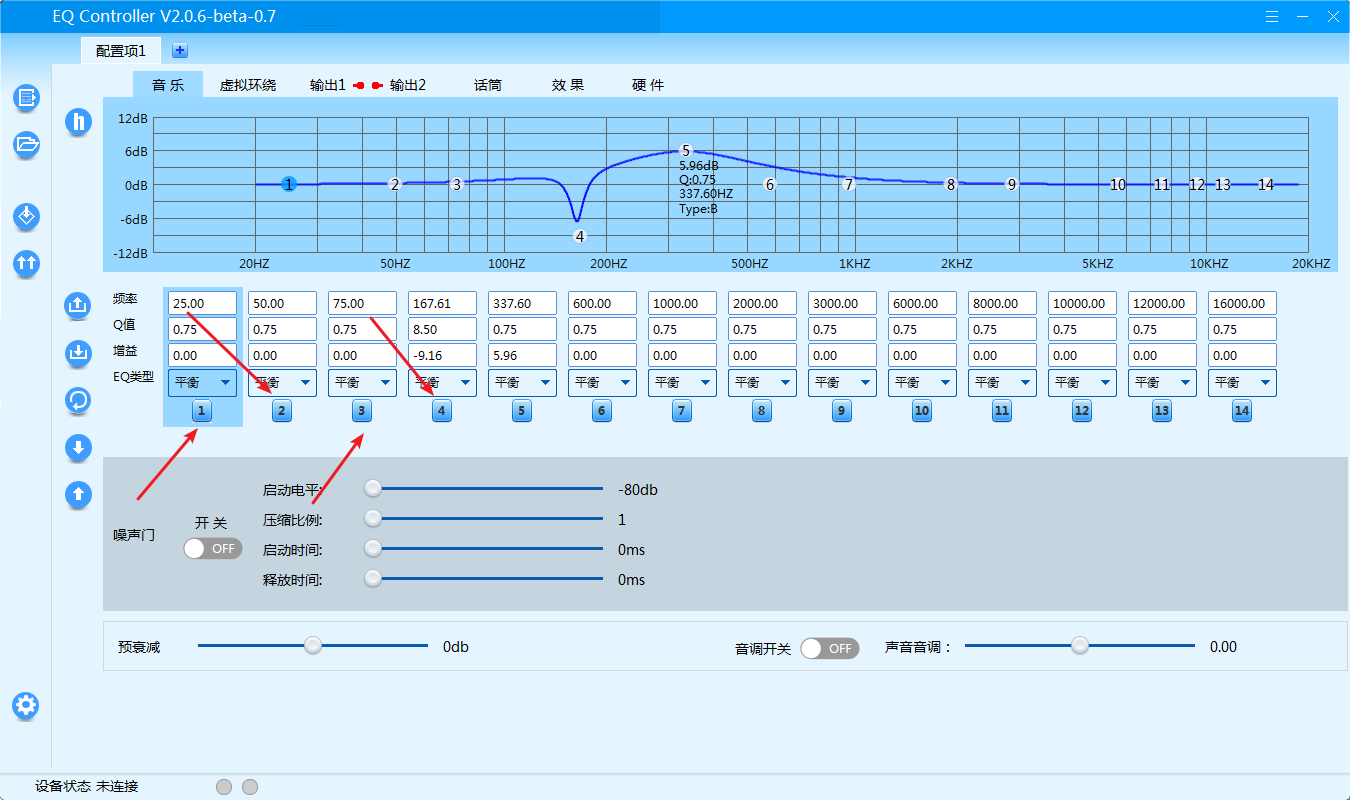
EQ类型目前共有六种，分别为低棚架，平衡，高棚架、高通带通、低通，可在EQ类型下拉菜单中选择，默认为“平衡”，如图21-1-2所示



**图21-1-2**

### EQ使能

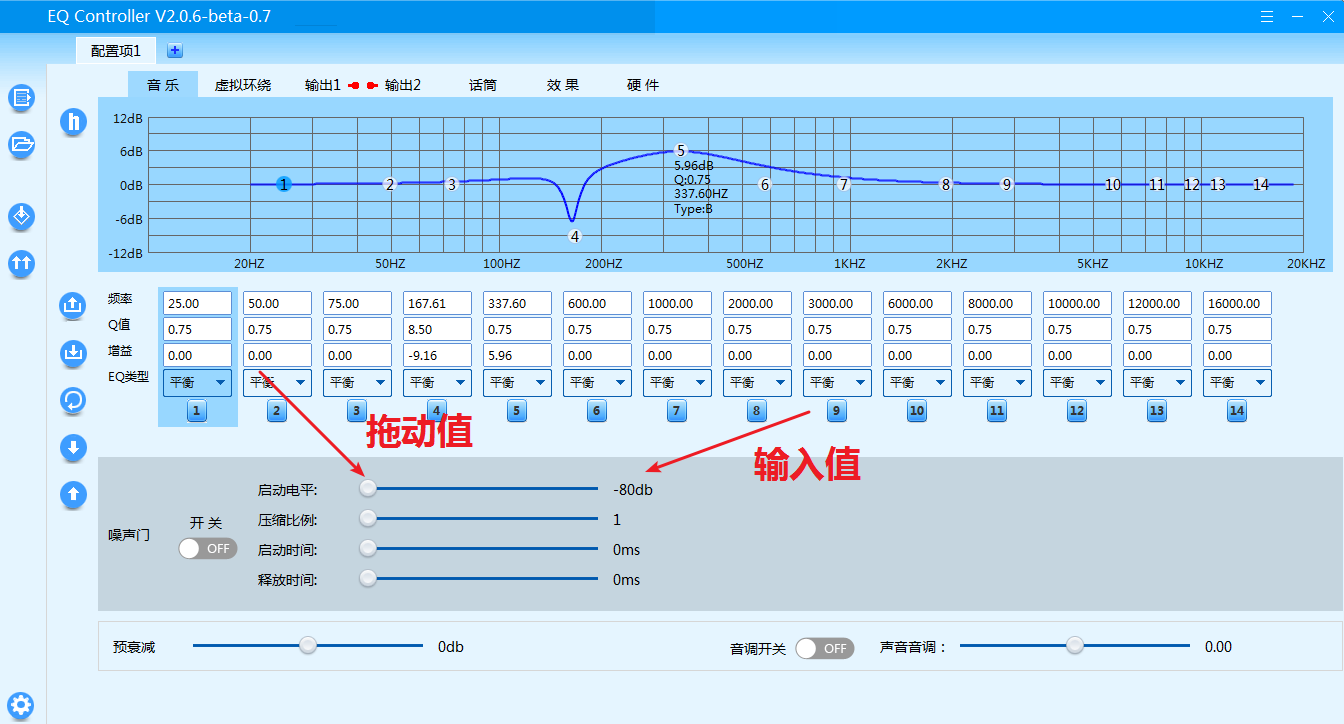
每一路EQ均有各自的使能按钮，如图21-1-3所示，可以相应的禁止或者打开该路EQ调节功能。



**图21-1-3**

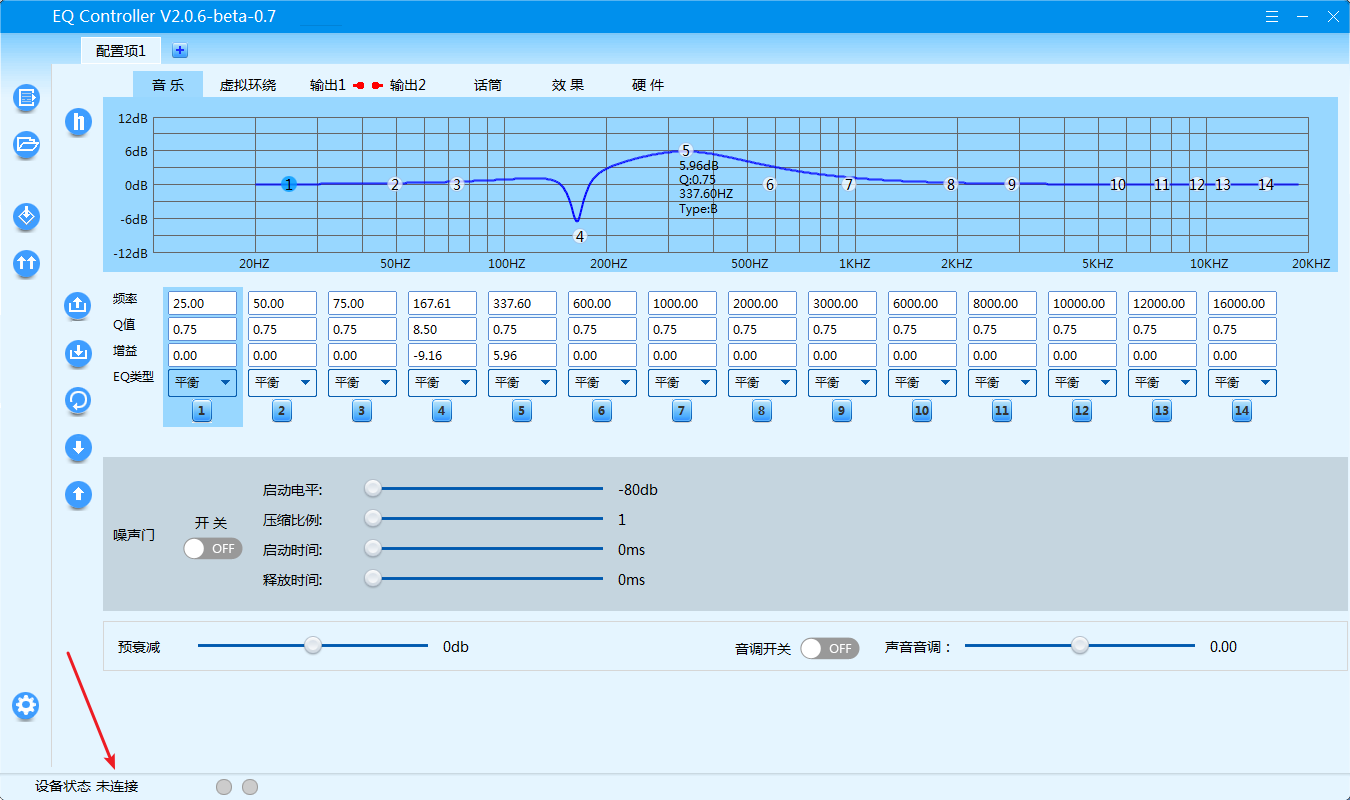
### 调节其它效果参数

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。



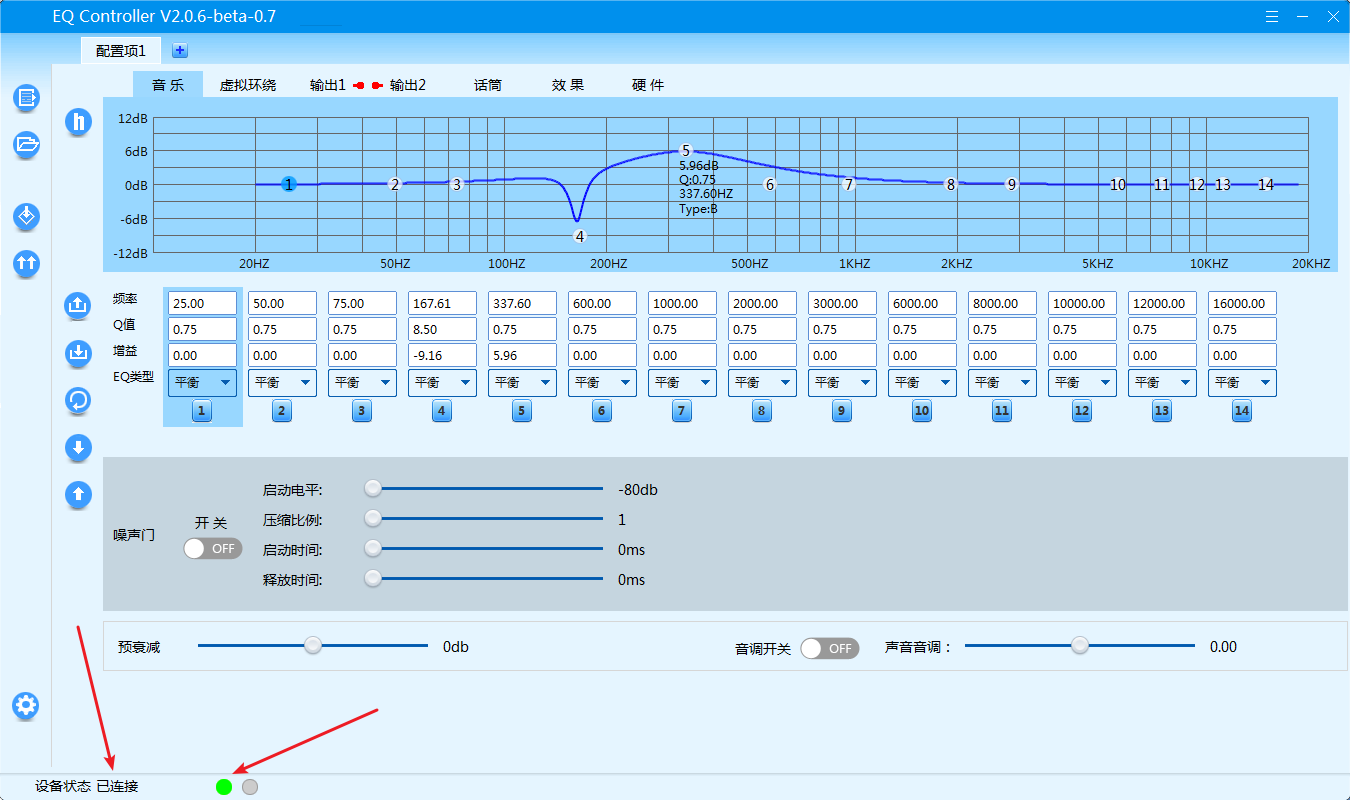
**图21-1-4**

### 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**

### 错误隐患

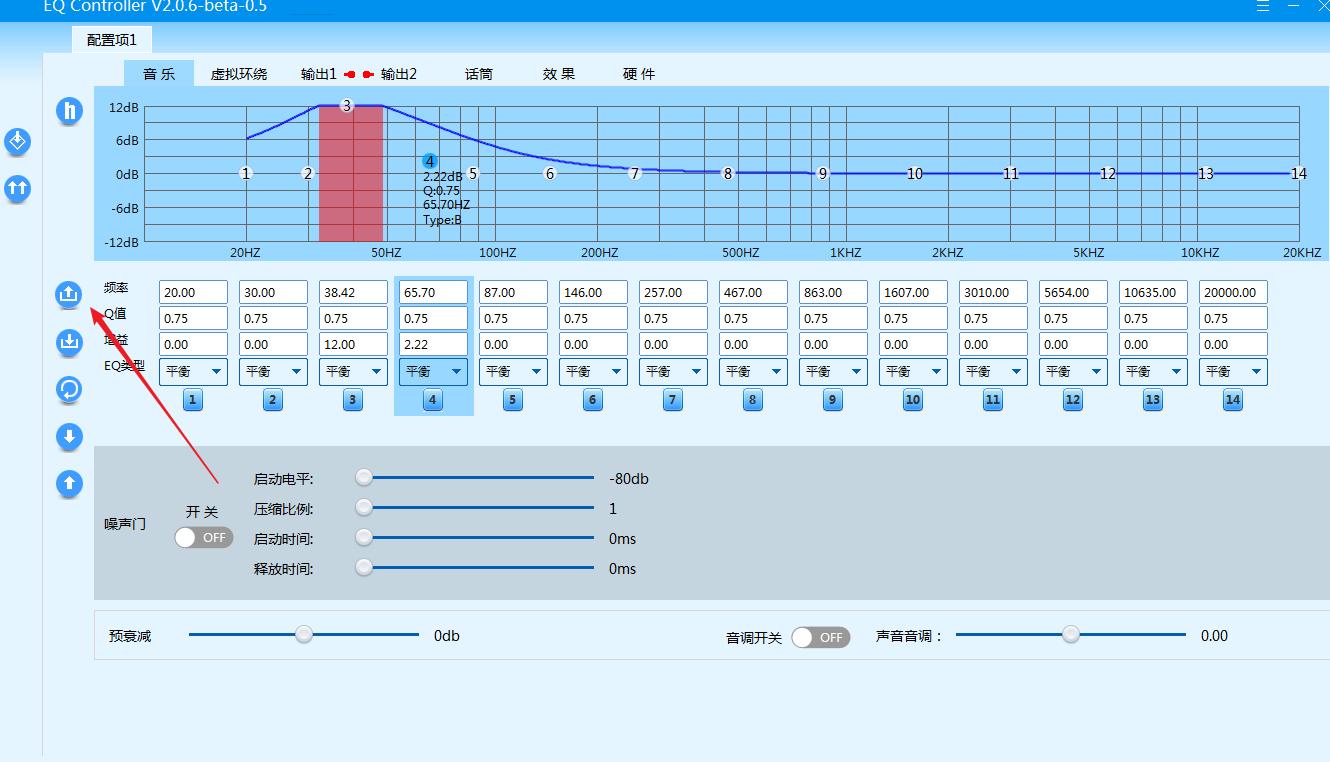
当设置的EQ参数导致EQ曲线出现削峰时，EQ插件会出现红色半透明区域，表明大致出现削峰的范围，需引起注意，如图21-1-6所示

****

**图21-1-6**

### 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**

### 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**

### 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**

### 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**

### 提取设备EQ

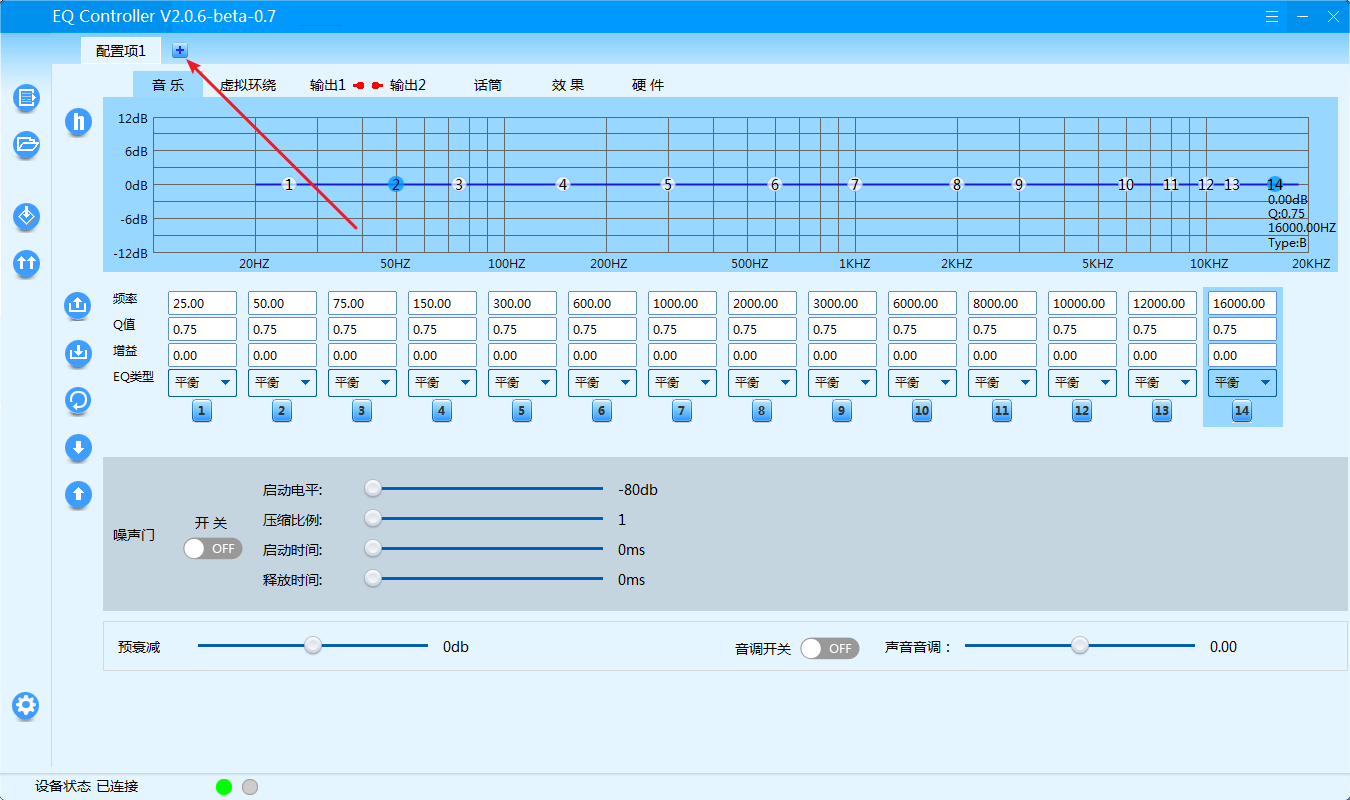
将设备中运行的EQ参数刷新到界面中



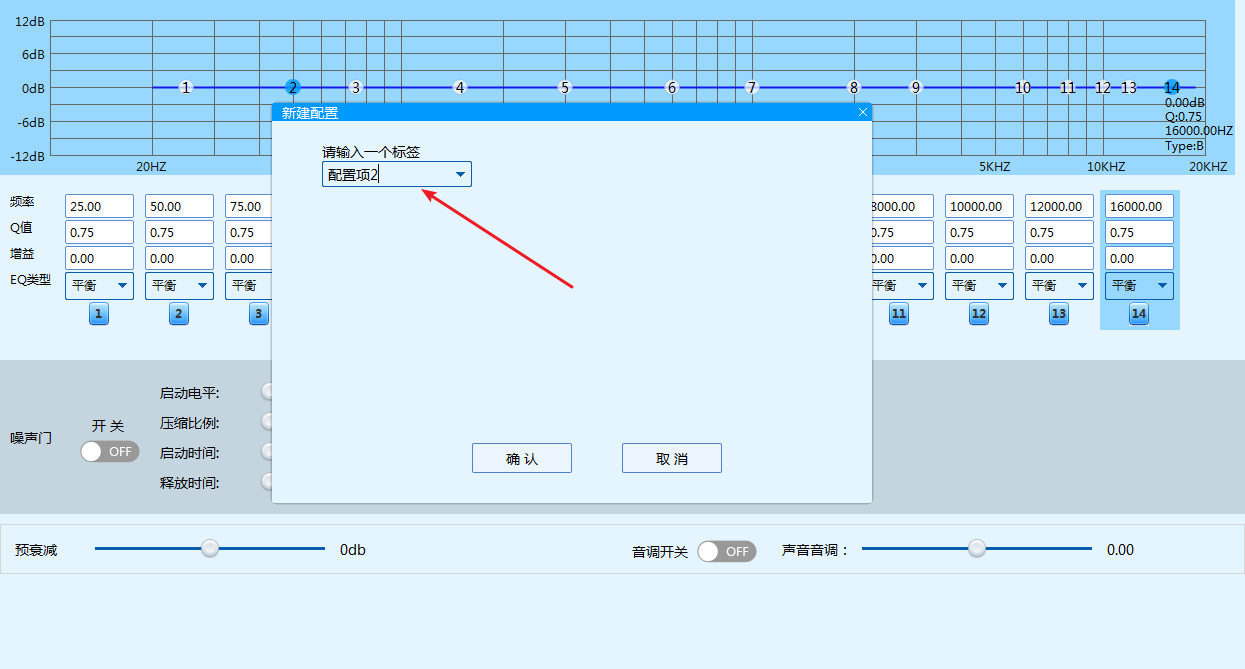
**图21-1-11**

### 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

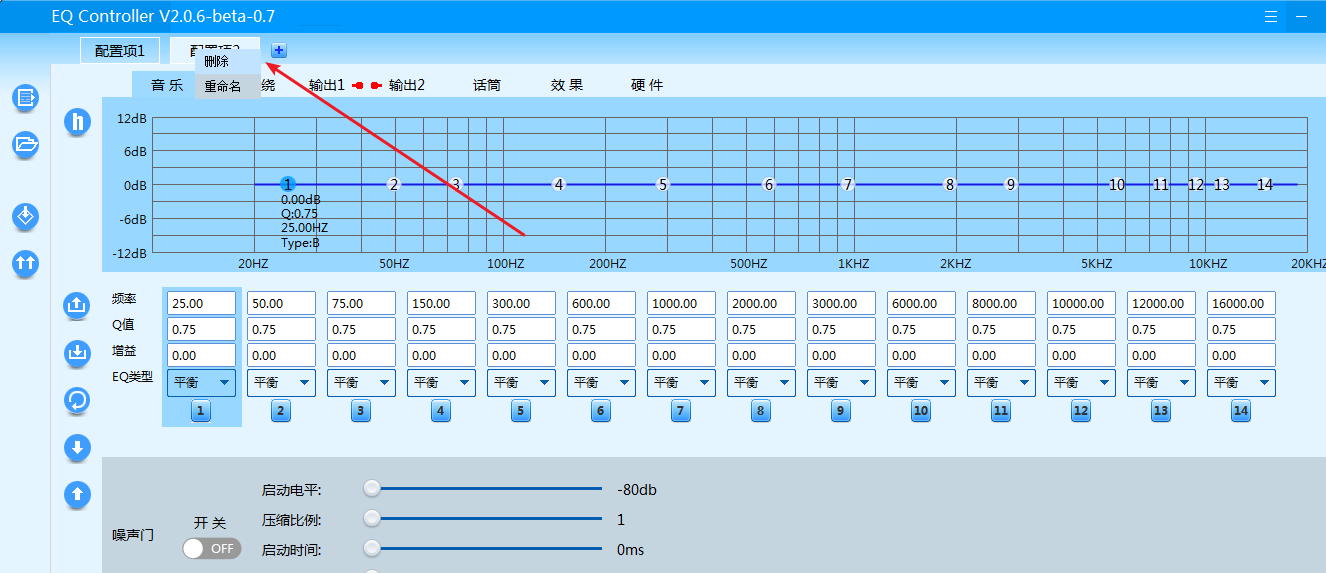


**图21-1-12-1**



**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**

### 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**

### 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**

### 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**

### 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**

### 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。

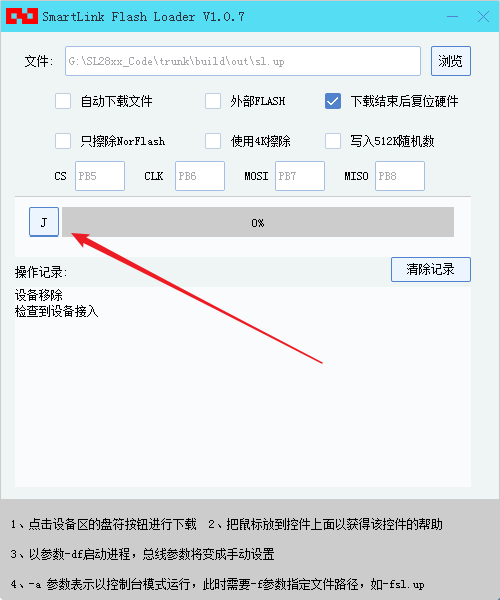


图21-1-17

## Flash\_loader工具

### 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**

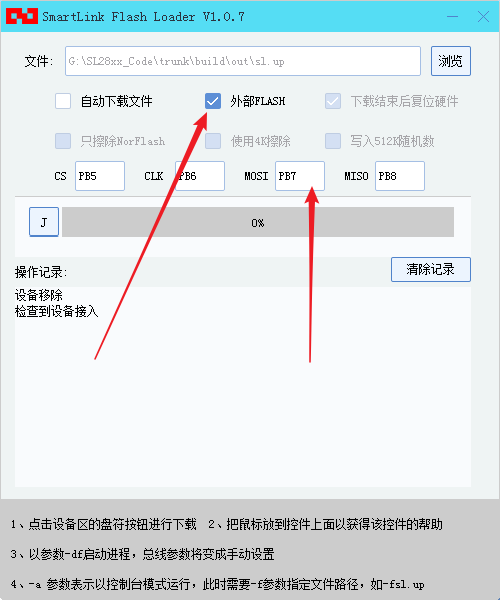
### 更新外部 FLASH

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**

### 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

## 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

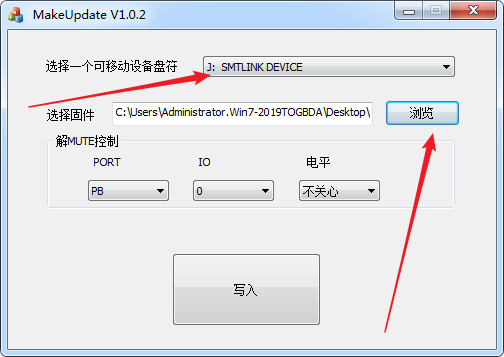
## 升级软件

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

## 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。

# 音效软件API

## 基本API说明

### 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);**

### 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);**

### 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);**

### 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);**

### 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);**

### 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);**

### 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);**

### 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);**

### 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);**

### 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);**

### 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);**

### 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);**

### 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);**

### 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);**

## 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) |