珠海市杰理科技有限公司
ZHUHAI JIELI TECHNOLOGY CO.,LTD

AC692X_SDK_介绍

AC692X_SDK_介绍 用户手册

Rev 1.5 —— 2018 年 9 月 05 日

目录

修改日志

版本	日期	描述
1.5	2018 / 9 /05	增加 BLE 透传功能说明
1.4	2018 / 8 /22	增加录音使用说明
1.3	2018 / 8 /11	增加串口升级协议说明
1.2	2018 / 5 /18	增加混响章节和 TWS 章节
1.1	2018 / 4 /08	添加 VM 使用的注意项
1.0	2018 / 3 / 30	AC692x 用户手册
更新：	● 建立初始版本 ● 定义文档格	

Chapter 1 SDK 开发包快速使用说明

1.1 编写目的

该文档主要描述 AC692x_SDK 开发包的使用方法及开发中注意的一些问题，为用户进行二次开发提供参考,其中包括：

开发环境搭建，安装 IDE 步骤，认证方式等开发前准备介绍；

目录结构，工程的结构介绍，用户允许修改的文件和用户不允许修改的文件；

1.2 IDE 开发工具的安装

❖ 编译环境安装

- 工具链安装

  开发 AC692x 必须重新安装工具链：

  1、codeblocks-16.01mingw-setup,跟 AC690x 的一样

  2、jl_toolchain_pi32v2_lto_2.1.6.exe

- 安装教程：

  jl_toolchain_pi32v2_lto_2.1.6

  工具需要注册，否则在链接的步骤出错，报”No Such File”

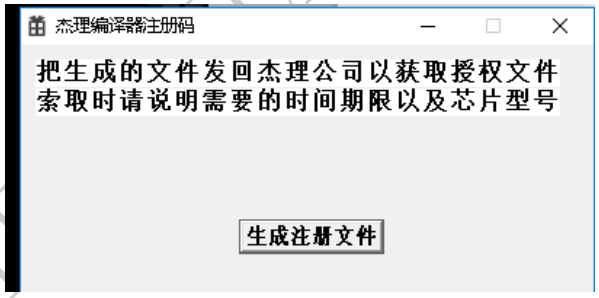
- 注册工具：

  ❖ 生成 key 文件

  为了注册首先需要生成一个 key 文件反馈给 杰理科技

  开始 --> 程序 --> JL toolchain --> 生成 License Key 文件 --> 《生成注册文件》

  然后保存 key 文件，并发给杰理科技，同时说明需要使用的项目（AC54，AC691X，AC692X 等）或后端（PI32，PI32V2，Q32S）



  ❖ 导入 lic 文件

  开始 --> 程序 --> JL toolchain --> 导入 License 文件 --> 《打开 License 文件》

  然后选择得到的 lic 文件

1.3AC692x_SDK 工程的打开

1、打开工程之前先要确保已经安装好杰理科技最新发布的 IDE 和工具链

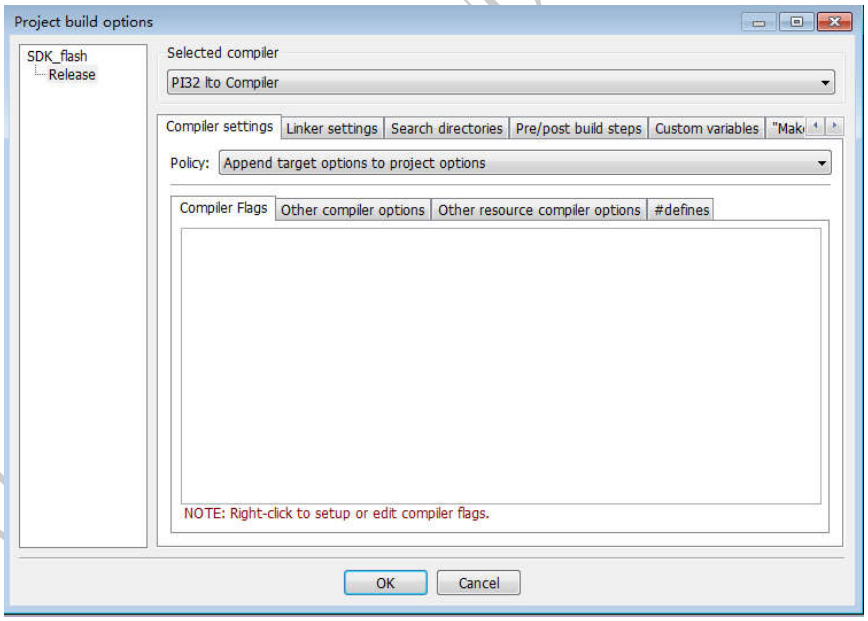
2、双击 apps/目录下的 ac69_app.cbp工程文件，或者把 ac69_app.cbp 文件拖动到 CodeBlocks 快捷方式下打开，AC692x SDK 工程将会被打开

注意 ：第一次安装好 CodeBlocks 如果编译不过，可能需要指定下 compiler (如果编译正常，则不需要指定)，方法：右击工程名-》build options-》Compiler settings，如下图所示：

1、打开工程之前先要确保已经安装好杰理科技最新发布的 IDE 和工具链，因为 AC692x 有一些新增的编译功能，所以一定要先安装（jl_toolchain_pi32v2_lto_2.1.6.exe）之后的编译器版本才能正常使用，否则 SDK 可以正常编译但会错误执行。

2、双击 apps/目录下的 sdk.cbp 工程文件，或者把 sdk.cbp 文件拖动到 CodeBlocks 快捷方式下打开，SDK 工程将会被打开

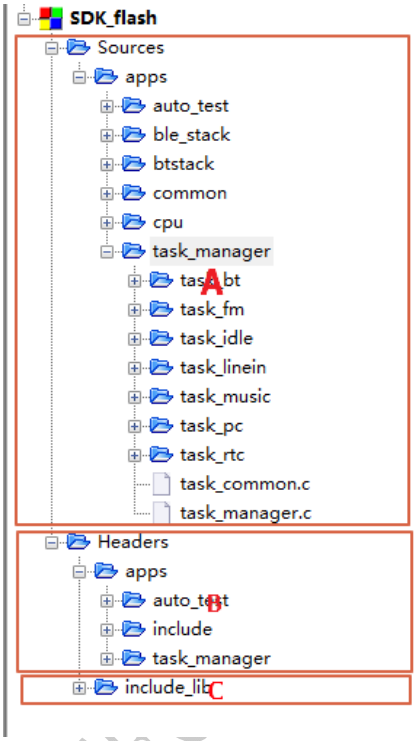
注意 ：如果第一次安装好 CodeBlocks 如果编译不过，可能需要指定下 compiler (如果编译正常，则不需要指定)，方法：右击工程名-》build options-》Compiler settings，如下图所示：



选好之后再重新编译即可。

1.4 工程目录介绍

工程打开后，工程目录如下所示：



截图 A 区：用户可看到，并且可修改的一些.c 文件，用户也可以自行添加自己的 c 文件

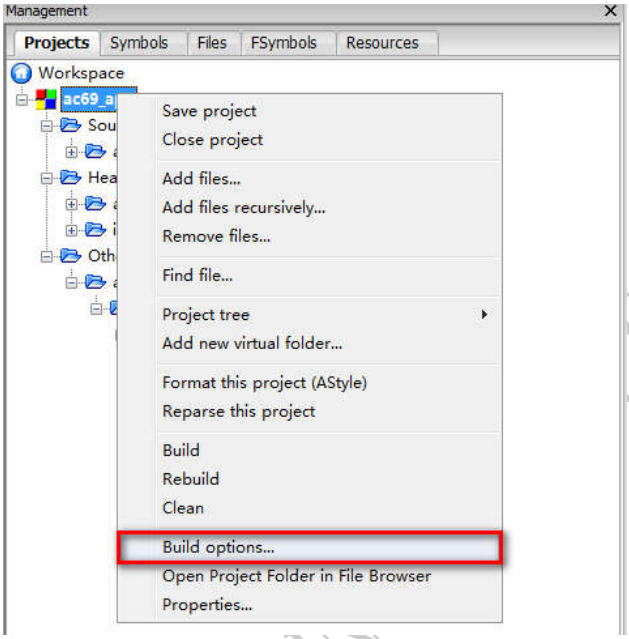
截图 B 区：apps 目录提供外面应用的头文件,用户可以随便修改

截图 C 区：include_lib 提供库用到的头文件，跟库的编译关系密切，客户尽量不要修改里面的内容

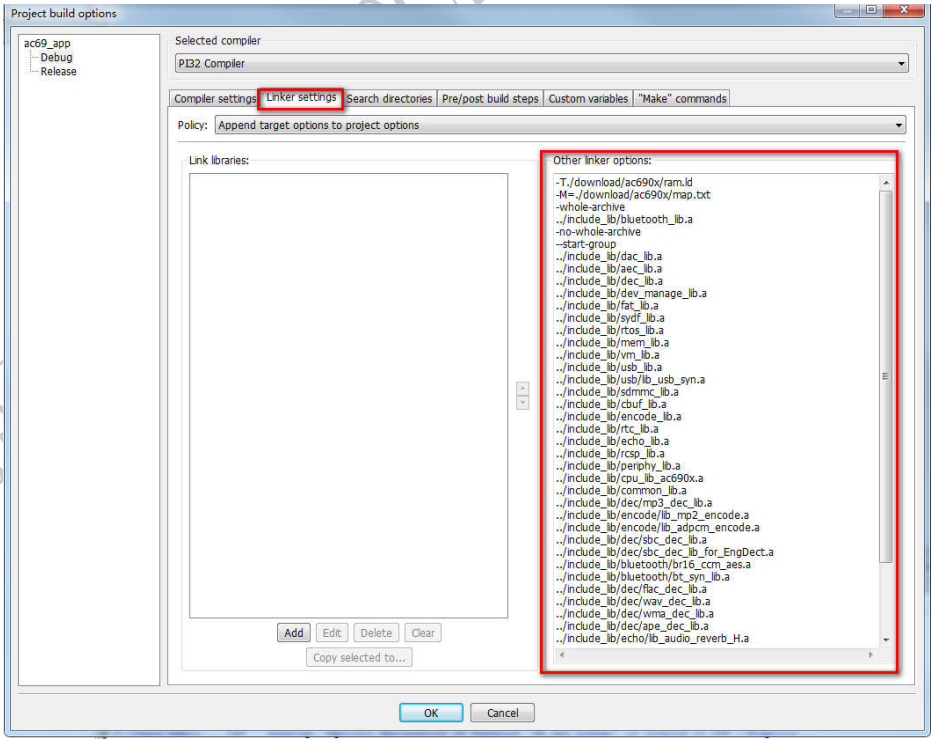
apps\download\post_build\flash 目录下 Map.txt 文件每次编译完成都会更新，内容主要是工程函数以及一些变量的映射关系，sdk.ld 主要表明 ram 的使用情况

1.5 一些常用的设置说明

工程选中右键设置如下图：



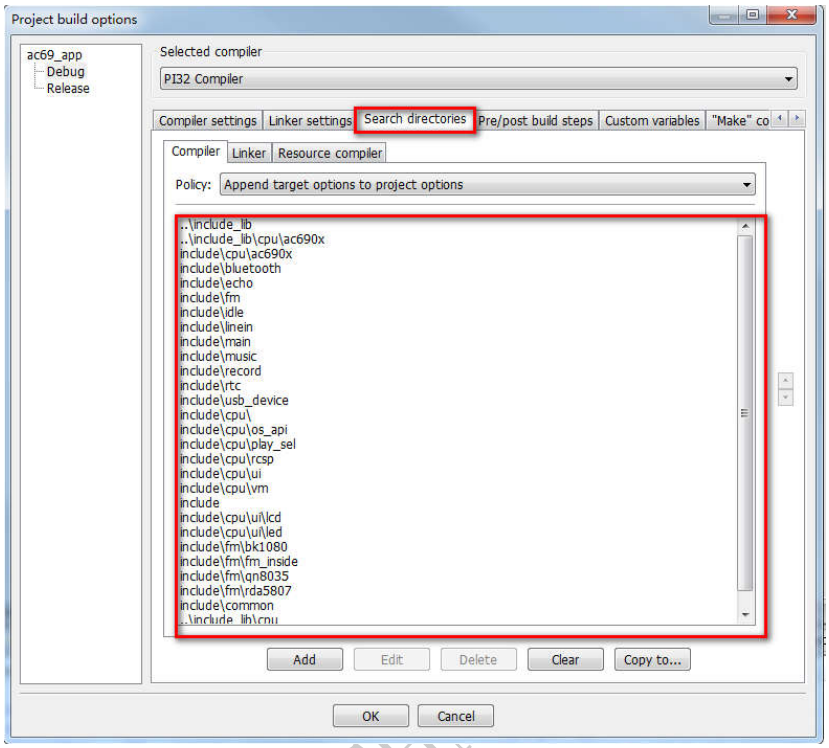
点击红色边框后将出现如下设置界面，选择 Linker settings 设置库文件：



说明：添加库文件时（更改库文件）都需要重新 rebuild，只修改.c 或者.h 文件执行 build 即可

右边红色框是所添加的库文件

选择 Search directories 设置路径：



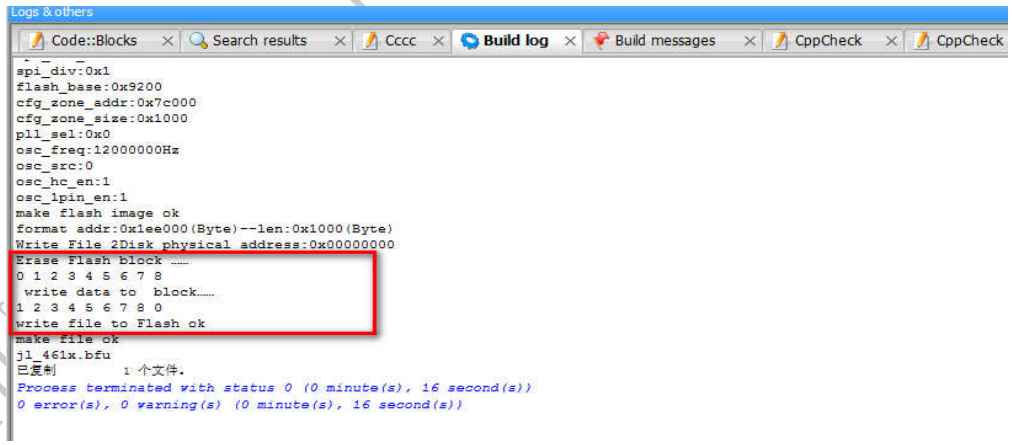
1.6 程序下载说明

download.bat 文件是一个下载批处理文件，对应配置，（A 部分是负责下载程序到 flash，B 是负责生生成烧写文件和升级文件（bfu）：

```
cd %~dp0


if exist uboot.boot del uboot.boot
type uboot.bin > uboot.boot

cd tone_resource
copy *.mp3 ..\
cd ..

isd_download.exe -tonorflash -dev br21 -boot 0x2000 -div6 -wait 300  -f uboot.boot sdk.app bt_cfg.bin  bt.mp3 music.mp3  linein.mp3 radio.mp3 pc.mp3 connect.mp3 disconnect.mp3 ring.mp3 warning.mp3 0.mp3 1.mp3 2.mp3 3.mp3 4.mp3 5.mp3 6.mp3 7.mp3 8.mp3 9.mp3


::-format cfg

:: -read flash_r.bin 0-2M

if exist *.mp3 del *.mp3
if exist *.PIX del *.PIX
if exist *.TAB del *.TAB
if exist *.res del *.res
if exist *.sty del *.sty
if exist jl_692x.bin del jl_692x.bin


rename jl_isd.bin jl_692x.bin
bfumake.exe -fi jl_692x.bin -ld 0x0000 -rd 0x0000 -fo updata.bfu


IF EXIST no_isd_file del jl_692x.bin
del no_isd_file

@rem format vm        //擦除VM 68K区域
@rem format cfg       //擦除BT CFG 4K区域
@rem format 0x3f0-2  //表示从第 0x3f0 个 sector 开始连续擦除 2 个 sector(第一个参数为16进制或10进制都可，第二个参数必须是10进制)

ping /n 2 127.1>null
::pause
```

A B C

最终显示 write file to Flash ok 表示下载成功



注意：编译器编译完成直接下载，下载之前需要复位 flash 上电（在开发板按着 flash 复位键上电，或者是按着 flash 复位键再按下 Reset 键进行复位），在 pc 上弹出盘符方可下载，下载完成后请断电重启启动

Chapter2 升级说明

2.1 下载升级说明

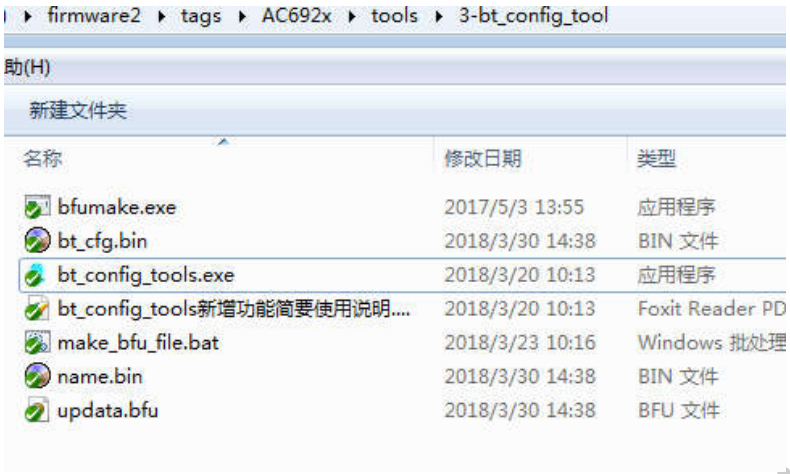
692x 系列下载更新下载程序，包括：PC 在线升级、U 盘/TF 卡升级、PC 强制升级、测试盒蓝牙无线升级。**[注]flash 分区：代码区、配置区、vm 区。**

- 生成升级的代码的升级文件 updata.bfu。如下图 AC692x\flash\download.bat,生成待升级文件 updata.bfu，此升级文件可以进行更新相关代码区。

```
isd_download.exe -tonorflash -dev br21 -boot 0x2000 -div6 -wait 300  -

::-format cfg

:: -read flash_r.bin 0-2M

if exist *.mp3 del *.mp3
if exist *.PIX del *.PIX
if exist *.TAB del *.TAB
if exist *.res del *.res
if exist *.sty del *.sty
if exist jl_692x.bin del jl_692x.bin


rename jl_isd.bin jl_692x.bin
ufumake.exe -fi jl_692x.bin -ld 0x0000 -rd 0x0000 -fo updata.bfu
```

烧写 bin 文件和升级 updata.bfu 文件都**必须**在样机或者开发板没有连接电脑的情况下生成，即**离线生成**！

- 生成单独升级蓝牙名字的 updata.bfu。如下图通过工具 bt_config_tool.exe，修改需要升级的蓝牙名字，保存。然后双击批处理 make_bfu_file.bat，最后生成升级蓝牙名字的升级文件 updata.bfu。（此升级文件升级时通过擦除 flash 配置区，然后把相关信息写入配置区）



- 升级需要擦除 flash 配置区和 VM。例如擦除 flash 配置区，需要更新配置信息,修改 AC692x\flash\isd_tools.cfg 如下图 1 所示。最后生成的 updata.bfu 会带有擦除 flash 配置区信息，升级时进行擦除。

```
########flash空间使用配置区域#############################################################
#PDCTNAME:    产品名，对应此代码，用于标识产品，升级时可以选择匹配产品名
#BOOT_FIRST:  1=代码更新后，提示APP是第一次启动；0=代码更新后，不提示
#UPVR_CTL:    0：不允许高版本升级低版本   1：允许高版本升级低版本
#XXXX_ADR:    区域起始地址        AUTO：由工具自动分配起始地址
#XXXX_LEN:    区域长度            CODE_LEN：代码长度
#XXXX_OPT:    区域操作属性
#操作符说明  OPT:
#       0:  下载代码时擦除指定区域
#       1:  下载代码时不操作指定区域
#       2:  下载代码时给指定区域加上保护
#############################################################################################

SPECIAL_AREA_START;
{
PDCTNAME=jl_692X;
BOOT_FIRST=1;
UPVR_CTL=0;
PRCT_ADR=0;
PRCT_LEN=CODE_LEN;
PRCT_OPT=2;
BTIF_ADR=AUTO;
BTIF_LEN=0x1000;
BTIF_OPT=0;
VMIF_ADR=AUTO;
VMIF_LEN=0x10000;
VMIF_OPT=1;
}
```

#操作符说明 OPT:

0: 下载代码时擦除指定区域

1: 下载代码时不操作指定区域

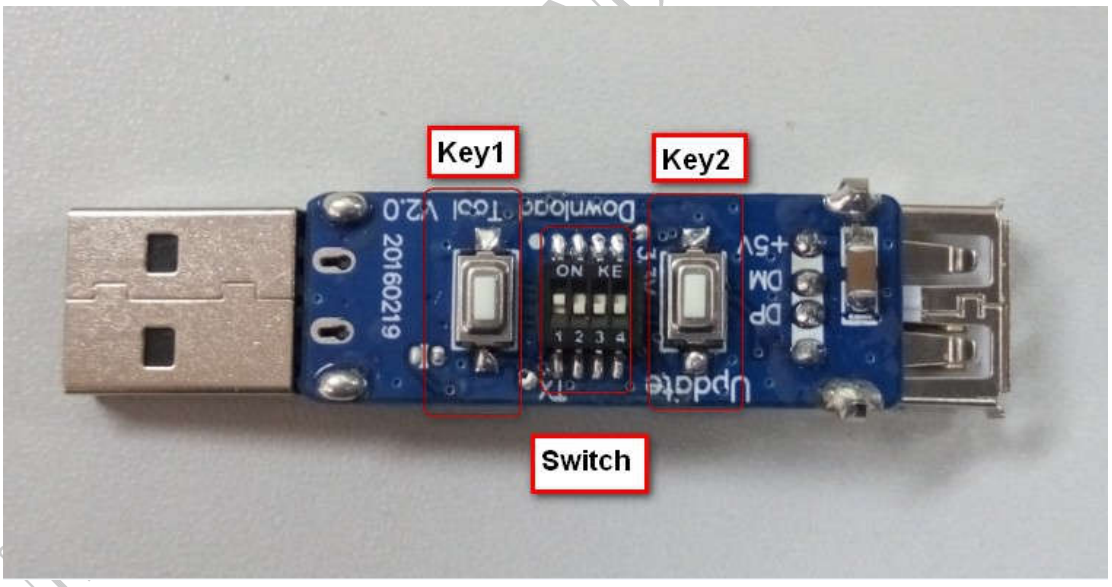
2: 下载代码时给指定区域加上保护

❖ PC 强制升级

- 使用我司定制的 USB 工具，按下“update”按钮，蓝灯亮，样机断电情况下，插入 USB 升级工具，蓝灯灭，即可进入升级模式，点击批处理 AC692x\flash\download.bat 下载。该升级模式是主要用在芯片跑不起来或者烧错程序的情况下使用！
- 升级工具使用 usb 接口，正常使用时候，母口连接电脑，公口连接样机的 usb 口。V2.0 版添加拨码开关，以便后续升级和兼容旧版本。

拨码开关	工作模式	说明
0-0-0-0	AC460X	用于升级 AC460X 系列芯片
1-0-0-0	AC69XX_USB 从机升级	用于升级 AC9XX 系列芯片
Other	保留	保留

如下图所示：



升级工具

- 本工具连接 USB 线后，默认为正常模式，不作任何操作，相当于一个 USB 延长线。
- 升级步骤说明

1、先把样机完全断电。

2、把本工具母口用过 USB 线连接电脑，等待 led1 亮起。

3、按一下 key2，升级指示灯 led2 常亮后（如果按一下 key2 后，led2 不亮，请检查硬件或者

断电后重试步骤 2）

4、把升级工具公口插入样机的 usb 中，把样机开机， 样机已经进入升级模式。 （如果样机正常开机而不能进入升级模式，请重试步骤 1）

5、成功进入升级模式后，重新编译代码后即可自动升级下载到样机。

❖ PC 在线升级

● 首先样机程序需要有 PC 从机功能，然后开机情况下，连接 PC，进入 PC 升级模式。点击批处理下载 AC692x\flasht\download.bat 。升级需要擦除 flash 配置区和 VM 参考上面描述进行配置。

❖ U 盘/TF 卡升级

● 首先样机程序需要有插 U 盘功能或者 TF 卡功能。然后开机的情况下，进入音乐模式，检测到 u 盘或者 TF 卡有 updata.bfu 文件进入升级模式。升级需要擦除 flash 配置区和 VM 参考上面描述进行配置。

❖ 测试盒蓝牙无线升级

● 首先样机程序需要有蓝牙模式进行可连接状态，然后开机情况下。用带有升级文件 updata.bfu 的 tf 卡插入到杰理一拖二测试盒，然后用测试盒进行连接样机，进入升级模式升级传输（测试盒需要更新到 AC690x_1T2 测试盒 V1.0.9 才支持打开 updata.bfu 进行升级）。详细操作可以参考（AC690x_1T2 测试盒使用说明.pdf）

U 盘/TF 卡/蓝牙无线升级 升级完成的处理结果用户可以修改 sdk 的 app/cpu/updata.c,如下图所示，升级完成会灯亮，同时会有按键音叮叮响。如果升级出错，会有报警声响。用户开发时记得要测试相关升级功能，防止批量生产时程序出错无法挽救。

```
void update_result_deal()
{
    u8 key_voice_cnt = 0;
    u16 result = 0;
    result = (g_updata_flag & 0xffff);
    printf("<---------update_result_deal=0x%x--------->\n", result);
    if (result == UPDATA_NON) {
        return;
    }
#ifdef UPDATE_VOICE_REMIND
    set_sys_vol(30, 30, FADE_ON);
#endif
#ifdef UPDATE_LED_REMIND
    if (result == UPDATA_SUCC) {
        led_update_finish();
    }
#endif
    while (1) {
        clear_wdt();
        key_voice_cnt++;
#ifdef UPDATE_VOICE_REMIND
        if (result == UPDATA_SUCC) {
            puts("<<<<<<UPDATA_SUCC");
            sin_tone_play(500);
            delay_2ms(500);
            puts(">>>>>>>>>>>\n");
        } else {
            printf("!!!!!!!!!!!!!!!updata waring !!!!!!!!!!=0x%x\n", result);
            sin_tone_play(1000);
            delay_2ms(500);
        }
#endif
        if (key_voice_cnt > 5) {
            key_voice_cnt = 0;
            delay_2ms(500);
            puts("enter_sys_soft_poweroff\n");
            enter_sys_soft_poweroff();
        }
    }
}
```

Chapter3 VM 使用说明

3.1 VM 概述

❖ **概述：**

VM(virtual memory)，是一套虚拟的记忆系统，主要用于保存信息到 flash 设备，由于记忆设备读次数有限，该系统能做到记忆设备平衡磨损，延长设备寿命。

VM 基本使用主要是分为以下四步：

- 初始化 VM：任何 VM 操作，都是必须保证 VM 已经初始化。
- 申请 VM：申请 VM 后获取 VM 句柄，该句柄提供 VM 读写使用
- 读 VM：通过 VM 句柄进行操作。
- 写 VM: 通过 VM 句柄进行操作。

VM 最小总容量为 8K，最大总容量为 128K。通过修改 isd_tools.cfg 文件配置。

3.2 VM 的基本使用

❖ **申请 VM：**

函数原型	void vm_open_all(void)
功能描述	申请 data_len 长度的 vm 存储空间
参数说明	index:申请 VM 区的序号 data_len:申请 VM 区的大小
返回	操作所申请 VM 区的句柄

```
void vm_open_all(void);
```

使用 VM 存储，必须先进行 open 操作，使用 vm_opoen 或者 vm_open_all，需要注意的是，新版 VM 在打开时候，不需要传长度参数。

```
enum {
    VM_REMOTE_DB = 1,
    VM_REMOTE_DB_END = (VM_REMOTE_DB + 20),

    //vm_start index
    VM_SYS_VOL,
    VM_SYS_EQ,
    VM_DEV0_BREAKPOINT,
    VM_DEV1_BREAKPOINT,
    VM_DEV2_BREAKPOINT,
    VM_DEV3_BREAKPOINT,
    VM_MUSIC_DEVICE,
```

```
    VM_PC_VOL,
    VM_FM_INFO,
    VM_PHONE_VOL,
    VM_BT_STEREO_INFO,
    VM_BT_OSC_INT_R,
    VM_BT_OSC_INT_L,

    //------请在次分界线下添加新的 VM 项-----//
    VM_TEST,

    VM_MAX_INDEX,
};
```

❖ **VM 读：**

函数原型	vm_err vm_read(u8 index, void * data_buf, u16 len)
功能描述	读 VM
参数说明	index:申请 vm 时候，对应的 index 枚举变量表 *data_buf:读 buff 指针
返回	len：成功 ret < 0：出错

用法举例:

```
err = vm_read(vm_hdl_vol,&sys_info_var.vol, 1);
```

❖ **VM 写：**

函数原型	vm_err vm_write(u8 index ,const void *data_buf, u16 len)
功能描述	写 VM
参数说明	index:申请 vm 时候，对应的 index 枚举变量表 *data_buf:要写的数据指针
返回	len：成功 ret < 0：出错

用法举例:

```
err = vm_write(vm_hdl_vol,&sys_info_var.vol);
```

关于 AC692X 的 VM 读写操作注意：

此版本 VM 读写需要提供长度参数。每一项 VM 内部仅存储最后一个有效数据。

读写长度大于内部数据长度，仅返回有效长度。读写取长度小于内部存取长度，返回读写长度

VM 与 DAC：

VM 在经行读/碎片整理期间，DAC 仍可继续工作

vm_init_api 函数中，参数为 0：vm 工作期间**不允许** DAC 工作，参数为 1：vm 工作期间**允许** DAC 工作

要注意的是,如果允许 DAC 工作,dac_isr_cb 里调用得所有函数或者常数都需要放在 audio_text 段中（使用 AT_AUDIO 定义即可）

Chapter4 AUX 模式

4.1 总体设计

❖ 系统：

本说明主要是基于 SDK 开发包来实现 AUX 的功能。

AUX 应用主要实现的功能包括：

- aux 输入通道选择
- aux_adc 转换

❖ 总体架构设计：

AUX 任务主要分为两个个功能模块：

①输入通道选择：aux 输入有 amux0，amux1，amux2 以及 dac_amux。每个通道都可以只用通道的其中一路输入，适用于 IO 比较少的 IC 封装。其中 dac_amux 即为 dac 的两个输出声道，其中一个作为输入，另外一个作为输出的功能。

②aux_adc 转换：该功能为了实现对 aux 模拟输入的数字化转换。可以实现能量值采样，以及将 aux 输入做 ad 采样，然后再做数字音效处理。

❖ 应用的启动

aux 模式进入可通过以下 2 个方式：

①插入 aux 数据线，跳转进入 aux 模式，前提是打开了 aux 的插入检测功能

②通过模式按键切换，直接进入 aux 模式，前提是关闭了 aux 插入检测功能

❖ 应用的退出：

当按下 Mode 键、aux 数据线拔出或者其他任务需要被激活时候，主线程会强制退出 aux 模式，会释放 Aux 任务资源和关闭提示音

Chapter5 蓝牙认证说明

5.1 FCC 认证说明

- **fcc 认证配置介绍：**

  - 开启蓝牙测试模式的宏介绍在文件 Bluetooth_api.h 如下图 1 所示

```
#define NORMAL_MODE          0
#define TEST_BQB_MODE        1        ///<测试bqb认证
#define TEST_FCC_MODE        2        ///<测试fcc认证
#define TEST_FRE_OFF_MODE    3        ///<测试频偏(使用频谱分析仪-手提测试仪-中心频率默认2422M)
#define TEST_PERFOR_MODE     4        ///<指标性能测试(使用MT8852A仪器测试,测试芯片性能的时候使用)
#define TEST_BOX_MODE        5        ///<测试盒测试
```

图 1

  - 开启 FCC 认证需要在 sdk_cfg.h 把 BT_MODE 宏配置为 TEST_FCC_MODE,如下图所示。（注意开了该功能后进行 pc 在线升级升级要使用把 usb 口拉高开机的方法才能再升级，download.bat 要使用-erase 配置把 flash 全部擦除方式。）

```
#define BT_MODE     TEST_FCC_MODE
```

图 2

  - Sdk 的 fcc 认证默认使用 USB 端口（DP 为 TX \DM 为 RX）来做 uart 串口与电脑通信(配置在 uart.c 如下图 3 所示)，为了确保串口与电脑正常通信，在 sdk_cfg.h 需要把使用 usb 口的程序都屏蔽（如下图 4 所示），注意各芯片的封装，如果 usb 口与其他脚有绑在一起时要把该脚设置为高阻，否则可能会影响到串口与电脑通信。如果用户想要使用其他硬件串口的话，可以自己修改 uart.c 文件的 void fcc_uart_init()函数里相应的硬件设置。

```
void fcc_uart_init()
{
    u32 status;
    puts("-----fcc_uart_init\n");
    fcc_uart_handle=NULL;
    status=uart_module_open(&uart1_handle,UART1_HARDWARE_NAME);
    if(!status)
    {
        __uart_param fcc_param;
        memset(&fcc_param,0,sizeof(__uart_param));
        fcc_param.baud_rate=9600;
        fcc_param.io=  UART_USBP_USBM;
        fcc_param.workmode=UART_WORKMODE_NORMAL;
        fcc_param.custom |= (BIT(14)|BIT(3));
        status=uart_module_init(&uart1_handle,&fcc_param);
        uart_reg_isr_callback_fun(&uart1_handle,5,fcc_uart_isr_callback);
        if(status)
        {
            puts("uart_module_init err\n");
        }
        if(!status)
        {
            status=uart_module_start(&uart1_handle);
            if(!status)
                fcc_uart_handle=&uart1_handle;
        }

    }
}
```

图 3

```
*/
#ifdef MINI_BT
    #define SDMMC0_EN       1
    #define SDMMC1_EN       0
    #define USB_DISK_EN     0
    #define USB_PC_EN       0
#else
    #define SDMMC0_EN       1
    #define SDMMC1_EN       1
    #define USB_DISK_EN     0
    #define USB_PC_EN       0
#endif
```

图 4

- **FCC 认证 PC 工具 FCCAssist_1.5.exe 介绍：**

  - 根据使用电脑的系统类型，首先要打开 REG_WINDOWS_UART 文件夹，点击 register_uart_WIN7.bat 或 register_uart_WINXP.bat 批处理来按照库文件；
  - 打开软件 FCCAssist_1.5.exe，点击 serialport 选择下拉框，选择正确的串口端口，如果正常旁边的红色标志会变成绿色，如图 5 所示；
  - 在测试过程中按需要设置相应配置项，然后按 sendconfiguration 按键即可，如果发送成功如图左下角显示框表示发送成功，否则发送不成功（如图 5 所示）。

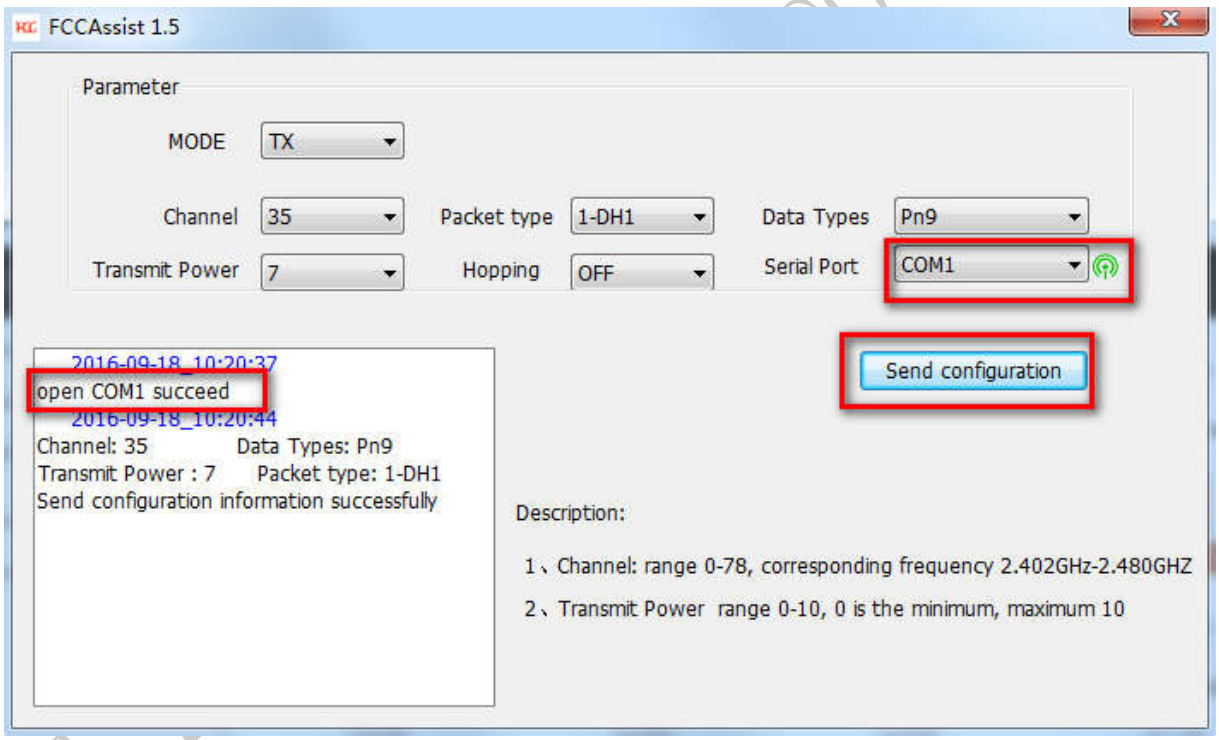


图 5

注意：Sdk 默认使用 usb 口做串口与电脑通信，如果该口与串口板链接死了，（系统在启动时会检测 usb 在线情况）会导致不能正常开机，因此要样机正常开机后再把 usb 口与串口板链接起来与电脑通信。

5.2 BQB

- **BQB 配置介绍**

❖ 开启 BQB 认证需要在 sdk_cfg.h 把 BT_MODE 宏配置为 TEST_BQB_MODE,如下图 6 所示。

```
#define BT_MODE    TEST_BQB_MODE
```

图 6

- **BQB 的 RF 测试**

❖ 开启 TEST_BQB_MODE 宏，样机就可以进入到 dut 测试，上电后手机可以搜索到所配置的蓝牙名字，手机是连接不上的，只为 BQB 测试连接。
❖ 样机的天线端要焊屏蔽线，而天线端的料可以先不用改。
❖ 去到实验室把样机和测试仪器链接好屏蔽线，上电后链接测试即可。
❖ 有一个设置项中心频率为 2M。
❖ 测试 RF 的时候可能有些项目要特殊处理才可以过，尽量先把全部测试项都过一次，最后拿报告看看那些项有问题，再看要怎么处理。

- **BQB 的 profile 测试**

❖ 开启 NORMAL_MODE 模式，样机天线不用焊屏蔽线;
❖ 根据客户的样机功能选择好测试项，该测试主要是测试功能如上下曲，暂停，接听，挂断，回链等。

Chapter6 蓝牙开发使用说明

6.1 术语和缩写词

提示：列出本文件中用到专门术语的定义和外文首字母组词的原词组。

缩写和术语	解 释
AC692x	杰理科技 AC692x 系列芯片
1 拖 1	蓝牙仅支持一个手机连接

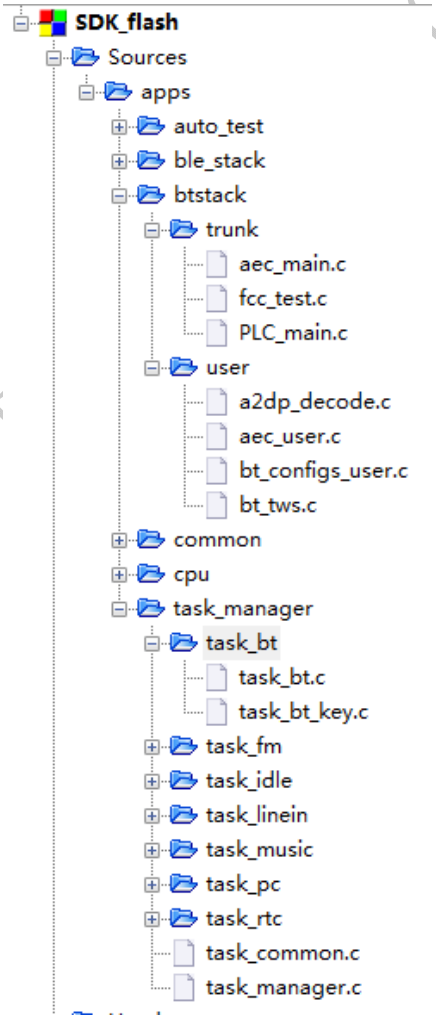
6.2 开发说明

❖ AC692X 工程打开：

- 打开工程之前先要确保已经安装好杰理科技最新发布的 jl_toolchain 和 CodeBlocks。
- 双击 apps 目录下的 sdk.cbp 文件，或者把 sdk.cbp 文件拖动到 CodeBlocks 快捷方式下打开，工程将会被打开。

❖ 工程目录介绍：

❖ 工程打开后，工程目录如下所示



Sources:用户可以修改的一些.c 文件，用户也可以自行添加自己的 c 文件

Headers：用来放头文件，apps 下的头文件用户可以修改，但是 include_lib 下的头文件是库对应的头文件。用户要注意不要随意修改，若要修改库一并更新头文件。

❖ 蓝牙部分的主要 C 文件

1、文件 aec_main.c

通话回声消除的一下流程代码，包括初始化，运算模块的添加

2、文件 fcc_test.c

FCC 认证模式的参数设置

3、文件 PLC_main.c

通话的丢包修复模块代码，一般情况都不需要改

4、文件 a2dp_decode.c

蓝牙音乐的一些解码流程，主要是 open 和 close 库外面的音乐模块。一般情况都不需要改

5、文件 aec_user.c

主要是用于设置回声消除参数，调回声消除就看这个文件吧

6、文件 bt_configs_user.c

主要是用于蓝牙的一些参数设置和回调函数注册，这个会经常修改的文件

协议选择，下面这个几个宏在 bt_configs_user.c,配置这几个宏可以选择蓝牙支持的协议。

```
///---sdp service record profile- 用户选择支持协议--///
#define USER_SUPPORT_PROFILE_SPP       1
#define USER_SUPPORT_PROFILE_HFP       1
#define USER_SUPPORT_PROFILE_A2DP      1
#define USER_SUPPORT_PROFILE_AVCTP     1
#define USER_SUPPORT_PROFILE_HID       0
```

bt_configs_user.c 内主要函数说明：

◎函数 static void bt_setup_init(u8 *adr, char *name, u8 idx, char *pin_code);是实现读取经典蓝牙配置文件的流程。

◎函数 static void bt_function_select_init();是一些参数配置的函数，此函数集中了大部分供二次开发设置的函数。有时候补丁新增一些新配置也应该放在这个函数里面。

◎函数 static void ble_config_select_init(void);是实现读取 BLE 蓝牙配置文件的流程。

◎函数 static void bredr_handle_register()集中了注册一些蓝牙库的回调函数。这些注册进去的函数库里面会在相应的流程使用。

◎函数 void bt_mode_init()就是初始化蓝牙的硬件和蓝牙的协议栈。有些会有顺序要求，不要随意调整初始化位置。

7、文件 bt_tws.c

对箱用到的一个流程处理文件，有些 SDK 版本不支持对箱

8、文件 task_bt.c

流程处理，消息处理，这个会经常修改的文件

主要函数说明：

◎函数 void bt_work_state_control(u8 enble);控制蓝牙的可发现可连接。

◎函数 void hook_hfp_incoming_phone_number(char *number, u16 length);如果有电话进来，库里面会调用这个回调函数给上层反馈电话号码。

◎函数 int btstack_status_update_deal(u8 *info, u16 len)，这个函数属于库里面的一个回调函数，蓝牙协议栈的流程直接调用，注意不要在这个函数里面直接加太多 delay 或者等待操作，会影响这个蓝牙命令收数的解析，建议判断到某些状态后，复杂的流程推消息到 void task_bt_deal(void *hdl)这个主循环的函数处理。其它回调函数也一样要注意

◎函数 void bt_discon_complete_handle(u8 *addr, int reason)跟以前一样是处理蓝牙连接上和断开的一些消息。

◎函数 static void *task_bt_init(void *priv)模式进入会调用。

◎函数 static void task_bt_exit(void **hdl)模式退出会调用

◎函数 void task_bt_deal(void *hdl)主循环函数，处理按键消息或其它情况产生的消息。

注意该文件的有一些说是后台的管理函数，留以后拓展使用，AC692x 不支持后台。

9、文件 task_bt_key.c

按键消息表

10、文件 bt_ui.c

有屏版本的显示处理

❖ 蓝牙最重要的头文件----avctp_user.h

这个头文件定义了很多蓝牙库的状态和蓝牙支持的命令。需要开发个性化蓝牙功能时多查询改文件的命令接口。很多情况下更新库要注意补丁里面有没有这个文件要更新。

Chapter7 音乐开发使用说明

7.1 总体设计

❖ 功能概述：

Music 任务实现的功能如下：

- 支持 MP3、WMA、WAV、FLAC、APE 文件的播放。
- 支持播放 SD 卡、U 盘、FLASH。
- 支持单通道解码。
- 支持歌曲播放时间、总时间、歌曲名等的获取。
- 支持歌曲断点播放。
- 支持快进、快退、复读、暂停、继续播放。
- 支持变采样、声道控制、EQ/频谱、音量/自动 mute 等音效。
- 支持多个音效同时打开。

7.2 总体架构设计

❖ 音乐模式整体软件流程图如图 1.2.1 所示：

- 音乐播放相关初始化主要包括文件播放模式，设备选择模式，可支持的文件后缀等的设置，解码错误处理等函数的注册。
- 解码前会对要进行操作的设备进行在线检查，并对要播放的文件进行文件格式等的检查，若设备不在线或设备中不存在文件、文件损坏、文件格式不支持等，都会进行对应的错误处理。
- 与解码相关的函数调用是在软中断的中断服务函数里面进行，通过注册软中断，配合 DAC 那边的数据输出情况，不定时的将软中断的中断标志置位，在软中断中断服务函数中进行解码与输出操作，并对解码过程中产生的错误进行处理。
- 因为解码等操作是在软中断服务函数中进行，所以在播放过程中支持上下曲、暂停\播放、插拔设备等响应。
- 本系统支持多个音效同时打开，解码后的数据通过音效处理叠加后输出。

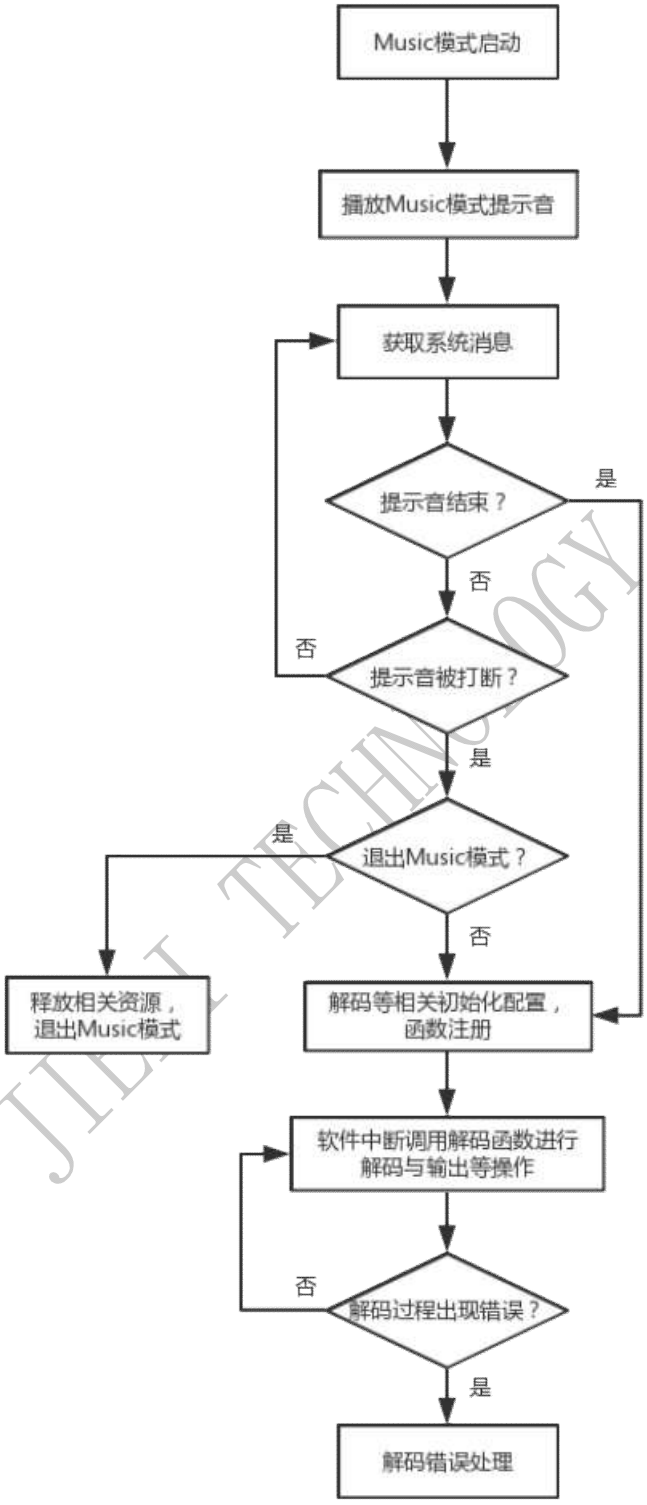


图 7.2.1 音乐模式流程图

7.3 解码通道说明

❖ 解码相关程序流程如下：

- 初始化解码系统参数，使能不同格式的解码库。
- 文件格式检查，调用对应的解码函数。
- 进行解码与解码输出。
- 解码出错时进行相关错误处理。
- 在播放过程中，接受消息进行处理，根据消息不同，有歌曲控制，歌曲切换，设备切换，音效控制等操作。
- 一首歌曲播放完成后，自动查找下一首歌曲播放。
- 如果有新设备插入，则进入最新插入的设备，查找歌曲进行播放。
- 如果正在播放的设备被拔出，发送 SYS_EVENT_DEC_DEVICE_ERR 消息，并查找下一个设备进行播放，如果没有设备了，则等待退出 music 模式。

7.4 部分 API 函数说明

函数原型：

MUSIC_PLAYER *music_player_creat(void)

功能描述： 音乐播放控制句柄的创建与设置

参数说明： 无

返回： NULL：创建失败

其他：创建成功，返回对应的句柄地址

tbool music_player_play(MUSIC_PLAYER *obj, MUSIC_PLAYER_BP *bp_info, u8 is_auto)

功能描述： 音乐播放流程控制

参数说明： *obj 音乐播放控制句柄

*bp_info 断点信息

is_auto 辅助参数

返回： false： 播放失败

True： 播放成功

Chapter8 收音开发使用说明

8.1 总体设计

❖ 系统：

本功能主要是基于 SDK 开发包来实现收音的功能。

FM 应用主要实现的功能包括：

- 自动搜台模式、手动搜台模式、半自动搜台。
- 支持暂停，播放电台。
- 支持断点记忆，可记忆上次播放的频点。

❖ 总体架构设计：

收音主要分为两个功能模块:

①收音主模式模块:初始化 FM 模块，播放当前频道、上下按键选择台播放。

②自动搜台,半自动搜台，手动搜台。

❖ 应用的启动、应用的退出

收音模式进入可通过以下方式：

按 Mode 键切换模式，进入 FM 模式，当再次按 Mode 键跳出 FM 模式。

8.2 收音设计说明

- 模块描述

收音流程中，采用统一的接口方式来兼容多种收音模块。不同收音模块，只需要提供以下功能函数即可添加到收音流程中：

- 获取模块 ID 函数
- 启动/初始化函数
- 关闭函数
- 设置音量函数
- 设置频点函数

- 流程功能

该收音流程已经添加好基本的收音使用功能，主需要接上模块基本驱动，即可实现以下功能：

- 全频点搜台
- 搜上/下一个频点
- 移动上/下一个频点
- 静音、解除静音

8.3 内置收音搜台参数说明

❖ 内置 FM 搜台参数简介

- FMSCAN_SEEK_CNT_MIN ：过零点记数判断最小值.
- FMSCAN_SEEK_CNT_MAX ：过零点记数判断最大值.
- FMSCAN_CNR ：信噪比阈值.

这三个参数均在函数 fm_inside_io_ctrl(SET_FM_INSIDE_SCAN_ARG1 中设置,如下:

fm_inside_io_ctrl(SET_FM_INSIDE_SCAN_ARG1,FMSCAN_SEEK_CNT_MIN, FMSCAN_SEEK_CNT_MAX, FMSCAN_CNR);

● 真台判断：扫描电台时,信号过零点统计值 seek_cnt 在[FMSCAN_SEEK_CNT_MIN, FMSCAN_SEEK_CNT_MAX]范围内，且当前电台的信噪 cnr>= FMSCAN_CNR 时，则判断为真台，反之则判断为假台.

❖清晰与不清晰电台特征:

● 清晰正常电台的特征：固定时间内的,对信号过零点统计值 seek_cnt 在相对较小的范围内,且信噪比 cnr 的值比较大.

● 无台(白噪声)或不清晰的电台特征：seek_cnt 一般比较大，cnr 相对较小.

❖ 内置 FM 一般调试方法:

1. 按默认的参数配置搜台，如果搜台不满足要求,先检测硬件，硬件没有问题,则继续第 2 步.

2. 执行一次搜台(按键消息 MSG_FM_SCAN_ALL_INIT)，程序中开打印,查看所有电台的 seek_cnt，cnr0，cnr1. 初步统计一下各电台的过零点 seek_cnt，信噪比 cnr0，cnr1.

典型搜台打印参数如下:

```
[freq: 875  seek_cnt: 554  cnr: 22]
[freq: 876  seek_cnt: 385  cnr:-21]
[freq: 877  seek_cnt: 545  cnr:  0]
```

根据需要调节信噪比阈值 FMSCAN_CNR，及过零点取值范围[FMSCAN_SEEK_CNT_MIN, FMSCAN_SEEK_CNT_MAX]，当

(cnr>=FMSCAN_CNR))&&(FMSCAN_SEEK_CNT_MIN<=seek_cnt)&&(seek_cnt<=FMSCAN_SEEK_CNT_MAX)时，则当前电台判断为真台，否则判断为假台.

注意：搜台时同时打印 seek_cnt/cnr 信息,需要先调用以下函数打开库中的相关打印.

fm_inside_io_ctrl (SET_FM_INSIDE_PRINTF, 1); //1 打开库中的打印. 0 关闭库中的打印.

若怀疑搜台时库中开打印,会影响到搜台参数，可以在搜台时关掉打印,搜完台时，再下面的函数统一把搜台参数打印出来(搜台时,已先把参数存到内部 RAM 中)

fm_insice_scan_info_printf(875,1080);//打印 87.5 到 108.0 的搜台参数.

❖ 常见搜台问题处理:

1) 收台数量不够：先调低 FMSCAN_CNR，若还不行，再放宽[FMSCAN_SEEK_CNT_MIN, FMSCAN_SEEK_CNT_MAX]

2) 假台数量过多：先调高 FMSCAN_CNR，再缩窄[FMSCAN_SEEK_CNT_MIN, FMSCAN_SEEK_CNT_MAX].

❖ 其它 API 函数:

1) void fm_inside_set_stereo(u8 set); //双声道(立体声)效果设置. set 取值范围[0,127].
   set 值为 0 时，相当于完全的单声道.
   set 的值越接近 127 时,双声道效果越明显,
   set 值为 127 时为双声道.
2) void fm_inside_set_abw(u8 set); //音频带宽设置, set 取值范围[0,128].
   set 的值越大，音频带宽越宽. 带宽可调范围[2k, 16k]
3) void fm_inside_deempasis_set(u8 set); //去加重参数设置. set 只能设置为 0 或 1.
   set 值为 0 时，去加重时间参数为 50us.
   set 值为 1 时，去加重时间为 75us.
4) s16 fm_inside_rssi_read(void); //收音 RSSI 值获取. 单位为 dB.

Chapter9 时钟开发使用说明

9.1 章节简介

该章节说明可以为开发 AC692x_SDK 应用的人员的提供设计开发文档，也可以为测试 RTC 应用的测试人员提供参考。文档中详细定义了 RTC 应用的总体功能、系统的接口和数据属性；对程序的基本结构、功能模块以及各个程序的名称进行了划分，以便于 RTC 应用的详细设计和编码。

9.2 总体设计

- 需求概述

本模块主要是基于 AC692x_SDK 系统开发包来实现 RTC 的功能。RTC 应用主要实现的功能包括：

（1）支持系统时间调整。

（2）支持闹钟设置处理。

本模块的开发实现基于杰理 AC692x_SDK 软件开发包及相应的硬件平台。

- 总体架构设计

RTC 主要分为三个功能模块:

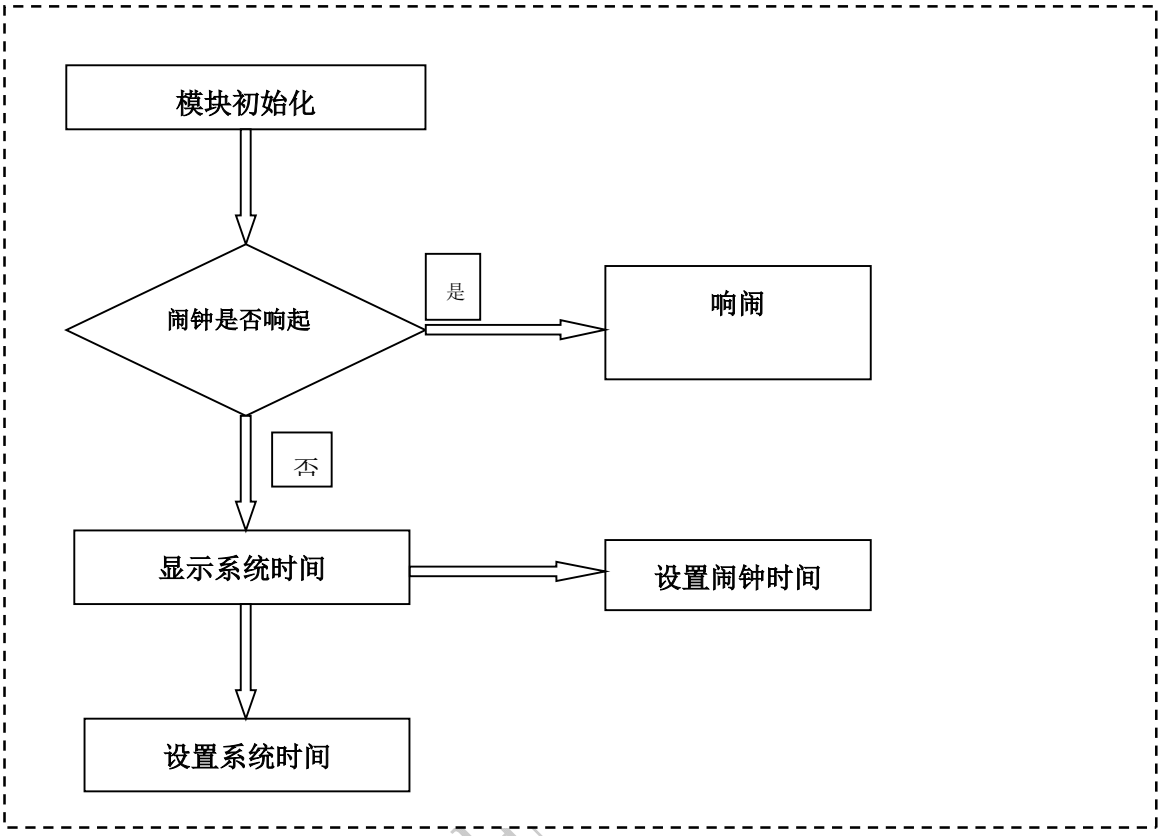
- 显示模块:主要显示当前时间
- 设置模块:设置系统时间和闹钟时间
- 闹钟响应模块：主要对闹钟进行响应

- 总体架构图

图 10.2.1 RTC 流程图

- 功能模块划分

模块名称	功能简述	对应文件
系统入口模块闹钟响应	设置系统相关信息,响应闹钟操作	rtc.c
显示模块	显示系统时间，设置系统时间和闹钟时间	rtc_ui.c
时钟设置模块	设置时间和闹钟操作流程	rtc_setting.c



❖ 功能模块划分

模块名称	功能简述	对应文件
系统入口模块闹钟响应	设置系统相关信息,响应闹钟操作	task_rtc.c
显示模块	显示系统时间，设置系统时间和闹钟时间	rtc_ui.c
时钟设置模块	设置时间和闹钟操作流程	rtc_setting.c

❖ 应用的生命周期

❖ 按 Mode 键切换模式，进入 RTC 模式，开始运行，当再次按 Mode 键跳出 RTC 模式，应用生命周期结束。

❖ 应用的启动

当切换到 RTC 任务时，就会调用 void rtc_info_init(void)函数，创建和初始化 RTC 所需的资源和硬件模块。

激活过程主要是以下几步：

（1）初始化 RTC 需要的资源，配置初始值

（2）初始化 RTC 信息，查看时钟，闹钟范围是否正常

❖ 应用的退出

当有切换任务或者设备插入的消息时，task_common 处理就会选择切换任务，退出 RTC 模式。

9.3 系统入口模块设计说明

❖ 模块描述：

对应的文件是 task_rtc.c 是 RTC 应用的入口,进入时负责系统的初始化。退出时负责关闭资源和硬件模块。

❖ 模块功能

该模块为应用程序的入口模块,主要进入时负责系统的初始化，屏幕初始化，DAC 初始化等。

❖ 模块接口设计

功能描述：注册给 RTC 中断的回调函数，回调时 flag 区分表示产生中断的情况。闹钟触发时要做处理的在 if(RTC_ISR_ALARM_ON == flag)分支。

函数原形：void rtc_isr_user_handler(u8 flag)

其它说明：

1）if(RTC_ISR_PCNT==flag)计数溢出产生的中断

2）if(RTC_ISR_LDO5V ==flag)是 LDO5V 检测产生的中断

9.4 设置时间模块设计说明

❖ 模块描述

该模块给客户调整时间和闹钟。

❖ 模块功能

该模块给客户调整时间和闹钟。

❖ 模块接口设计

功能描述：根据设置模式，设置 RTC 模式下时间或者是闹钟

函数原形：void rtc_setting(int msg)

其它更多 RTC 功能设置的接口请查询 rtc_api.h 文件使用

❖ 异常处理

超时，主界面消息，设备插入会退出设置模式。

9.5 闹钟模块设计说明

❖ 模块描述

该模块对闹钟进行响应。

❖ 模块功能

该模块对闹钟进行响应，显示闹钟界面，闹钟声音。

❖ 模块接口设计

功能描述：闹钟、时钟设置

函数原型：void rtc_setting(int msg)

功能描述：闹钟开关

函数原型：void rtc_sw(u8 flag)

其它更多 RTC 功能设置的接口请查询 rtc_api.h 文件使用

❖ 异常处理

超时，主界面消息，设备插入会退出设置模式。

9.6RTC 模块特殊功能说明

❖ 普通 IO 口

RTC 有独立的 IO 口，PORTR0~3，PORTR1 默认输出 0，其中 PORTR0 复用 32.768KHz 晶振引脚，供 RTC 内部时钟模块。PORTR3 复用 12-26MHz 的晶振，可连接外部晶振作为系统时钟。

PORTR0~3 都能唤醒 RTC 低功耗模式。

设置 IO 口的函数如下所示，每次只能这只一个 IO 口的状态：

```
void PORTR_DIR(u8 port, u8 val);
void PORTR_OUT(u8 port, u8 val);
void PORTR_HD(u8 port, u8 val);
void PORTR_PU(u8 port, u8 val);
void PORTR_PD(u8 port, u8 val);
u8 PORTR_IN(u8 port);
```

Port：PORTR0、PORTR1、PORTR2、PORTR3　　Val：0 或者 1

NOTE:

1. *PORT3 引脚必须外挂 32.768KHz 晶振，时钟功能才能正常工作*
2. *PORTR3 如果使用晶振，不能设置为输出，否则时钟将停止*
3. *进入低功耗后，PORTRIO 还能正常工作，其他普通 IO 口为高阻状态*

❖ PORTR1 和 PORTR2 可以做 ADkey 使用，下面提供代码示例

```
/*1: PR1 port voltage  drive ADC  , 0: no this function*/
//if you want to use portl to be ADKEY io, you can set as below
PORTR1_ADCEN_CTL(1);
PORTR_PD(PORTR1 , 0);
PORTR_PU(PORTR1 , 0);
PORTR_DIR(PORTR1 , 1);
```

```
PORTR_DIE(PORTR1 , 0);
```

❖ RTC 模块还支持计数器溢出唤醒和产生相应的中断。

功能描述：设置计数唤醒的相关配置。

```
函数原型：void pcnt_init(u8 port, u8 edge);
```

功能描述：设置计数寄存器的计数值。

```
函数原型：void set_pcnt_value(u8 value);
```

功能描述：获取计数寄存器的计数值。

```
函数原型：int  get_pcnt_value();
```

❖ PORTR 口长按复位功能接口

功能描述：设置按键复位的配置

```
函数原型：int rtc_port_reset(u8 mode, u8 port, u8  enable, u8 edge);
```

❖ 低功耗

RTC 支持独立低功耗模式，进入改低功耗时候，主控电源将会关闭，进入低功耗后，只有 RTC 模块工作，请参考低功耗的相关章节。

Chapter10 PC 从机开发使用说明

10.1 总体设计

❖ 系统：

本说明主要是基于 SDK 开发包来实现 PC 的功能。

PC 应用主要实现的功能包括：

- 支持读卡器功能。
- 支持声卡功能。
- 支持 HID 功能

❖ 总体架构设计：

PC 任务主要分为三个功能模块：

①读卡器功能模块：从机模式，PC 能够对设备中的存储设备进行操作。

②声卡功能模块：从机模式，充当 PC 音乐播放器的扬声器。

③HID 操作模块：主要是从机充当声卡，发出 PC 音乐播放器播放的音乐声音时，能通过控制从机上的上/下一曲，播放来控制 PC 播放器的相应操作。

❖ 应用的启动

PC 模式进入可通过以下 2 个方式：

①USB 从机线插入设备后，设备管理器检测到，就会向主线程发送 SYS_EVENT_PC_IN，主线程会强制退出当前模式，并激活 PC 模式。

②通过模式按键切换，主线程会强制退出当前模式，并激活 PC 模式。

❖ 应用的退出：

当按下 Mode 键、PC 线拔出或者其他任务需要被激活时候，主线程会强制退出 PC 模式

释放 PC 任务资源

关闭提示音

❖ 应用依赖库及其接口说明

usb_lib.a	——USB 模块设备库
lib_usb_syn.a	——PC 模式 AUDIO 同步库

10.2 系统入口模块设计说明

❖ 模块描述

对应的文件是 task_pc.c 是 PC 应用的入口,进入时负责 PC 模式的变量申请和硬件的初始化，退出时负责释放 PC 模式占用的变量资源。

❖ 模块功能

该模块为应用程序的入口模块，主要进入时负责系统的初始化，如读卡器功能初始化、AUDIO 功能的 DAC 初始化、HID 初始化等。

10.3 读卡器模块设计说明

❖ 模块描述

该模块主要实现对大容量存储设备的读取。card_reader_io.c 提供读卡器与设备的 IO 连接。

❖ 模块功能

该读卡器功能。

❖ 模块接口设计

读卡器功能实现功能描述：

- 函数原型：s32 app_usb_slave_card_reader(u32 cmd)
- 函数说明：PC 读卡器执行流程函数
- 参数说明：cmd 命令
- 返回值说明：执行状态

- 函数原型：sUSB_DEV_IO *get_card_read_io(DEV_TYPE dev_type)
- 函数说明：获取读卡器操作设备的 IO
- 参数说明：dev_type 设备类型
- 返回值说明：设备的操作 IO

10.4HID 操作模块设计说明

❖ 模块描述

该模块主要实现用连接 PC 线的设备来控制 PC 上的播放器的相应操作。

❖ 模块功能

该模块实现 HID 功能。

❖ 模块接口设计

HID 功能描述：

- 函数原型：void usb_slave_hid(u32 key)
- 函数说明：PC HID 功能函数
- 参数说明：key：PC HID 按键命令
- 返回值说明：无

10.5USB_SPK 模块设计说明

❖ 模块描述

该模块主要实现把本设备当作 PC 声卡功能。

❖ 模块功能

该模块实现 USB_AUDIO 功能。

❖ 模块接口设计

PC_AUDIO 声卡

功能描述：

- 函数原型：u8 pc_set_speaker_vol(u32 pc_mute_status)
- 函数说明：PC AUDIO 功能音量设置函数
- 参数说明：pc_mute_status：mute 状态
- 返回值说明：当前声卡音量值

- 函数原型：void pc_dac_mute(bool mute_status, u8 fade_en)
- 函数说明：PC AUDIO mute 设置函数
- 参数说明：mute_status：mute 状态；fade_en：淡入淡出设置
- 返回值说明：无

10.6PC 检测功能

在的每个模式下，插入 PC 线的时候将自动跳到 PC 模式，也可以通过 Mode 键切换到 PC 模式。

- 函数原型：void pc_check_api(void)
- 函数说明：PC 在线检测函数
- 参数说明：void
- 返回值说明：void

在设备检测任务里加入 PC 检测函数，如果检测到有 PC 线接入，设备检测任务会当前任务发出 EVENT_PC_IN 事件，由 task_common 控制切换到 PC 模式。

Chapter11　F1A 提示音文件

11.1 F1A 提示音概述

F1A 提示音文件是本公司专门针对提示音文件特点而定制的音频格式。在不影响音质前提下，提高压缩率，尽可能使提示音占用更少的代码空间。

开启应功能开启。

对应功能宏在 sdk_cfg.h 文件中

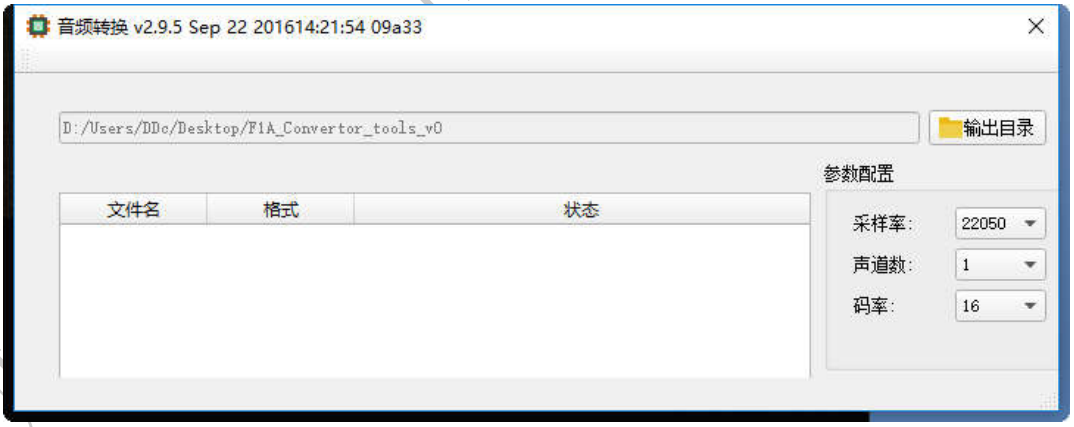
#define DEC_TYPE_F1A_ENABLE　　///<30K_code_space

下载对应提示音文件。

apps\download\ac692x\post_build\download.bat 文件中

isd_download.exe …. power_off.f1a bt.f1a music.f1a record.f1a linein.f1a radio.f1a pc.f1a wait.f1a connect.f1a disconnect.f1a ring.f1a ….

F1A_Convertor_tools_v1.0 可以把普通音频文件转换为 F1A 文件。



图

Chapter12 混响模式以及混响功能

12.1 总体设计

❖ 系统：

本说明主要是基于 SDK 开发包来实现混响的功能。

混响应用主要实现的功能包括：

- 深度，强度自定义。
- 支持多个模式下开启混响功能。

❖ 总体架构设计：

混响任务主要分为两个功能模块：

- ❖ 音源模块：混响音源输入仅支持 mic 通道。
- ❖ 混响处理模块：获取音源后，进行混响效果处理，然后输出到回调函数。

❖ 应用的启动

混响模式进入可通过以下方式：

- ❖ 通过按键开启混响模式。
- ❖ 当按下 Mode 键、切换进入混响模式

❖ 应用的退出：

混响模式进入可通过以下方式：

- 通过按键关闭混响模式。
- 当按下 Mode 键、退出混响模式

❖ 应用依赖库及其接口说明

- lib_audio_reverb_H.a ——混响处理库

❖ 参数说明

```
// ECHO 效果参数初始化.
ECHO_CONTROL_VAR  echo_var = {
    .music_vol = 13384,             //0 ~ 16384   //背景音乐音量大小. 16384(0 db)
    .mic_vol_digital = 13384,       //0 ~ 16384   //MIC数字音量大小.  16384(0 db)
    .deep = 900,                    //0 ~ 1024    //值越大, 混响深度越深
    .mic_vol = 50,                  //0 ~ 63      //MIC模拟音量(放大)大小
    .pitch = -2,                    //-127 ~ 127  //变调, 0为正常声音, 负值越小频率越低, 正值越大频率越高. (howlingsupression)
    .decay = 110,                   //0 ~ 128     //混响强度衰减: 值越大, 混响的后一声比前一声衰减越多.
    .run_flag = 0,
    .first_echo_sel = FIRST_ECHO_ANALOG,
};

// ECHO 效果参数初始化.
REVERB_PARM_SET rev_parm = {
    /*-------------------------------------------------------*/
    /* 以下参数用于调试使用, 请勿修改 */

    /* reserved */

    /*-------------------------------------------------------*/
    .ef_reverb_parm_2.deepval       = 1024,     /* 混响深度, 范围:(0-1024) , (1024->max_ms)) */
    .ef_reverb_parm_2.decayval      = 128,      /* 混响强度, 范围:(0-128) */
    .ef_reverb_parm_2.gainval       = 4000,     /* 音量增益, 范围:(0-4096) */
    .ef_reverb_parm_2.rs_mode       = 0x100,    /* reserved */
    /*-------------------------------------------------------*/
};
```

参数说明 (reserved 仅用于调试，不建议修改)

注意：692X 由于 ram 资源问题，SDK 默认配置的混响深度最大为 200ms，对应约需要 8K 空间。

12.2 系统入口模块设计说明

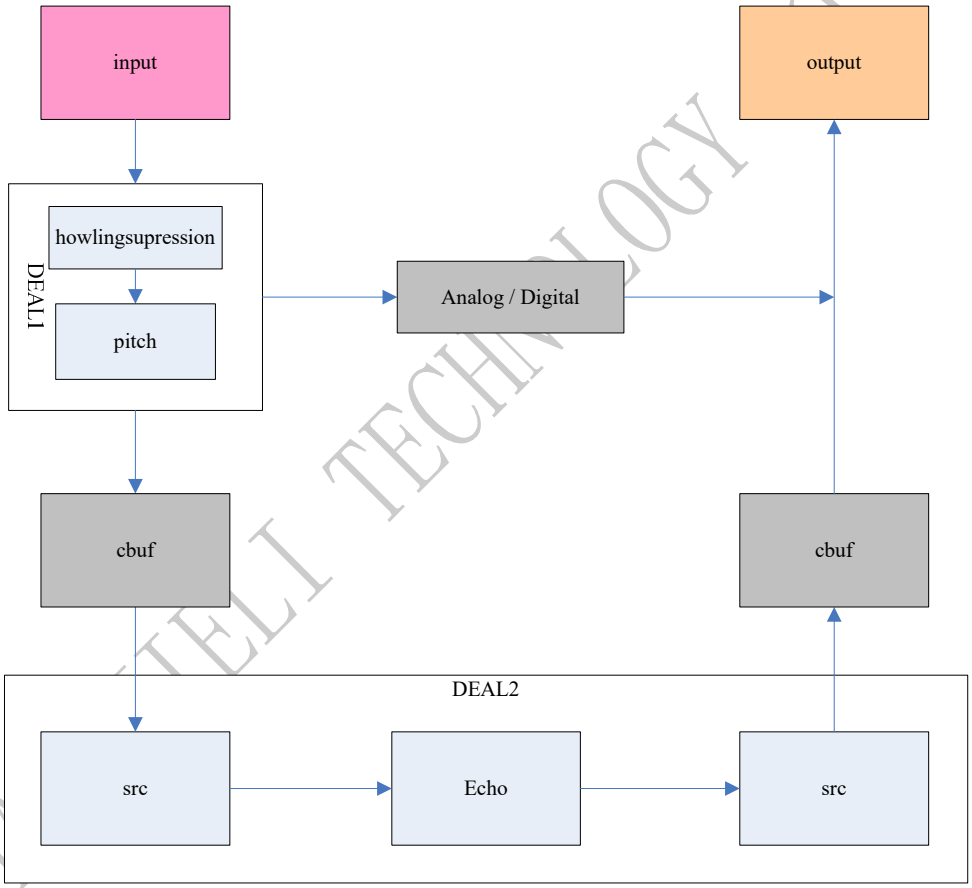
❖ 模块描述

混响综合消息处理，支持开启/关闭混响。

❖ 模块功能

在不同模式下，按下混响开启/关闭按键即可开启/关闭混响功能。

流程图：



流程图 1

Chapter13 蓝牙对箱开发使用说明

13.1 编写目的

该文档主要描述 AC692x TWS 开发方法及开发中要注意的一些问题，为用户进行二次开发提供参考。

TWS 开发和普通单台蓝牙音箱开发兼容，其他开发文档和工具使用说明文档通用。下面是针对蓝牙 TWS 开发使用进行说明。

13.2 术语和缩写词

提示：列出本文件中用到专门术语的定义和外文首字母组词的原词组。

缩写和术语	解 释
AC692x	杰理科技 AC692x 系列芯片
TWS（turewireless）	蓝牙对箱，基于主、从两个蓝牙设备
TWS_ROLE_MASTER	蓝牙对箱主机
TWS_ROLE_SLAVE	蓝牙对箱从机
Inquiry	发现设备状态
Inquiry scan/i_s	可被发现状态
page	连接设备状态
page scan/p_s	可被连接状态

13.3 TWS 开发说明

一、TWS 配置选项

1. **对箱使能配置**

```
///可选配置：0(普通音箱)/BT_TWS_TRANSMIT(对箱使能)
#define BT_TWS          BT_TWS_TRANSMIT
```

二、操作设置 tws 选项配置

1、按键搜索设备： MSG_BT_SEARCH_DEVICE

初次配对，需要使用该消息来搜索并配对对箱设备

user_send_cmd_prepare(USER_CTRL_SEARCH_DEVICE,0,NULL);

```
case MSG_BT_SEARCH_DEVICE:
    puts("MSG_BT_INQIRY_DEVICE\n");
    if (get_tws_device_role() == 0) {
        user_val->inquiry_flag = 1;
        bt_work_state_control(0);
        user_send_cmd_prepare(USER_CTRL_SEARCH_DEVICE, 0, NULL);
    } else {
        puts("TWS_connected,ignore INQUIRY\n");
    }
    break;
```

注：默认当对箱连接的时候，会自动忽略该设备搜索信息

//设置蓝牙搜索的时间，实际时间=N*1.28s

__set_search_timeout_value(0x06);/*搜索设备时间,Range:0x01-0x30,Time = N * 1.28s*/

2、打开可发现可连接

case MSG_BT_PAGE_SCAN:

puts("MSG_BT_PAGE_SCAN\n");

bt_work_state_control(1);

break;

3、开机是否回连 tws 从机

__set_tws_start_conn(1); /*对箱回连，0：不回连，1：回连*/

4、设置对箱搜索标识,搜索到相应的标识才允许配对连接

/*设置对箱搜索标识，inquiry 时候用,搜索到相应的标识才允许连接*/

char tws_device_indicate[2] = {'J', 'L'};/*只能两个字符，可以自定义内容*/

5、TWS 连接成功之后，主设备进行呼叫处理和控制，将音频流传送到从设备，主从设备配合实现无线声道同步播放。主从设备进行音频同步播放，可以将左右声道分离播放。也可以合成进行主从同步播放。

单台使用连接手机时会自动进行立体声播放。

位置：audio_sync/sync_tws.c:void*get_tws_sync_param()

```
tws_sync.tws_sync_parm.set_user_ch = 0;              //0:master left   1:master right
tws_sync.tws_sync_parm.set_ch_mode = 0;              //0:声道独立      1:opL = opR = (L+R)/2
tws_sync.tws_sync_parm.sbc_bitpool_value = 31;       //对箱音频压缩率
```

6、蓝牙电话在主设备播放，从设备不进行建立通话连接。即通话只在主机实现。

7、当前设备搜索配对状态显示（用于调试）

void bt_baseband_state_debug()；

该接口可以显示当前是否可发现可连接，是否正在配对或者搜索设备

8、avctp_user.h 声明了一些常用的 tws 接口

9、对箱连接成功，保存当前的设备角色（主机或者从机）

```
/**
 *保存对箱连接角色
 *用来判断下次开机是否发起回连
 */
void bt_tws_info_save(u8 info)
{
    u8 tws_info = 0;
    vm_read(VM_TWS_INFO, &tws_info, VM_TWS_INFO_LEN);
    if (tws_info == info) {
        puts("same tws_info,return\n");
    }
    tws_info = info;
    update_bt_tws_info(info);
    log_printf("TWS_info W:%x\n", info);
    vm_write(VM_TWS_INFO, &tws_info, VM_TWS_INFO_LEN);
}
```

10、设备开机，通过读取设备的记忆设备角色，执行相应的动作

```
/**
 *获取对箱上次连接角色，用来判断开机是否发起回连
 *主机发起回连，从机等代连接
 */
u8 get_tws_mem_role(void)
{
    u8 info;
    vm_read(VM_TWS_INFO, &info, VM_TWS_INFO_LEN);
    log_printf("TWS_info R:%x\n", info);
    //return 0;
    return info;
}
```

三、TWS 连接成功之后可以主从设备按键关联控制

1、TWS 连接之后，按键消息会通过以下接口进行关联

```
#if BT_TWS
        if (tws_key_cmd_send(msg[0], 0)) {
            continue;
        }
#endif
```

注：有些消息主机实现，从机不需要同步实现的会被过滤掉。

2、TWS 连接之后，信息同步

```
/**
 *对箱连接成功，同步一些必要的配置，如eq模式，音量级数等
 *如果不想同步，不注册该接口或者直接返回即可
 */
static void bt_tws_info_sync(void)
{
    tws_cmd_send(MSG_BT_TWS_EQ_SYNC, sound.eq_mode);
    tws_cmd_send(MSG_BT_TWS_VOL_SYNC, sound.vol.sys_vol_l);
}
```

这里根据实际需要，可以同步更多的信息

3、TWS 主机从机上层通信接口

void tws_cmd_send(int msg, u8 value)

比如同步关机，可以在关机前发一个关机的命令给对方即可。

四、TWS 连接成功之后状态获取处理

1、获取当前连接或者断开的是什么设备

```
    case BT_STATUS_FIRST_CONNECTED:
    case BT_STATUS_SECOND_CONNECTED:
#if BT_TWS
        /*判断当前连接的设备类型:对箱或者手机*/
        newest_bd = get_cur_conn_bd(1);
        log_printf("BT_CONNTCTED:%d\n", newest_bd);
        if (newest_bd == BT_CUR_CONN_PHONE) {
            puts("[connect_dev]phone\n");
        } else if (newest_bd == BT_CUR_CONN_TWS_MASTER) {
            puts("[connect_dev]tws_master\n");
        } else if (newest_bd == BT_CUR_CONN_TWS_SLAVE) {
            puts("[connect_dev]tws_slave\n");
        }
#endif
        led_fre_set(C_BLED_FAST_ONE_5S_MODE);
        /* led_fre_set(C_BLED_FAST_DOBLE_5S_MODE); */
        task_post_msg(NULL, 1, MSG_BT_TONE_CONN);
        break;
    case BT_STATUS_FIRST_DISCONNECT:
    case BT_STATUS_SECOND_DISCONNECT:
#if BT_TWS
        /*判断当前断开的设备类型: 对箱或者手机*/
        newest_bd = get_cur_conn_bd(0);
        log_printf("BT_DISCONN:%d\n", newest_bd);
        if (newest_bd == BT_CUR_CONN_PHONE) {
            puts("[disconn_dev]phone\n");
        } else if (newest_bd == BT_CUR_CONN_TWS_MASTER) {
            puts("[disconn_dev]tws_master\n");
        } else if (newest_bd == BT_CUR_CONN_TWS_SLAVE) {
            puts("[disconn_dev]tws_slave\n");
        }
#endif
        task_post_msg(NULL, 1, MSG_BT_TONE_DISCONN);
        break;
```

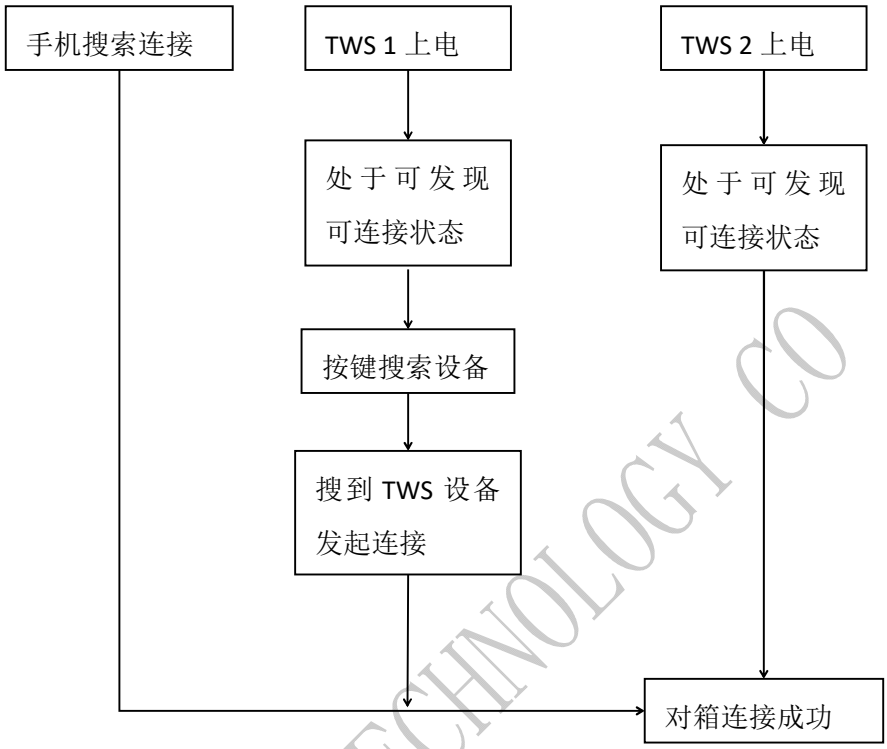
2、手动连接和断开对箱

```
    /*对箱连接控制*/
    case MSG_BT_TWS_TEST:
    case MSG_BT_TWS_CONNECT_CTL:
        if ((get_tws_connect_status() == BT_STATUS_CONNECTING)    ||
            (get_tws_connect_status() == BT_STATUS_TAKEING_PHONE) ||
            (get_tws_connect_status() == BT_STATUS_PLAYING_MUSIC)) {
            //puts("disconn_tws\n");
            user_send_cmd_prepare(USER_CTRL_TWS_DISCONNECTION_HCI, 0, NULL);
        } else {
            //puts("conn_tws\n");
            user_send_cmd_prepare(USER_CTRL_TWS_START_CONNECTION, 0, NULL);
        }
        break;
```

1.4 蓝牙 TWS 通信流程

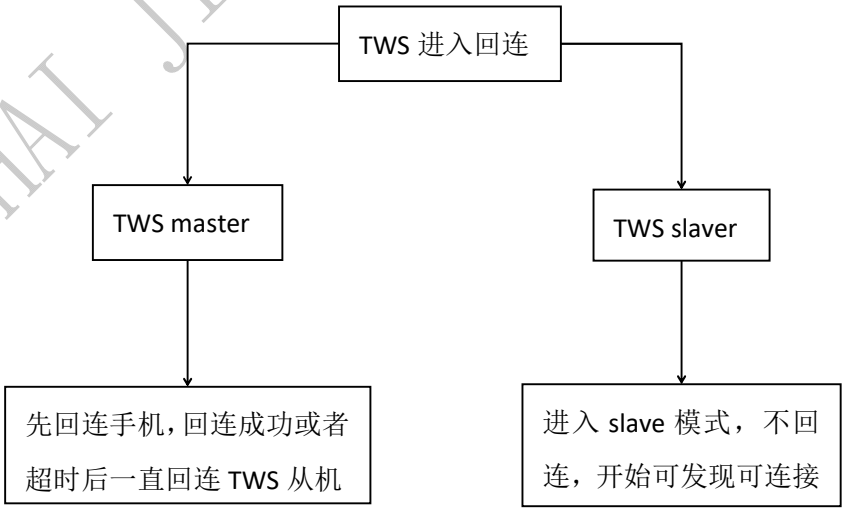
- **tws 连接流程**

  实际连接流程根据配置去选项。



- **tws 开机回连流程**

  根据连接上次连接角色，TWS 主机开机发起回连



Chapter14 AC692X 串口升级协议

14.1 编写目的

该文档主要描述 AC692x 开发串口升级的方法及开发的通信协议和通信命令。该功能只在 AC692x_SDK_release_V2.3.1 以后的版本才支持。该功能需要 uboot 的支持，建议需要该功能的在 AC692x_SDK_release_V2.3.1 以后的版本开发。

14.1 配置

1）打开配置宏：

```
#define UART_UPDATA_EN                1
```

2）配置结构体：

```
UPDATA_UART updata_uart = {
    .control_io = UART0_TXPA5_RXPA6,//只有 UART0 有 DMA，所以只能选择 UART0 的 IO
    .control_baud = 460800,//波特率
    .control_timeout = 300,//超时退出和重发，单位 10ms
};
```

14.2 文件使用

升级需要使用的文件是.bfu 格式的文件，例如 updata.bfu 文件

14.3 协议

命令	发出端	接收端
升级开始	工具端(12byte):UPDATE_STAR!	Mcu(12byte):RECEIVE_CMD!
进入升级	工具端(12byte):UART_UPDATE!	Mcu(12byte):RECEIVE_CMD!
读文件命令	Mcu(12byte):READ+addr+len	工具端:crc(4byte)+data
完成升级	Mcu(12byte):UPDATE_SUCC!	工具端(12byte):RECEIVE_CMD!
升级失败	Mcu(12byte):UPDATE_FAIL!	工具端(12byte):RECEIVE_CMD!

协议解释：

升级时，Mcu（AC692x）是被升级的的，所以是从机端，发起升级的一端是主机端，比如工具、其他芯片等，以下内容皆使用主机代称发起升级的一端。

1、当需要升级时，命令是由主机发送到从机（mcu：AC692x）：主机发送：UPDATE_STAR！共 12byte，从机回应：RECEIVE_CMD！共 12byte，确认通信正常；

2、确认通信后进入升级，主机发送 UART_UPDATE！共 12byte 到从机（mcu：AC692x），进入 uboot 进行升级；

3、当进入升级后，命令是由从机（mcu：AC69）发送到主机：从机发送：READ+文件读取的地址（4byte）+读取长度（4byte）共 12byte，长度 len 进行 4byte 对齐，对齐部分数据使用 0（比如需要长度为 22byte 时候，需要补齐为 24byte，最后两 byte 为 0），主机发送：数据区的 16 位 CRC 校验+数据；通常数据为 512byte，但是需要根据读取命令 len 来适配，CRC 放在数据头，使用 u16 的 crc，占用 4byte，即回复总长度为 4+len，数据为小端模式（即当数据为 0x87654321 时候，发送应为 0x21436587），读取一包数据超过 5 次视为升级失败，退出升级，mcu 将重启。

例如：读取文件从 0x00 09 D8 E0 之后的 34（0x22）byte 数据，而数据为的二进制为：4C 15 BD 6E 55 E0 EB 75 CA 24 78 05 B2 DA 46 05 28 6B F9 AE 1B 55 7B E9 4B CD E4 69 2C 82 5B 55 B3 A9

则过程如下：

读文件数据命令（READ+addr 小端格式+len 小端格式）：52 45 41 44 E0 D8 09 00 22 00 00 00

回复命令（crc 小端格式+34byte 数据+数据对齐补 0）：C3 5F 00 00 4C 15 BD 6E 55 E0 EB 75 CA 24 78 05 B2 DA 46 05 28 6B F9 AE 1B 55 7B E9 4B CD E4 69 2C 82 5B 55 B3 A9 00 00

4、当升级完成后，命令是由从机（mcu：AC69）发送到主机：从机发送：UPDATE_SUCC！共 12byte，主机回应：RECEIVE_CMD！共 12byte，从机启动，升级完成；当升级失败时，命令是由从机（mcu：AC69）发送到主机：从机发送：UPDATE_FAIL！共 12byte，主机回应：RECEIVE_CMD！共 12byte，从机启动，

Chapter 15 录音模式以及录音功能

15.1 总体设计

❖ **系统：**

本说明主要是基于 SDK 开发包来实现录音的功能。

录音应用主要实现的功能包括：

- 支持 WAV 和 MP3 格式录音。
- 支持 U 盘和 SD/TF 卡 录音。
- 支持最高 9999 个录音文件编号。
- 支持 BT 录音、FM 录音、aux 录音、录音模式下为 mic 录音。
- 支持直接从 DAC 拿数据或者直接从数据源拿数据。
- 支持录音回放。

❖ **总体架构设计：**

录音任务主要分为三个功能模块：

❖ 音频输入源：在不同模式，音频源不同，但输入数据都是 PCM 数据。

❖ 编码模块：WAV 和 MP3 使用不同的编码器。

❖ 数据输出：编码后的数据从输出模块流出，默认存放在 U 盘或者 SD/TF 卡中，只支持 fat 文件系统。可根据需求修改输出目标。

❖ **应用的启动**

❖ 在对支持录音的模式下，若存储设备在线、按下录音按键即可进行录音。

❖ 通过模式按键切换，若存储设备在线，可进入录音模式。

❖ **应用的退出：**

当按下 Mode 键、拔出设备或者其他任务需要被激活时候，主线程会强制关闭录音或者退出录音模式

❖ **应用依赖库及其接口说明**

- lib_adpcm_encode.a ——WAV 编码库
- lib_mp2_encode.a ——MP3 编码库

15.2 系统入口模块设计说明

❖ 模块描述

录音功能模式可添加在任意模式下的消息处理当中，当收到录音消息后，即可进入对应的消息处理。

❖ 模块功能

在不同模式该模块为应用程序的入口模块，只需要改变输入音源即可支持不用的录音。

录音模式支持以下处理：

- 开始录音。
- 暂停/恢复录音。
- 结束录音。

15.3 录音参数配置说明

❖ 采样率参数描述

该模块主要实现对大容量存储设备的读取。

(44100), (48000), (32000) 采样率可配置以下比特率：

32,48,56,64,80,96,112,128

(44.1K/2,/4), (48k/2,/4), (32K/2,/4) 采样率可配置以下比特率：

8,16,24,32,40,48,56,64,80,96,112,128

❖ 录音文件夹

录音文件夹，仅支持一层目录：

```
const char rec_folder_name[] = "/JL_REC";
```

❖ 录音文件文件

```
const char rec_file_name[] =    "/JL_REC/FILE0000.MP3";        //MP3 录音文件名（含路径）
const char rec_file_name[] =    "/JL_REC/FILE0000.WAV";        //ADPCM 录音文件名（含路径）
```

❖ 录音资源配置

由于 AC692N 没有内存分配功能，需要传一块内存给录音使用。调用接口 void rec_alloc_init(void *alloc_buffer, u32 buffer_len);默认 MP3 编码需要 16K 的资源，WAV 编码需要 8K 的 RAM。但注意 WAV 编码后的数据量更大，需要更快的设备写入速度。目前蓝牙录音只支持 WAV 录音，因为蓝牙下资源不够 MP3 录音。

❖ 优化录音丢帧

由于资源紧张问题，目前录音是以最少资源运行。当某些写入较慢的设备或运行代码过多会出现丢帧现象，以下方法可进行优化，前提是资源可用的情况。

1、加大输入和输出缓存。void encode_input_buf_len(u32 len); 和 void encode_output_buf_len(u32 len);是分别加大输入和输出缓存的接口。必须在编码器初始化之前设置，并且传给录音使用的资源必须相应加大。

2、降低 MP3 编码的比特率，减小写入设备的数据。但必须选取合适的比特率。

3、使用两个软中断来分开跑编码流程和编码输出流程。但编码流程软中断优先级必须高于编码输出，从而减小跑录音流程的时间，提高效率。这个做法需要充分考虑整个系统的中断优先级，防止其他中断优先级问题，

Chapter16 蓝牙 BLE 透传功能说明

16.1 编写目的

该文档主要描述 AC692x BLE 数据透传开发方法及开发中要注意的一些问题，为用户进行二次开发提供参考。

BLE 是一个独立的蓝牙低功耗蓝牙，支持单独使用，也支持与单台蓝牙音箱同时使用，其他开发文档和工具使用说明文档通用。下面是针对蓝牙 BLE 开发使用进行说明。

16.2 术语和缩写词

提示：列出本文件中用到专门术语的定义和外文首字母组词的原词组。

缩写和术语	解 释
BLE（Bluetooth Low Energy）	低功耗蓝牙
GAP (Generic Access Profile)	通用访问应用，用来控制设备连接和广播。
GATT (Generic Attribute Profile)	是一个在蓝牙连接之上的发送和接收很短的数据段的通用规范，这些很短的数据段被称为属性（Attribute）。
ATT(Attribute Protocol)	属性层,是 GATT 和 GAP 的基础，它定义了 BLE 协议栈上层的数据结构和组织方式。
ADV(Advertising)	可被发现状态
SCAN(Scanning)	发现设备状态
CONNECT	设备连上状态
MASTER	主机，处于主动发起蓝牙连接的角色
SALVE	从机，处于被动蓝牙连接的角色
SERVER	GATT 定义的服务器角色，就是数据中心；可以被客户端读写，也有通知客户端功能，SDK 对应是 SLAVE
CLIENT	GATT 定义的客户端角色，就是访问数据者；可以读写服务器，也接收服务器通知功能，SDK 对应是 MASTER
UUID	就是用来唯一识别一个特征值的 ID
HANDLE	就是对应的 attribute 的一个句柄

16.3 BLE 的配置说明

1. **BLE 使能配置**

```
//可选配置：BT_BREDR_EN/BT_BLE_EN/(BT_BREDR_EN|BT_BLE_EN)
#define BLE_BREDR_MODE                  (BT_BREDR_EN|BT_BLE_EN)//双模配置

//GAP 定义的 GATT 角色配置
#if (BLE_BREDR_MODE&BT_BLE_EN)
    //可选配置：0--server ,1--client
    #define BLE_GAP_ROLE                0
#endif
```

BRDER 和 BLE 可以各自单独打开，相当于单模功能。

也可以同时打开，相当于双模功能。

16.4 BLE 的 SERVER 角色

对应文件 le_server_module.c，负责模块初始化、处理协议栈事件和命令数据控制发送等；server 角色在 SDK 中是以从机 slave 的方式实现，被主机（如手机 app）搜索和连接。

1、协议栈初始化

接口：void ble_stack_config_init(void)

功能：定义协议栈配置初始化，profile 数据注册，事件注册处理，read 和 write 注册处理。

2、profile 生成配置

在 le_server_module.h 文件上，profile_data[]为 server 使用的 profile 数据；该结构是由由杰理的 profile 工具 make_gatt_services 根据配置 cfg 文件生成的。详细用法，请查看工具文档《make_gatt_services 工具说明》。

下图为 SDK 对应的 CFG 配置和生成的 profile_data 部分截图。

```
PRIMARY_SERVICE, 1800
CHARACTERISTIC,  2a00, READ | WRITE | DYNAMIC,

PRIMARY_SERVICE, ae00
CHARACTERISTIC,  ae01, WRITE_WITHOUT_RESPONSE | DYNAMIC,
CHARACTERISTIC,  ae02, NOTIFY,

CHARACTERISTIC,  ae03, WRITE_WITHOUT_RESPONSE | DYNAMIC,
CHARACTERISTIC,  ae04, NOTIFY,

CHARACTERISTIC,  ae05, INDICATE,

CHARACTERISTIC,  ae10, READ | WRITE | DYNAMIC,

PRIMARY_SERVICE, 1812
```

```
const uint8_t profile_data[] = {
    ////////////////////////////////////////////////////
    //
    // 0x0001 PRIMARY_SERVICE  1800
    //
    ////////////////////////////////////////////////////
    0x0a, 0x00, 0x02, 0x00, 0x01, 0x00, 0x00, 0x28, 0x00, 0x18,

    /* CHARACTERISTIC,  2a00, READ | WRITE | DYNAMIC, */
    // 0x0002 CHARACTERISTIC 2a00 READ | WRITE | DYNAMIC
    0x0d, 0x00, 0x02, 0x00, 0x02, 0x00, 0x03, 0x28, 0x0a, 0x03, 0x00, 0x00, 0x2a,
    // 0x0003 VALUE 2a00 READ | WRITE | DYNAMIC
    0x08, 0x00, 0x0a, 0x01, 0x03, 0x00, 0x00, 0x2a,

    ////////////////////////////////////////////////////
    //
    // 0x0004 PRIMARY_SERVICE  ae00
    //
    ////////////////////////////////////////////////////
    0x0a, 0x00, 0x02, 0x00, 0x04, 0x00, 0x00, 0x28, 0x00, 0xae,

    /* CHARACTERISTIC,  ae01, WRITE_WITHOUT_RESPONSE | DYNAMIC, */
    // 0x0005 CHARACTERISTIC ae01 WRITE_WITHOUT_RESPONSE | DYNAMIC
    0x0d, 0x00, 0x02, 0x00, 0x05, 0x00, 0x03, 0x28, 0x04, 0x06, 0x00, 0x01, 0xae,
    // 0x0006 VALUE ae01 WRITE_WITHOUT_RESPONSE | DYNAMIC
    0x08, 0x00, 0x04, 0x01, 0x06, 0x00, 0x01, 0xae,

    /* CHARACTERISTIC,  ae02, NOTIFY, */
    // 0x0007 CHARACTERISTIC ae02 NOTIFY
    0x0d, 0x00, 0x02, 0x00, 0x07, 0x00, 0x03, 0x28, 0x10, 0x08, 0x00, 0x02, 0xae,
    // 0x0008 VALUE ae02 NOTIFY
    0x08, 0x00, 0x10, 0x00, 0x08, 0x00, 0x02, 0xae,
    // 0x0009 CLIENT_CHARACTERISTIC_CONFIGURATION
    0x0a, 0x00, 0x0a, 0x01, 0x09, 0x00, 0x02, 0x29, 0x00, 0x00,

    /* CHARACTERISTIC,  ae03, WRITE_WITHOUT_RESPONSE | DYNAMIC, */
    // 0x000a CHARACTERISTIC ae03 WRITE_WITHOUT_RESPONSE | DYNAMIC
    0x0d, 0x00, 0x02, 0x00, 0x0a, 0x00, 0x03, 0x28, 0x04, 0x0b, 0x00, 0x03, 0xae
```

3、广播配置初始化

接口：void advertisements_setup_init()

功能：定义广播周期、广播包内容和扫描应答包的内容。

广播周期参数的是以 0.625ms 为单位，如下图：

```
#define ADV_INTERVAL_MIN          160
#define ADV_INTERVAL_MAX          160
```

广播包组成，用户可修改添加类型，如下图：

```
static int make_set_adv_data(void)
{
    u8 offset = 0;
    u8 *buf = adv_data;

    offset += make_eir_packet_val(&buf[offset], offset, HCI_EIR_DATATYPE_FLAGS, 6, 1);
    offset += make_eir_packet_val(&buf[offset], offset, HCI_EIR_DATATYPE_COMPLETE_16BIT_SERVICE_UUIDS, 0x1812, 2);

    if (offset > sizeof(adv_data)) {
        puts("***adv_data overflow!!!!!!\n");
        return -1;
    }
    /* printf_buf(adv_data,offset); */
    gap_advertisements_set_data(offset, (uint8_t *)adv_data);
    return 0;
}
```

扫描应答包组成，用户可修改添加类型，如下图：

```
static int make_set_rsp_data(void)
{
    u8 offset = 0;
    u8 *buf = scan_rsp_data;

    u8  tag_len = sizeof(user_tag_string);
    offset += make_eir_packet_data(&buf[offset], offset, HCI_EIR_DATATYPE_MANUFACTURER_SPECIFIC_DATA, (void *)user_tag_string, tag_len);

    u8 name_len = strlen(gap_device_name);
    u8 vaild_len = ADV_RSP_PACKET_MAX - (offset + 2);
    if (name_len > vaild_len) {
        name_len = vaild_len;
    }
    offset += make_eir_packet_data(&buf[offset], offset, HCI_EIR_DATATYPE_COMPLETE_LOCAL_NAME, (void *)gap_device_name, name_len);

    if (offset > sizeof(scan_rsp_data)) {
        puts("***rsp_data overflow!!!!!!\n");
        return -1;
    }
    /* printf_buf(scan_rsp_data,offset); */
    gap_scan_response_set_data(offset, (uint8_t *)scan_rsp_data);
    return 0;
}
```

4、协议栈事件处理

接口：void cbk_packet_handler(...)

功能：处理协议栈回调的事件消息处理，例如连接断开，发送完成等。

协议栈初始化完成事件处理，如下图：

```
case BTSTACK_EVENT_STATE:
    if (btstack_event_state_get_state(packet) != HCI_STATE_WORKING) {
        break;
    }
    log_info("init complete\n");
    set_ble_work_state(BLE_ST_IDLE);
```

连接成功事件处理，确定设备已被连上，如下图：

```
case HCI_SUBEVENT_LE_CONNECTION_COMPLETE: {
    set_adv_enable(0, 0);
    con_handle = little_endian_read_16(packet, 4);
    log_info("HCI_SUBEVENT_LE_CONNECTION_COMPLETE : %0x\n", con_handle);
```

连接断开事件处理，确定设备已断开连接，如下图：

```
case HCI_EVENT_DISCONNECTION_COMPLETE:
    log_info("---ble disconnect!\n");
    ble_test_flag = 0;
    con_handle = 0;
```

5、read 和 write 回调处理

当 profile 的 cfg 配置文件中 read 和 write 定义属性有 DYNAMIC 关键字时，会有回调处理。

接口：uint16_t att_read_callback(...)

功能：根据 read 操作 handle，返回对应的读的内容。

当 buffer 为 NULL，返回 read 属性的总长度。

当 buffer 不为 NULL，返回 read 获取的内容。

```
static uint16_t att_read_callback(hci_con_handle_t connection_handle, uint16_t att_handle, uint16_t offset,
```

接口：int att_write_callback(...)

功能：根据 write 操作 handle，接收 write 的内容，包括使能开关 notify&indicate 功能。

```
static int att_write_callback(hci_con_handle_t connection_handle, uint16_t att_handle, uint16_t transaction_mode,
```

6、连接参数更新

从机可以请求更改连接的参数，但必须经过主机同意才能更改成功。

主要分 3 步如下：

（1）从机发参数请求处理，如下图：

```
static void send_request_connect_parameter(u8 table_index)
{
    struct conn_update_param_t *param = (void *)&connection_param_table[table_index];

    log_info("update_request:-%d-%d-%d-%d-\n", param->interval_min, param->interval_max, param->latency, param->timeout);
    if (con_handle) {
        gap_request_connection_parameter_update(con_handle, param->interval_min, param->interval_max, param->latency, param->timeout);
    }
}
```

```
static const struct conn_update_param_t connection_param_table[] = {
    {16, 24, 0, 600},//11
    {12, 28, 0, 600},//3.7
    {8,  20, 0,  600},
```

连接参数 interval 是以 1.25ms 为单位，timeout 是以 10ms 为单位；用户慎重修改，必须按照蓝牙的协议规范来定义修改。

（2）主机响应请求结果，事件回调，如下图：

```
case L2CAP_EVENT_CONNECTION_PARAMETER_UPDATE_RESPONSE:
    tmp = little_endian_read_16(packet, 4);
    log_info("-update_rsp: %02x\n", tmp);
    if (tmp) {
        connection_update_cnt++;
        log_info("remoter reject!!!\n");
        check_connetion_updata_deal();
    } else {
        connection_update_cnt = CONN_PARAM_TABLE_CNT;
    }
    break;
```

（3）连接参更改成功，事件回调，如下图：

```
case HCI_SUBEVENT_LE_CONNECTION_UPDATE_COMPLETE:
    connection_update_complete_success(packet);
    break;
```

7、对外提供接口调用

接口有广播使能，断开连接，获取发送 buffer 空间，发送数据等，详细查看接口具体实现。

```
struct ble_server_operation_t {
    int(*adv_enable)(void *priv, u32 enable);
    int(*disconnect)(void *priv);
    int(*get_buffer_vaild)(void *priv);
    int(*send_data)(void *priv, void *buf, u16 len);
    int(*regist_wakeup_send)(void *priv, void *cbk);
    int(*regist_recieve_cbk)(void *priv, void *cbk);
    int(*regist_state_cbk)(void *priv, void *cbk);
};
void ble_get_server_operation_table(struct ble_server_operation_t **interface_pt);
```

```
static const struct ble_server_operation_t mi_ble_operation = {
    .adv_enable = set_adv_enable,
    .disconnect = ble_disconnect,
    .get_buffer_vaild = get_buffer_vaild_len,
    .send_data = (void *)app_send_user_data_do,
    .regist_wakeup_send = regiest_wakeup_send,
    .regist_recieve_cbk = regiest_recieve_cbk,
    .regist_state_cbk = regiest_state_cbk,
};
```

注意：数据发送接口使用 server 的 notify&indicate 通知功能来实现，需要主机 client 端使能开通知，才能发送出去数据。

16.5 BLE 的 CLIENT 角色

对应文件 le_client_module.c，负责模块初始化、处理协议栈事件和命令数据控制发送等；client 角色在 SDK 中是以主机 master 的方式实现，主动发起搜索和连接其他 BLE 设备。搜索 server 的 profile 列表数据。

SDK 的例子是以杰理的 server 中 profile 为搜索目标，用户根据需求也可自行搜索过滤 profile。

1、协议栈配置初始化

接口：void ble_stack_config_init(void)

功能：定义协议栈配置初始化，注册事件处理和流程处理。

2、协议栈事件处理

接口：void cbk_packet_handler(...)

功能：处理协议栈回调的事件消息处理，例如连接断开，发送完成等。

协议栈初始化完成事件，如下图：

```
case BTSTACK_EVENT_STATE:
    if (btstack_event_state_get_state(packet) != HCI_STATE_WORKING) {
        break;
    }
    log_info("init complete\n");
    set_ble_work_state(BLE_ST_IDLE);
    client_scan_set_param(1, SET_SCAN_INTERVAL, SET_SCAN_WINDOW);
    ACT_SET_FLAG(CLI_SCAN_ENABLE);
    break;
```

连接成功事件处理，确定设备已被连上，如下图：

```
case HCI_SUBEVENT_LE_CONNECTION_COMPLETE: {
    if (tc_state != TC_W4_CONNECT) {
```

连接断开事件处理，确定设备已断开连接，如下图：

```
case HCI_EVENT_DISCONNECTION_COMPLETE:
    tc_state = TC_IDLE;
    gc_handle = 0;
    set_ble_work_state(BLE_ST_IDLE);
    client_puts("client disconn\n");
    ACT_SET_FLAG(CLI_SCAN_ENABLE);
```

扫描返回广播事件处理，如下图：

```
case GAP_EVENT_ADVERTISING_REPORT:
    /* client_puts("GAP_LE_ADVERTISING_REPORT\n"); */
    /* putchar('V'); */
    if (tc_state != TC_W4_SCAN_RESULT) {
        client_printf("SCAN ACTIVE err,%d\n", tc_state);
        break;
    }
    client_report_adv_data((void *)&packet[2], packet[1]);
    break;
```

3、创建连接控制

使能扫描后，可以根据搜索的设备广播信息，可以发起连接。

设备信息返回接口 void client_report_adv_data(adv_report_t *report_pt, u16 len) 。

创建连接可根据名字，地址，标识等发起连接对应的设备，用户可修改创建连接条件，如下图是可选择的条件：

```
enum {
    CLI_CREAT_BY_ADDRESS = 0,
    CLI_CREAT_BY_NAME,
    CLI_CREAT_BY_TAG,
    CLI_CREAT_BY_LAST_SCAN,
};
```

```
static const u8 create_conn_mode = BIT(CLI_CREAT_BY_LAST_SCAN);// BIT(CLI_CREAT
static const u8 create_conn_remoter[6] = {0x11, 0x22, 0x33, 0x88, 0x88, 0x88};
static const u8 create_remoter_name[32] = "AC692x_testAa18";
static u8 adv_last_remoter_name[32];
static u8 conn_last_remoter_flag = 0;
static u8 create_conn_flag = 0;
```

找到对应的设备，可以发起创建连接的动作，如下图：

```
if (find_remoter) {
    if (create_conn_flag) {
        puts("\n*********creat_connection**********\n");
        printf("***remote type %d,addr:", report_pt->address_type);
        put_buf(report_pt->address, 6);
        puts("\n");
        create_conn_flag = 0;
        client_creat_connection(report_pt->address, report_pt->address_type);
        ACT_SET_FLAG(CLI_SCAN_DISABLE);
    }
}
```

4、查找对应的 UUID 和 HANDLE

client 端会搜索目标的 UUID，记录对应 HANDLE，之后会使用 HANDLE 进行对 server 服务器的 read、write、notify 和 indicate 等操作。若有 notify&indicate，会自动使能打开通知功能。

搜索服务分 4 步操作，如下述。

（1）设置搜索目标的 UUID，如下图：

```
static const target_service_t target_services = {
    .service_uuid16 = 0xae00,
    .read_uuid16 = 0xae10,
    .write_uuid16 = 0xae01,
    .notify_uuid16 = 0xae02,
    .indicate_uuid16 = 0,
};
```

（2）启动服务搜索操作，如下图：

```
static u16  client_search_profile_start(u8 search_pfl_type, u32 search_data)
{
    u16 res = 0;
```

（3）搜索 profile 的过程中事件处理，如下图：

```
static void s_client_gatt_event(uint8_t packet_type, uint16_t channel, uint8_t *packet, uint16_t size)
{
    u32 tmp32, ret_type;
    /* printf("client cbk gatt event %x \n",tc_state);   */
    switch (tc_state) {
```

（4）搜索 profile 完成操作，如下图：

```
static void client_search_profile_complete(void)
{
    u16 mtu;
```

5、扫描参数和连接参数配置

扫描参数以 0.625ms 为单位，设置如下图：

```
#define SET_SCAN_WINDOW     16
#define SET_SCAN_INTERVAL   48
```

连接参数 interval 是以 1.25ms 为单位，timeout 是以 10ms 为单位，如下图：

```
#define SET_CONN_INTERVAL_MIN   0x10
#define SET_CONN_INTERVAL_MAX   0x20
#define SET_CONN_TIMEOUT        0x180
```

以上两组参数请慎重修改，必须按照蓝牙的协议规范来定义修改。

6、对外提供接口调用

接口有搜索使能，断开连接，获取发送 buffer 空间，发送数据等，详细查看接口具体实现。

```
struct ble_client_operation_t {
    int(*scan_enable)(void *priv, u32 enable);
    int(*disconnect)(void *priv);
    int(*get_buffer_vaild)(void *priv);
    int(*write_data)(void *priv, void *buf, u16 len);
    int(*read_do)(void *priv);
    int(*regist_wakeup_send)(void *priv, void *cbk);
    int(*regist_recieve_cbk)(void *priv, void *cbk);
    int(*regist_state_cbk)(void *priv, void *cbk);
};
void ble_get_client_operation_table(struct ble_client_operation_t **interface_pt);
```

```
static const struct ble_client_operation_t client_operation = {
    .scan_enable = app_client_scan_enable,
    .disconnect = app_client_disconn,
    .get_buffer_vaild = app_client_get_buffer_vaild_len,
    .write_data = (void *)app_client_write_without_respond_send,
    .read_do = (void *)app_client_read_send,
    .regist_wakeup_send = app_client_regiest_wakeup_send,
    .regist_recieve_cbk = app_client_regiest_recieve_cbk,
    .regist_state_cbk = app_client_regiest_state_cbk,
};


void ble_get_client_operation_table(struct ble_client_operation_t **interface_pt)
{
    *interface_pt = (void *)&client_operation;
}
```

注意：数据发送接口使用 client 的 write 功能来实现。

16.6 BLE 的 ATT 发送机制

对应文件 att_send.c，兼容 server 和 client 角色的数据发送调用，负责 ATT 的命令 read、write、notify 和 indicate 等命令发送控制。工作原理主要是使用 cbuff 缓存 ATT 的发送命令，根据协议栈的发送机制实时填入数据发送。

1、配置 cbuff 的缓存大小，根据需求可修改大小，如下图：

```
#define USER_BUF_LEN          (512*4)      //发送buf大小，可根据需求修改
cbuffer_t user_send_cbuf;
static u8 user_data_buf[USER_BUF_LEN] __attribute__((aligned(4)));
```

2、获取 cbuff 剩余空间大小，如下图：

```
u32 user_data_cbuf_is_write_able(u32 len)
{
    u32 buf_space, w_len;
    u16 head_size = sizeof(user_send_head_t);
    u16 pack_size = sizeof(user_send_head_t) + att_payload_size;

    if (!att_conn_handle) {
        return 0;
    }
```

3、获取 cbuff 是否为空，如下图：

```
u32 user_data_cbuf_is_null(void)
{
    return !cbuf_get_data_size(&user_send_cbuf);
}
```

4、用户数据发送接口，如下图：

```
u32 user_data_att_send(u16 handle, u8 *data, u16 len, u8 send_type)
{
    u16 wlen;
    u32 ret_val;
    user_send_head_t send_head;
    u16 packet_cnt = 0;
```

附录 A　AC692x 与 AC690x、AC460x 功能差异

AC692x与AC690x、AC460x功能差异				
序号	功能	AC692x	AC690x	AC460x
1	内置FM模式下蓝牙跑后台	不支持	支持	支持
2	K歌宝：混响模式下MIC和AUX是否支持同时录音	不支持	支持	支持
3	MIC和AUX是否支持分开独立调音量	不支持	支持	支持
4	内置充电	不支持	支持	不支持
5	对耳，对箱	支持	支持	不支持
6	DC-DC	不支持	支持	不支持
7	NFC	不支持	支持	支持
8	在线调EQ	支持	支持	不支持
9	普通IO口内部上下拉	上拉10K，下拉10K	上拉10K，下拉60K	上拉10K，下拉60K
10	SPDIF（数字音频输出）	支持	不支持	不支持

附录 B　JL 蓝牙通话调试说明

文档说明

该文档用来说明如何调试蓝牙通话，适用于绝大部分情景。

调试说明

1、 **dac_analog_gain**：调试近端样机喇叭音量大小（*模拟增益*）

2、 **mic_analog_gain**：调试远端手机听到的音量大小（*模拟增益*）

3、**NDT_max_gain**：远端手机听到声音音量最大数字增益（*放大倍数=NDT_MAX_gain/64*）

NDT_min_gain：远端手机听到声音音量最小数字增益（*放大倍数=NDT_min_gain/64*）

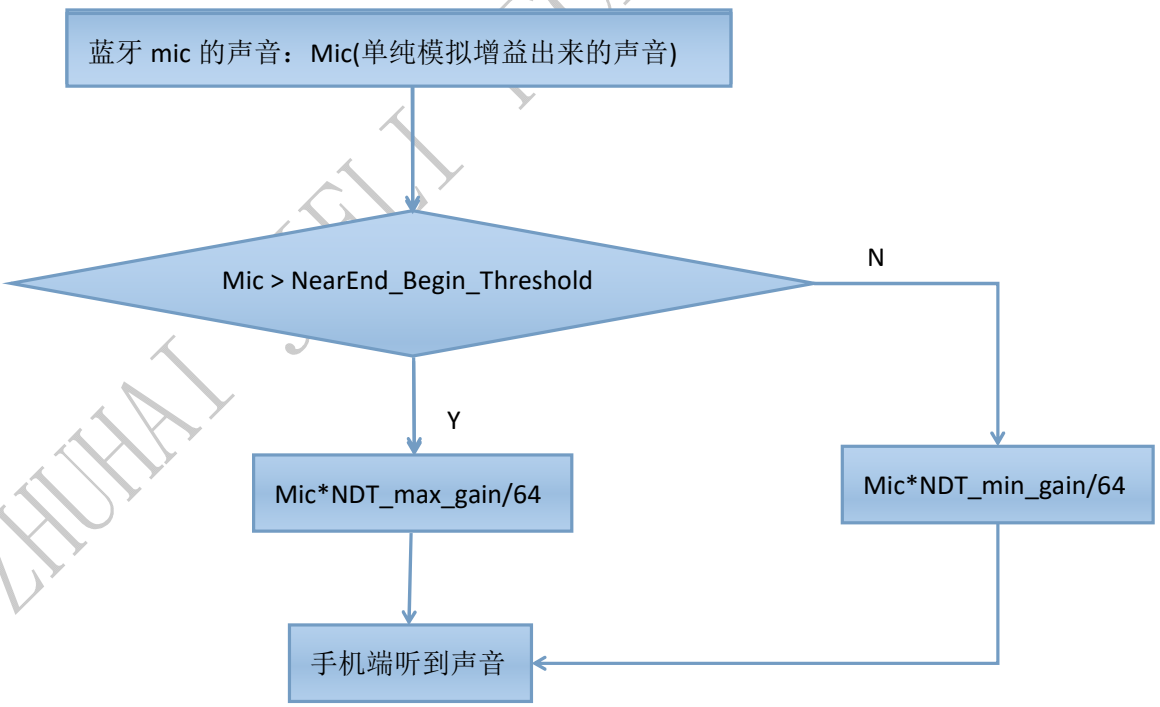
NDT_Fade_Speed：淡入淡出速度

NearEnd_Begin_Threshold：放大阈值

以上四个参数属于数字参数，配套一起使用，用来调节远端手机听到的声音效果。之所以不能直接使用 mic 的模拟增益来控制音量大小，是因为 mic 的模拟增益太大，1 是回音太大，不好处理；2 是周围的环境声音容易被采集到，影响听感。

注意：mic 的增益也不能太小，太小清晰度不够

操作流程如下：



NearEnd_Begin_Thrashold 阈值太大，容易放不大，太小远端手机容易听到近端周围环境声。调试过程以 10~20 为单位加减，具体多大跟 mic 的模拟增益有关。

NDT_Fade_Speed 表示 NDT_Max_gain 和 NDT_Min_gain 之间淡入淡出的速度

4、aec_ctl

AEC_ADVANCE:使能全部运算模块，适用于回声大的方案，比如音箱

AEC_REDUCE:关闭部分运算模块（节省功耗，提升双工效果），适用于回声小的方案，比如耳机

5、**suppress_coeff1**：回音压制粗调（压制级数），效果较为明显

这个值越大，回声压制越大，相应的通话也容易出现断断续续，因为人讲话的声音也有可能被压制；这个值越小，回声压制越小，相应的通话效果也会提高，变得均匀，太小的话有可能有回音。

调试过程以 1 位单位加减，建议调试范围（0~5）

6、**suppress_coeff2**：回音压制细调（清零阈值），主要用来优化通话效果

这个值的效果跟 suppress_coeff1 一样，该值大小跟回音压制强度成正比。

如果是耳机方案，这个参数应该设置为 0，才能达到双工效果。

调试过程以 100 为单位加减，建议范围（500~1500）

注意 1：以上参数中只有参数 1（dac_analog_gain）用来调节近端样机喇叭音量大小的，其余的都是用来调节远端手机听到的效果。

注意 2：远端手机端听到的音量大小可以通过模拟增益和数字增益调试，区别是模拟增益可以提高清晰度和辨析度，太大会有回音或者容易听到近端环境声。数字增益不会带来回音，但是如果模拟增益太小，仅仅通过数字增益来放大，清晰度会差一点。