CI130X 系列离在线大模型前端方案

文件历史跟踪 DOCUMENT HISTORY PAGE			
版本号 Rev. NO.	发起者 Originator	描述 Description	日期 Date
0.1	启英泰伦	新建文档	2025/05/20
0.2	启英泰伦	1、新增 iis_sample、hpout_sample 2、新增大模型采音以及 AEC 调试文档、OTA v4 文档链接 3、新增 G722 编码解码 4、新增常见问题解决提示	2026/01/23

目录

一、方案介绍

1.1 方案背景与目标

25 年初，随着 DeepSeek 的广泛应用，大量的产品新增对在线大模型的需求，启英泰伦作为 BNPU 的发明者，采用 BNPU 的高效 AI 计算能力为在线大模型方案提供即时、干净、准确的语音入口。

1.2 硬件方案框架

本方案中，启英泰伦 AI 语音芯片 CI130X 系列（CI1303/CI1302/CI1306） 与 wifi 通过高速串口进行数据交互，CI130X 负责本地唤醒、降噪后的声音压缩上传（G722/SPEEX/OPUS），流媒体播放（PCM/G722/MP3），播放打断，VAD 端点检测等算法功能，wifi 负责将语音数据发到云端及将云端音频及控制数据下发播放，默认使用 uart_sample（音频数据串口上下行）。

方案中还提供了 hpout_sample（音频数据 hpout 上行）和 iis_sample（音频数据 iis 上下行)供用户选择。目前 iis_sample 只支持单麦，iis 格式为 16khz、16bit、双通道、wifi 端 iis 输入作从机，输出作主机，下行播放音频时，需给语音端发送开始播放、停止播放指令，控制声卡开关避免底噪影响体验；hpout_sample 不支持播报，需要 wifi 端进行播报，并给语音端发送播放状态，控制 AEC 处理。各个 sample 具体框图如下：

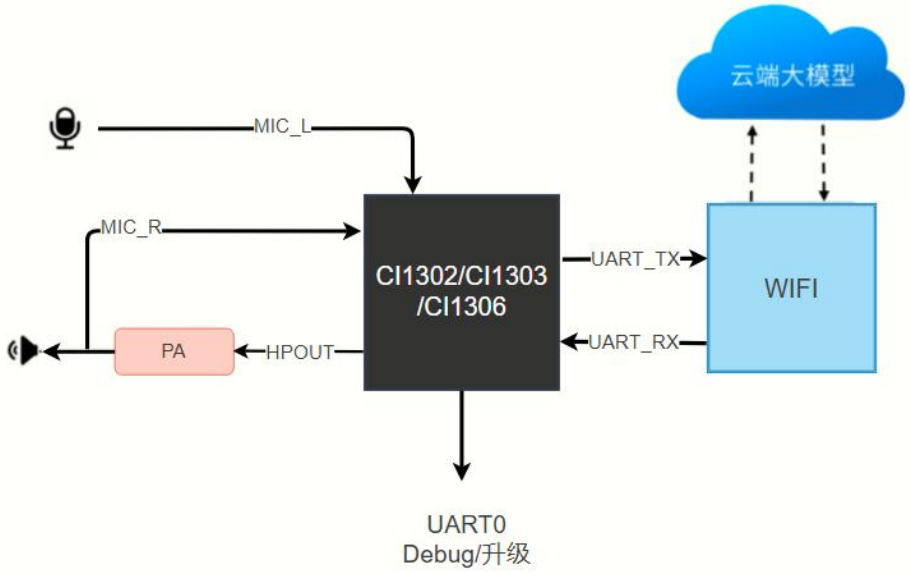


图 1　CI130X+wifi（uart_sample 单麦）离在线大模型方案框图

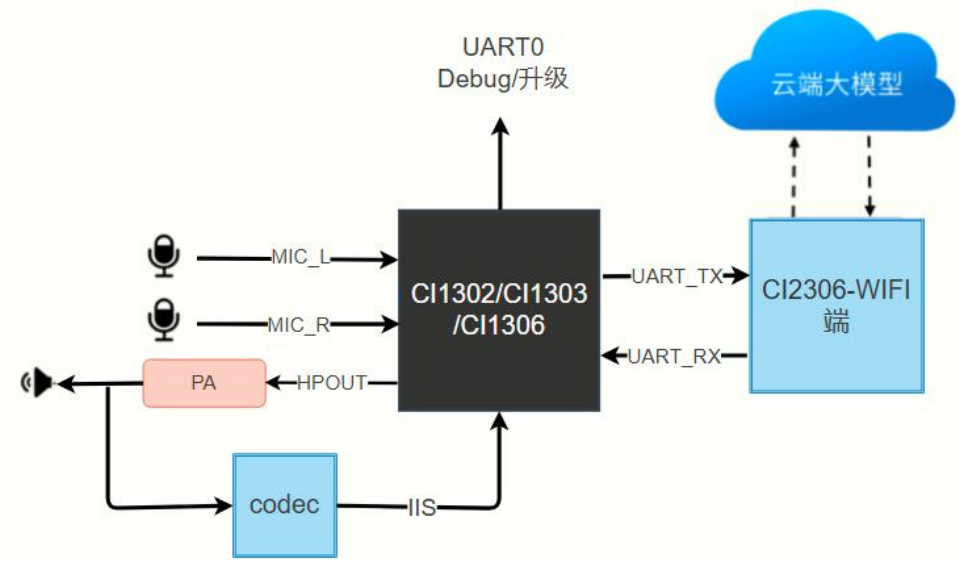


图 2　CI130X+wifi（uart_sample 双麦）离在线大模型方案框图

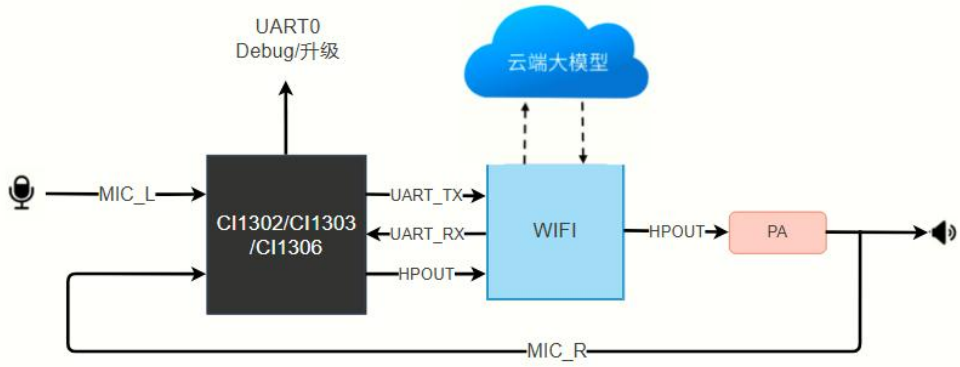


图 3　CI130X+wifi（hpout_sample 单麦）离在线大模型方案框图

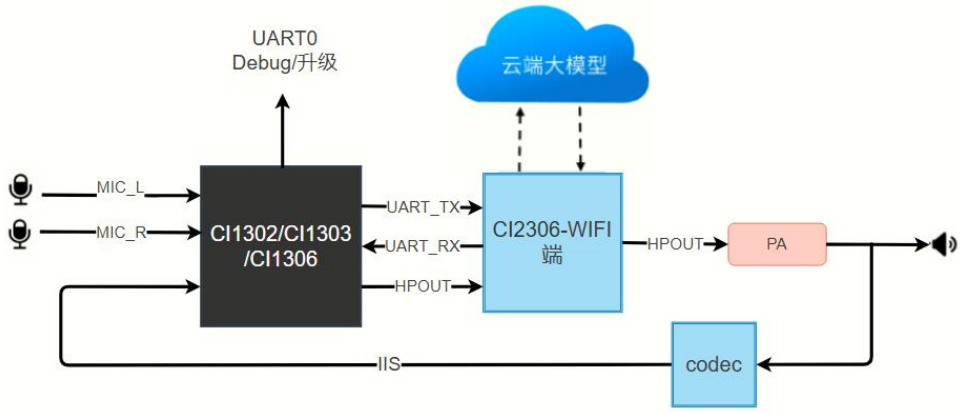


图 4　CI130X+wifi（hpout_sample 双麦）离在线大模型方案框图

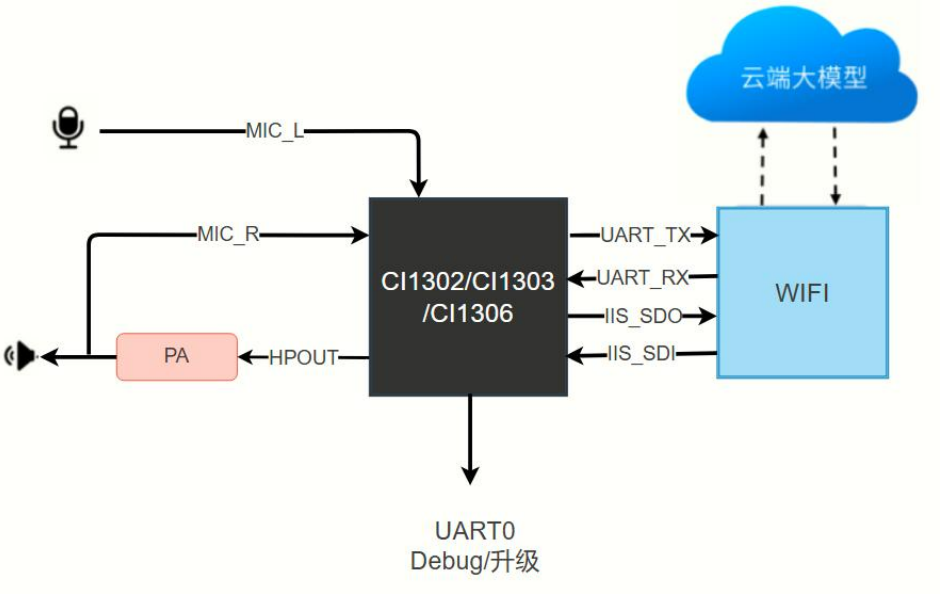


图 5　CI130X+wifi（iis_sample 单麦）离在线大模型方案框图

1.3 130X 软件功能框图

该方案的软件功能运行框图如下：

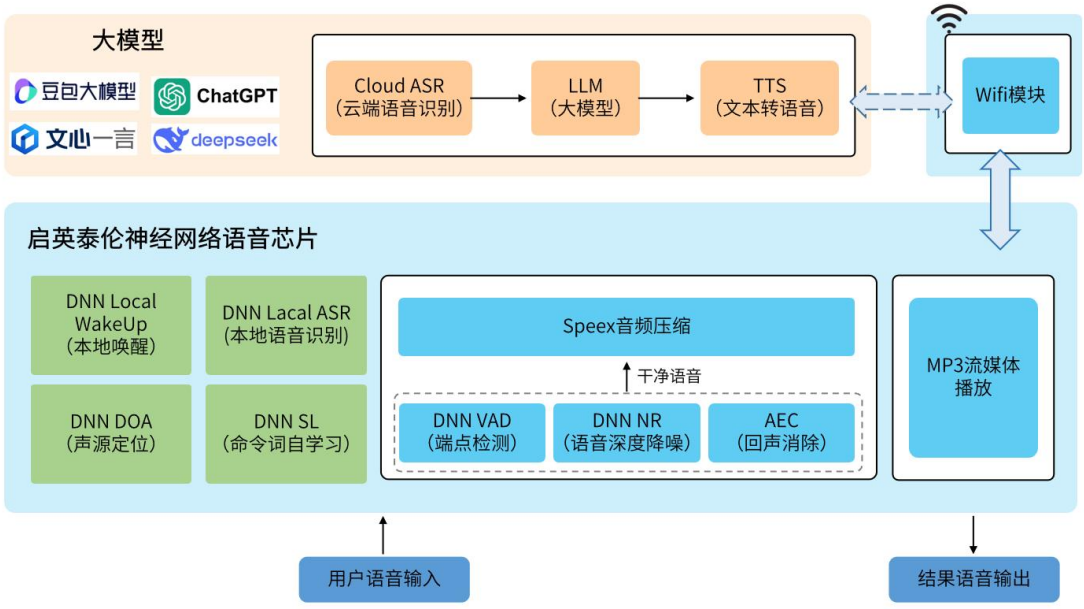


图 6　130X 软件功能框图

在离在线语音对话大模型方案应用中，CI130X 的主要功能特点如下：

1、DNN NR：神经网络语音降噪，可以过滤稳态噪声，瞬态噪声及远距离的背景人声，提供给云端干净的人声。

2、DNN VAD：神经网络端点检测，精准的人声检测，使 wifi 只上传需要的人

声，降低云端的服务成本和数据带宽

3、DNN AEC：神经网络回声消除，过滤掉喇叭播放的声音，保留干净的说话人声；

4、DNN Local Wakeup/ASR：神经网络的本地唤醒和识别，可以识别本地唤醒词和命令词，使产品在未联网时，也具备基本的离线语音控制功能。

5、DNN DOA（选配）：神经网络的声源定位，基于双麦克支持 0~180 度，精度±10 度。

6、Speex/Opus/G722/PCM 音频压缩：用户可以选择 Speex 或 Opus 其一进行压缩上传；

7、MP3/G722/PCM 音频播放：用户可以选择 MP3、G722、PCM 播放，或者 MP3+PCM 播放，或者 MP3+G722；

8、支持唤醒词用户自学习（选配）：允许用户通过对话的方式更改唤醒词，让产品更好玩；

9、支持唤醒打断和全双工自然对话：联合 AEC 功能消除喇叭播放的声音，进行唤醒识别和自然对话。

10、可以支持十多个国家的小语种识别；

备注：如采用双麦 DNN DOA 功能，在打开声源角度识别识别功能时候，硬件需要外加音频 ADC（推荐 ES7243）做 AEC 的参考声音回采；并且因内存原因不支持唤醒词自学习

1.4 对 wifi 的要求

1. 支持协议：根据云端接口情况，一般需要支持 http，mqtt，websocket 格式
2. Wifi 主要完成功能
   A、将语音的声音进行缓存及上传到云端
   B、通过网络协议对接到云端
   C、云端回复的内容缓存及下发到 CI130X 进行播放

1.5 对话方式描述

对整机离在线大模型的产品来说，有以下 2 种启动对话方式：

1.5.1 按键方式

1、按下按键（长按）后，设别开始上传声音，按键松开结束声音上传，云端经过大模型生成播放内容下发后播放，按下按键进行播放打断，并开启新的上传声音到大模型识别的流程

2、短按按键后，开始上传声音，当人讲完话后，结束声音上传，云端经过大模型生成播放内容下发后播放，短按按键进行播放打断，并开启新的上传声音到大模型识别的流程

1.5.2 对话方式

下图为本大模型方案支持的几种对话模式（开发者可进行配置选择或者通过串口协议选配）：

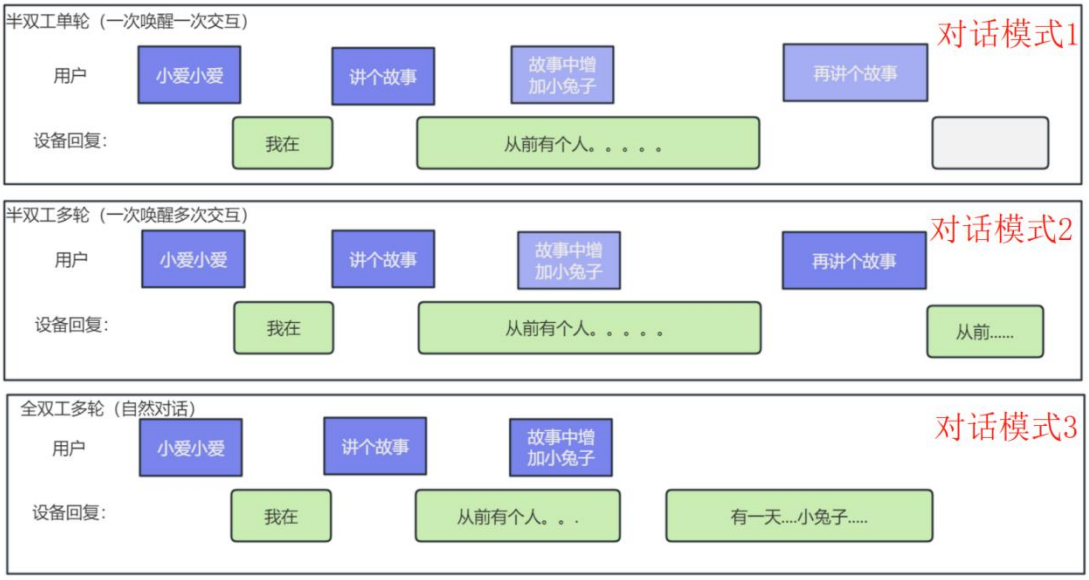


图 7：多种对话模式：模式 1 半双工单轮对话，模式 2 半双工多轮，模式 3 全双工多轮自然对话

对话模式 1：单轮对话，一次唤醒一次交互，每次交互完后，必须唤醒词才能说下；

对话模式 2【推荐】：单轮对话，一次唤醒多次交互，每次交互完后，可以继续进行新的对话交互；

对话模式 3：全双工多轮自然对话，一次唤醒多次交互，在设备播放云端的声音的时候，用户说的话也可以上传云端进行识别；

备注：全双工多轮自然对话对应用环境有要求，如果环境多人声，则可能出现人声一直打断云端内容播放的情况，导致体验下降，所以产品中建议该功能作为 APP 配置的可选功能，默认不开启；

二、开发准备工作

开发前，请先准备好相关如下硬件，软件，如涉及量产，请从查看文档中心生产及联系启英获取生产支持：

2.1 开发硬件准备

1、本方案建议使用硬件模块 D02GS01J 进行：

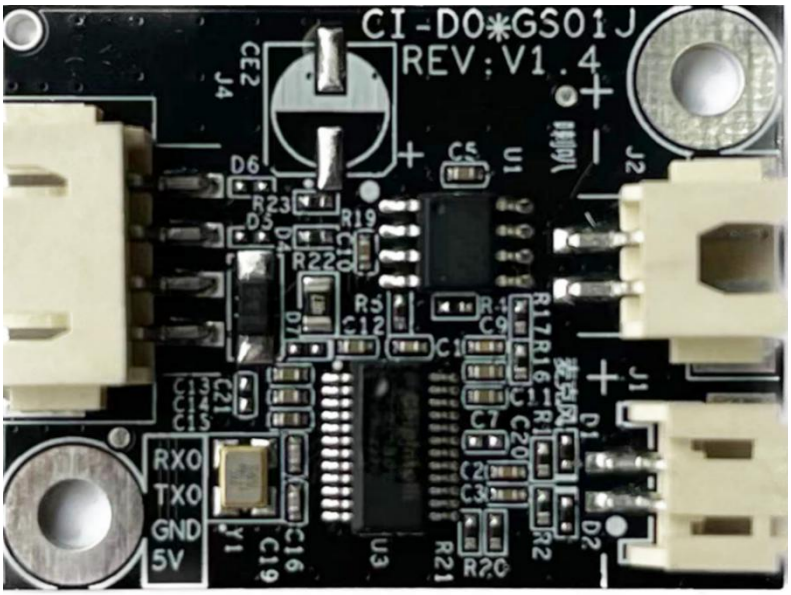


图 2.1 D02GS01J 实物图

备注：本模块贴不同芯片，对应的模块编号不同，总共 3 中型号：D02GS01J（贴 1301），D02GS01J（1302），D03GS01H（1303），点击获取：D02GS01J 模块板说明、点击获取购买链接

2、下图是串口烧录工具

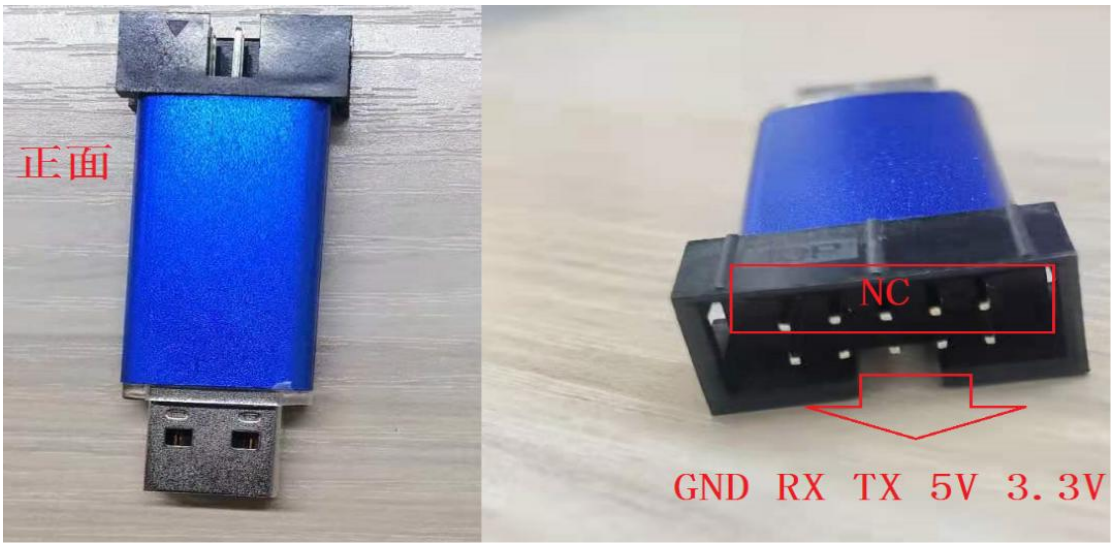


图 2.2 串口工具实物图

备注：开发者可以采用通用的串口工具进行固件烧录和查看日志，或进行点击购买链接

特别注意：为了数据稳定，尽量选择带晶振的串口，推荐选择 CH341 的串口（驱动下载链接 

）

3、采音板，用于分析产品的底噪及声音情况

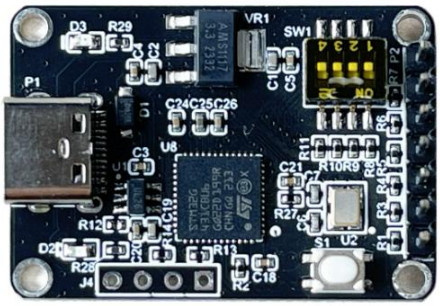


图 2.3 采音板实物图

本采音板主要是通过 IIS 采音和 uart 对被测试语音模块进行采音分析（需要特定的软件配置编译的固件），使用方法（需要增加 uart 采音的资料）

注意：

1、本离在线大模型前端方案中，因系统资源原因，本固件无法使用该采音板

2、关闭大模型前端的部分功能，开启对应宏定义，可以进行录音判断硬件底噪和参考信号灯情况

2.2 软件准备

1、开发环境搭建：VScode 下载及使用：

2、SDK 代码包获取：在启英泰伦 AI 平台开发资料里搜索 SDK CI13XX_SDK_LLM_AIOT_V2.0.10

3、启英泰伦 AI 平台账号：用于命令词模型的生成，播放音的生成等，注册后认证企业用户，可以获取更多功能

4、误识测试工具：用于分析误识别，误唤醒的时候的优化；**使用方法带补充**

GetCmdVoice_20250427.rar

5、录音及分析工具：Audacity 或其他工具

6、串口日志工具：SSCOM 或者串口调试工具 或其他开发者习惯的串口工具主要用于 debug 查看日志

三、开发验证准备

3.1 用于对话大模型前端语音处理开发的 CI1302 语音模块板

开发者可以采用基于 CI1302 的 D02GS01J 模块板，用于大模型前端语音处理方案的开发；通过飞线与 wifi 连接用于快速开发和验证；软件验证好后烧录到 CI1302，可将 CI1302 和 wifi 贴在一起进行量产。

下图是 D02GS01J 的模块板示意框图：

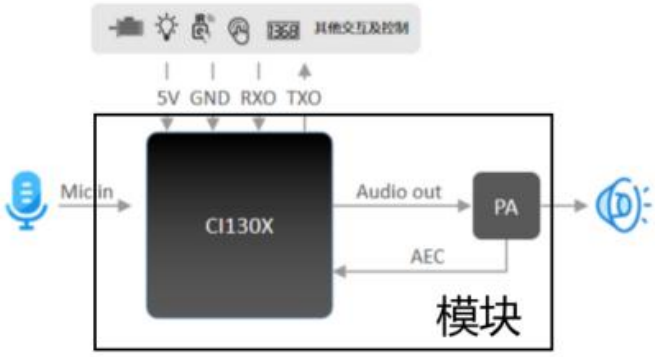


图 4 D02GS01J 的模块板示意图

该模块板的支持单路麦克接入，支持 AEC 回声消除，5V 供电，UART 串口通讯，支持 8 欧 2W 的喇叭；

下图是 D02GS01J 模块板实物图【模块板资料链接】：

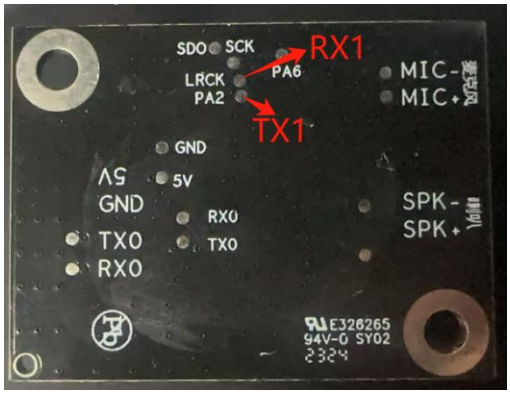


图 5a D02GS01J 模块板反面

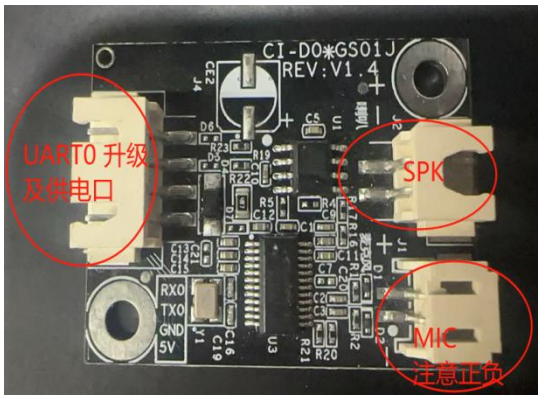


图 5b、D02GS01J 模块板正面

方案可以使用该模块和 wifi 进行串口调试

图 5a 中的 tx1，rx1 通过焊接跳线连接到 wifi 的 uart 串口的 rx 和 tx 脚；默认速度是 921600bps

图 5b 中 uart0 用于升级和 log 打印，同时也是语音模块的供电口，默认 921600bps

图 5b 中的 SPK 用于接喇叭；

图 5b 中的 MIC 用于接麦克风，将麦克风的“+”对应接口的“+”，避免接反影

响识别效果；
连接后的实物图如下：

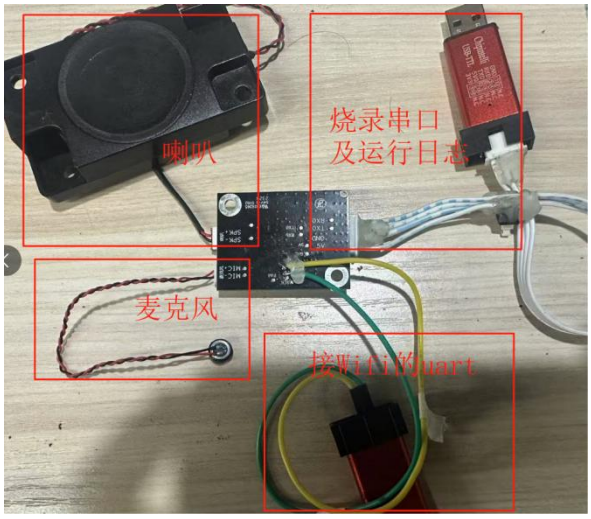


图 6 D02GS01J 模块板连接实物图

备注：如果使用其他的模块如 D02GS02S（须注意代码中的板级配置也要配置成这个模块），仍参考上图连接，uart0 升级，uart1 接 wifi 进行通讯；

四、对话大模型方案 CI1302 语音前端处理部分的硬件参考设计：

4.1 对话大模型前端语音处理参考原理图

本应用大部分为电池类供电，需要考虑功耗，故需要使用电池+dcdc 进行方案供电，使用串口 1 与 wifi 进行 uart 串口对接；

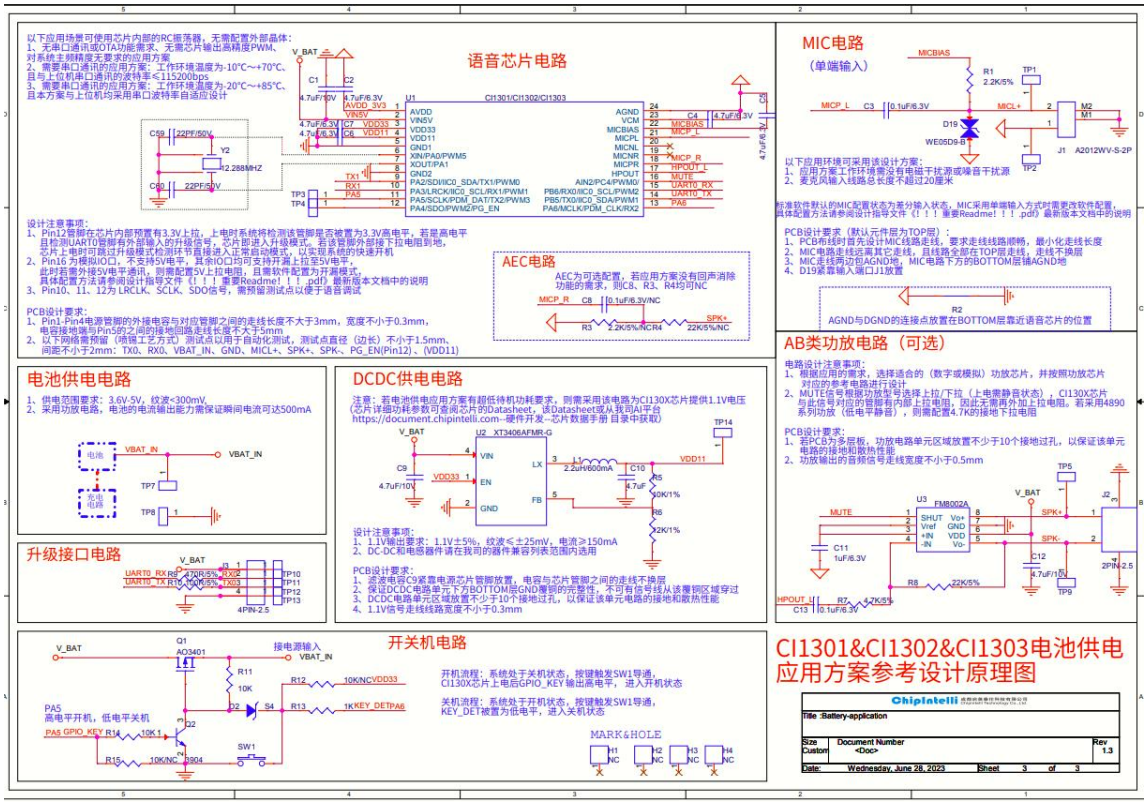


图 7 大模型前端语音处理参考原理图

原理图中以下部分需要注意：

4.1.1 麦克风的单端线路和差分线路

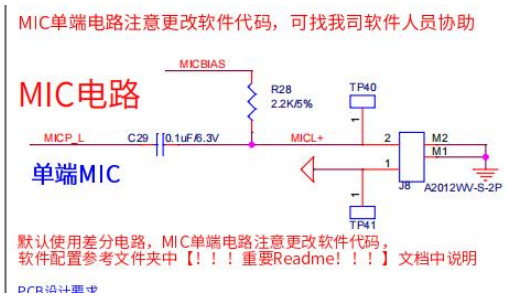


图 8 麦克风的单端线路

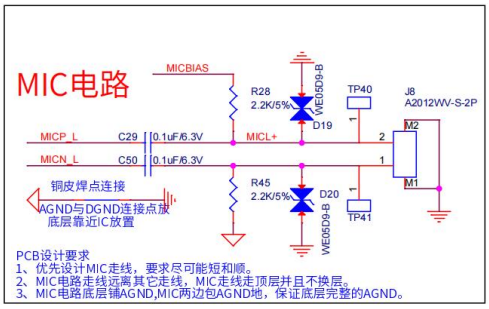


图 9 麦克风的差分线路

图 8：单端线路，麦克“+”信号接语音芯片的 MICP_L 麦克的“-”接 GND，此线路适合 A、硬件干扰小 B、硬件干扰大同时使用屏蔽线麦克的应用场景；

图 9：差分线路（推荐），麦克的“+”接语音芯片的 MICP_L，麦克的“-”接语音芯片的 MICP_N，在语音芯片内进行差分运算，可以消除常见的一些共模干扰（例如一个信号干扰到麦克“+”和“-”，差分运算后就将该干扰信号消除了）,此线路+双绞线麦克，可以解决绝大部分的通用外部干扰

注意事项：两种方式麦克风线路在语音芯片内接收到的信号幅度存在差异，

软件为保证识别效果，需要调整宏定义；（user_config.h 文件）

```
//**麦克风电路模式配置
#define MIC_DIFF_SINGLE                 0    /*1,单端。0，差分
```

4.1.2 uart0 预留升级线路

Uart0 为升级和 debug 的日志口，需要预留便于升级和调试

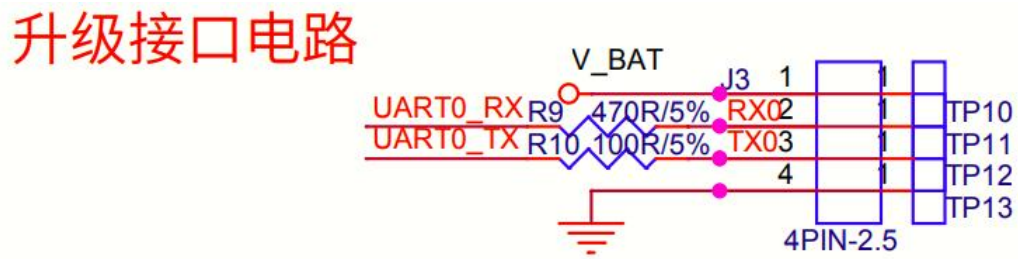


图 10 uart0 预留升级口

4.1.3 AEC 参考线路

本方案需要进行云端内容的播放，并且需要 AEC 打断唤醒，必须要有 AEC 的参考信号，参考信号通过 MICP_R 进入 CI1302；

图 11 AEC 参考线路

4.2 硬件验证

硬件 PCB 做好后，需要进行整机底噪测试和 AEC 效果测试，依此保证产品为最佳效果状态

4.2.1 硬件底噪测试

为了保证识别效果，做好的硬件安装在产品结构中后，需要检查底噪是否正常；底噪正常意味着整个产品的麦克灵敏度正常及有较好的声学结构；

底噪主要采集如下场景：
1、整机供电无麦克，
2、整机供电安静环境（40~50db）
3、整机供电 60db 稳态噪音，
详细的采音方法和底噪判断链接：

4.2.2 AEC 效果测试

将 PCB 安装到结构中，安装固定好麦克和喇叭，组成整个产品后，再来测试 AEC 效果；

定性测试：安静环境 3 米，设备播放云端内容时麦克处的分贝不超过 85db 情况下，正常讲话音量 70db 左右，唤醒率为 90%以上，麦克分贝接近 90db 情况，大声讲话可以正常打断；

声音判断：在正常播放云端内容时，录制参考声音和 AEC 后的声音，分析是否 AEC 后的声音能消除掉产品本身的播放音；

详细测试方法见 AEC 说明；

五、结构注意事项及重要物料选配

5.1 麦克风结构注意

麦克风类似是人的耳朵，本质是将一个微小声压信号进行放大，然后进入语音芯片进行识别，所以麦克风在安装时候，要特别产品开孔和注意远离噪音避免干扰,详情查看连接

针对本方案产品需要在播放时仍可以正常识别唤醒词，所以还应特别注意 AEC 应用注意事项；

麦克规格（推荐）：灵敏度-32db，信噪比 65db 以上；

5.2 喇叭注意事项

喇叭是设备发音的器件，很多时候为了成本选择不带音腔的喇叭，此时设计上需要特别注意密封和开孔；【新增视频：无结构的视频链接】；

喇叭的结构注意事项见文档中心链接

5.3 影响 AEC 效果的结构注意事项

AEC 为在设备播放声音时，用户仍可以通过唤醒词将播放状态打断，在本方

案中是个特别重要的功能，在产品设计时需要特别注意：

1、麦克和喇叭严格按照建议进行相关设计,注意好密封

2、麦克和喇叭尽量远离，背向，使喇叭到达 mic 的声音尽量小，最好不大于 85db

3、麦克和喇叭的相对空间位置不能变化；【AEC 会自适应估计喇叭位置、喇叭发音特性等情况】

4、参考信号按启英给出的分压参考，如果喇叭声音变大，需要适当调整分压电阻，保证最大音量下参考分压电路后的电压幅值范围为 100-150mv；

详细的测试方法参考文档中心（单麦 AEC 串口采音及问题分析）：

六、大模型方案的软件配置组合说明及调试说明

方案中，原始的麦克采音为 16k，16bit，单声道，为了减少带宽，提升效率，一般会进行压缩后再上传声音，目前支持 speex 、opus 和 g722 3 种压缩方式，通过代码宏定义进行配置，并且配置压缩模式后，对应的上传数据的格式中，也有相应的参数标识当前是 speex 压缩、opus 压缩还是 opus 压缩；

在公版 SDK 的例程中，\projects\offline_asr_llm_aiot_uart_sample\project_file\makefile 中可选择算法组合、播放器类型、以及音频压缩方式

```
########单一算法定义(注意:下面算法变量的值用户不可修改，值后更不能有空格)########
#AEC+NN DENOISE
USE_AEC_DENOISE_NN    := 0
#ANY+AEC+DENOISE  需要外部挂7243e codec，-双mic算法，仅双mic可用
USE_ANY_MIC_AEC_DENOISE_NN = 1
#自学习+AEC+DENOISE
USE_CWSL_AEC_DENOISE_NN    := 2
#DOA+AEC+DENOISE 该方案只支持上下行PCM
USE_AI_DOA_AEC_DENOISE_NN  := 3
#########################################################################
CI_ALG_TYPE := $(USE_AEC_DENOISE_NN)
```

```
#---播放器类型：0-精简播放器(相比老播放器节约内存15KB,功能简单) 1-老播放器(占内存较多，功能完善)
USE_AUDIO_PLAYER_TYPE          := 0
```

```
#---AIOT 音频压缩类型  #0-null or g722  1-speex  2-opus
AIOT_AUDIO_COMPRESS_TYPE        := 1
```

注意：精简播放器 MP3 格式音频的 ID3 头最大支持 512 字节，老播放器可支持

\projects\offline_asr_llm_aiot_uart_sample\app\app_main\user_config.h，配置组合（DNN 建议：1458）（默认 sdk 参考，实际可能会有差异）：

语音压缩上传格式	流媒体播放格式	内存剩余资源	预计词条（**根据实际情况而定**）	算法	其他备注： 1、lds 文件中 SYS_HEAP_SIZE 大小

Speex	MP3	asr :11KB system:22KB	50	USE_CWSL_AEC_DENOISE_NN	1、ci130x_alg_pro_cwsl_aec_denoise_nn.lds 的 SYS_HEAP_SIZE=110*1024 2、开启自学习功能 3、本地 160s 的播放音 4、关重采样
Speex	MP3	asr :26KB system:12KB	100+	USE_AEC_DENOISE_NN	1、ci130x_alg_pro_aec_denoise_nn.lds 的 SYS_HEAP_SIZE=1024*110 2、未开启自学习功能 3、本地 160s 的播放音
Speex	PCM	asr :36KB system :26KB		USE_AEC_DENOISE_NN	1、ci130x_alg_pro_aec_denoise_nn.lds 的 SYS_HEAP_SIZE=1024*130 2、无法支持自学习功能 3、此方案中，所有的播放声音都是 wifi 转成 pcm 下发给 1302 进行播放，1302 本地不做任何播放
Speex	PCM+本地 mp3	asr :10KB system:15KB		USE_AEC_DENOISE_NN	1、ci130x_alg_pro_aec_denoise_nn.lds 的 SYS_HEAP_SIZE=1024*120 2、未开启自学习功能 3、此方案中，所有的播放声音都是 wifi 转成 pcm 下发给 1302 进行播放，1302 本地不做任何播放
Speex	PCM+本地 mp3	asr:6KB system:15KB		USE_CWSL_AEC_DENOISE_NN	1、ci130x_alg_pro_cwsl_aec_denoise_nn.lds 的 SYS_HEAP_SIZE=1024*110 2、开启自学习功能 3、此方案中，所有的播放声音都是 wifi 转成 pcm 下发给 1302 进行播放，1302 本地不做任何播放 4、关闭重采样
Speex	G722	asr:39KB system:30KB		USE_AEC_DENOISE_NN	1、ci130x_alg_pro_aec_denoise_nn.lds的SYS_HEAP_SIZE=120 2、无法支持自学习功能 3、此方案中，所有的播放声音都是 wifi 转成 g722 下发给 1302 进行播放，1302 本地不做任何播放

Speex	G722+本地mp3	asr:22KB system:18KB		USE_AEC_DENOISE_NN	1、ci130x_alg_pro_aec_denoise_nn.ld 的 SYS_HEAP_SIZE=1024*120 2、未开启自学习功能 3、此方案中，所有的播放声音都是 wifi 转成 pcm 下发给 1302 进行播放，1302 本地不做任何播放
Speex	G722+本地mp3	asr:6KB system:18KB		USE_CWSL_AEC_DENOISE_NN	1、ci130x_alg_pro_cwsl_aec_denoise_nn.lds 的 SYS_HEAP_SIZE=1024*110 2、开启自学习功能 3、此方案中，所有的播放声音都是 wifi 转成 pcm 下发给 1302 进行播放，1302 本地不做任何播放 4、关闭重采样
G722	MP3	asr:7KB system:27KB		USE_CWSL_AEC_DENOISE_NN	1、ci130x_alg_pro_cwsl_aec_denoise_nn.lds 的 SYS_HEAP_SIZE=1024*130 2、开启自学习功能 3、本地 160s 的播放音
G722	MP3	asr:42KB system:28KB		USE_AEC_DENOISE_NN	1、ci130x_alg_pro_aec_denoise_nn.lds 的 SYS_HEAP_SIZE=1024*130 2、未开启自学习功能 3、本地 160s 的播放音
G722	PCM	asr:72KB system:22KB		USE_AEC_DENOISE_NN	1、ci130x_alg_pro_aec_denoise_nn.lds 的 SYS_HEAP_SIZE=1024*130 2、无法支持自学习功能 3、此方案中，所有的播放声音都是 wifi 转成 pcm 下发给 1302 进行播放，1302 本地不做任何播放
G722	PCM+本地mp3	asr:45KB system:22KB		USE_AEC_DENOISE_NN	1、ci130x_alg_pro_aec_denoise_nn.lds 的 SYS_HEAP_SIZE=1024*130 2、未开启自学习功能 3、此方案中，所有的播放声音都是 wifi 转成 pcm 下发给 1302 进行播放，1302 本地不做任何播放

G722	PCM+本地mp3	asr:12KB system:10KB		USE_CWSL_AEC_DENOISE_NN	1、ci130x_alg_pro_cwsl_aec_denoise_nn.lds 的 SYS_HEAP_SIZE=1024*120 2、开启自学习功能 3、此方案中，所有的播放声音都是 wifi 转成 pcm 下发给 1302 进行播放，1302 本地不做任何播放
G722	G722	asr:75KB system:27KB		USE_AEC_DENOISE_NN	1、ci130x_alg_pro_aec_denoise_nn.lds 的 SYS_HEAP_SIZE=1024*130 2、无法支持自学习功能 3、此方案中，所有的播放声音都是 wifi 转成 g722 下发给 1302 进行播放，1302 本地不做任何播放
G722	G722+本地mp3	asr:48KB system:24KB		USE_AEC_DENOISE_NN	1、ci130x_alg_pro_aec_denoise_nn.ld 的 SYS_HEAP_SIZE=1024*130 2、未开启自学习功能 3、此方案中，所有的播放声音都是 wifi 转成 pcm 下发给 1302 进行播放，1302 本地不做任何播放
G722	G722+本地mp3	asr:15KB system:13KB		USE_CWSL_AEC_DENOISE_NN	1、ci130x_alg_pro_cwsl_aec_denoise_nn.lds 的 SYS_HEAP_SIZE=1024*120 2、开启自学习功能 3、此方案中，所有的播放声音都是 wifi 转成 pcm 下发给 1302 进行播放，1302 本地不做任何播放
PCM	MP3	asr:20KB system:35KB		USE_CWSL_AEC_DENOISE_NN	1、ci130x_alg_pro_cwsl_aec_denoise_nn.lds 的 SYS_HEAP_SIZE=1024*120 2、开启自学习功能 3、本地 160s 的播放音
PCM	MP3	asr:53KB system:27KB		USE_AEC_DENOISE_NN	1、ci130x_alg_pro_aec_denoise_nn.lds 的 SYS_HEAP_SIZE=1024*110 2、未开启自学习功能 3、本地 160s 的播放音

PCM	PCM	asr:83KB system:20KB		USE_AEC_DENOISE_NN	1、SYS_HEAP_SIZE=110*1024 2、无法支持自学习功能 3、此方案中，所有的播放声音都是 wifi 转成 pcm 下发给 1302 进行播放，1302 本地不做任何播放
PCM	PCM+本地 mp3	asr:46KB system:30KB		USE_AEC_DENOISE_NN	1、ci130x_alg_pro_aec_denoise_nn.lds 的 SYS_HEAP_SIZE=120*1024 2、未开启自学习功能 3、此方案中，所有的播放声音都是 wifi 转成 pcm 下发给 1302 进行播放，1302 本地不做任何播放
PCM	PCM+本地 mp3	asr:13KB system:18KB		USE_CWSL_AEC_DENOISE_NN	1、ci130x_alg_pro_cwsl_aec_denoise_nn.lds 的 SYS_HEAP_SIZE=120*1024 2、开启自学习功能 3、此方案中，所有的播放声音都是 wifi 转成 pcm 下发给 1302 进行播放，1302 本地不做任何播放
PCM	G722	asr:78KB system:45KB		USE_AEC_DENOISE_NN	1、ci130x_alg_pro_aec_denoise_nn.lds 的 SYS_HEAP_SIZE=130*1024 2、无法支持自学习功能 3、此方案中，所有的播放声音都是 wifi 转成 g722 下发给 1302 进行播放，1302 本地不做任何播放
PCM	G722+本地 mp3	asr:43KB system:42KB		USE_AEC_DENOISE_NN	1、ci130x_alg_pro_aec_denoise_nn.ld 的 SYS_HEAP_SIZE=130*1024 2、未开启自学习功能 3、此方案中，所有的播放声音都是 wifi 转成 pcm 下发给 1302 进行播放，1302 本地不做任何播放
PCM	G722+本地 mp3	asr:28KB system:21KB		USE_CWSL_AEC_DENOISE_NN	1、ci130x_alg_pro_cwsl_aec_denoise_nn.lds 的 SYS_HEAP_SIZE=110*1024 2、开启自学习功能 3、此方案中，所有的播放声音都是 wifi 转成 pcm 下发给

					1302 进行播放，1302 本地不做任何播放
Opus	MP3	asr :16KB system:17KB	20	USE_AEC_DENOISE_NN	1、ci130x_alg_pro_aec_denoise_nn.ld 的 SYS_HEAP_SIZE=120*1024 2、无法支持自学习功能 3、本地 160s 的播放音 4、关闭重采样
Opus	PCM	asr :9KB System :9KB	10	USE_AEC_DENOISE_NN	1、ci130x_alg_pro_aec_denoise_nn.ld 的 SYS_HEAP_SIZE=130*1024 2、无法支持自学习功能 3、3、此方案中，所有的播放声音都是 wifi 转成 pcm 下发给 1302 进行播放，1302 本地不做任何播放
Opus	G722	asr :14KB system:11KB	10	USE_AEC_DENOISE_NN	1.ci130x_alg_pro_aec_denoise_nn.ld 的 SYS_HEAP_SIZE=128*1024 2.无法支持自学习功能 3.关闭本地播报 4、此方案中，所有的播放声音都是 wifi 转成 g722 下发给 1302 进行播放，1302 本地不做任何播放
Opus	其他组合				***内存不足，无法支持***

备注：

1、上表仅描述了单麦算法内存分配，若需使用**双麦算法**或者 **Opus 压缩/其他组合播放**内存不足时，请先尝试在 sdk_default_config.h 中关掉重采样 INNER_CODEC_AUDIO_IN_USE_RESAMPLE 宏，降低 4 个点识别率来节省 24KB 内存后，还是不能使用请联系本司技术支持人员确认

2、资源原因，仅适合做大模型对话类产品（16k 采样率，16bit，单声道），不适合做对音质要求高的音箱类产品；

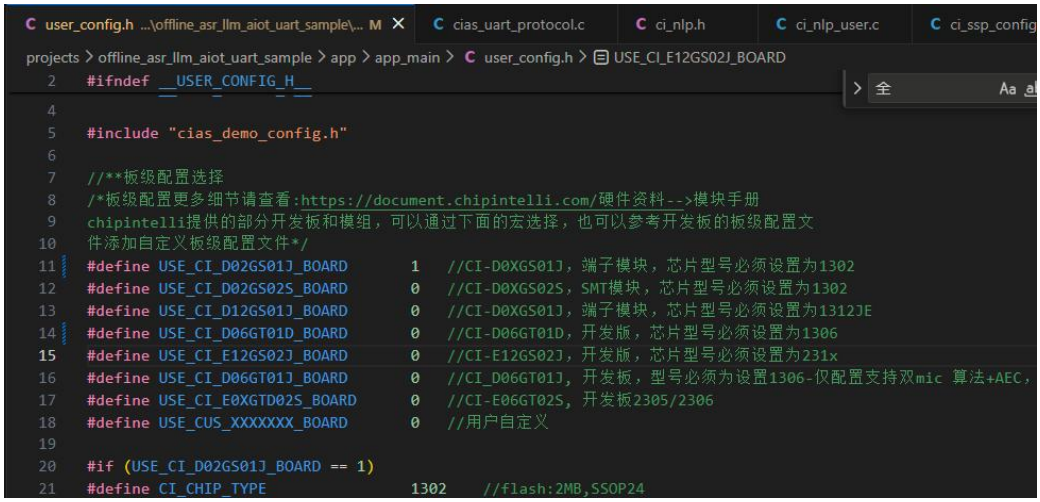
3、本章节后续以 SPEEX+MP3 播放 和 OPUS+PCM 播放为例进行介绍方案的宏定义

6.1 不同软件配置中的公用部分

不同的软件配置中，板级配置、固件编译打包、烧录；流程是一样的，如下：

1、板级配置：

本方案采用D02GS01J模块，需要在projects\projectsoffline_asr_llm_aiot_uart_sample\app\app_main\user_config.h中修改板级配置如下：



2、修改配置完成后，在VScode中按如下操作进行代码编译和和合成分区文件：

3、通过上图step5的打包固件工具按如下步骤打包固件；

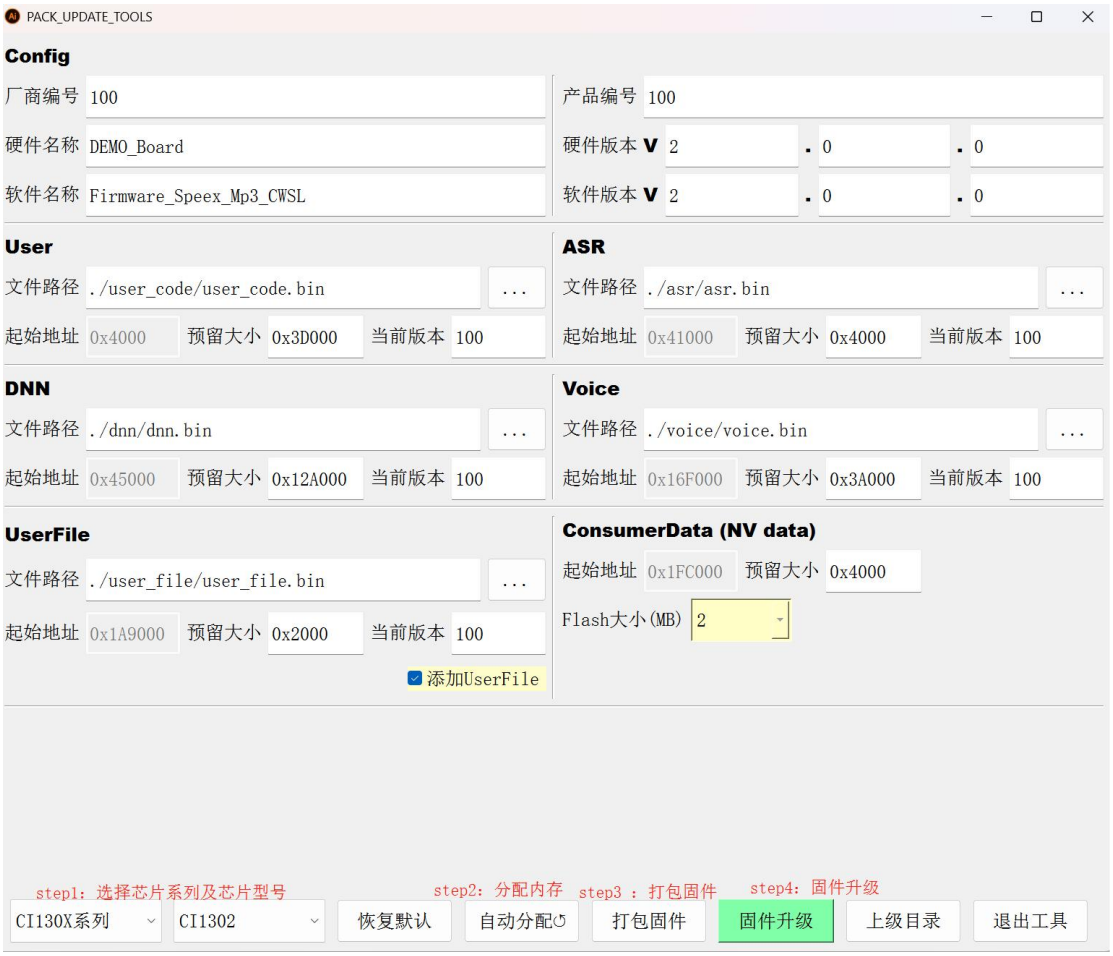


图 6.1 固件打包界面

4、打包固件成功后，将硬件串口接到 PC 后，再给 Ci1302 进行供电；下图为固件升级成功的界面；更多细节[请查看文档中心链接]

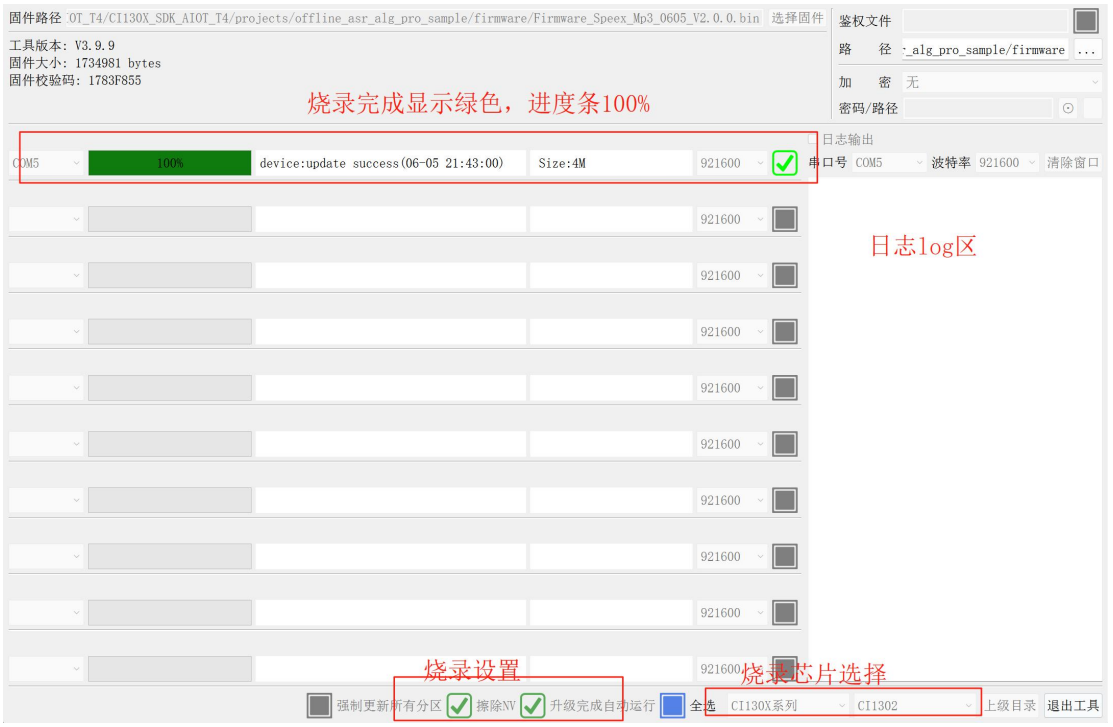


图 6.2 固件打包升级界面

6.2 SPEEX 压缩上传语音+MP3 播放

本方案内存资源适中，为在线提供良好语料基础上，也可以做自学习和较多的本地磁条，为优先建议方案：

6.2.1 方案代码配置

1、在文件（projects\projectsoffline_asr_llm_aiot_uart_sample\app\app_main\user_config.h）中，按下图配置 SPEEX 压缩数据上传使能 和 MP3 播放使能：

```
// 若使用的组合内存不够，可关闭重采样，节省24KB内存，但是识别效果会下降5个点左右
//压缩算法选择-pcm和speex和opus只能三选一,不能同时支持; opus压缩+mp3播放不支持自学习，推荐推荐设置SYS_HEAP_SIZE = (1024
#define AUDIO_COMPRESS_RECORD_DISABLE                   0           //(务必在makefile中配置宏AIOT_AUDIO_COMPRESS
#define AUDIO_COMPRESS_SPEEX_ENABLE                     1           //(务必在makefile中配置宏AIOT_AUDIO_COMPRESS
#define AUDIO_COMPRESS_OPUS_ENABLE                      0           //(务必在makefile中配置宏AIOT_AUDIO_COMPRESS
#define AUDIO_COMPRESS_G722_ENABLE                      0           //(务必在makefile中配置宏AIOT_AUDIO_COMPRESS
//网络数据播放类型选择-mp3和pcm和opus只能三选一，不能同时支持;
#define NET_AUDIO_PLAY_BY_MP3                           1           //云端播放通过mp3格式(注意:推荐设置SYS_HEAP_S
#define NET_AUDIO_PLAY_BY_PCM                           0           //云端播放通过pcm格式(注意:推荐设置SYS_HEAP_S
#define NET_AUDIO_PLAY_BY_G722                          0           //云端播放通过G722格式播放,必须使用精简播放器
```

1、在文件（projects\projectsoffline_asr_llm_aiot_uart_sample\project_file\makefile）中，改为 speex 格式压缩：

```
26  #---AIOT 音频压缩类型  #0-null or g722  1-speex  2-opus
27  AIOT_AUDIO_COMPRESS_TYPE        := 1
```

3、在文件（projects\projectsoffline_asr_llm_aiot_uart_sample\project_file\makefile）中，如果需要自学习功能，需要设置为：USE_CWSL_AEC_DENOISE_NN

```
4   include $(SDK_PATH)/utils/common_head.mk
5
6   ########单一算法定义(注意:下面算法变量的值用户不可修改，值后更不能有空格)########
7   #AEC+NN DENOISE
8   USE_AEC_DENOISE_NN    := 0
9   #ANY+AEC+DENOISE  需要外部挂7243e codec，-双mic算法，仅双mic可用
10  USE_ANY_MIC_AEC_DENOISE_NN = 1
11  #自学习+AEC+DENOISE
12  USE_CWSL_AEC_DENOISE_NN    := 2
13  #DOA+AEC+DENOISE 该方案只支持上下行PCM
14  USE_AI_DOA_AEC_DENOISE_NN  := 3
15  ##########################################################################
16  CI_ALG_TYPE := $(USE_AEC_DENOISE_NN)
17
```

4、在文件（projects\projectsoffline_asr_llm_aiot_uart_sample\firmware\合成分区 bin 文件.bat）中合成分区时，选择 MP3 格式（下图输入 4）

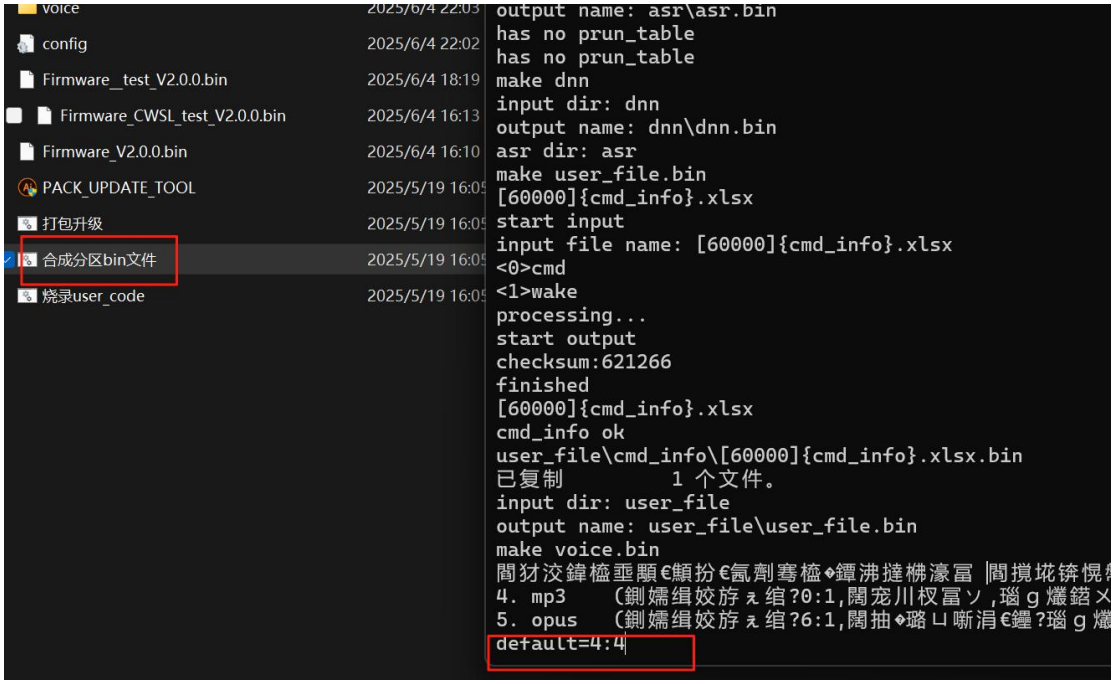


图 6.3 合成分区根据播放不同，选择不同格式

5、配置完成后，即可进行代码编译、代码打包和固件下载测试

6.2.2 本类型方案中的数据格式描述

1、SPEEX 上传格式：

speex 参数配置可参考离在线 SDK 中 audio_speex_task_init（）函数

```
/* Split-band wideband CELP mode*/
const SpeexSBMode sb_wb_mode = {
   &speex_nb_mode,
   NB_FRAME_SIZE,     /*frameSize*/
   NB_SUBFRAME_SIZE,     /*subframeSize*/
   8,     /*lpcSize*/
#ifdef FIXED_POINT
   29491, 19661, /* gamma1, gamma2 */
#else
   0.9, 0.6, /* gamma1, gamma2 */
#endif
   QCONST16(.0002,15), /*lpc_floor*/
   QCONST16(0.9f,15),
   {NULL, &wb_submode1, NULL, NULL, NULL, NULL, NULL, NULL},
   3,
   {1, 8, 2, 3, 4, 5, 5, 6, 6, 7, 7},
   {1, 1, 1, 1, 1, 1, 2, 2, 3, 3, 4},
#ifndef DISABLE_VBR
   vbr_hb_thresh,
#endif
   5
};
```

图 6.4 Speex 格式代码参考

音频数据为 20m 数据（640 字节）压缩为 43 字节，约 14 倍压缩率；43 字节数据中，第一个字节为 2A（表示数据长度），串口通信协议头 16 个字节上传(共 59 字节);语音发给 wifi 的完整一帧数据如下：

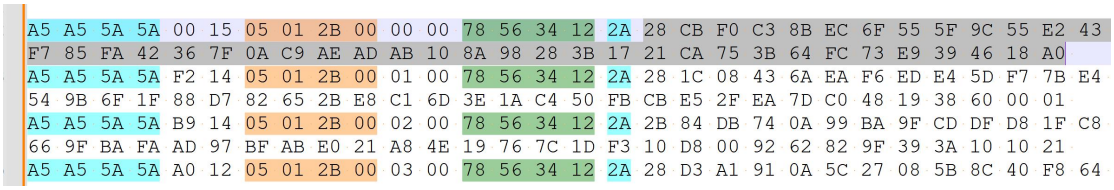


图 6.5 speex 串口上传数据截图

上图中 2A 前的部分为 16 字节帧头，05 01 为上传音频的命令号码。002B 为数据长度，2A 及后面的数据未压缩后的 speex 数据；

2、MP3 流媒体播放格式：

资源原因，目前只支持：16k 采样率，16bit，单声道（mono）的声音，服务器可以使用：mp3PRO?(FhG) MPEG Layer-3，码率在 32 Kbps 以下 (Mono)的配置；

6.3 OPUS 音频压缩上传+PCM 播放

本方案内存资源紧张，可以支持 20 词条左右，建议使用 1M 的通用模型，本方案应用时，本地不支持播放音，所有声音有 wifi 下发播放，不支持唤醒词自学习；

6.3.1 方案代码配置

1、在文件（projects\projectsoffline_asr_llm_aiot_uart_sample\app\app_main\user_config.h）中，按下图配置 SPEEX 压缩使能 和 MP3 播放使能：

```
// 若使用的组合内存不够，可关闭重采样，节省24KB内存，但是识别效果会下降5个点左右
//压缩算法选择-pcm和speex和opus只能三选一,不能同时支持; opus压缩+mp3播放不支持自学习，推荐推荐设置SYS_HEAP_SIZE = (1024*110))
#define AUDIO_COMPRESS_RECORD_DISABLE                   0           //(务必在makefile中配置宏AIOT_AUDIO_COMPRESS_TYPE=0
#define AUDIO_COMPRESS_SPEEX_ENABLE                     0           //(务必在makefile中配置宏AIOT_AUDIO_COMPRESS_TYPE=1
#define AUDIO_COMPRESS_OPUS_ENABLE                      1           //(务必在makefile中配置宏AIOT_AUDIO_COMPRESS_TYPE=2
#define AUDIO_COMPRESS_G722_ENABLE                      0           //(务必在makefile中配置宏AIOT_AUDIO_COMPRESS_TYPE=0
//网络数据播放类型选择-mp3和pcm和opus只能三选一，不能同时支持;
#define NET_AUDIO_PLAY_BY_MP3                           0           //云端播放通过mp3格式(注意:推荐设置SYS_HEAP_SIZE = (
#define NET_AUDIO_PLAY_BY_PCM                           1           //云端播放通过pcm格式(注意:推荐设置SYS_HEAP_SIZE = (
#define NET_AUDIO_PLAY_BY_G722                          0           //云端播放通过G722格式播放,必须使用精简播放器 推荐设置
```

2、在文件（projects\projectsoffline_asr_llm_aiot_uart_sample\project_file\makefile）中，改为 speex 格式压缩：

```
#---AIOT 音频压缩类型  #0-null or g722  1-speex  2-opus
AIOT_AUDIO_COMPRESS_TYPE        := 2
```

3、在文件（projects\projectsoffline_asr_llm_aiot_uart_sample\project_file\makefile）中，如果需要自学习功能，需要设置为：USE_CWSL_AEC_DENOISE_NN

```
include $(SDK_PATH)/utils/common_head.mk

########单一算法定义(注意:下面算法变量的值用户不可修改，值后更不能有空格)########
#AEC+NN DENOISE
USE_AEC_DENOISE_NN    := 0
#ANY+AEC+DENOISE  需要外部挂7243e codec，-双mic算法，仅双mic可用
USE_ANY_MIC_AEC_DENOISE_NN = 1
#自学习+AEC+DENOISE
USE_CWSL_AEC_DENOISE_NN    := 2
#DOA+AEC+DENOISE 该方案只支持上下行PCM
USE_AI_DOA_AEC_DENOISE_NN  := 3
#########################################################################
CI_ALG_TYPE := $(USE_AEC_DENOISE_NN)

```

4、在文件（projects\projectsoffline_asr_llm_aiot_uart_sample\firmware\合成分区bin 文件.bat）中合成分区时，选择 MP3 格式（下图输入 4）

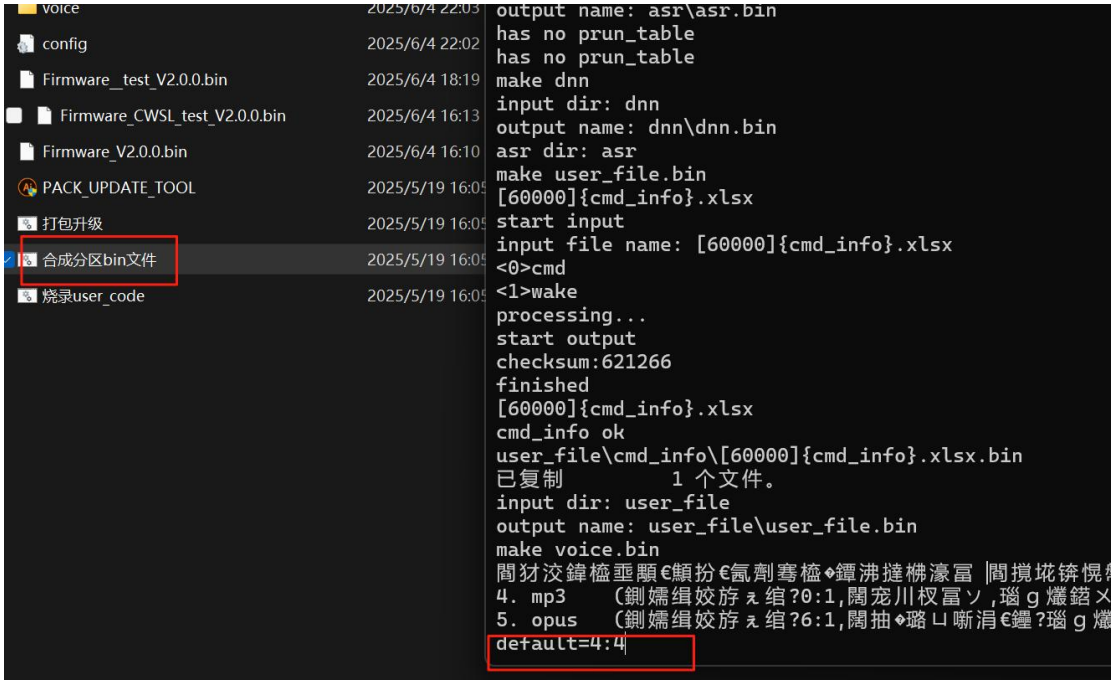


图 6.7 合成分区根据播放不同，选择不同格式

5、配置完成后，即可进行代码编译、代码打包和固件下载测试

6.3.2 本类型方案中的数据格式描述

1、Opus 压缩上传，

OPUS 的压缩格式，参数配置可参考离在线 SDK 中 opus_encode_task（）函数；如下：

```
projects > offline_asr_alg_pro_sample > app > app_audio_handle > C cias_voice_upload.c > opus_encode_task(void)
526  #if AUDIO_COMPRESS_OPUS_ENABLE
528  void opus_encode_task(void)
534      int8_t opus_rx_temp[COMPRESS_NEED_PCM_LEN * 2] = {0};
535      OpusEncoder *opusEncoder = NULL;
536      int size = opus_encoder_get_size(1);
537      mprintf("==opus_encoder malloc size = %d\r\n", size);
538      opusEncoder = pvPortMalloc(size);
539      opus_encoder_init(opusEncoder, 16000, 1, OPUS_APPLICATION_RESTRICTED_LOWDELAY); // 最低延
540      opus_encoder_ctl(opusEncoder, OPUS_SET_BITRATE(16000));
541      opus_encoder_ctl(opusEncoder, OPUS_SET_EXPERT_FRAME_DURATION(OPUS_FRAMESIZE_40_MS));
542      opus_encoder_ctl(opusEncoder, OPUS_SET_VBR(0));           // 强制CBR
543      opus_encoder_ctl(opusEncoder, OPUS_SET_VBR_CONSTRAINT(1)); // 约束VBR
544      opus_encoder_ctl(opusEncoder, OPUS_SET_BANDWIDTH(OPUS_BANDWIDTH_WIDEBAND));
545      opus_encoder_ctl(opusEncoder, OPUS_SET_COMPLEXITY(10));
546      opus_encoder_ctl(opusEncoder, OPUS_SET_SIGNAL(OPUS_SIGNAL_VOICE));
547      // 16bit
548      opus_encoder_ctl(opusEncoder, OPUS_SET_LSB_DEPTH(16));
549      opus_encoder_ctl(opusEncoder, OPUS_SET_PACKET_LOSS_PERC(0));
550      opus_encoder_ctl(opusEncoder, OPUS_SET_DTX(0));
551      int skip = 0;
552      opus_encoder_ctl(opusEncoder, OPUS_GET_LOOKAHEAD(&skip));
553      opus_encoder_ctl(opusEncoder, OPUS_SET_FORCE_CHANNELS(OPUS_AUTO));
554      mprintf("opus_encoder_init_ok\r\n");
```

音频数据为 40ms（1280 字节）压缩一帧，压缩为 80 字节，16 倍的压缩率，CBR 格式；质量为 Audio 格式，带宽是 8，深度 16，通道为 1：格式 CELT；语音发给 wifi 的数据如下：

```
A5 A5 5A 5A B9 28 05 01 50 00 00 00 78 56 34 12 BB 42 01 B5 10 9D E7 C5 21
DD CD F8 3A 30 84 D1 BE 43 F4 09 E2 95 33 46 E3 D9 B7 E8 DD 14 93 5E 43 F7
DA 5B 6C 06 E3 21 21 B2 82 A8 72 B3 52 7F 5B E6 3F 37 24 43 B0 A3 BF A3 5D
83 9C 81 BB 58 A4 D3 50 09 67 FF 5C A3 A8 66 A2 DA 20 45 21 00
A5 A5 5A 5A 48 25 05 01 50 00 01 00 78 56 34 12 BB 42 01 8A 06 3F BD E0 E5
67 39 E1 DE 96 E9 44 DB F9 C0 8E 90 97 70 81 47 2B 2B FB 2B 3A 5C CC EB 2A
6B 39 BB D5 CB 7B 43 DF AC 8D A8 53 1D 67 3F 91 4B 38 19 55 9E 3B 8A 25 13
28 53 1A D1 39 4C 0E F6 BB AB D3 37 E6 A8 1B 94 14 8D 0D 69 00
A5 A5 5A 5A A5 27 05 01 50 00 02 00 78 56 34 12 BB 42 01 D7 EB D3 6D EF E1
DE 9A 87 A7 7B 45 2A A8 A4 61 49 CE 5D 8E 46 49 86 AF A4 CC C8 98 78 AE 5F
E3 A1 6E 63 DB 36 75 D6 37 68 41 40 72 7E B7 B2 B0 7F 80 8D 1F 1A 0A 09 0A
01 A5 F9 61 93 86 AC 92 BA E5 C0 71 85 13 C5 1B 25 7A 0E 75 00
```

图 6.8 Opus 串口上传数据截图

图中 BB 前的部分为 16 字节帧头，BB 及后面的数据为压缩后的 Opus 数据；一帧数据 80 字节

2、MP3+PCM 流媒体播放格式：

资源原因，MP3 目前只支持：16k 采样率，16bit，单声道（mono）的声音，服务器可以使用：mp3PRO?(FhG) MPEG Layer-3, 码率在 32Kbps 以下 (Mono)的配置；数据格式参考前面 6.2.2 的数据格式；

七、数据流描述

本章节描述语音芯片和 WIFI 进行数据流交互的细节

7.1 CI130X 数据发给 wifi

CI130x 将处于唤醒状态时，将用户的声音降噪压缩后，通过高速串口（921600bps）发给 wifi：

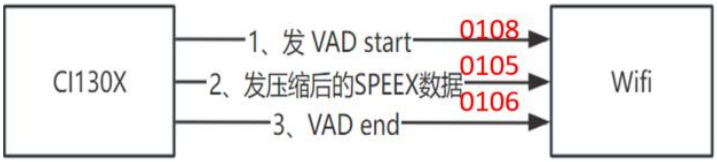


图 7.1 录音压缩数据发给 wifi 的流程框图

流程如下：

步骤 1、CI130X 唤醒后，进行深度降噪和深度学习的 VAD 判断，然后进行音频数据压缩为 speex/opus；

步骤 2、数据上传流程如右图，1，2.....2， 3 的方式进行数据上传 wifi,每帧数据长度：speex 为 XXbyte，opus 为 xbyte

步骤 3、上传完毕后；

1、在一段时间内等待云端下发播放音（5s），该等待时间内，如果有新的 VAD 数据不进行上传【如有 vad 打印：xxxx】，同时播放“云端处理中”（也可以设置为不播放）

2、超过 5s 后播放“未听清请再说一遍”（安抚作用），并重新按二的步骤进行数据上传；

3、如果上传一段声音，云端正常播放完后，则按步骤二检测 vad 及进行数据上传

4、如 wifi 上传到云端，并且开始有一些云端数据进行播放，此时产生新的 VAD 数据时，

a、全双工开启后：会暂停当前播放，并上传新的 VAD 数据；【此处最好改为 130X 进行数据上传，不停止播放内容，由云端根据本次识别的结果进行是否播放新内容的判定，修改方法：#define VAD_START_STOP_PLAY_ENABLE 0(user_config 文件)】

b、全双工关闭后：不进行数据上传

图 7.2 录音压缩数据发给 wifi 的实际数据图及 1302 本阶段的运行 log

注意事项：

1、播放过程中可以唤醒词打断当前播放内容，，并返回到上面流程的步骤 1 中；

2、声音数据使用 SPEEX 数据压缩，云端也注意需解析这个格式的数据；

3、唤醒时 CI130X 会同步发 0x0102，退出唤醒会发 0x0109，识别到其他命令词会发 0x0101，wifi 可以根据识别情况做相关逻辑。

4、全双工弊端：播放云端内容时候，任意讲话声音（办公室，会场、卖场等多人交流环境）可以产生 vad（可能并不想打断当前播放），导致停止当前的播放内容，体验不佳

备注：
1、为了产品控制人性化感觉更好，可以在一段 VAD 数据上传到云端时，播放“叮”，降低用户的等待焦虑，如果有 LED，也可以给个闪烁提示替代叮的声音；【降低延迟是关键，这个只能起部分作用】
2、全双工模式时建议通过 APP 进行开关，此功能仅限于个人安静环境使用；默认使用半双工多轮对话；

7.2 WIFI 接收 CI 130X 数据的接口参考代码

7.3 wifi 下发数据到 CI130X 进行流媒体播放流程

本章节重点介绍流媒体播放流程及遇到的常见问题

7.3.1 播放流程详细：

当 wifi 拿到云端播放内容后，告知 CI130X 开始播放，之后根据 CI130X 获取播放数据的请求，不断的将数据发给 CI130x；

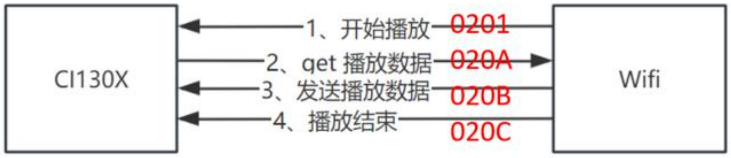


图 7.3 WIfi 下发播放数据流程框图（箭头表示数据方向）

1、wifi 拿到云端要播放的内容，进行数据缓存，并告知 CI130x 开始播放;
2、上图步骤 2,3，循环进行，1302 会向 wifi 发获取 1kbyte 数据，wifi 将数据（可以不是 1k，但是协议中必须是实际发送的数据长度）发给 130x 进行播放；
3、wifi 将播放的最后数据发完后，告知 CI130X 播放数据播放完；
4、CI130X 播放将本身缓存数据播放完。

```
[wifi msg:]recv type: 0201
NET_PLAY_START.......
requst one frame2, play buffer data size =0
[wifi msg:]recv type: 020b
pheader->len = 1024
play start
audio_play_state = 1
requst one frame2, play buffer data size =0
[wifi msg:]recv type: 020b
pheader->len = 1024
requst one frame2, play buffer data size =320
[wifi msg:]recv type: 020b
pheader->len = 1024
requst one frame2, play buffer data size =1344
[....中间数据手工删除.....]
[wifi msg:]recv type: 020b
pheader->len = 1024
requst one frame2, play buffer data size =3968
[wifi msg:]recv type: 020b
pheader->len = 1024
requst one frame2, play buffer data size =3840
[wifi msg:]recv type: 020b
pheader->len = 78
[wifi msg:]recv type: 020c
auido paly finish...
pheader->len = 0
play stop sync state to wifi ....
```

图 7.4 WIfi 下发播放数据时 1302 的打印 log

解析如下：

1、[wifi msg:]recv type 0201 表示收到 wifi 指令，0201 为开始播放

2、NET_PLAY_START..... 和 auido paly finish... 为播放的起始和结束位置

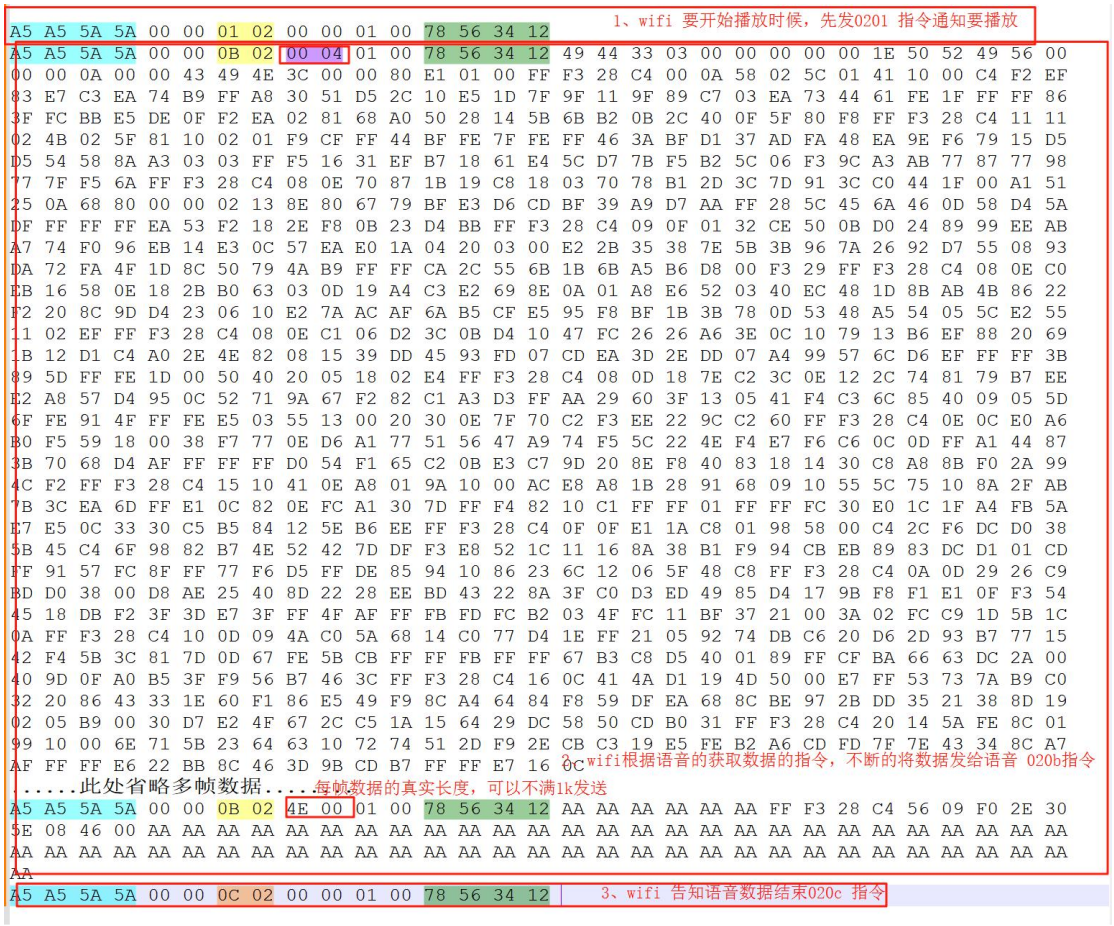
3、requst one frame2, play buffer data size =0：表示当前 1302 播放 buffer 中数据长度是 0，并向 wifi 获取数据，其他处的 play buffer data size =320/1344/...都是当前播放 buffer 的已有数据；当前播放默认总大小：AIOT_AUDIO_PLAY_BUF_SIZE 为 6k，通过此数值可以知道播放 buffer 还有多少数据可以播放

4、 [wifi msg:]recv type: 020c 表示收到 wifi 指令，020c 为 wifi 发送数据完成，此时播放器会继续播放未播完的 play buffer data size 的音频

5、play stop sync state to wifi .... 为播放完播放 buffer 的数据，此时播放器同步本地和 wifi 的播放状态

7.3.2 语音芯片收到 wifi 播放真实数据分析：

下图为 WIfi 发给语音播放的真实数据格式：



7.3.1 播放中的注意事项：

1、当前版本 V1.1.10，CI130X 在播放中，如开启全双工工功能，新的 VAD Start 信号到来时，VAD 数据会上传；

2、云端下发给 wifi 的播放格式确定 CI130x 是支持的

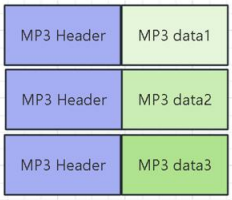
3、在未唤醒状态，130X 仍然可以响应流媒体播放的指令

4、对接到云端后，如果 wifi 播放还未播放完中，出现 play buffer data size =0 或者很小的情况，意味着 wifi 未及时拿到云端数据，播放器已经无数据播放，此时会表现为卡顿情况

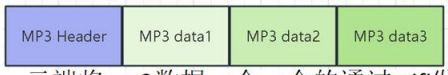
5、wifi 可以先缓存一点数据（例如 1k~2K），再给语音芯片发播放数据，避免刚开始播放的时候网络抖动导致播放 buffer 很快无数据，引起播放卡顿的情况

6、云端下发的数据，如果是多段合成的 MP3，操作如下图：

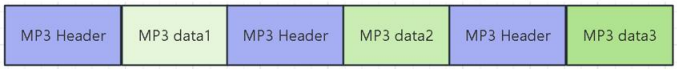
一次对话内容上传后，云端将数据给大模型，产生的MP3播放内容，
可能是多段MP3文件(一次交互回复)如下：



以下两种wifi发给语音的方式都支持【均只需要发一次0201 开始播放指令，根据语音get数据不断给数据给语音芯片，直到播放完，再发020c给语音】
1、云端将数据只保留一个数据头，有效内容拼接在一起通过wifi发给语音



2、云端将mp3数据一个一个的通过wifi发给语音进行播放



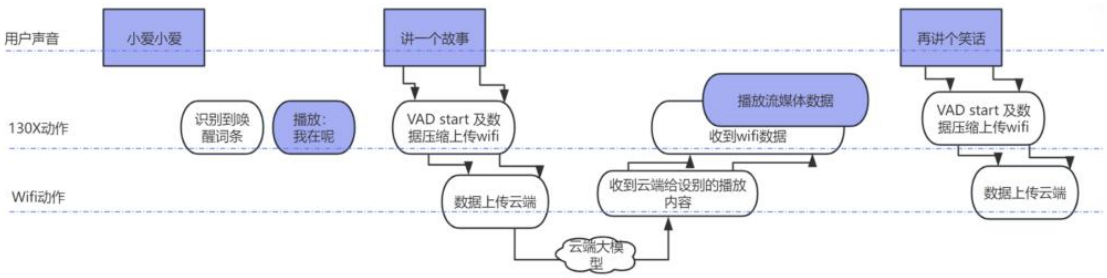
7、在线内容播放完后，本地播放完后，继续播放云端内容的方法

1、语音发给 wifi 本地播放完成
2、Wifi 继续发播放数据下来

7.4 WIFI 端下行流媒体的接口代码参考如下

待补充

7.5 数据流-多轮对话时序流程

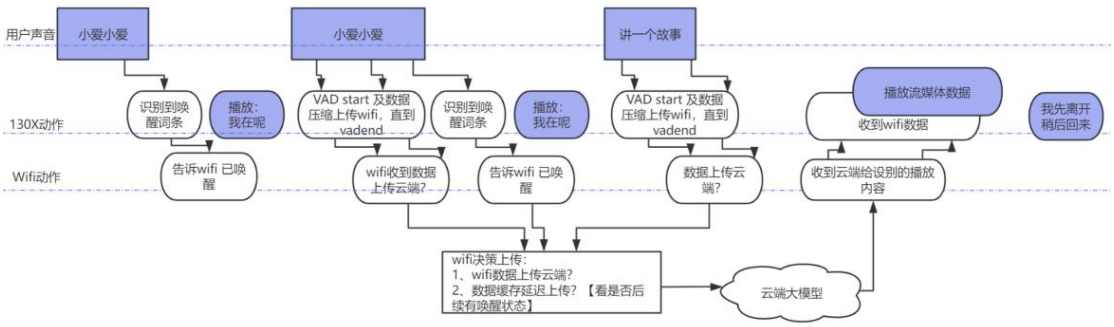


正常多轮对话的时序配合：
1、唤醒后启动 vad 检测，语音播放一次对话的云端数据后，再进行下一次的对话；
2、每次的播放完成后，将唤醒时间清零，并重新唤醒时长的计时，从而实现多轮对话。

注意：
1、可以用唤醒词打断正在进行的云端在线内容播放

2、AEC 需要特别注意硬件设计的参考信号和结构，详情查看；
3、唤醒反馈“我在呢”播放的时候，会停止打断掉当前的在线流媒体播放内容；
4、VAD 的超时时间目前设置为 5s；【130X 代码中宏定义 VAD_FORCE_OVER_NUM_TIME】。

7.6 数据流-多轮的话中离线词条的处理机制



多轮对话中连续说唤醒/命令词的处理：
1、唤醒之后再说唤醒词的语音数据，CI1302 会先传给 WIFI，随后将唤醒词的消息发给 WIFI，由 WIFI 和云端决定如何处理这笔数据；
2、离线识别词条时，如果说唤醒词，CI130X 会发 0x0102 给 wifi，wifi 收到数据&&等待一段时间无识别结果再进行数据上传；
3、唤醒后，如果说命令词，仍然采用 2 的逻辑，CI130X 会给 wifi 发 **LOCAL_ASR_RESULT_NOTIFY（0101）**指令给 wifi，wifi 可以识别离线词条同时忽略云端的处理结果。

Wifi 及云端的决策方式：
wifi 收到本地命令词 NOTIFY 后，告诉云端本次的数据无效，云端清除本的交互数据，并且不会再从服务器拿本次的播放数据；

云端处理：
正常云端有播放数据的时候，云端告诉 wifi，wifi 申请云端数据，再拿到云端数据播放，当 wifi 收到 NOTIFY 指令后，不从 wifi 拿数据

八、功能模块描述及调优

本章节描述本方案设计的各种功能：

8.1 VAD（Voice Activity Detection）

VAD 是检测人声的起始和结束，起具备以下特性：

1、VAD start 是有滞后性的
2、VAD end 是连续一段时间低能量或者非人声产生 VAD end
3、本身算法复杂，启英采用 DNN+传统算法结合的方式进行 VAD 准确性的判断

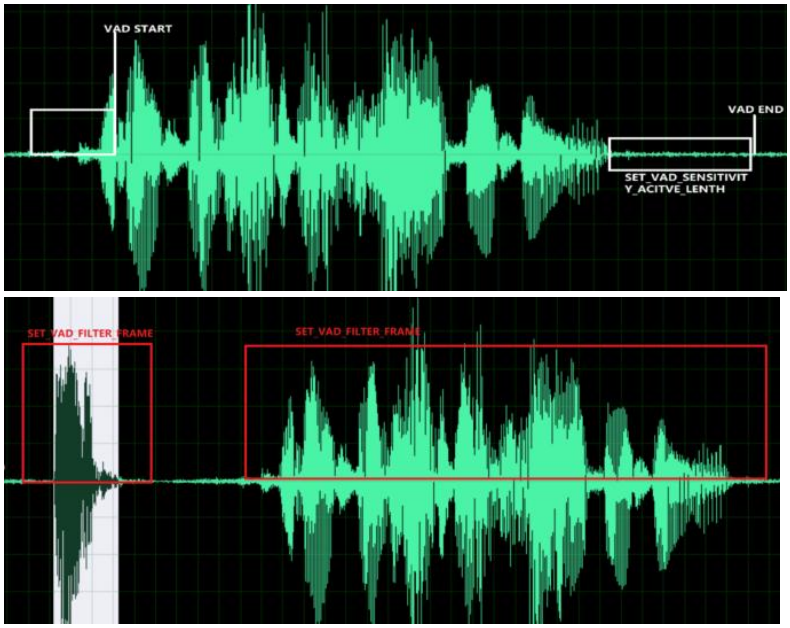


图 8.1 vad 波形说明

Wifi 可以通过串口协议控制 VAD 的一些参数：

1、SET_VAD_FILTER_FRAME：如果 VAD 的持续时间小于该宏定义，则不进行声音上传；【无特殊情况默认就好】

2、SET_VAD_SENSITIVITY_ACITVE_LENTH：VAD 结束灵敏度(停顿间隔)，数值非线性(等效时常为估计长度不精准)：参数 6=300ms，15=400ms，20=500ms，50=1S，80= 1.5S，120=2S

3、SET_VAD_START_MAX_TIMEOUT：VAD 最大超时时间，例如一直说话不停，到达该时间后强制结束 VAD 进行上传数据【默认为 5s】

4、SET_VAD_SENSITIVITY：VAD 灵敏度设置，灵敏度高的时候，表现为更小的声音也可以产生 vad，能即使上传声音，识别距离更远，灵敏度低时，意味着能过滤更多的噪声，识别距离更近，需要根据产品灵活选择

8.2 AEC

AEC 是声学回声消除(Acoustic Echo Cancellation)的简称，开启该功能后，当存在设备在播放内容时，也可以进行语音识别打断。

AEC 效果依赖硬件参考声音的回采、喇叭的频响以及产品的结构，详情查看文档中心

8.3 DNN 深度降噪

通过深度学习的方式过滤掉一些噪音，例如风吹，油烟机，敲桌子，抚摸毛绒玩具的表面，部分电机类声音，平缓音乐及其他稳态噪声；过滤后再进行 vad 判断，

避免所有声音都传到云端导致云端负担过重；
注意：实际产品中，受限于磨具结构及电机的噪声，可能对产品本身的噪声过滤不够好，此种情况需要特别就产品的噪音进行录音训练降噪模型；来减缓这种问题；

8.4 语音 UI 设计

语音 UI 是指设备对话中使用的唤醒词，命令词及播放音：好的命令词和唤醒词，更容易将产品的效果做好，详情链接

本方案中有以下建议：
1、唤醒后，当本地声音上传完成后，播放“叮”的声音，以便用户知道声音传完，或者指示灯闪烁指示
2、当唤醒后或当一轮用户交互结束后，提示“叮咚”的声音，以便用户知道可以跟大模型对话了，或者指示灯常亮指示
3、当出现网络异常或者服务器异常时，进行播放“网络不稳定”之类的提示语，避免网络不好时，用户无法判断能否对话
4、其他状态提示，例如充电，网络连接，断开等状态，建议有个提示音便于用户知道产品的状态

8.5 唤醒词及命令词制作

通启英 AI 平台可以更换唤醒词和命令词，以及产品的播放音，详细操作流程参考链接
备注：
1、本方案资源紧张，建议使用 1M 的 DNN 模型（DNN 建议选择：1458）
2、如需支持小语种，需要按中文格式进行制作 ASR，CMD_info，Voice，替换 DNN，重新打包即可支持对应语种的唤醒识别和播放；当选择较大模型时，对应的词条数量应少一些

8.5 唤醒效果优化

唤醒词需要兼顾唤醒率和误唤醒，可以通过以下方式调优：
8.5.1 唤醒词调优
调优有以下方法：
1、阈值调节：cmd_info 表格中阈值越大，要求用户这个词条说的越准确，同时也会更少的误唤醒；
2、阈值确定方法：
A、基于启英平台产生的默认分数，用户可以在调低一些（例如 30），然后在产品的实际使用距离和一些人声干扰噪声下（如电脑播放新闻，麦克处分贝测

量 55db 左右），多人多次喊唤醒词，看下识别的分数，确定合适的阈值

B、误唤醒挂测：电脑播放新闻，麦克处分贝测量 65db 左右环境，将语音模块晚上挂测保存 log，第二天查看是否有误识和及误识的阈值情况，

C、经历后续的正性词、负性词、误唤醒防误测试等流程后，最终确定合适阈值

3、正性词：为了提升词条的适应性和兼容更多口音，有时需要增加正性词，例如：小爱小爱，我们可以增加“小盖小盖”作为正性词说法，当用户说道“小盖小盖”时，也认为是唤醒了；【该词条可以使用分词表上传的方式做，或者使用命令词表格上传的方式做（找启英获取分词优化方法）】

4、负性词：为了降低唤醒，我们可以增加一些有些相似有不是非常像的声音，作为唤醒词防误使用，例如唤醒词是小爱小爱，我们可以增加“小美小美”作为防误，当用户说类似小 X 小 X 的词条时候，就不会容易误识小爱小爱了；

5、垃圾词：不做任何反馈或操作的词条，通常与命令词的编辑距离远或毫无相关的词，这种词能降低一些词条的误识（AI 平台做的模型默认会添加一部分）；

备注：

1、每个词条的阈值跟训练时词条的词频是相关的，所以阈值会有所差异

2、词条长度和发音习惯，也会影响词条的阈值，一般情况，当词条较长时，分数可以设置的相对小一些

3、负性词可以使用启英泰伦的误唤醒测试工具进行测试，然后通过听什么声音备误唤醒的方法，产生负性词加入命令词表格

特别注意：

1、当有些口音特别重的时候，不建议针对性作为正性词，否则可能加打误唤醒

2、如有些词条经过上述优化仍不好识别，可以联系启英泰伦 FAE 获取更多帮助；

8.5.2 唤醒词防误优化

用尽量多的一些新闻综艺节目，来对被测模块进行误唤醒的激发测试，

1、电脑本省具备麦克输入功能，可以打开系统的录音机进行录制，如果录制后能播放，则电脑可以正常录音（笔记本电脑，或者台式电脑接麦克风）

2、将模块通过串口工具接到电脑，电脑记录器运行 log，加上时间戳

3、开启工具 GetCmdCoice，选择模块对接的电脑串口号

4、电脑播放新闻或者综艺节目，麦克处分贝测量 65db 左右，长时间挂测

5、操作正确会产生如下文件：

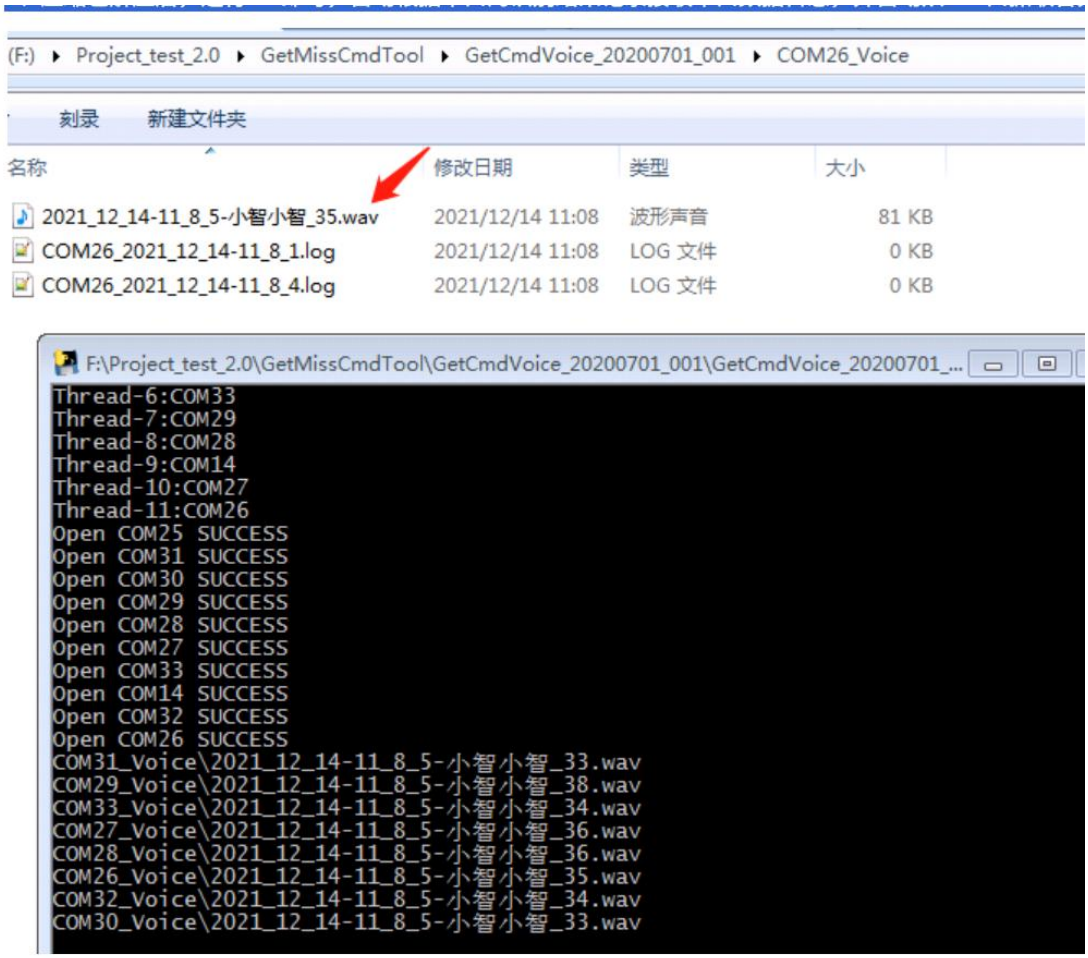


图 8.2 误唤醒测试记录

6、听产生误识的内容，判断其与唤醒词“小爱小爱”的相似度，将不应该备误识的声音作为负性词进行重新做模型和固件测试

注意：

1、可以多个模块同时测试，如上图 8.2 同时挂测了 8 个模块

2、环境准备好后，可以先喊下唤醒词测试是否正常工作

3、不要使用相同的测试集重复播放，尽量每次换不同测试集进行测试

更多细节参考链接：语音识别使用说明

8.8 声学模型选择

如本方案用在不同的家电，则需选择对应家电的模型，更适配产品噪声，声学模型的选取方法

8.10 OTA

详细见《启英泰伦离在线大模型方案 OTA V4 使用说明 V1.1》

8.11 产测模式

产测方式分为 2 种类：
方式 1：使用启英泰伦的标准生产治具，可以完整的测试电流，电压，串口通讯，asr 识别，声音回路等功能，详情链接
方式 2：简化测试，只测试声音通路及串口通讯，如下图

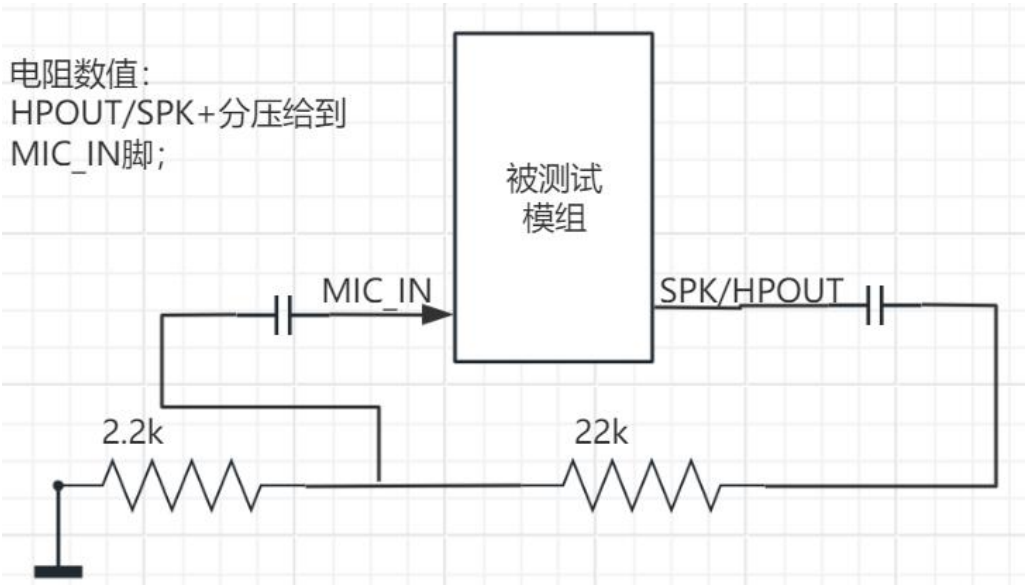


图 8.3 简化产测测试方案

简化产测方案说明，默认使用串口 1 接 wifi：
1、硬件上，将 hpout 分压接到 mic 输入；
2、软件上：wifi 发开始测试给 CI130X，CI130X 测试后返回测试成功和测试失败的指令给 wifi；

1、上位机发： CIAS_FACTORY_SELF_TEST_START
2、Wifi 设置能量值：CIAS_FACTORY_TEST_ENG_THR_SET
3、Wifi 发： CIAS_FACTORY_START ，
4、语音回复： CIAS_FACTORY_OK / CIAS_FACTORY_FAIL
5、wifi 可以通过：CIAS_FACTORY_TEST_ENG_GET 再次查询测试结果

测试过程观测音频能量方法：
Wifi 发：CIAS_FACTORY_TEST_REAL_VAL_GET ，如参数是 1，则语音会一直回复能量数值给 wifi 【调试使用】
具体操作
备注：以上代码写在应用逻辑中，测试的时候，无法进行语音识别等功能。

九、串口协议说明

9.1 串口命令格式概述

CI1302 与 wifi 采用串口进行数据通讯，串口设置为 921600 bps，无奇偶校验，8bit 数据，1bit 停止位，协议内的数据对齐方式为小端模式（低字节在前，高字节在后） 数据内容；串口协议内容如下：

名称	长度(字节)	模型描述
head	4	帧标识头，固定为:0x5a5aa5a5； WiFi 端收到的数据为：a5 a5 5a 5a
checksum	2	校验和（只校验数据部分）
cmd type	2	指令类型
len	2	数据长度 n
version	2	版本信息
fill_data	4	填充数据：控制指令用（DEF_FILL） DEF_FILL = 0x12345678, SPEAK_FILL = 0x12345677, INVAILD_SPEAK_FILL = 0x12345666, MUSIC_START_FILL = 0x12345699, MP3_FILL = 0x12345688, M4A_FILL = 0x123456aa, WAV_FILL = 0x123456bb, REPEAT_FILL = 0x123456ab,
data	n byte	数据部分

图 9.1 通讯协议内容表

下图为实际的一个通讯内容：

```
A5 A5 5A 5A 00 00 04 01 00 00 00 01 78 56 34 12
A5 A5 5A 5A 00 00 02 01 00 00 00 01 78 56 34 12
A5 A5 5A 5A 00 00 08 01 00 00 00 01 78 56 34 12
A5 A5 5A 5A B9 28 05 01 50 00 00 00 78 56 34 12 BB 42 01 B5 10 9D E7 C5
21 DD CD F8 3A 30 84 D1 BE 43 F4 09 E2 95 33 46 E3 D9 B7 E8 DD 14 93 5E
43 F7 DA 5B 6C 06 E3 21 21 B2 82 A8 72 B3 52 7F 5B E6 3F 37 24 43 B0 A3
BF A3 5D 83 9C 81 BB 58 A4 D3 50 09 67 FF 5C A3 A8 66 A2 DA 20 45 21 00
```

图 9.2 通讯数据解析

图中：A5 A5 5A 5A 00 00 04 01 00 00 00 01 78 56 34 12
每个部分的内容如下：
A5 A5 5A 5A：数据帧头
00 00 ： 校验码
04 01 ： 指令类型
00 00 ： 数据部分长度（本协议为指令，不包含数据，固为 00）
00 01 ： 版本号 0001

78 56 34 12 ：填充数据格式

9.2 指令类型说明

更多内容详见文档：
CI130X_SDK_ALG_PRO_AIOT_V1.0.10(1)(1)\CI130X_SDK_ALG_PRO_AIOT_V1.0.10\对接协议\启英泰伦离在线应用对接说明.pdf

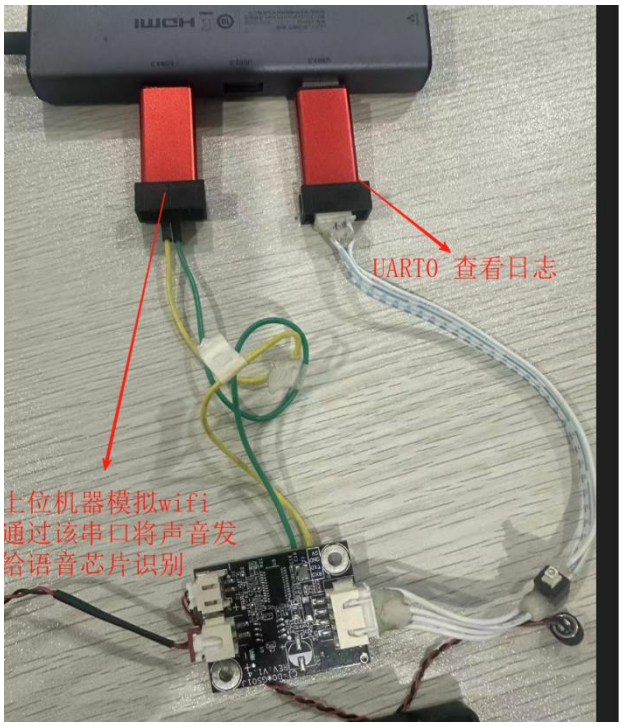
十、方案调试细节

10.1、串口采音及转为 wav 查看（以 Speex 为例）

见大模型采音文档

10.2 CI1302 播放 wifi 发送流媒体数据（以 MP3 为列）

启英提供上位机测试程序，模拟 wifi 发流媒体数据给语音芯片进行播放：
步骤 1、做合适的固件
代码配置的播放内容的格式与测试的模式一致，测试 MP3 时，需要设置 NET_AUDIO_PLAY_BY_MP3 1。
步骤 2、硬件连接及说明



步骤 3、开启上位机工具 chipintelli-audio-tools，按下图步骤导入正常可以用的测试音频，目前支持的三种格式：

MP3：16k，16bit，单声道，建议 32kbps 的码率

PCM：16k，16bit，单声道，PCM 是原始数据，wifi 将云端内容解析好发给语音芯片进行播放

OPUS：音频数据为 40ms（1280 字节）压缩一帧，压缩为 80 字节，16 倍的压缩率，CBR 格式；质量为 Audio 格式，带宽是 8，深度 16，通道为 1：格式 CELT；

从 UART0 看到如下 log：

```
[wifi msg:]recv type: 0201
NET_PLAY_START.......
requst one frame2, play buffer data size =0
[wifi msg:]recv type: 020b
pheader->len = 1024
play start
audio_play_state = 1
requst one frame2, play buffer data size =0
[wifi msg:]recv type: 020b
pheader->len = 1024
requst one frame2, play buffer data size =320
[wifi msg:]recv type: 020b
pheader->len = 1024
requst one frame2, play buffer data size =1344
[....中间数据手工删除.....]
[wifi msg:]recv type: 020b
pheader->len = 1024
requst one frame2, play buffer data size =3968
[wifi msg:]recv type: 020b
pheader->len = 1024
requst one frame2, play buffer data size =3840
[wifi msg:]recv type: 020b
pheader->len = 78
[wifi msg:]recv type: 020c
auido paly finish...
pheader->len = 0
play stop sync state to wifi ....
```

解析如下：

6、[wifi msg:]recv type 0201 表示收到 wifi 指令，0201 为开始播放

7、NET_PLAY_START..... 和 auido paly finish... 为播放的起始和结束位置

8、requst one frame2, play buffer data size =0：表示当前 1302 播放 buffer 中数据长度是 0，并向 wifi 获取数据，其他处的 play buffer data size =320/1344/...都是当前播放 buffer 的已有数据；当前播放默认总大小：AIOT_AUDIO_PLAY_BUF_SIZE 为 6k，通过此数值可以知道播放 buffer 还有多少数据可以播放

9、 [wifi msg:]recv type: 020c 表示收到 wifi 指令，020c 为 wifi 发送数据完成，此时播放器会继续播放未播完的 play buffer data size 的音频

10、play stop sync state to wifi .... 为播放完播放 buffer 的数据，此时播放器同步本地和 wifi 的播放状态

注意事项：

1、对接到云端后，如果 wifi 播放还未播放完中，出现 play buffer data size =0 或者很小的情况，意味着 wifi 未及时拿到云端数据，播放器已经无数据播放，此时会表现为卡顿情况

2、如打印 mp3 decord err -9,bad frame!,remove 2byte,detect next，类似 log，可以查看是否数据中断导致异常

10.3AEC 录音及调优

离在线的 sdk 内存紧张，建议使用算法 sdk 先对结构和 aec 调好，再通过在

线抓到的音频数据判断 asr 的生效情况

1、如果是有 wifi/4G 进行播放在线内容，当播放的时候，需要通过指令告诉语音模块处在播放状态(当前 ALG_PRO_AIOT_V2.0.10 sdk 无对应指令)

2、当产品结构较小，喇叭 mic 距离近的情况，可以调节 alc_off_codec_adc_gain_mic 调小，避免喇叭声音太大的时候 mic 消顶导致 aec 效果异常；（备注：本参数是在 aec 运行过程中，mic 的增益设置是多大）；

如果产品结构上 mic 和喇叭远离较大，可以调大该参数；

```
// aec模块配置
const aec_config_t aec_config =
{
    .alg_enable = true,
    #if (USE_BEAMFORMING_MODULE && BF_DEEPSE_MODE)
    .mic_channel_num = 2,                                   //麦克风信
    #else
    .mic_channel_num = 1,
    #endif
    .ref_channel_num = 1,                                   //参考
    .aec_control_mode = ENABLE_PLAYING_STATE_MODE,          //ENA
    .aec_gain = 1.0f,                                       //增益
    .aec_enable_threshold = 2000.0f,                        //参考
    .aec_mic_div_ref_thr = 0.0f,
    .aec_ref_db_thr = 130.0f,
    .nlp_flag = 2,
    .aggr_mode = 1,
    .fft_size = 256,        //频域处理频点数
    .alc_off_codec_adc_gain_mic = 20,   //可调
    .alc_off_codec_adc_gain_ref = 0     //可调
};
```

10.4 内存查看

代码每段时间会打印如下内容：

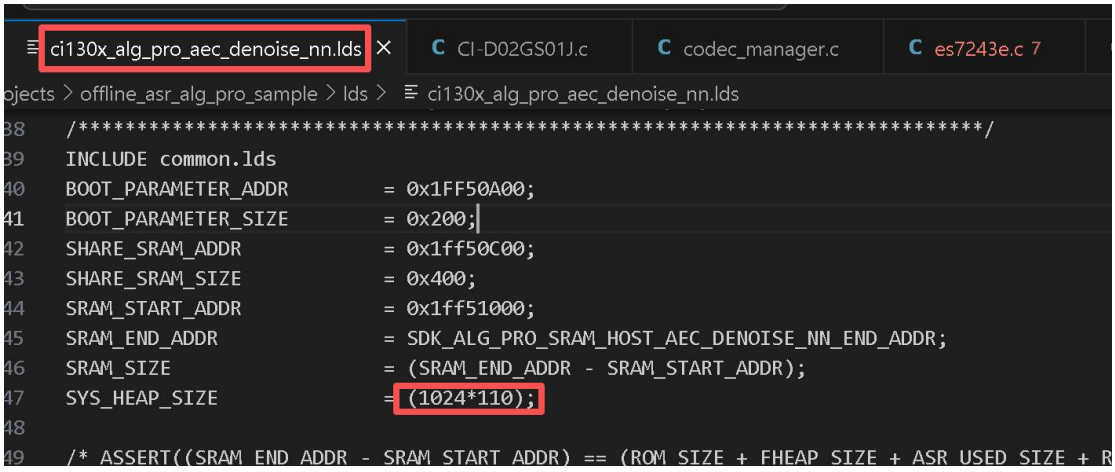
Asr heap min free 是识别部分的最小剩余内存；【表示的是历史上最小的时候内存的大小，不是现在内存大小】

System heap min free：是 rtos 运行的最小内存；【表示的是历史上最小的时候内存的大小，不是现在内存大小】

```
TaskName            Priority        TaskNumber      MinStk  13
pcm_data_play_t     4               12              262
init task           4               1               26
request_play_da     4               8               402
IDLE                0               2               99
record_data_upl     4               10              254
network_recv_da     4               7               310
audio_in_manage     4               6               102
decoder_manage_     4               5               250
vad_state_handl     4               9               154
Tmr Svc             5               3               90
remote call tas     4               4               194
UserTaskManageP     4               13              306
network_send_da     4               11              378

asr heap min free:96KB
system heap min free:11KB
system heap free:11KB
```

如需调整大小，需要在 makefile 中配置的对应 lds 文件，修改 sys heap size，调整后剩余的空间就是给 asr 识别使用的：



10.5 VAD 调优

1、vad 灵敏度，修改 sdk_default_config 的 VOX_VAD_THRE_DB_DEFAULT 如下：

#define VOX_VAD_THRE_DB_MIN 45

#define VOX_VAD_THRE_DB_MAX 60

#define VOX_VAD_THRE_DB_DEFAULT 50.0f //值越大，vad 越不容易触发，语音检测能量阈值(45~60dB) 高灵敏度 45~48dB 中灵敏度 49~52dB 低灵敏度 53~60dB

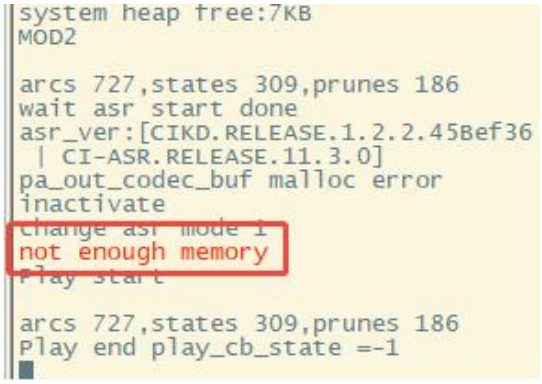
应用注意：
A、产品的应用距离为较近距离，希望尽量多的过滤噪音和远距离人声，此时应该调大改参数；
B、如果产品需要远距离使用，需要尽量灵敏的上传人讲话的内容，此时英调小改数值：

特别注意：唤醒词为离线识别，是另外一套独立的 vad，其效果近场远程都可以，不受到改参数的影响;

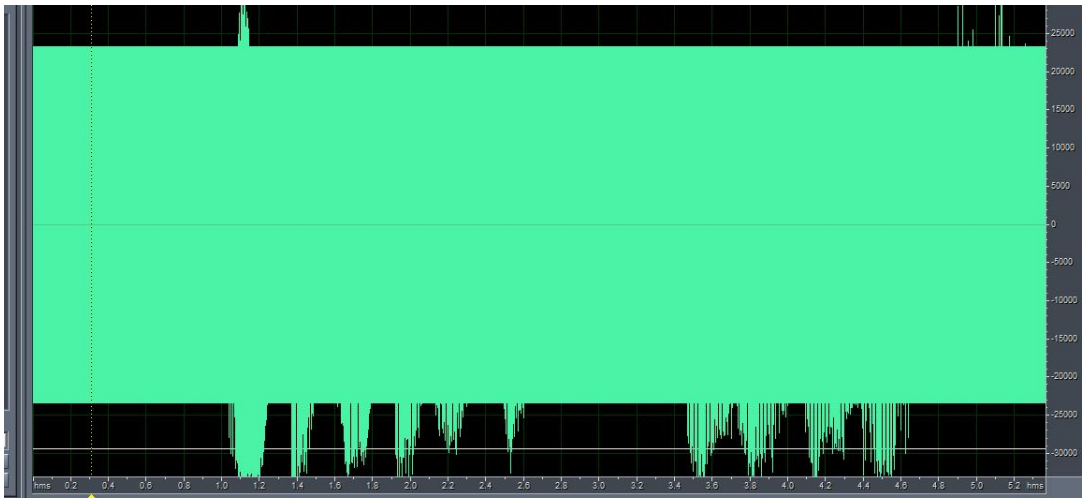
十一、调试中常见问题解决提示

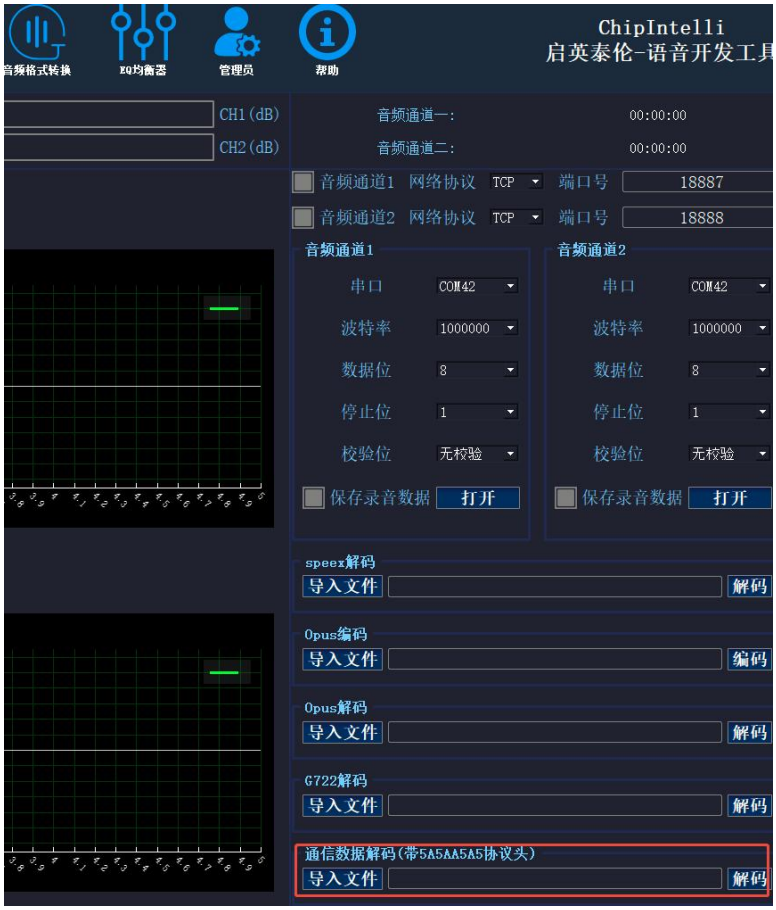
- **串口解析错误，检查串口配置，以及发送端的数据是否正确（使用启英泰伦-语音开发工具则重启工具）**

- 内存问题打印 Heap overflow asr 堆不够，not enough memory 系统堆不够，（本地识别、本地播放 MP3、SPEEX/OPUS 压缩使用 asr 堆，云端播放 PCM、G722 编码解码使用系统堆）



- 用启英泰伦-语音开发工具采音出现全程截幅问题，需先解析协议头，再根据压缩方式（SPEEX\OPUS\G722）解码，或者未压缩，则解析头后则可直接使用





- **更改压缩方式后，sdk 编译失败，需要在 makefile 中对应修改**

```
15  ######################################################################
16  CI_ALG_TYPE := $(USE_ANY_MIC_AEC_DENOISE_NN)
17
18
19  ########可组合算法使能(注意:设置变量值后面不能有空格，只能是0/1/2...)########
20  #---单麦就近唤醒:1-开启/0-关闭
21  USE_PWK_ENABLE                  := 0
22  #---是否开启动态ALC功能，1:开启，0:关闭; -单mic算法，仅单mic可用
23  USE_ALC_AUTO_SWITCH_MODULE      := 0
24  #---意图数量:1-单意图   >1为多意图，最大支持3意图，除USE_NULL、USE_DENOISE_NN和USE_DEREVERB可以支持多意图外，其他算法选项都只支持单意图
25  MULT_INTENT                     := 1
26  #---AIOT 音频压缩类型   #0-null or g722  1-speex  2-opus
27  AIOT_AUDIO_COMPRESS_TYPE        := 1
28  #---音频上传云端接口类型：#1 串口输出  2 hpout输出 不可修改
29  AUDIO_UPLOAD_TYPE               := 1
30  #---播放器类型：0-精简播放器(相比老播放器节约内存15KB,功能简单) 1-老播放器(占内存较多，功能完善)
31  USE_AUDIO_PLAYER_TYPE           := 0
32  ######################################################################
33
34  #算法功能编译配置
35  ifeq ($(CI_ALG_TYPE), $(USE_NULL))
```

- **在线播放卡顿，检查 WIFI 端收到数据请求 020a 后是否下发 020b 数据**