

AC696X 软件问题整理

链接: <https://pan.baidu.com/s/1kjhBSPTfegAm3xRpuVv8NA>

提取码: fxq3

以下是杰理客户问题反馈的二维码图片, 可以通过微信扫二维码填写反馈!

也可以通过访问链接填写反馈: <https://www.wjx.cn/vj/O1EbrN.aspx>

我们将第一时间安排对应工程协助解决!



1.AC696X DCDC 模式时需要设置 CLOCK_MODE_IGNORED , LDO 模式时就设置 CLOCK_MODE_ADAPTIVE 20200407	7
2.AC696 系列应用层关闭长按复位功能 20200504	7
3.AC696 系列测试盒修改复位脚 20200504	8
4.AC696 系列耳机 SDK 版本 V0.1.3 版本注意 20200520	9
5.AC696 系列 SDK0.1.3 关充电拔出开机和和 BLE 功能后, 进入低功耗模式程序会进入睡眠问题 20200516	12
6.AC696 系列 SDK0.1.3 关掉 TWS 功能后蓝牙播歌没有声音问题修改 20200516	13
7.AC696 系列 软件或电阻检测分左右耳之后, 用测试盒配对, 能显示配对成功但实际没有配对上处理 20200602	14
8.AC696 系列音箱版本 sdk 模拟的 DAC 初始化外置定义 20200610	15
9.AC696 版本支持 MTY 格式提示音 20200618	18
10.AC696 系列蓝牙耳机和音箱关于 PD4 IO 使用注意 20200619	20
11.AC696N 2M 程序烧写到 4M 容量芯片注意事项 20200619	21
12.动态开关长按复位 20200619	22
13.使用 AC6969D 调式样机出现通话远端没有声音问题处理方法 20200619	22
14.AC696X 跑 LDO15 若出现部分机子通话时样机复位处理方法 20200619	22
15.AC696 系列 2M 的 SDK 开蓝牙一拖二出现奇怪的问题 20200624	24
16.AC696 系列开启 Gsenor 编译不过 20200701	24
17.AC696 测试盒配对的 TWS 清配对注意点 20200701	25
18.AC696 系列耳机版本 V014-P1 优化 20200721	26
19.AC696 系列音箱版本 V024 和耳机 SDK031 获取连接手机的蓝牙名和地址 20220707---WTS 更新	27
20.关于弱 VDDIO 档位为 2.4V 给触摸 IC 的供电引起开关机漏电问题, 重要!! 20200721	28
21.AC696X SDK0.2.3-p1 录音录到外部 flash 后, 播放不了问题 20200711	29
22.AC696X SDK014-p1 对耳连苹果手机开 SIRI 有时蓝牙会断连问题处理 20200711	29
23.AC696X 外挂 FM 时, FM 的 IIC 口和卡口的 CMD 脚及 DATA 脚复用问题 20200713	30
24.AC696X 耳机和音箱 SDK 的 pwm 功能 20200714	31
25.AC696X 在蓝牙播放歌曲的时候播发 msbc 提示音注意点 20200714	32
26.AC696X 混响和回声, 有变量选择 20200716	32
27.AC696X EQ 在线调式注意点 20200716 CCB	33
29.AC696 系列用于音箱应用注意点 20200721 YWP	33
30.AC696X 软件编码器实现方法 20200724 LZK	33
31. linein 复用 CMD 检测配置例子 20200724 LZK	35
32. GPIO 相关函数参数传参不正常会导致 USB 功能不正常 20200724 LZK	35
33. Linein 单声道模拟输入变双声道输出函数,(DAC 声道跟 linein 声道不一致也适用) 20200724 LZK	36
34. SD_PG/PA0 脚无法正常控制 20200724 LZK	36
35.AC696 添加设备(优先)独立模式功能 20200724 LZK	37
36.AC696 音箱版本直接播放 MP3 提示音方法 20200801 YWP	42
37.696 系列 K 歌宝解决第一声延时长的问题 20200803 YWP	43
38. music 模式下 LED7 显示、播放对应设备名, 并在播放完之后显示文件号 20200808 SQ	45
39. 报号前播放指定一个提示音 20200808 SQ	55
40. 固定音量播放提示音 20200808 SQ	56
41.Linein 模式走模拟使用抢断方式播放提示音带有 Linein 声音 20200808 SQ	57

42.AC696 系列 V024V025 版本的 FM 背噪杂音的库优化	20200808	PYP	59
43.【音箱】AC696 系列 DAC 输出前后有 PO 声 (DAC 设置高阻引起)	202008014	LZK	59
44.【音箱】AC696 系列 AUX 走模拟时播叠加提示音时, 音乐出现卡顿处理方法	202008015	WTS	60
45.AC696 系列单独调节系统叠加提示音音量方法	202008015	WTS	62
46.AC696 系列直接在代码中修改叠加提示音方法	202008015	WTS	63
47.AC696 系列音箱 SDK2.5 判断录音录到哪个设备	202008015	WTS	66
48.AC696 系列音箱 SDK2.5 快速按遥控有时不灵敏修改方法	202008015	WTS	67
49.AC696 系列如果误拿 ROM 中用到的 IO 口去做唤醒口, 出现充电脚 5V 拔出后, 耳机不会自动开机问题解决 方法	202008020	WTS	68
50.AC696X 音箱 SDK 在对箱时红外按键处理	2020 0822	LHY	75
51. AC696X 音箱 SDK 低电检测使用复位脚问题	LHY 2020 0822		76
52. AC696X 耳机 SDK016 面条耳机, 把按键消息配置在工具里面, 按键的消息会错位	XNW 20200824		76
53. AC696X 音箱 SDK025 使用外部 eq 配置文件后程序中的 6 个 eq 效果不起作用	SQ 20200825		77
54. AC696X 音箱 SDK025 程序中实时修改 eq 某频点增益	SQ 20200825		80
55. AC696X ac696n_sdk_release_v0.1.6_2M 包加入来电铃声和来电报号的问题	YP		83
56. AC696X 程序上设置自定义蓝牙名【通用】	LZK		85
57. 025 版本带混响功能有概率死机	20200910	xin	86
58.AC696 的 V025 版本 SDK 进入蓝牙会自动退出蓝牙模式问题	20200911	YWP	86
59.AC696 系列外插 MIC 做混响或扩音设计注意	20200915	YWP	87
60.AC6965E 的 AUX/LINEIN 设计注意	20200915	YWP	88
61.AC696N 的 LADC 和蓝牙后台设计注意	20200915	YWP	89
62.AC696N 的蓝牙连接成功自动播放	20200916	FSW	89
63.AC696N 混响开发注意点	LZK		90
64. AC696 1.1.0 SDK 媒体音量与麦克风音量独立实现方法	20200925	LHY- 20210817 WGH 修改	93
65. AC696 1.1.0 SDK U 盘/SD 卡提示音独立实现方法	20200925	LHY	97
66.AC696 系列 LINEIN 左右声道与 DACLR 输入出的注意	20201015	YWP	98
67.AC696 系列 外挂 FLASH 使用方法与注意点	20201016	LHY	98
68. AC696 系列 SDK1.1.1 音乐模式频繁播放提示语导致播放异常	20201016	LHY	102
69. AC696 系列 SDK1.1.1 蓝牙模式提示语未播放完被连接提示语打断解决方法	20201016	LHY	103
70. AC696 系列 SDK1.1.1 指定拔出 U 盘切换到特定模式解决方法	20201016	LHY	104
71. AC696 系列 SDK1.1.1 开机优先设备(SD 卡/U 盘)上线实现方法	20201017	LHY	104
72. AC696 系列 SDK1.1.1 关闭 TWS 对箱回连解决方法	20201017	LHY	106
73. AC696 系列 SDK1.1.1 按键断开对箱后会回连的解决方法	20201017	LHY	106
74. AC696 系列 SDK1.1.1 对箱设置单声道时, 播歌下关其中一个音箱后, 另外一个有时会复位需要更换的 库	-----WTS 20201022		108
75. AC696 系列 SDK1.1.1 播完 U 盘或卡后, 不会自动播放下一个设备音乐的修改方法	-----WTS 20201022		108
76. AC696 系列 SDK1.1.1 切蓝牙模式播提示音的时候(蓝牙初始化未完成)插入 TF 卡,无法切入 TF 卡模式, 程序卡死问题处理方法	-----LAQ 20201022		109
77.AC696 系列 SDK1.1.1 数码管显示变化较快时, 出现某一段亮一些, 某一段暗一些的处理方法	-----LJW 20201023		110
78. AC696 系列 SDK1.1.1 一边调回声延迟, 一边用 MIC 说话会有沙沙声的处理方法	-----LJW 20201023		110
79. AC696 系列 SDK1.1.1 插卡无法扫描到隐藏文件和带.的文件名的文件的处理方法	-----LJW		

20201023	111
80.AC696 系列 SDK1.1.1 由于获取电量值改变的时间间隔太长导致电压检测不准的处理方法 -----LJW	
20201027	111
81.AC696 系列 SDK1.1.1 自动 mute 使用方法 -----LHY 20201028	111
82.AC696X SDK1.1.1 设置为单声道后, 有的机子有时播放 U 盘或卡音乐声音非常小解决方法 ----WTS	
20201102	112
83.AC696 系列 SDK1.1.1 快速插拔设备出现死机、无法切模式解决方法 -----LHY 20201102	113
84.AC696 系列 PC 模式插上电脑后默认 MIC 音量和系统音量值设置 -----WTS 20201103	114
85.AC696 系列 SDK1.1.1 DACR 做 linein 输入时没声音的处理方法 -----LJW 20201103	116
86.AC696 系列 SDK1.1.1 打开 PC 模式时某些 U 盘无法读取的处理方法 -----LJW 20201103	116
87.AC696 系列 SDK1.1.1 播放 MP3 提示音回复位的处理方法 -----LJW 20201105	117
88.AC696 系列 SDK1.1.1 单声道开启人声消除无效果的处理方法 FSW 20201105	117
89.AC696 系列 SDK1.1.1 播放 wav 音频时 SD 卡复用 AUX 失败补丁 WTS 20201106	119
90.AC696 系列 脉冲型驱动功放使用(us 级别) LHY 20201110	120
91.AC696 系列 优化插麦克风底噪问题 LHY 20201110	120
92.AC696 系列 高关功放在软关机下的处理 LHY 20201110	121
93.AC696 系列音箱 V1.1.1 获取音频数据的方法 FSW 20201110	122
94.AC696 系列音箱 V1.1.1 添加自定义的数字音量控制的方法 FSW 20201110	123
95.AC696 系列 SDK1.1.1 上下曲切歌有破声问题需要更换的库 WTS 20201111	125
96.AC696 系列 SDK1.1.1 无法读取 MMC 卡的问题处理方法 LJW 20201116	125
97.AC696 系列 SDK1.1.1 断开对箱后自动回链问题解决方法 LHY 20201117	125
98.AC696 系列 SDK1.1.1 连接手机状态下无法连接或者断开对箱解决方法 LHY 20201117	126
99.AC696/AC695 修改蓝牙配对码 PINCODE 20201118 YWP	127
100.AC696X 音箱 SDK 自动调节 VDDIO 电压 20201119 YP	128
101.AC696X 音箱 SDK 内置充电有时会处于很小(20mA)左右的处理方法 20201119 YP	128
102.AC696X 音箱 SDK 1.1.1 硬件微妙延时 SQ 20201123	129
103.耳机工程升级到 SDK020, 发现同样的参数, 回音比以前的版本明显, 调回音压制参数效果不明显的处理方法 XNW 2020112	131
104.696x 音箱 SDK 添加定时器扫描函数后有几率搜不到蓝牙问题 20201124 YP	132
105.696x 音箱 SDK 蓝牙模式下打开 LINEIN 进行叠加 20201127 LHY	133
106.696x 音箱 SDK 搜不到蓝牙问题(补充) 20201127 LZK	133
107.AC696X 大音量播放歌曲声音会抖 20201201 HJY	134
108.AC696 系列 SDK1.1.1 开启内置充电后红外遥控不起作用的处理方法 LJW 20201203	135
109AC696 系列 SDK1.1.1 录音的同时 DAC 输出录音数据的处理方法 LJW 20201203	135
110.AC696 系列 SDK1.1.1 在 music 模式下开混响, 喊麦会卡音的处理方法 LJW 20201203	137
111.AC696 系列 SDK1.1.1 外部 EQ 文件替换 EQ 模式中的任何一种模式 HJY 20201203	137
112.AC696 系列 SDK1.1.1 外部添加双 EQ 文件的方法 HJY 20210419	138
113.AC696 系列 SDK1.1.1 如果出现莫名其妙的自动关机(非复位死机), 可以试下按下图修改 20201204 FSW	141
114.AC696X 111 SDK 版本使用 U 盘或者 SD 卡不能进行跳转升级的问题 20201204 YP	141
115.AC69 系列 sdk V111 版本无缝循环播放的处理 20201207 YWP	143
116.AC69 系列 sdk V111 插两驱动类型的(第一驱动类型为光盘模式) U 盘死机 20201207 SQ	144
117.AC696 系列 sdk 1.1.1 麦克风(混响)无法调数字音量解决方法 20201211 LHY	147
118.AC696 系列 sdk 1.1.1 主机连上手机后, 副机发起配对无法连接主机处理方法 20201211 WTS	148

119.AC696 系列 TWS 音箱同时按左右两边配对：一边连手机的情况下，配对慢	20201211	LZK	148
120.AC696N SDK1.1.1 录音模式播放录音文件时插入 U 盘播歌，再回录音模式播放录音文件时无法播放问题	20201211	LAQ	150
121.AC696N SDK1.1.1 退出 FM 之后功耗大	20201214	SQ	150
121.AC696N SDK1.1.1 利用定时器保存音乐断点	20201214	LJW	151
122.AC696N SDK1.1.1 音乐模式下获取播放设备的音乐文件数目	20201214	LJW	151
123. 配置多个唤醒脚时开机唤醒时判断哪个 IO 唤醒	20201214	xin	152
124. SDK1.1.1 开 TWS 同时开混响有一边音乐断续问题	20201218	WTS	152
125. 软开机有时候需要按几次才能正常开机解决方法	20201218	LHY	153
126. 低电检测报低电提示音后，电压拉高还会继续报提示音解决办法	20201221	LHY	154
127. 开机判断是哪个唤醒口唤醒	20201222	xin	155
128. SDK1.1.1 实时修改自定义 EQ 效果的指定频点增益	20201224	SQ	155
129. SDK1.1.1 对箱功能需要与 692x 一样	20201224	SQ	156
130. SDK1.1.1 打开混响 TCFG_MIC_EFFECT_ENABLE 这个宏后，使用测试盒进行通话测试会有啸叫的处理方法	20201226	LJW	158
131. SDK1.1.1 打开 linein 录音，播放的声音不正常的处理方法	20210104	LJW	160
132. SDK1.1.1 麦克风能量获取方法	20210105	LHY	160
133. SDK1.1.1 的 SD 选择 C 组、CMD 检测，录音设备为 SD 卡时，录音时 SD 卡会掉线的处理方法	20210108	LJW	161
134. SDK 1.2.0 linein 为 DAC 进使用打断方式播放提示音，一直有 linein 背景音	20210112	SQ	161
135.AC696X V111 版本的 SDK 对箱配对后来电接听近端无声音的问题	20210121	YP	164
136.AC696X V111 版本的 SDK 连接手机后，无法控制手机拍照的处理方法	20210126	LJW	165
136.AC696X SDK121 暂停播放提示音	20210309	XNW	166
137.AC696X V121 版本的 SDK 遥控功能和收音功能同时开异常处理方法	20210311	lzk	167
138.AC696X V020 版本的 SDK 普通包关 TWS，设置立体声输出，通话结束后会复位的处理方法	20210317	LJW	168
139.AC696X V020 版本的 SDK 设置立体声输出，播左右声道歌曲后 DAC 没有声音或声音变小的处理方法	20210317	LJW	169
141.AC6965 系列设置 DAC 管脚高阻态	20210326	YWP	169
142.AC696N 系列的左右调反实现	20210402	YWP	170
143.AC696Nmusic 模式下文件解码成功后保存断点时没有清除“时间”信息	20210412	SQ	177
144. ac696n_soundbox_sdk_v1.2.0 只能获取到 DAC_L 能量的问题	20210413	YP	177
145. 696n_sdk_v122 控制 PA1 脚(MIC 脚)电平状态，AUX 输入没有声音	20210419	WGH	179
146. 使用 MIC 脚做普通 IO 的注意事项	WGH		179
147.AC696X V122 版本的 SDK 音频文件属性不勾选“可以存档文件”的不能播放的问题			180
148. 20210430	LJW		180
149.AC696X V122 版本的 SDK 打开设备提示音，插 SD 卡再插 U 盘有啧啧声的问题			181
150. 20210430	LJW		181
151.AC696X V122 版本的 SDK 插 U 盘播歌速度变快的问题处理	20210430	LJW	181
152.AC696X 音箱工程 V123 版本安卓手机不能拍照或者手机回连后不能拍照	20210430	XNW	182
153.AC696X 音箱工程 V0.2.6、1.1.1 有概率出现软关机功耗大	20210506	LHY	182
154.AC696X 音箱工程 V1.2.3 部分手机不能拍照--红米 K30	20210513	XNW	182
AC696X 音箱工程 V1.2.3U 盘和 TF 卡复用，一直切设备，某些 U 盘会掉线的处理方法	20210513	LJW	183

155. 在主控的 flash 中存放 N 个文件，读取这些文件需要的时间有很大的差异	20210514 SQ	183
156. AC696x 1.2.2sdk 串口通信接收数据时产生一次中断之后，接下来接收数据异常	20210514 SQ	184
157. ac696n_soundbox_sdk_v1.3.0 开启串口 test 功能会复位问题	20210611 YP	186
158. ac696n_soundbox_sdk_v1.2.3 对箱配对后无法连接其他对箱解决方法	20210623 LCP	189
157.ac696n_soundbox_sdk_v1.3.0 linein 走 ADC,并复用 fm 引脚，使用省电容接法时，进 linein 出来后，麦声音变得很小解决方法	20210623 LCP	189
158. ac696n_soundbox_sdk_v1.1.1 TWS 非公共 mac 地址，去除配对限制	20210726 SQ	190
159. 公版 V1.3.0sdk2M 板级配置关掉 eq 之后打入电话长时间响铃死机	20210702 SQ	191
160. 1.2.2SDK 定时器使用注意点	20210702 SQ	192
162. 1.2.2SDK 读 U 盘 txt 文件案例	20210702 SQ	193
163. 1.2.3 SDK 打开 TWS 跳转升级无法维持 IO 电平	20210702 lzk	194
164. 1.3.0 MCU uart 串口升级注意事项	20210705 HJY	194
165. AC696 音箱 TWS 配对慢优化方法	20210705 LZK	195
166. ac696n_soundbox_sdk_v1.2.1-v1.3.0 蓝牙开混合录音 WAV 格式，录音音频出现卡音问题	20210707 WGH	196
167. ac696n_soundbox_sdk_v1.2.2-v1.3.0 对箱 TWS 通话有回音，噗噗声，卡顿问题	20210805 WGH	200
168. ac696n_soundbox_sdk_v1.2.2-v1.2.3 系统音量选择独立通道数字音量，播放提示音低概率复位问题	20210805 WGH	200
169. ac696n 系列耳机工程，如果用外置触摸，触摸通过主控的 IO 口供电，测试盒串口升级，升级到 70%，又重新升级问题修改方法	20210810 XNW	201
170. AC696X AUX 输入信号没有大于 1.2V，但 DAC 输出会失真问题处理	20210811 WTS	202
171. AC6969D 芯片使用内置 flash 录音修改文件	20210823 LCP	202
172. AC696X SDK130 对箱配对上后想保持音箱都输出立体声修改方法	20210824 WTS	203
173. 音箱工程 V1.3.1 版本 SDK 在连接上 TWS 后播歌时不断拨打挂断几次后有概率出现从箱无声音 A2DP 链路切换不成功的问题	20210830 MJW	205
174. 注意!! 注意!! 量产程序需要关掉异常中断!! 注意!! 注意	20210907 LHY	205
175. AC696 音箱蓝牙发射器和接收器之间进行对讲的 demo	20211028 FSW	206
176. AC696 0.2.0 耳机 SDK 版本 使用联合音量对箱后出现左右耳音量不同步解决方法	20211109 LHY	207
177. AC696 耳机 SDK 版本 使用 mic_rec_play_start()测试 mic 无声的解决方法	20211110 FSW	208
178. AC696 音箱 1.5.0 SDK 版本 Linein 走数字通道 Automute 不起作用	20220211 SQ	209
179. AC696 音箱 1.5.0 或者之前的 SDK 版本包括 695 的音箱 SDK 出现 Linein 检测有概率出现检测不了拔出	20220114 MJW	214
180. AC696N_soundbox_sdk_release_1.4.0 SDK 配成单声道会出现声音破音，波形失真	LZK	214
181. AC696 耳机 SDK020 开立体声时，通话前 20S 左右远端听不到声音处理方法	20201201 WTS	214
182. AC696_earphone_v030 版本 SDK 添加连接确认功能	---YP 20220506	214
182. AC696 音箱 SDK 使用 ANCS 功能		216
183.695V310 以及旧版本 SDK 当歌曲内容恰好存放于设备地址的底部，播放歌曲会异常退出设备	20220614 WGH	217
184.AC696 1.3.3SDK 打开人声消除的宏定义，声音失真		219
185. ac696n_sdk_release_v0.3.0 SDK VIVO 手机连接打开 SIRI 蓝牙断连问题		220
186. AC696X AUX 输入信号有效值没有超过 600MV，但从 DAC 输出出现失真问题处理	-----20220705 WTS	220
187. AC696X_170SDK_打开 TWS 以及公共地址使用左声道会出现 MAC 地址无法随机生成	-----20220709	

LZK 222
188.AC696X_170SDK 某些 IO 无法输出打印信息（DP,DM,PC4 等） WGH 223
189.AC696n_earphone_v0.2.0 设置立体声通话有整句回音处理方法 20220714---- WTS 224
190.AC696n_earphone_v1.7.0 替换 eq_cfg_hw.bin 的正确位置 20220815---- WGH 224

1.AC696X DCDC 模式时需要设置 CLOCK_MODE_IGNORED ， LDO 模式 时就设置 CLOCK_MODE_ADAPTIVE 20200407

```
ifig.h x board_config.h x board_ac6963a_tws_cfg.h x
L */
#define TCFG_CHARGE_MA          CHARGE_mA_60

//*****
//                                LED 配置                                //
//*****
#define TCFG_PWMLED_ENABLE      ENABLE_THIS_MOUDLE          //是否支持PMW LED推灯模块
#define TCFG_PWMLED_IOMODE      LED_ONE_IO_MODE            //LED模式, 单IO还是两个IO
#define TCFG_PWMLED_PIN         IO_PORTB_06                //LED使用的IO口
//*****
//                                时钟配置                                //
//*****
#define TCFG_CLOCK_SYS_SRC       SYS_CLOCK_INPUT_PLL_BT_OSC //系统时钟源选择
#define TCFG_CLOCK_SYS_HZ        24000000                 //系统时钟设置
#define TCFG_CLOCK_OSC_HZ        24000000                 //外界晶振频率设置
#define TCFG_CLOCK_MODE          CLOCK_MODE_IGNORED
//*****
//                                低功耗配置                                //
```

2.AC696 系列应用层关闭长按复位功能 20200504

```
// P33_CON_SET(P3_PINR_CON, 0, 1, 1); ///开启 8S 复位
P33_CON_SET(P3_PINR_CON, 0, 1, 0); ///关闭 8S 复位
```

```
Board_ac696x_demo.c 全部关闭
00958:
00959: static void board_power_wakeup_init(void)
00960: {
00961:     power_wakeup_init(&wk_param);
00962:     // P33_CON_SET(P3_PINR_CON, 0, 1, 1);    ///开启 8S复位
00963:     P33_CON_SET(P3_PINR_CON, 0, 1, 0);    ///关闭 8S复位
00964:
00965: #if TCFG_POWER_ON_NEED_KEY
00966:     extern u8 power_reset_src;
00967:     if ((power_reset_src & BIT(0)) || (power_reset_src & BIT(1))) {
00968: #if TCFG_CHARGE_ENABLE
00969:         log_info("is_ldo5v_wakeup:%d\n", is_ldo5v_wakeup());
00970:         if (is_ldo5v_wakeup()) {
00971:             return;
00972:         }
00973:         if (get_ldo5v_online_hw()) {
00974:             return;
00975:         }
00976:         /*LDO5V检测上升沿，用于检测ldoin插入*/
00977:         LDO5V_EN(1);
00978:         LDO5V_EDGE_SEL(0);
00979:         LDO5V_PND_CLR();
00980:         LDO5V_EDGE_WKUP_EN(1);
00981: #endif
00982:         power_set_callback(TCFG_LOWPOWER_LOWPPOWER_SEL, sleep_enter_callback, sleep_exit_callback, boa
00983:         power_set_soft_poweroff();
00984:     }
00985: #endif
00986: } ? end board_power_wakeup_init ?
00987: early_initcall(board_power_wakeup_init);
00988: #endif /* #ifndef CONFIG_BOARD_AC696X_DEMO */
```

3.AC696 系列测试盒修改复位脚 20200504

```
/**
 * @param pin 引脚号，如 IO_PORTB_01, IO_PORTA_00,,, 等等
 * @param level 有效电平，0: 低电平有效。 1: 高电平有效
 * @param time 长按时间有 00、01、02、04、08 三个值可选，单位为秒，
当长按时间为 00 时，则关闭长按复位功能。
 */
```

```
void reset_pin_init(u32 pin, u32 level, u32 time)
{
    if(time) {
        gpio_set_dieh(pin, 1);
        gpio_set_die(pin, 1);
        gpio_set_direction(pin, 1);
        if(level) {
            gpio_set_pull_up(pin, 0);
            gpio_set_pull_down(pin, 1);
        } else {
            gpio_set_pull_up(pin, 1);
            gpio_set_pull_down(pin, 0);
        }
    }
}
```

```
}  
switch (time) {  
case 1: //1s  
time = 0;  
break;  
case 2: //2s  
time = 1;  
break;  
case 4: //4s  
time = 2;  
break;  
case 8: //8s  
time = 3;  
break;  
}  
p33_tx_1byte(P3_PINR_CON, (level << 2) | (time << 4));  
p33_tx_1byte(P3_PORT_SEL10, pin);  
p33_or_1byte(P3_PINR_CON, BIT(0));  
} else { //0s  
p33_tx_1byte(P3_PINR_CON, 0);  
}  
}
```



```
power_manage.h Board_ac696x_demo.c 全部关闭  
00958:  
00959: static void board_power_wakeup_init(void)  
00960: {  
00961:     power_wakeup_init(&wk_param);  
00962:     // P33_CON_SET(P3_PINR_CON, 0, 1, 1); //开启 8S复位  
00963:     // P33_CON_SET(P3_PINR_CON, 0, 1, 0); //关闭 8S复位  
00964:     reset_pin_init(IO_PORTA_10, 0, 8);  
00965:     //  
00966:     #if TCFG_POWER_ON_NEED_KEY  
00967:     extern u8 power_reset_src;  
00968:     if ((power_reset_src & BIT(0)) || (power_reset_src & BIT(1))) {  
00969:     #if TCFG_CHARGE_ENABLE  
00970:     log_info("is_ldo5v_wakeup:%d\n", is_ldo5v_wakeup());  
00971:     if (is_ldo5v_wakeup()) {  
00972:         return;  
00973:     }  
00974:     if (get_ldo5v_online_hw()) {  
00975:         return;  
00976:     }  
00977:     /*LDO5V检测上升沿，用于检测ldoin插入*/  
00978:     LDO5V_EN(1);  
00979:     LDO5V_EDGE_SEL(0);  
00980:     LDO5V_PND_CLR();  
00981:     LDO5V_EDGE_WKUP_EN(1);  
00982:     #endif  
00983:     power_set_callback(TCFG_LOWPOWER_LOWPPOWER_SEL, sleep_enter_callback, sleep_;  
00984:     power_set_soft_poweroff());  
00985:     }  
00986:     #endif  
00987:     } ? end board_power_wakeup_init ?  
00988:     early_initcall(board_power_wakeup_init);  
00989:     #endif /* #ifdef CONFIG_BOARD_AC696X_DEMO */  
00990:
```

4.AC696 系列耳机 SDK 版本 V0.1.3 版本注意 20200520

如下方法解决 AC696 的播放歌曲偶尔咔音和通话咔音问题。也解决关闭 TWS 开单耳通话没声音问题。

版权所有，侵权必究

在 sdk_ld.c 文件加入如下语句, `*(.sbc_eng_code)`



```
00081:
00082: ENTRY(_start)
00083:
00084: SECTIONS
00085: {
00086: /*****
00087:  . = ORIGIN(code0);
00088:  .text ALIGN(4):
00089:  {
00090:      PROVIDE(text_rodata_begin = .);
00091:
00092:      *(.entry_text)
00093:      *startup.o(.text)
00094:
00095:      *(.text*)
00096:      bank_stub_start = .;
00097:      *(.bank.stub.*)
00098:      bank_stub_size = . - bank_stub_start;
00099:
00100:      *(.LOG_TAG_CONST*)
00101:      *(.rodata*)
00102:      . = ALIGN(4);
00103:      _SPI_CODE_START = .;
00104:      *(.spi_code)
00105:      . = ALIGN(4);
00106:      _SPI_CODE_END = .;
00107:      *(.sbc_eng_code)
00108:
00109:      . = ALIGN(4);
00110:      a2dp_sink_media_probe_begin = .;
00111:      KEEP(*(a2dp_sink_media_probe))
00112:      a2dp_sink_media_probe_end = .;
00113:
00114:      a2dp_sink_media_codec_begin = .;
00115:      KEEP(*(a2dp_sink_media_codec))
00116:      a2dp_sink_media_codec_end = .;
00117:  }
```

注意: 同时看下本文档的第 15 点的相关点。
把系统时钟提升到 96M。

```

l_ac696x_demo_cfg.h  Sdk_ld.c  App_config.h  全部关闭
00238: //*****
00239:
00240: #define PHONE_CALL_USE_LDO15    0
00241:
00242: //*****
00243: //          时钟切换配置          //
00244: //*****
00245:
00246: #define BT_NORMAL_HZ             96 * 100000L//24 * 100000L
00247: #define BT_CONNECT_HZ           96 * 100000L
00248:
00249: #define BT_A2DP_HZ               96 * 100000L//24 * 100000L
00250: #if TCFG_BT_SUPPORT_AAC
00251: #define BT_A2DP_STEREO_EQ_HZ    96 * 100000L//48 * 100000L
00252: #else
00253: #define BT_A2DP_STEREO_EQ_HZ    96 * 100000L//32 * 100000L
00254: #endif
00255: #define BT_A2DP_AAC_HZ           96 * 100000L//48 * 100000L
00256: #define BT_A2DP_TWS_AAC_HZ      96 * 100000L//64 * 100000L
00257: #define BT_A2DP_MONO_EQ_HZ      96 * 100000L//32 * 100000L
00258: #define BT_A2DP_ONLINE_EQ_HZ    96 * 100000L//48 * 100000L
00259:
00260: #define BT_CALL_HZ               96 * 100000L//80 * 100000L
00261: #define BT_CALL_ADVANCE_HZ      96 * 100000L//80 * 100000L
00262: #define BT_CALL_SIMPLEX_HZ      96 * 100000L
00263: #ifdef CONFIG_ANS_V2
00264: #define BT_CALL_16k_HZ           96 * 100000L
00265: #define BT_CALL_16k_ADVANCE_HZ  120 * 100000L
00266: #else
00267: #define BT_CALL_16k_HZ           96 * 100000L//80 * 100000L
00268: #define BT_CALL_16k_ADVANCE_HZ  96 * 100000L
00269: #endif
00270: #define BT_CALL_16k_SIMPLEX_HZ  120 * 100000L
00271:
00272: #ifdef CONFIG_CPU_BR26
00273: #undef BT_CALL_16k_HZ
00274: #undef BT_CALL_16k_ADVANCE_HZ
00275: #define BT_CALL_16k_HZ           96 * 100000L
00276: #define BT_CALL_16k_ADVANCE_HZ  96 * 100000L
00277: #endif
00278:
00279: #ifdef CONFIG_CPU_BR23
00280: #undef BT_A2DP_STEREO_EQ_HZ
00281: #define BT_A2DP_STEREO_EQ_HZ    96 * 100000L//48 * 100000L
00282: #undef BT_A2DP_MONO_EQ_HZ
00283: #define BT_A2DP_MONO_EQ_HZ      96 * 100000L//48 * 100000L
00284: #endif
00285:
00286: #ifdef CONFIG_FPGA_ENABLE
00287:
00288: #undef TCFG_CLOCK_OSC_HZ
00289: #define TCFG_CLOCK_OSC_HZ       12000000
00290:

```

注意：同时看下本文档的第 15 点的相关点。

5.AC696 系列 SDK0.1.3 关充电拔出开机和和 BLE 功能后，进入低功耗模式程序会进入睡眠问题 20200516

如果用 SDK013 设置 TCFG_CHARGE_OFF_POWERON_NE 为 0，同时 TCFG_USER_BLE_ENABLE 也设置为 0 时。出现样机一进入低功耗模式就会关机问题。解决方法如下：

在函数 charge_init(const struct dev_node *node, void *arg) 中加入：

```
//原因:softoff后没有rc时钟唤醒的滤波逻辑不工作给不出唤醒信号
struct port_wakeup port;
port.pullup_down_enable = 0;
port.edge = RISING_EDGE;
port.attribute = 0;
port.iomap = IO_LDOIN_DET;
power_wakeup_set_wakeup_io(7, &port);
power_wakeup_index_disable(7);

if (check_charge_state()) {
    if (__this->ldo5v_timer == 0) {
        __this->ldo5v_timer = sys_hi_timer_add(0, ldo5v_detect, 2);
    }
} else {
    CHARGE_WKUP_PND_CLR();
    CHGBG_EN(0);
    CHARGE_EN(0);
    charge_flag = BIT_LDO5V_OFF;
}
charge_set_callback(charge_wakeup_isr, sub_wakeup_isr);
/* sys_hi_timer_add(0, test_func, 1); */

__this->init_ok = 1;

return 0;
}

const struct device_operations charge_dev_ops = {
    .init = charge_init,
};
```

加入这句话

6.AC696 系列 SDK0.1.3 关掉 TWS 功能后蓝牙播歌没有声音问题修改 20200516

修改 ac696n_earphone_v0.1.3\sdk\cpu\br25\sdk_ld.c 文件如下:

```
2020/5/7 15:08:05 9,491 字节 C, C++, C#源代码 UTF-8 UNIX 2020/4/20 19:30:23 9,469 字节 C, C++, C#源代码 UT
```

```
91     *(.entry_text)
92     *startup.o(.text)
93
94     *(.text*)
95     bank_stub_start = .;
96     *(.bank.stub.*)
97     bank_stub_size = . - bank_stub_start;
98
99     *(.LOG_TAG_CONST*)
100    *(.rodata*)
101    . = ALIGN(4);
102    _SPI_CODE_START = .;
103    *(.spi_code)
104    . = ALIGN(4);
105    _SPI_CODE_END = .;
106
107    *(.sbc_eng_code)
108
109    . = ALIGN(4);
110    a2dp_sink_media_probe_begin = .;
111    KEEP(*(.a2dp_sink_media_probe))
112    a2dp_sink_media_probe_end = .;
113
114    a2dp_sink_media_codec_begin = .;
115    KEEP(*(.a2dp_sink_media_codec))
116    a2dp_sink_media_codec_end = .;
117
118    a2dp_event_handler_begin = .;
119    KEEP(*(.a2dp_event_handler))
120    a2dp_event_handler_end = .;
121
122
```

在sdk_ld.c文件中加入这句话

7.AC696 系列 软件或电阻检测分左右耳之后，用测试盒配对，能显示配对成功但实际没有配对上处理 20200602

修改如下：

```
board_config.h  x  app_chargestore.c  x  board_ac6956a_tws_cfg.h  x  key_event_deal.c  x  key_driver.c  x  event.h  x
112  #endif
113
114  #if TCFG_USER_TWS_ENABLE
115  bool chargestore_set_tws_remote_info(u8 *data, u8 len)
116  {
117      u8 i;
118      bool ret = true;
119      u8 remote_addr[6];
120      u8 common_addr[6];
121      u8 local_addr[6];
122      CHARGE_STORE_INFO *charge_store_info = (CHARGE_STORE_INFO *)data;
123
124
125      puts("\n-----chargestore_set_tws_remote_info-----\n");
126
127
128
129      #if (CONFIG_TWS_CHANNEL_SELECT == CONFIG_TWS_AS_LEFT_CHANNEL) \
130          || (CONFIG_TWS_CHANNEL_SELECT == CONFIG_TWS_AS_RIGHT_CHANNEL) \
131          || (CONFIG_TWS_CHANNEL_SELECT == CONFIG_TWS_EXTERN_DOWN_AS_LEFT) \
132          || (CONFIG_TWS_CHANNEL_SELECT == CONFIG_TWS_EXTERN_UP_AS_LEFT)
133      if (len > (sizeof(CHARGE_STORE_INFO)+1)) {
134
135          log_info("len err\n");
136          return false;
137      }
138      #else
139      if (len > sizeof(CHARGE_STORE_INFO)) {
140
141          log_info("len err\n");
142          return false;
143      }
144      #endif
145
146      //set remote addr
147      syscfg_read(CFG_TWS_REMOTE_ADDR, remote_addr, sizeof(remote_addr));
148      if (memcmp(remote_addr, charge_store_info->tws_local_addr, sizeof(remote_addr))) {
149          ret = false;
150      }
151      syscfg_write(CFG_TWS_REMOTE_ADDR, charge_store_info->tws_local_addr, sizeof(charge_sto:
152
153      //set common addr
154      bt_get_tws_local_addr(local_addr);
155      syscfg_read(CFG_TWS_COMMON_ADDR, common_addr, sizeof(common_addr));
156
157      for (i = 0; i < sizeof(common_addr); i++) {
```

8.AC696 系列音箱版本 sdk 模拟的 DAC 初始化外置定义 20200610

```
static void __dac_analog_init(struct audio_dac_hdl *dac, u8 LR_on)
{
    if (dac->analog.inited) {
        return;
    }

    SFR(JL_ANA->DAA_CON2, 4, 1, 0); // LPF_MUTE_11v
    delay(1000);
    SFR(JL_ANA->DAA_CON2, 14, 1, 1); // TRIM_EN_11v
    delay(1000);
    SFR(JL_ANA->DAA_CON2, 22, 1, 1); // PNS10K_EN_11v

    SFR(JL_ANA->DAA_CON0, 29, 1, 0); // ZCEN
    SFR(JL_ANA->DAA_CON0, 28, 1, 0); // FIR_FILTER_ENB_11v
    SFR(JL_ANA->DAA_CON0, 23, 1, 1); // DACVCM_RSEL_11v
    SFR(JL_ANA->DAA_CON0, 21, 1, 0); // ADCVCM_2_DAC_EN_11v
    SFR(JL_ANA->DAA_CON0, 10, 3, 0); // RESERVED_BIT_11v[2:0]
    SFR(JL_ANA->DAA_CON0, 9, 1, 0); // ZERO_RC_EN_11v
    SFR(JL_ANA->DAA_CON0, 8, 1, 0); // ZERO_LC_EN_11v
    SFR(JL_ANA->DAA_CON0, 7, 1, 0); // LPF_2_ZERO_EN_11v
    SFR(JL_ANA->DAA_CON0, 4, 1, 0); // DACVDD_TEST_EN_11v
    SFR(JL_ANA->DAA_CON0, 3, 1, 0); // CTADCREF_TEST_11v
    SFR(JL_ANA->DAA_CON0, 2, 1, 0); // MICLDO_TEST_11v
    SFR(JL_ANA->DAA_CON0, 1, 1, 0); // VOUTR_TEST_EN_11v
    SFR(JL_ANA->DAA_CON0, 0, 1, 0); // VOUTL_TEST_EN_11v

    // --- DAA_CON1 ---
    // -----
    SFR(JL_ANA->DAA_CON1, 31, 1, 0); // LPFVCM_SEL_11v
    SFR(JL_ANA->DAA_CON1, 30, 1, 0); // AMUX_MUTE_11v
    SFR(JL_ANA->DAA_CON1, 29, 1, 0); // AMUX_G_11v
    SFR(JL_ANA->DAA_CON1, 28, 1, 0); // AMUX_EN_11v
    SFR(JL_ANA->DAA_CON1, 27, 1, 0); // AMUX_BIAS_EN_11v
    SFR(JL_ANA->DAA_CON1, 26, 1, 0); // LIN1R_EN_11v
    SFR(JL_ANA->DAA_CON1, 25, 1, 0); // LIN1R_BIAS_EN_11v
    SFR(JL_ANA->DAA_CON1, 24, 1, 0); // LIN1L_EN_11v
    SFR(JL_ANA->DAA_CON1, 23, 1, 0); // LIN1L_BIAS_EN_11v
    SFR(JL_ANA->DAA_CON1, 22, 1, 0); // LIN0R_EN_11v
    SFR(JL_ANA->DAA_CON1, 21, 1, 0); // LIN0R_BIAS_EN_11v
    SFR(JL_ANA->DAA_CON1, 20, 1, 0); // LIN0L_EN_11v
    SFR(JL_ANA->DAA_CON1, 19, 1, 0); // LIN0L_BIAS_EN_11v
    SFR(JL_ANA->DAA_CON1, 18, 1, 0); // LR_2_R_11v
    SFR(JL_ANA->DAA_CON1, 17, 1, 0); // LR_2_L_11v
}
```

```
SFR(JL_ANA->DAA_CON1, 5, 5, 0); // RG_SEL_11v[4:0]
SFR(JL_ANA->DAA_CON1, 0, 5, 0); // LG_SEL_11v[4:0]
// --- DAA_CON2 ---
// -----
SFR(JL_ANA->DAA_CON2, 24, 1, 0); // PNS_EN_11v
SFR(JL_ANA->DAA_CON2, 23, 1, 0); // PNS50K_EN_11v
SFR(JL_ANA->DAA_CON2, 22, 1, 0); // PNS10K_EN_11v
SFR(JL_ANA->DAA_CON2, 21, 1, 0); // BIAS_2_OUT_EN_11v
SFR(JL_ANA->DAA_CON2, 20, 1, 0); // VCM_RISETIME_EN_11v
SFR(JL_ANA->DAA_CON2, 18, 1, 0); // TRIM_SW_11v
SFR(JL_ANA->DAA_CON2, 16, 2, 0); // TRIM_SEL_11v[1:0]
SFR(JL_ANA->DAA_CON2, 15, 1, 0); // TRIM_LPF_EN_11v
SFR(JL_ANA->DAA_CON2, 5, 2, 0); // LS_DTSEL_11v[1:0]
SFR(JL_ANA->DAA_CON2, 4, 1, 0); // LPF_MUTE_11v
// --- DAA_CON4 ---
// -----
SFR(JL_ANA->DAA_CON4, 3, 1, 0); // LIO_DATA_IN_11v
SFR(JL_ANA->DAA_CON4, 2, 1, 0); // LIO_MODE_EN_11v
SFR(JL_ANA->DAA_CON4, 1, 1, 0); // RIO_DATA_IN_11v
SFR(JL_ANA->DAA_CON4, 0, 1, 0); // RIO_MODE_EN_11v

delay(200);
SFR(JL_ANA->DAA_CON2, 19, 1, 1); // VCM_EN_11v

SFR(JL_ANA->DAA_CON2, 0, 4, dac->pd->lpf_ise1 & 0xf); // LPF_ISEL_11v[3:0]

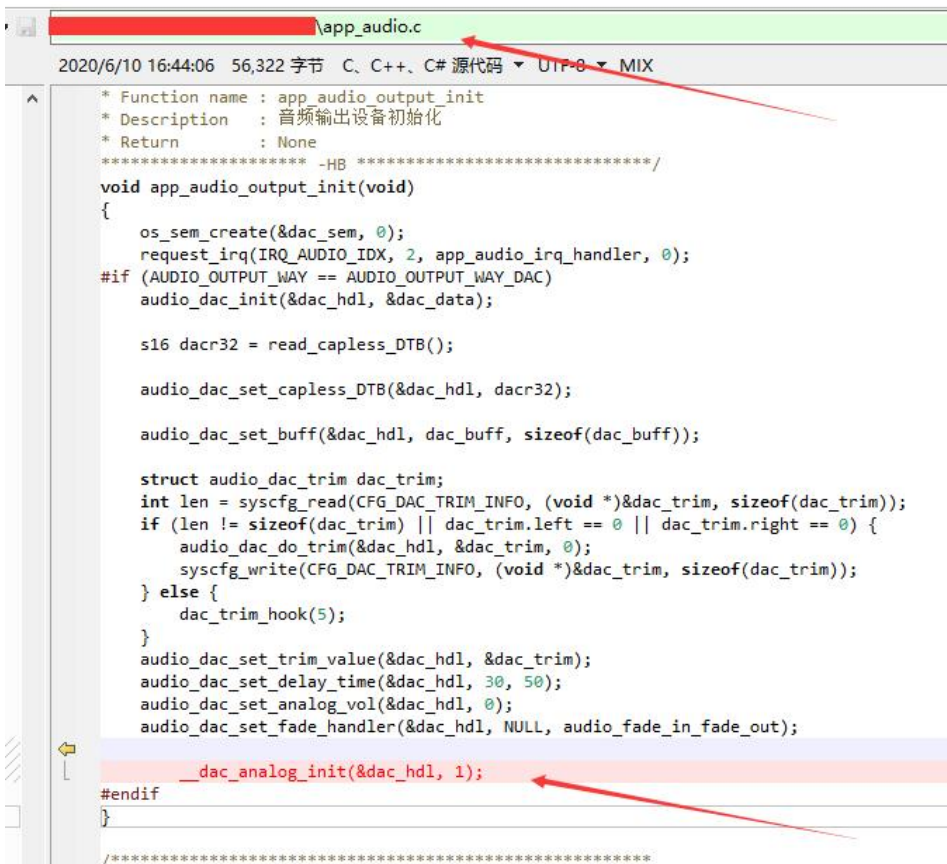
delay(1000);
switch (dac->pd->output) {
case DAC_OUTPUT_MONO_L:
    SFR(JL_ANA->DAA_CON1, 13, 1, 1); // L_RDAC_EN_11v
    SFR(JL_ANA->DAA_CON1, 10, 1, 1); // HP_L_EN_11v
    break;
case DAC_OUTPUT_MONO_R:
    SFR(JL_ANA->DAA_CON1, 14, 1, 1); // R_RDAC_EN_11v
    SFR(JL_ANA->DAA_CON1, 11, 1, 1); // HP_R_EN_11v
    break;
case DAC_OUTPUT_LR:
// case DAC_OUTPUT_SPECAIL_LR:
case DAC_OUTPUT_MONO_LR_DIFF:
    SFR(JL_ANA->DAA_CON1, 14, 1, 1); // R_RDAC_EN_11v
    SFR(JL_ANA->DAA_CON1, 13, 1, 1); // L_RDAC_EN_11v
    SFR(JL_ANA->DAA_CON1, 11, 1, 1); // HP_R_EN_11v
    SFR(JL_ANA->DAA_CON1, 10, 1, 1); // HP_L_EN_11v
    break;
```

```
default:
    break;
}

SFR(JL_ANA->DAA_CON1, 15, 2, dac->pd->ldo_fb_isel); // LDO_FB_ISEL_11v[1:0]
SFR(JL_ANA->DAA_CON2, 7, 4, 3); // SDDAC_DTSEL_11v[3:0]
SFR(JL_ANA->DAA_CON2, 11, 3, dac->pd->ldo_isel); // LDO_ISEL_11v[2:0]
SFR(JL_ANA->DAA_CON0, 25, 3, (dac->pd->ldo_volt & 0x7)); // DACVDD_LDO_VSEL_11v[2:0]
SFR(JL_ANA->DAA_CON0, 22, 1, 1); // DACVCM_EN_11v
os_time_dly(100);
SFR(JL_ANA->DAA_CON0, 5, 1, 1); // AUDIO_IBIAS_EN_11v
SFR(JL_ANA->DAA_CON0, 13, 1, 1); // CHOPPER_CLK_EN_11v
SFR(JL_ANA->DAA_CON0, 24, 1, 1); // DACVDD_LDO_EN_11v <-----这里是 DAC 上电
os_time_dly(3);
SFR(JL_ANA->DAA_CON0, 6, 1, 1); // AUDIO_VBIAS_EN_11v

delay(1000);
SFR(JL_ANA->DAA_CON2, 22, 1, 0); // PNS10K_EN_11v
SFR(JL_ANA->DAA_CON2, 14, 1, 0); // TRIM_EN_11v

dac->analog.inited = 1;
}
```



```
app_audio.c
2020/6/10 16:44:06 56,322 字节 C、C++、C# 源代码 UTF-8 MIX
* Function name : app_audio_output_init
* Description : 音频输出设备初始化
* Return : None
***** -HB *****
void app_audio_output_init(void)
{
    os_sem_create(&dac_sem, 0);
    request_irq(IRO_AUDIO_IDX, 2, app_audio_irq_handler, 0);
    #if (AUDIO_OUTPUT_WAY == AUDIO_OUTPUT_WAY_DAC)
        audio_dac_init(&dac_hdl, &dac_data);

        s16 dacr32 = read_capless_DTB();

        audio_dac_set_capless_DTB(&dac_hdl, dacr32);

        audio_dac_set_buff(&dac_hdl, dac_buff, sizeof(dac_buff));

        struct audio_dac_trim dac_trim;
        int len = syscfg_read(CFG_DAC_TRIM_INFO, (void *)&dac_trim, sizeof(dac_trim));
        if (len != sizeof(dac_trim) || dac_trim.left == 0 || dac_trim.right == 0) {
            audio_dac_do_trim(&dac_hdl, &dac_trim, 0);
            syscfg_write(CFG_DAC_TRIM_INFO, (void *)&dac_trim, sizeof(dac_trim));
        } else {
            dac_trim_hook(5);
        }
        audio_dac_set_trim_value(&dac_hdl, &dac_trim);
        audio_dac_set_delay_time(&dac_hdl, 30, 50);
        audio_dac_set_analog_vol(&dac_hdl, 0);
        audio_dac_set_fade_handler(&dac_hdl, NULL, audio_fade_in_fade_out);

        _dac_analog_init(&dac_hdl, 1);
    #endif
}
/*****
```

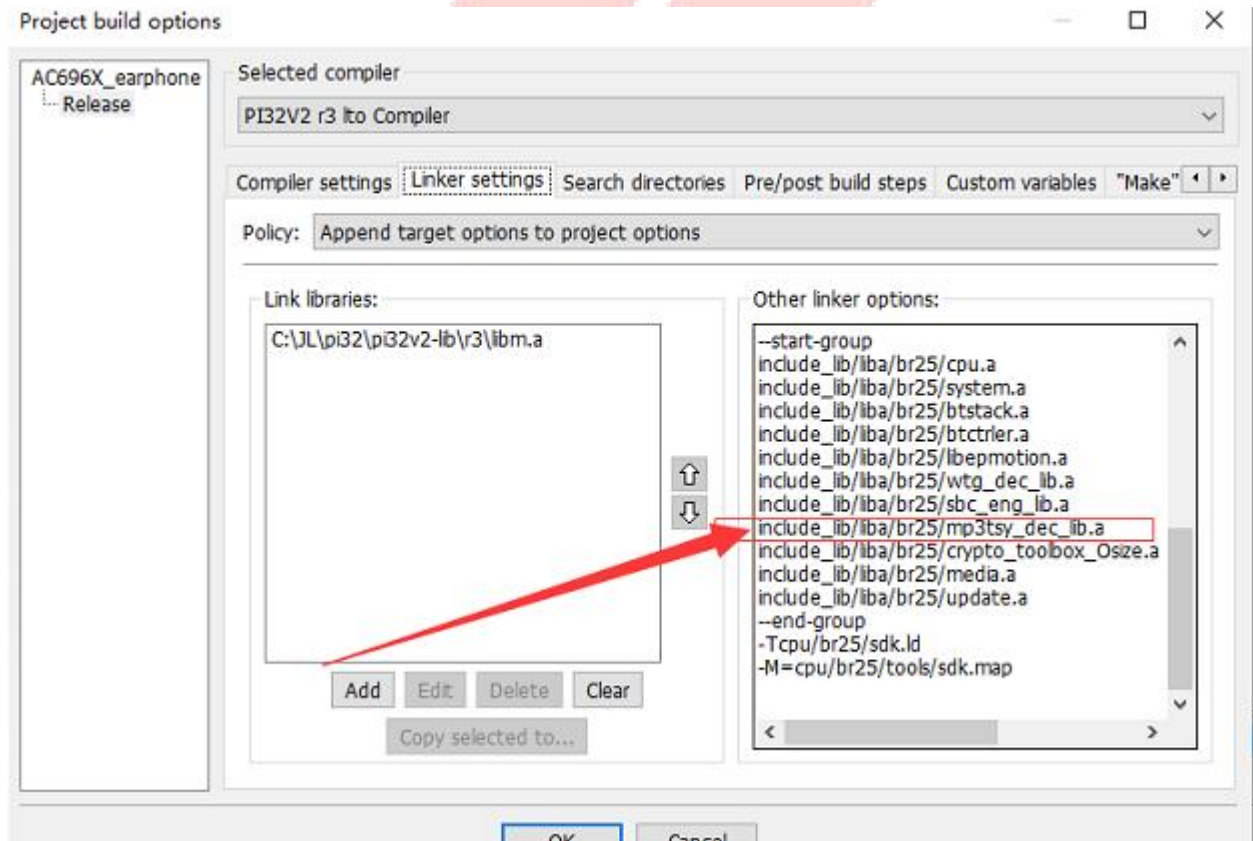
9.AC696 版本支持 MTY 格式提示音 20200618

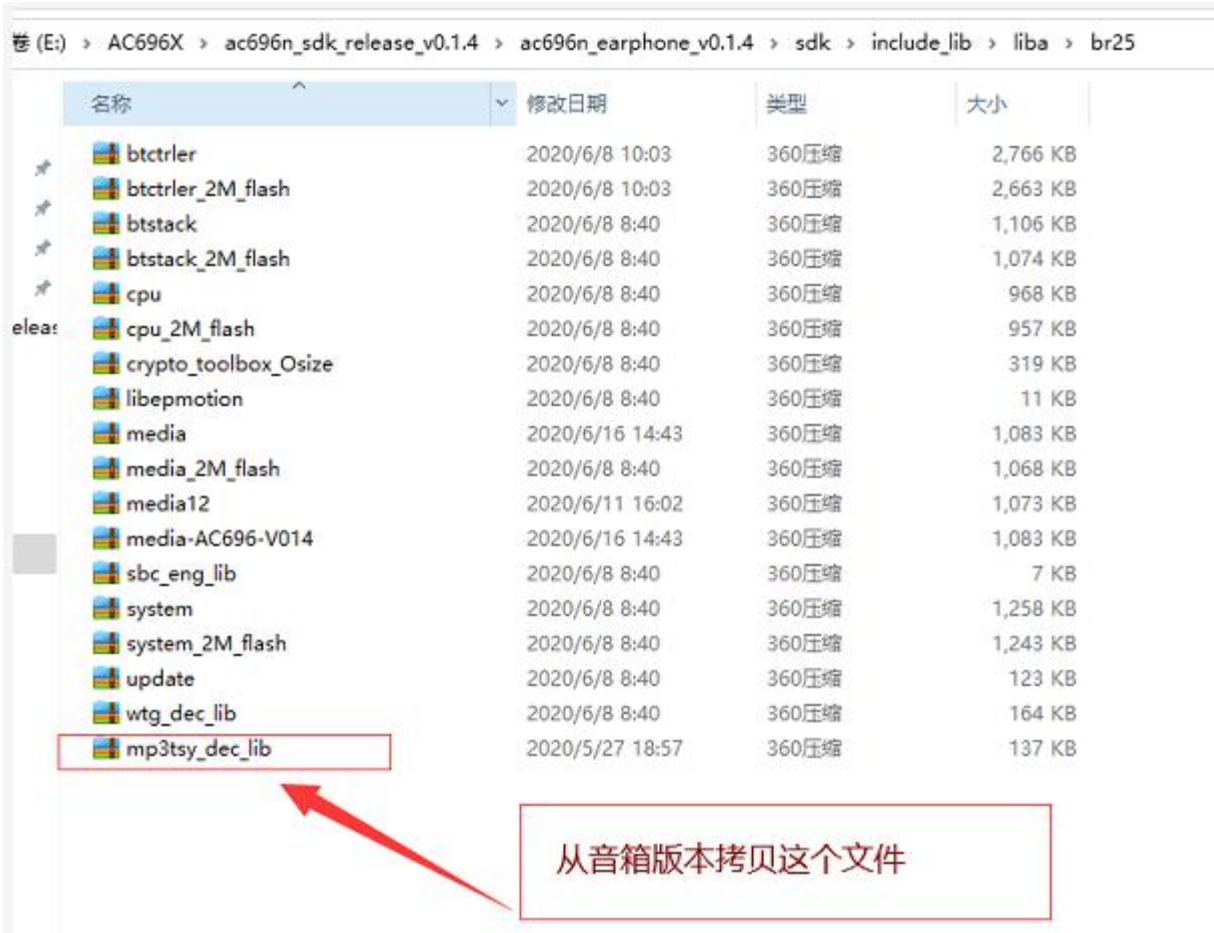
```

c6963a_tws_cfg.h Search Results | App_audio.c Board_config.h Bt_tws.c App_charge.h App_config.h Audio_plat
c696X_demo_cfg.h App_main.c 全部关闭
in.c
00094:     }
00095: } ? end check_power_on_key ?
00096:
00097:
00098: extern int cpu_reset_by_soft();
00099:
00100: static void audio_module_probe(void)
00101: {
00102:     /*解码器*/
00103:     AUDIO_DECODER_PROBE(sbc);
00104:     AUDIO_DECODER_PROBE(msbc);
00105:     AUDIO_DECODER_PROBE(cvsd);
00106: #ifndef CONFIG_LITE_DECODER
00107:     AUDIO_DECODER_PROBE(mty);
00108: #endif
00109: #if TCFG_BT_SUPPORT_AAC
00110:     AUDIO_DECODER_PROBE(aac);
00111: #endif
00112:
00113:     /*编码器*/
00114:     AUDIO_ENCODER_PROBE(msbc);
00115:     AUDIO_ENCODER_PROBE(cvsd);
00116: }
00117:
00118: void app_main()
00119: {

```

直接改为 #if 1





从音箱版本拷贝这个文件

(.media..text)代码段定义

```

cfg.h App_main.c Board_ac696x_demo.c Sdk_ld.c Tone_player.h 全部关闭
00422: //common bank code addr
00423: common_code_run_addr = .;
00424:
00425: .common ALIGN(4):
00426: {
00427:     * (.common*)
00428:     . = ALIGN(4);
00429:
00430:     media_code_begin = .;
00431:     /* *(.media.*.text)*/
00432:     media_code_end = .;
00433:     . = ALIGN(4);
00434:     media_code_size = media_code_end - media_code_begin;
00435:     . = ALIGN(4);
00436:
00437: #if 0
00438:     *(.classic_tws_const)
00439:     *(.classic_tws_code)

```

注释掉这里

```

fg.h App_audio.c Bt_tws.c App_charge.h App_config.h Audio_platform.h Audio_dec_server.h Se
fg.h App_main.c Board_ac696x_demo.c Sdk_ld.c Tone_player.h 全部关闭
00155:
00156:     * (.sbc_eng_code )
00157:     * (.sbc_eng_const)
00158:
00159:     . = ALIGN(4);
00160:     #include "media/media.ld"
00161:     * (.tech_lib.aec.text)
00162:     * (.media.*.text)
00163:     . = ALIGN(4);
00164:     #include "system/system.ld"
00165:
00166:     . = ALIGN(4);
00167:     #include "update/update.ld"
00168:     . = ALIGN(4);
00169:

```

这句话加在这

10.AC696 系列蓝牙耳机和音箱关于 PD4 IO 使用注意 20200619

PD3	PF2	ROM 中					spi0_CSA	SFC_CS
PD4 (超强驱动)		ROM 中	Flash 供电脚					
USBDP (下拉)		ROM 中				SDTAP_CLKB	ISP_CLK (mode_det0)	
UDBDM (下拉)		ROM 中		SDODAT_E		SDTAP_DATB	ISP_DI (mode_det1)	

PD4 是内部 flash 的一个供电脚，不能被操作到，操作到会影响 flash 的通讯。引起的问题：

1、写 VM 出现错误；2、flash 升级有问题；

2: 对应的板及文件: apps/earphone/board/br25/board_xxx.c

```

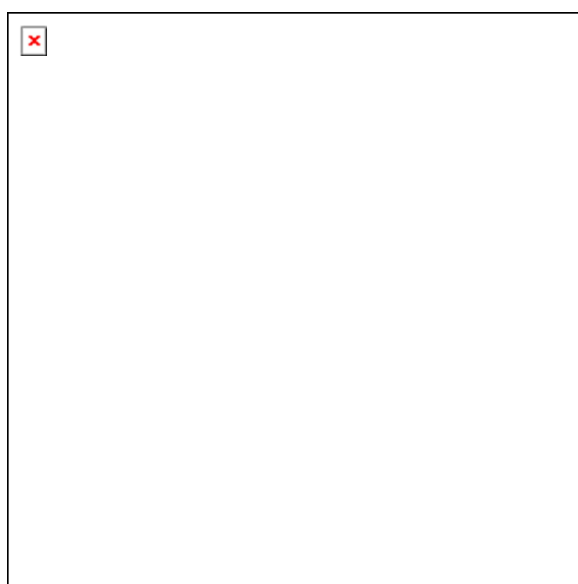
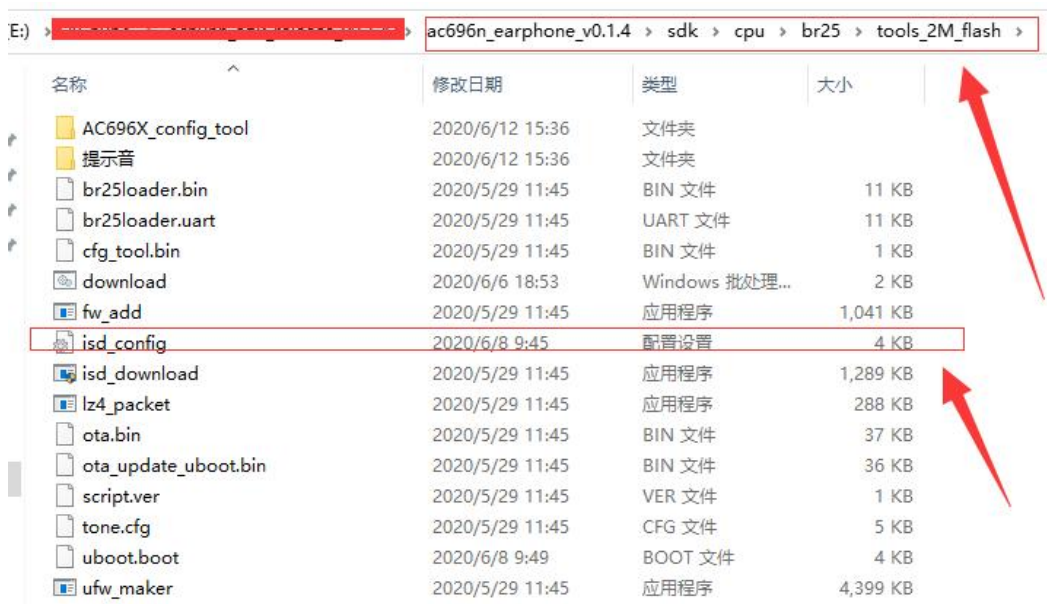
560
561 void board_power_init(void)
562 {
563     log_info("Power init : %s", __FILE__);
564
565     JL_PORTD->HD &=~BIT(4);
566     JL_PORTD->HD0 &=~BIT(4);
567
568     power_init(&power_param);
569

```

注释掉这两句话，没调用到就不用管它

11.AC696N 2M 程序烧写到 4M 容量芯片注意事项 20200619

若 2M 版本需要兼容烧录到 2M 和 4M 的 flash 芯片中，需要修改 isd_config.ini 文件：isd_config.ini 需要配置 BTIF 不小于 4Kbyte 与 VM 大小不小于 8K(如下图所示)。重新生成的烧写文件即可。



问：那么有疑问，改为支持 4Mbit 芯片后的代码，可以直接烧录进原来 2Mbit 的芯片？

答：要看整体空间够不够，因为了兼容 2Mbit 和 4Mbit 的芯片，VM 必须要 8KByte。2M 的代码编译出来烧录文件对于 2M 的芯片基本所剩无几，所以请分类好烧录文件 2M 和 4M，烧录 2M 的芯片就用 2M 的烧录文件。烧录 4M 芯片就用改 isd_config.ini 的烧录文件烧录。

12.动态开关长按复位 20200619

动态开关长按复位功能

///关闭长按复位

```
p33_and_1byte(P3_PINR_CON, 0);
```

///开启长按复位

```
p33_and_1byte(P3_PINR_CON, 1);
```

13.使用 AC696D 调式样机出现通话远端没有声音问题处理方法 20200619

```
532     gpio_set_pull_down(IO_PORT_DM, 0);  
533     gpio_set_direction(IO_PORT_DM, 1);  
534     gpio_set_die(IO_PORT_DM, 0);  
535 }  
536 }  
537  
538 void board_power_init(void)  
539 {  
540     log_info("Power init : %s", __FILE__);  
541     power_init(&power_param);  
542  
543     sdpd_config(1);  
544  
545     JL_PORTA->PU &= ~BIT(0);  
546     JL_PORTA->PD &= ~BIT(0);  
547     JL_PORTA->DIR |= BIT(0);  
548  
549     power_set_callback(TCFG_LOWPPOWER_LOWPPOWER_SEL, sleep_enter_callback, sleep_exit_call)  
550  
551     power_keep_dacvdd_en(0);  
552  
553     /* power_wakeup_init(&wk_param); */  
554  
555     #if (!TCFG_IOKEY_ENABLE && !TCFG_ADKEY_ENABLE)  
556     charge_check_and_set_pinr(1);  
557     #endif  
558 }  
559  
560 void board_power_wakeup_init(void)
```



14.AC696X 跑 LDO15 若出现部分机子通话时样机复位处理方法 20200619

备注：处理思路是提高内核电压，提高抗干扰能力

```

board_ac6969a_tws_cfg.h x setup.c x board_ac6969a_tws.c x ui_manage.c x user_cfg.h x key_event_c
253  /*
254  #define TCFG_CHARGE_MA          CHARGE_mA_60
255
256  //*****
257  //                      LED 配置
258  //*****
259  #define TCFG_PWMLED_ENABLE      ENABLE_THIS_MOUDLE      //是否支持
260  #define TCFG_PWMLED_IOMODE      LED_ONE_IO_MODE        //LED模式,
261  #define TCFG_PWMLED_PIN         IO_PORTB_03            //LED使用
262  //*****
263  //                      时钟配置
264  //*****
265  #define TCFG_CLOCK_SYS_SRC       SYS_CLOCK_INPUT_PLL_BT_OSC //系统时钟
266  #define TCFG_CLOCK_SYS_HZ        24000000              //系统时钟
267  #define TCFG_CLOCK_OSC_HZ        24000000              //外界晶振
268  #define TCFG_CLOCK_MODE          CLOCK_MODE_USR//CLOCK_MODE_ADAPTIVE
269
270  //*****
271  //                      低功耗配置
272  //*****
273  #define TCFG_LOWPOWER_POWER_SEL   PWR_LDO15            //电源模
274  #define TCFG_DCDC_PORT_SEL        NO_CONFIG_PORT      //DCDC
275  #define TCFG_LOWPOWER_BTOSC_DISABLE 0                //低功耗
276  #define TCFG_LOWPOWER_LOWPOWER_SEL SLEEP_EN          //SNIFF状
277  /*强VDDIO等级配置,可选:
278  VDDIOM_VOL_20V  VDDIOM_VOL_22V  VDDIOM_VOL_24V  VDDIOM_VOL_26V
279  VDDIOM_VOL_30V  VDDIOM_VOL_30V  VDDIOM_VOL_32V  VDDIOM_VOL_36V*/

```

```

board_ac6969a_tws_cfg.h x setup.c x board_ac6969a_tws.c x ui_manage.c x user_cfg.h x key_event_deal.c
172
173
174  void memory_init(void);
175  void setup_arch()
176  {
177      memory_init();
178
179      /* memset(stack_magic, 0x5a, sizeof(stack_magic)); */
180      /* memset(stack_magic0, 0x5a, sizeof(stack_magic0)); */
181
182      wdt_init(WDT_16S);
183      /* wdt_close(); */
184
185      u8 mode = TCFG_CLOCK_MODE;
186      if (TCFG_LOWPOWER_POWER_SEL == PWR_DCDC15) {
187          mode = CLOCK_MODE_USR;
188      }
189      mode = CLOCK_MODE_USR;
190
191      clk_voltage_init(mode, SYSVDD_VOL_SEL_120V, VDC13_VOL_SEL_135V);
192
193      // clk_voltage_init(mode, SYSVDD_VOL_SEL_120V, VDC13_VOL_SEL_110V);
194
195      clk_early_init(TCFG_CLOCK_SYS_SRC, TCFG_CLOCK_OSC_HZ, TCFG_CLOCK_SYS_HZ);
196
197      /*interrupt init()*/
198
199
200

```

修改这个几个地方

15.AC696 系列 2M 的 SDK 开蓝牙一拖二出现奇怪的问题 20200624

696 的 2M 程序版本开一拖二，有如下几个奇怪问题需要注意，

第一，通话杂音；

第二，播放提示音杂音；

如果有这些问题，请关闭一拖二功能，2M 的 SDK 蓝牙一拖二是不支持的。

同时 2M 的 SDK 也不支持测试盒的频偏测试和测试盒的 OTA 空中升级。

16.AC696 系列开启 Gsenor 编译不过 20200701

```

tion.h Search Results Board_ac696x_demo_cfg.h Da230.c 全部关闭
)0293:
)0294:
)0295: //*****
)0296: //                                g-sensor配置                                //
)0297: //*****
)0298: #define TCFG_GSENSOR_ENABLE                1 //gSensor使能
)0299: #define TCFG_DA230_EN                    1
)0300: #define TCFG_SC7A20_EN                    0
)0301: #define TCFG_STK8321_EN                  0  开启Gsensor
)0302: #define TCFG_GSENON_USER_IIC_TYPE        1 //0:软件IIC 1:硬件IIC
)0303:
)0304: //*****
)0305: //                                系统配置                                //
)0306: //*****
)0307: #define TCFG_IIC_USER_IIC_TYPE            0 //没有开启硬件IIC

```

编译后如下出错。

```

cpu\br25\iic_soft.c x
36     gpio_direction_output(sda, 1)
37
38     #define IIC_SDA_L(sda) \
39         gpio_direction_output(sda, 0)
40
41     #define IIC_SDA_READ(sda) \
42         gpio_read(sda)
43
44     #define iic_get_id(iic)    (iic)
45
46     static inline u32 iic_get_scl(soft_iic_dev iic)
47     {
48         u8 id = iic_get_id(iic);
49         return soft_iic_cfg[id].scl;
50     }
51
52     static inline u32 iic_get_sda(soft_iic_dev iic)
53     {
54         u8 id = iic_get_id(iic);
55         return soft_iic_cfg[id].sda;
56     }
57
58     static inline u32 iic_get_delay(soft_iic_dev iic)
59     {
60         u8 id = iic_get_id(iic);
61         return soft_iic_cfg[id].delay;
62     }
63

```

```

Logs & others
Code::Blocks Search results Build log Build messages Debugger
int _EXFUN(strcmp, (const char *, const char *));
E:\AC696X\ac696n_sdk_release_v0.1.4\ac696n_earphone_v0.1.4\sdk\apps\earphone\user_cfg.o:432:26: warning: passing 'u8 [32]' to parameter of type 'const char *' oor
types with different sign [-Wpointer-sign]
    if (strcmp(new_name, bt_cfg.edr_name)) {
C:\JL\pi32\pi32v2-include\string.h:28:47: note: passing argument to parameter here
int _EXFUN(strcmp, (const char *, const char *));
8 warnings generated.
cpu\br25\tools\sdk.elf.o: In function 'no symbol':
E:\AC696X\ac696n_sdk_release_v0.1.4\ac696n_earphone_v0.1.4\sdk\apps\common\gSensor\da230.c:(.text+0x498): undefined reference to `EFMotion_Set_Debug_level'
Process terminated with status 1 (0 minute(s), 8 second(s))
1 error(s), 54 warning(s) (0 minute(s), 8 second(s))

```

解决方法:

```

EPMotion.h Search Results Board_ac696x_demo_cfg.h Da230.c 全部关闭
00209: };
00210: u8 da230_init(void)
00211: {
00212:     u8 data = 0;
00213:
00214:     JL_PORTB->DIR &= ~BIT(2);
00215:     JL_PORTB->DIE |= BIT(2);
00216:     JL_PORTB->OUT |= BIT(2);
00217:     da230_register_read(NSA_REG_WHO_AM_I, &data);
00218:
00219:     if (data != 0x13) {
00220:         log_e("da230 init err1!!!!!!\n");
00221:         return -1;
00222:     }
00223:
00224:     if (EPMotion_Init(&ops_handle) ) {
00225:         log_e("da230 init err2!!!!!!\n");
00226:         return -1;
00227:     }
00228:
00229: // EPMotion_Set_Debug_level(DEBUG_ERR);
00230: EPMotion_Tap_Set_Parma(0x0d, LATCHED_100MS, 0x00); //0x05~0x1f
00231: //EPMotion_D_Tap_Set_Filter(1, 130);
00232: EPMotion_M_Tap_Set_Dur(70, 500);
00233: // EPMotion_M_Tap_Set_Dur(60,150);
00234: da230_register_mask_write(0x10, 0xe0, 0xc0);
00235: EPMotion_Control(M_TAP_T, ENABLE_T);
00236: // EPMotion_Tap_Set_Parma(0x08, LATCHED_25MS, 0x00); //0x05~0x1f
00237:

```

注释掉这里，这是DEBUG的

17.AC696 测试盒配对的 TWS 清配对注意点 20200701

手动清除 TWS 配对信息后，无法配对。

可能要注意测试盒配对优先的宏配置引起的问题。

```

t_tws.c App_audio.c Bt_profile_cfg.h App_config.c Setup.c App_main.c App_config.h Board_ac696x_demo.c Board
otion.h Search Results Board_ac696x_demo_cfg.h Da230.c 全部关闭
00139: #undef CONFIG_TWS_CHANNEL_SELECT
00140: #define CONFIG_TWS_CHANNEL_SELECT CONFIG_TWS_SECECT_BY_CHARGESTORE
00141: #endif //TCFG_CHARGESTORE_ENABLE
00142:
00143: #if TCFG_TEST_BOX_ENABLE && (!TCFG_CHARGESTORE_ENABLE)
00144: #if (CONFIG_TWS_CHANNEL_SELECT == CONFIG_TWS_EXTERN_DOWN_AS_LEFT) || \
00145:     (CONFIG_TWS_CHANNEL_SELECT == CONFIG_TWS_EXTERN_UP_AS_LEFT) || \
00146:     (CONFIG_TWS_CHANNEL_SELECT == CONFIG_TWS_AS_LEFT_CHANNEL) || \
00147:     (CONFIG_TWS_CHANNEL_SELECT == CONFIG_TWS_AS_RIGHT_CHANNEL)
00148: //如果硬件或软件固定左右耳, 则不采用测试盒指定左右耳信息
00149: #define CONFIG_TWS_SECECT_CHARGESTORE_PRIO 0
00150: #else
00151: //否则, 其他情况优先采用测试盒指定左右耳信息
00152: #define CONFIG_TWS_SECECT_CHARGESTORE_PRIO 1 //测试盒配置左右耳优先
00153: #endif
00154: #else
00155: #define CONFIG_TWS_SECECT_CHARGESTORE_PRIO 0
00156: #endif //TCFG_TEST_BOX_ENABLE
00157:
00158: //如果支持测试盒命令, 配置测试盒断开的行为
00159: #if TCFG_TEST_BOX_ENABLE
00160: #define ACTION_MODE_CPU_RESET 0 //复位
00161: #define ACTION_MODE_POWEROFF 1 //关机

```

18.AC696 系列耳机版本 V014-P1 优化 20200721

如果不改，进入软关机或者进入低功耗时候，vddio 电压会设置到最低档 2.1V 用 **SDK 默认**的电压值 **2.8V** 即可。

```
Power_interface.h Setup.c Board_ac6963a_tws.c Adc_api.c Search Results 全部关闭
00343:
00344: } ? end adc_scan ?
00345: static u16 vbg_value, vbg_value_w_32, vbg_value_w_28;
00346: static u8 vddiom_level, vddio_trim_ok;
00347: static u8 vddiow_level = TCFG_LOWPOWER_VDDIOW_LEVEL;
00348:
00349: u8 adc_get_trim_vddiow()
00350: {
00351:     return vddiow_level;
00352: }
00353: BANK_INIT
00354: static void udelay_osc(u32 usec)
00355: {
00356:     JL_TIMER0->CON = BIT(14) | BIT(3); //sel OSC
```

赋初始值，不然会导致刚开始开机运算按照默认 2.0V 计算。会导致第一次焊接电池外挂触摸 IC 工作失灵风险。

```
#include"board_config.h"
vddiow_level = TCFG_LOWPOWER_VDDIOW_LEVEL;
```

```
Adc_api.h Adsp.c Adc_api.c 全部关闭
00553:
00554: }
00555: } ? end adc_trim_vddio_record ?
00556: BANK_INIT
00557: void _adc_init(u32 sys_lvd_en)
00558: {
00559:     memset(adc_queue, 0xff, sizeof(adc_queue));
00560: |
00561:     JL_ADC->CON = 0;
00562:     JL_ADC->CON = 0;
00563:
00564:
00565:     adc_pmu_detect_en(1);
00566:
00567: #if (CONFIG_VDDIO_KEEP)
00568:
00569:     //get vddiom
00570:     JL_ADC->CON |= (0xf << 12); //启动延时控制，实际启动延时为此
00571:     JL_ADC->CON |= BIT(3);
00572:     JL_ADC->CON |= BIT(4);
00573:     JL_ADC->CON |= (0b1111) << 8;
00574:     JL_ADC->CON |= BIT(6);
00575:     hw_vddio_trim();
00576:     adc_trim_vddio_record();
00577: #else
00578:     #include"board_config.h"
00579:     vddiow_level = TCFG_LOWPOWER_VDDIOW_LEVEL;
00580: #endif
00581:     u32 vbat_queue_ch = adc_add_sample_ch(AD_CH_VBAT);
00582:     adc_set_sample_freq(AD_CH_VBAT, 30000);
```

19.AC696 系列音箱版本 V024 和耳机 SDK031 获取连接手机的蓝牙名和地址 20220707---WTS 更新

```
u8 btaddr[6];
put_buf(bt->args, 6);
memcpy(btaddr, bt->args, 6);
user_send_cmd_prepare(USER_CTRL_READ_REMOTE_NAME, 6, btaddr);
```

```
01243:
01244:     case BT_STATUS_ENCRY_COMPLETE:
01245:         break;
01246:
01247:     case BT_STATUS_SECOND_CONNECTED:
01248:         clear_current_poweron_memory_search_index(0);
01249:     case BT_STATUS_FIRST_CONNECTED:
01250:         log_info("BT STATUS CONNECTED\n");
01251:         u8 btaddr[6];
01252:         put_buf(bt->args, 6);
01253:         memcpy(btaddr, bt->args, 6);
01254:         user_send_cmd_prepare(USER_CTRL_READ_REMOTE_NAME, 6, btaddr);
01255:
01256:         sys_auto_shut_down_disable();
01257:     #if (defined(SMART_BOX_EN) && SMART_BOX_EN)
01258:         memcpy(__this->last_connecting_addr, bt->args, 6);
01259:         extern void smartbox_update_bt_emitter_connect_state(u8 state, u8 *addr);
01260:         smartbox_update_bt_emitter_connect_state(1, __this->last_connecting_addr);
01261:     #endif
01262:
01263:
01264:     #if (TCFG_BLE_DEMO_SELECT == DEF_BLE_DEMO_ADV)
```

```
00530:     music_vol_change_handle_register(bt_set_music_device_volume, phone_get_device_vol)
00531: #endif
00532: #if BT_SUPPORT_DISPLAY_BAT
00533:     get_battery_value_register(bt_get_battery_value); /*电量显示获取电量的接口*/
00534: #endif
00535:
00536:     bt_dut_test_handle_register(bt_dut_api);
00537:
00538:     read_remote_name_handle_register(bt_read_remote_name);
00539:
00540: #if TCFG_USER_EMITTER_ENABLE
```

```
static void bt_read_remote_name(u8 status, u8 *addr, u8 *name)
{
    if (status) {
        printf("remote_name fail \n");
    } else {
        printf("remote_name : %s \n", name);
    }
    put_buf(addr, 6);
    #if TCFG_USER_EMITTER_ENABLE
    emitter_search_noname(status, addr, name);
    #endif
}
```

替换 AC696 系列 V024 版本的 btctrlr.a 库，记得 rebuild.

链接: <https://pan.baidu.com/s/1giA-f-HbZ9ubVBCmsw72bg>

提取码: i2in

耳机 SDK031 需要替换的 btctrler.a 库，记得 rebuild:

【金山文档】 耳机 SDK031 获取蓝牙名获取功能
<https://kdocs.cn/l/cuExtklcvw5r>

20.关于弱 VDDIO 档位为 2.4V 给触摸 IC 的供电引起开关机漏电问题，重要！！ 20200721

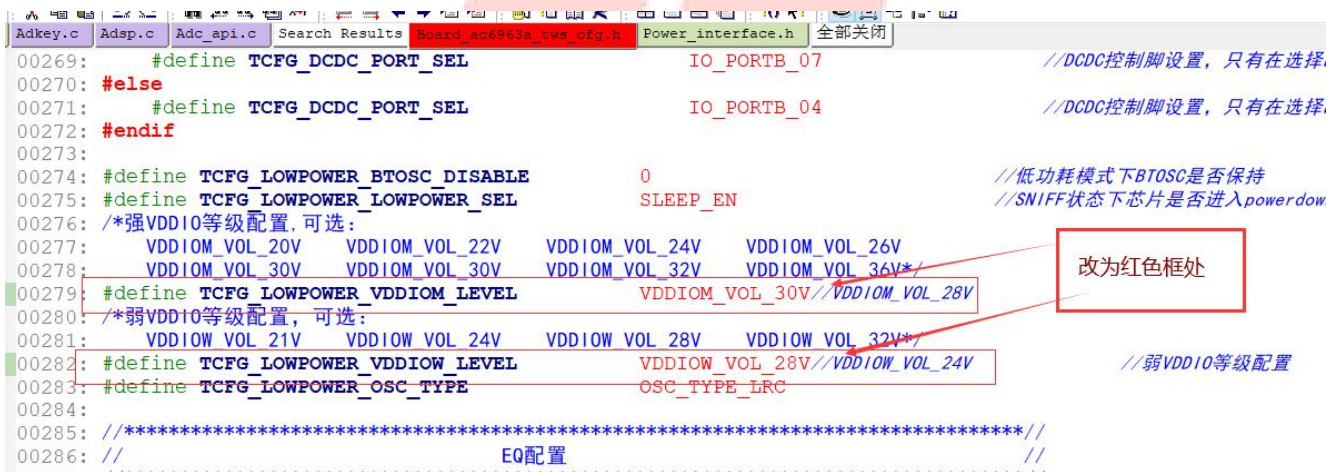
本文档第 18 点要修改，同时本点第 20 点也要改。

引起问题：如果弱 VDDIO 工作电压档位为 2.4V，从强档位到弱档位，VDDIO 会容易被拉扯或者瞬态拉扯到 2.0V，例如对于触摸 IC 工作在 2.4V 以上范围的，导致触摸 IC 工作错乱，误判，从而引起我们主控会被误操作，例如唤醒，然后进入重复开关机的情况，表现出来就是漏电严重。

解决方法：AC696 系列发出的所有 SDK 如下宏

```
#define TCFG_LOWPOWER_VDDIOM_LEVEL          VDDIOM_VOL_28V
#define TCFG_LOWPOWER_VDDIOW_LEVEL         VDDIOW_VOL_24V
全部改为
#define TCFG_LOWPOWER_VDDIOM_LEVEL        VDDIOM_VOL_30V
#define TCFG_LOWPOWER_VDDIOW_LEVEL        VDDIOW_VOL_28V
```

如类似下图



```
00269: #define TCFG_DCDC_PORT_SEL          IO_PORTB_07          //DCDC控制脚设置，只有在选择
00270: #else
00271: #define TCFG_DCDC_PORT_SEL          IO_PORTB_04          //DCDC控制脚设置，只有在选择
00272: #endif
00273:
00274: #define TCFG_LOWPOWER_BTOSC_DISABLE    0                //低功耗模式下BTOSC是否保持
00275: #define TCFG_LOWPOWER_LOWPOWER_SEL    SLEEP_EN          //SNIFF状态下芯片是否进入powerdown
00276: /*强VDDIO等级配置，可选：
00277: VDDIOM_VOL_20V VDDIOM_VOL_22V VDDIOM_VOL_24V VDDIOM_VOL_26V
00278: VDDIOM_VOL_30V VDDIOM_VOL_30V VDDIOM_VOL_32V VDDIOM_VOL_36V*/
00279: #define TCFG_LOWPOWER_VDDIOM_LEVEL    VDDIOM_VOL_30V//VDDIOM_VOL_28V
00280: /*弱VDDIO等级配置，可选：
00281: VDDIOW_VOL_21V VDDIOW_VOL_24V VDDIOW_VOL_28V VDDIOW_VOL_32V*/
00282: #define TCFG_LOWPOWER_VDDIOW_LEVEL    VDDIOW_VOL_28V//VDDIOW_VOL_24V //弱VDDIO等级配置
00283: #define TCFG_LOWPOWER_OSC_TYPE        OSC_TYPE_LRC
00284:
00285: //*****EQ配置*****
00286: //
```

重点：变量 vddiow_level 初始化为 static u8 vddiow_level = TCFG_LOWPOWER_VDDIOW_LEVEL; 本文档第 18 点。主要解决焊接电池第一次上电 VDDIO 弱档位电压位 2.0V 的问题。

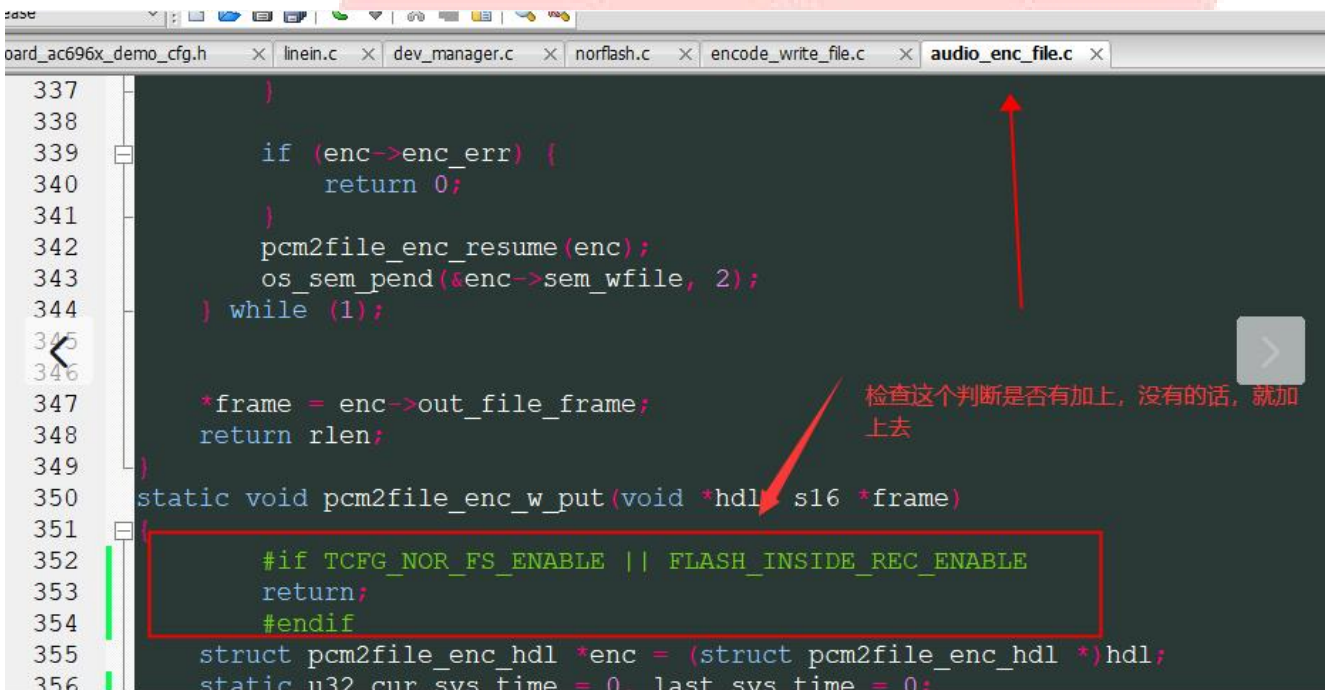
原因：我们主控 AC696 有强 VDDIO 电压档和弱 VDDIO 电压档位之分，弱 VDDIO 电压档位的驱动电流带载能力比较弱，容易被拉扯，例如用 VDDIO 或者 IO 口给触摸 IC 供电，如果弱 VDDIO 工作电压档位为 2.4V，从强档位到弱档位，VDDIO 会被拉扯，导致触摸 IC 工作错乱，误判，从而引起我们主控会被唤醒，然后进入重复开关机的情况，表现出来就是漏电严重。

其他疑惑：所生产的机器是按键开机的，弱电压是 2.4V 有没有问题。

答：没问题！继续生产！！

如果没有用 VDDIO 或者 IO 口作为外部器件（例如触摸或者 Gsensor 等）的供电，也就是说跟 VDDIO 不关联的是没问题的。但建议还是及时修改，防止后续用此程序用在其他项目带触摸等的漏改了引起问题！！

21.AC696X SDK0.2.3-p1 录音录到外部 flash 后，播放不了问题 20200711



```
337     )
338
339     if (enc->enc_err) {
340         return 0;
341     }
342     pcm2file_enc_resume(enc);
343     os_sem_pend(&enc->sem_wfile, 2);
344     } while (1);
345
346
347     *frame = enc->out_file_frame;
348     return rlen;
349 }
350 static void pcm2file_enc_w_put(void *hdl, s16 *frame)
351 {
352     #if TCFG_NOR_FS_ENABLE || FLASH_INSIDE_REC_ENABLE
353         return;
354     #endif
355     struct pcm2file_enc_hdl *enc = (struct pcm2file_enc_hdl *)hdl;
356     static u32 cur_sys_time = 0, last_sys_time = 0;
```

检查这个判断是否有加上，没有的话，就加上去

22.AC696X SDK014-p1 对耳连苹果手机开 SIRI 有时蓝牙会断连问题处理 20200711

处理该问题，需要更新两个库：[btctrlr.a](#)、[btctrlr_2M_flash.a](#)（换库后需要 rebuild）

备注：如果测试到通话时，有时蓝牙会断开的问题，也可以使用这两个库替换测试，可能也是同一个问题。

库存于百度网盘中：“AC696X SDK014-p1 对耳连苹果手机开 SIRI 有时蓝牙会断连，需要更新的库.zip”，提取信息如下：

链接：<https://pan.baidu.com/s/1U5ZyiOtxGEefZBof5pg-Kg>

提取码：mq75

23.AC696X 外挂 FM 时，FM 的 IIC 口和卡口的 CMD 脚及 DATA 脚复用问题 20200713

外挂 FM 时，如果 FM 的 IIC 通信脚需要跟卡口复用，那么卡的检测要选择用 CLK 检测方式。

因为如果选 CMD 检测，会导致两个问题：1、CMD、CLK 脚一直有信号（发送检测数据），这样会对 FM 产生一些干扰信号。2、CMD、CLK 脚一直要通信发数据，分时复用就会很麻烦，不好处理。

选用 CLK 检测的好处：1、进入 FM 时，卡口没有信通，没有信号干扰。程序仅检测 CLK 脚高低电平变化。2、卡的 CMD 脚和 DATA 脚 没有通信，因此只要释放卡口控制。就不会影响 IIC 通信。而且 IIC 通信时，不需要把卡检测停掉。

软件修改（IIC 通信时把卡口控制挂起（相当于设置卡口高阻态），通信完后恢复）：

```
Q8035.c x iic_soft.c x timer.h x file_api.c x board_ac696x_demo.c x app_common_key.c x iic
129 extern s32 sd_io_suspend(u8 sd_io);
130 extern s32 sd_io_resume(u8 sd_io);
131
132
133
134 void soft_iic_start(soft_iic_dev iic) {
135     u32 scl, sda, dly_t;
136
137     sd_io_suspend(0);
138     scl = iic_get_scl(iic);
139     sda = iic_get_sda(iic);
140     dly_t = iic_get_delay(iic);
141
142     IIC_SDA_H(sda);
143     delay(dly_t);
144
145     IIC_SCL_H(scl);
146     delay(dly_t * 2);
147
148     IIC_SDA_L(sda);
149     delay(dly_t);
150
151     IIC_SCL_L(scl);
152     delay(dly_t);
153
154 }
155
156 void soft_iic_stop(soft_iic_dev iic) {
157     u32 scl, sda, dly_t;
158
159     scl = iic_get_scl(iic);
160     sda = iic_get_sda(iic);
161     dly_t = iic_get_delay(iic);
162
163     IIC_SDA_L(sda);
164     delay(dly_t);
165
166     IIC_SCL_H(scl);
167     delay(dly_t * 2);
168
169     IIC_SDA_H(sda);
170     delay(dly_t);
171
172     sd_io_resume(0);
173
174 }
175
176
177 static u8 soft_iic_check_ack(soft_iic_dev iic) {
```

24.AC696X 耳机和音箱 SDK 的 pwm 功能 20200714

对于使用 AC696X 需要 PWM 功能的，可以见如下参考例子，详情请见网盘链接：

链接：<https://pan.baidu.com/s/166ibbi08er-tHo5LVHr11Q>

提取码：s2qq

特别需要注意的是，需要先看里面的 readme.txt 文档说明，另外一个音箱可以支持三路 PWM，耳机的 SDK 有且仅有一路 PWM。

名称	修改日期	类型	大小
AC696X耳机SDK_PWM功能.zip	2020/7/2 18:57	360压缩 ZIP 文件	50,165 KB
AC696X音箱SDK_PWM功能.rar	2020/6/24 17:01	360压缩 RAR 文件	37,506 KB

25.AC696X 在蓝牙播放歌曲的时候播发 msbc 提示音注意点 20200714

696x 系列在蓝牙播歌的时候播发 msbc 提示音的时候会播放不了或者出现死机问题，因为程序默认设置是叠加方式的，并且 **msbc 和蓝牙 sbc 是同一个解码器，不能同时使用，要做互斥使用**，所以在蓝牙播歌的时候播放 msbc 的提示音需要设置为打断方式即可。具体参考如下设置：将播放提示音的接口的第二个参数传递 1 表示打断方式，0 表示叠加方式。

```

1881     extern void sys_enter_soft_poweroff(void *priv);
1882     sys_enter_soft_poweroff(NULL);
1883     } else if (reason == SYNC_CMD_MAX_VOL) {
1884 #if TCFG_MAX_VOL_PROMPT
1885     STATUS *p_tone = get_tone_config();
1886     if (p_tone->max_vol != IDEX_TONE_NONE) {
1887         tone_play_index(p_tone->max_vol, 1);
1888     } else {
1889         /* tone_sin_play(150, 1); */
1890     }
1891 #endif
1892 #if (defined(TCFG_TONE2TWS_ENABLE) && (TCFG_TONE2TWS_ENABLE))

```

接口的第二参数改成 1，表示打断方式

26.AC696X 混响和回声，有变量选择 20200716

```

SDK > cpu > br25 > audio_effect > C audio_reverb.c > open_reverb(REVERB_PARM_SET *, u16)
52     if (reverb_setting) {
53         memcpy(&reverb_api_obj->parm, reverb_setting, sizeof(REVERB_PARM_SET));
54     } else {
55         reverb_api_obj->parm.decayval = 2000; //衰减系数 [0:4096]
56         reverb_api_obj->parm.deepval = 0;//4096; //调节混响深度，影响pre-delay:[0-4096],4096 代表
57         reverb_api_obj->parm.filtsize = 0;//4096; //[0-4096],如果要回声效果 置0;混响效果建议4096
58         reverb_api_obj->parm.wetgain = 4096; //湿声增益: [0:4096]
59         reverb_api_obj->parm.drygain = 4096; //干声增益: [0:4096]
60         reverb_api_obj->parm.sr = 16000; //配置输入的采样率，影响need_buf 大小

```

27. AC696X EQ 在线调式注意点 20200716 CCB

开了 eq 在线调试, eq_cfg_hw.bin 默认不起作用, 所以如果需要将 eq_cfg_hw.bin 加到程序中试听, 就必须在软件中关闭 eq 在线调试。

28. AC696X 音箱版本供电低于 3.5V DAC 出现底噪 20200718 CCB

主控供电电压: 一般要保证 VBAT>VDDIO>DACVDD, 且保证压差要 0.2V 或以上。音箱版本默认 VDDIO 是 3.4V, 当供电低于 3.5V 时候, 明显观察到 VDDIO 有纹波, 从而出现底噪。软件把 VDDIO 从 3.4V 降到 3.2V, DACVDD 维持 3.0V 不变。把 VDDIO 降低到 3.2V 后, 如果 VDDIO 还给 SD 卡供电, 那么需要串 4.7R+106 电容, 保证 SD 供电正常。

注意: 因为 VDDIO 要给 TF 卡供电, 所有 VDDIO 不能降得太低, 不能降到 3.0V, 这样 TF 卡的供电会有问题, 有可能会复位重启或者不读卡。

29.AC696 系列用于音箱应用注意点 20200721 YWP

AC696 只有一路 LADC, 同时 FM 不支持蓝牙后台。

对于一路 LADC 跟之前的 AC692 系列又有区别, LINEIN 进来的立体录音在 AC692 录音出来是 AMUXL/R 的混合, 但在 AC696 录音出来只有一路 AMUXL 或者 AMUXR。

30.AC696X 软件编码器实现方法 20200724 LZK

(PS:可直接复制)

```
#include "app_action.h"
/**
 * 使用方法
 * @param:(必填)
 * sin_port0 : 编码器的 io
 * sin_port1 : 编码器 io;
 * msg : 对应消息
 * @description:
 * 1, 填写 user_rdeckey_list 结构体的值
 * 2, 在系统初始化后, 调用 void user_rdec_device_init(void)
 * 3, 若有编码器按钮触发, 会通过 app_task_msg_post 函数将 msg 发送出去
 */
struct rdec_device_soft {
    unsigned char sin_port0;    //采样信号端口 0
    unsigned char sin_port1;    //采样信号端口 1
    int msg;                    //发送的消息
    //不用初始化以下变量
    unsigned char io[2];       //定义了两个变量用来储蓄上一次调用此方法是编码开关两引脚的电平
    unsigned char st[2];       //定义了一个变量用来储蓄以前是否出现了两个引脚都为高电平的状态
};
```

```
//定义变量
unsigned char rdec_device_num = 0; //使能的编码器数量
struct rdec_device_soft user_rdeckey_list[] = {
    {
        .sin_port0 = IO_PORTA_03,
        .sin_port1 = IO_PORTA_04,
        .msg = USER_MSG_TEST,
    },
};
void rdec_device_scan(void)
{
    unsigned char tp0,tp1;
    unsigned char index = 0;
    struct rdec_device_soft* rdec;
    for(index = 0; index < rdec_device_num; index++) {
        rdec = &user_rdeckey_list[index];
        tp0 = gpio_read(rdec->sin_port0);
        tp1 = gpio_read(rdec->sin_port1);
        if((tp0 && tp1) || ((!tp0) && (!tp1))) {
            if(rdec->io[0]) {
                printf("1\n");
                app_task_msg_post(rdec->msg,1,0);
            } else if(rdec->io[1]) {
                printf("2\n");
                app_task_msg_post(rdec->msg,1,1);
            }
            rdec->st[0] = tp0;
            rdec->st[1] = tp1;
            rdec->io[0] = 0;
            rdec->io[1] = 0;
        } else if(rdec->io[0] || rdec->io[1]) {
            return;
        } else if(tp0 != rdec->st[0]) {
            rdec->io[0] = 1;
        } else if(tp1 != rdec->st[1]) {
            rdec->io[1] = 1;
        }
    }
}
void user_rdec_device_init(void)
{
    unsigned char index = 0;
    rdec_device_num = sizeof(user_rdeckey_list)/sizeof(user_rdeckey_list[0]);
    unsigned char enable = 0;
```

```

struct rdec_device_soft* rdec;
for(index = 0; index < rdec_device_num; index++) {
    rdec = &user_rdeckey_list[index];
    gpio_direction_input(rdec->sin_port0);
    gpio_set_pull_down(rdec->sin_port0, 0);
    gpio_set_pull_up(rdec->sin_port0, 1);
    gpio_set_die(rdec->sin_port0, 1);
    gpio_direction_input(rdec->sin_port1);
    gpio_set_pull_down(rdec->sin_port1, 0);
    gpio_set_pull_up(rdec->sin_port1, 1);
    gpio_set_die(rdec->sin_port1, 1);
    enable = 1;
}
if(enable) {
    sys_hi_timer_add(NULL, rdec_device_scan, 2);
}
printf("rdec device init :%d\n",rdec_device_num);
}
    
```

31.linein 复用 CMD 检测配置例子 20200724 LZK

```

//***** linein配置 *****//
//***** linein配置 *****//
#define TCFG_LINEIN_ENABLE 1//ENABLE_THIS_MOUDLE // linein使能
// #define TCFG_LINEIN_LADC_IDX 0 // linein使用的ladc通道, 对应ladc_list
#define TCFG_LINEIN_LR_CH AUDIO_LR01_CH
#define TCFG_LINEIN_CHECK_PORT IO_PORTC_04 // linein检测IO
#define TCFG_LINEIN_POR1_UP_ENABLE 1 // 检测IO上拉使能
#define TCFG_LINEIN_POR1_DOWN_ENABLE 0 // 检测IO下拉使能
#define TCFG_LINEIN_AD_CHANNEL AD_CH_PC4 // 检测IO是否使用AD检测
#define TCFG_LINEIN_VOLTAGE 940 //(根据实际设置) // AD检测时的阈值
#ifndef TCFG_REVERB_ENABLE
#define TCFG_LINEIN_INPUT_WAY LINEIN_INPUT_WAY_ANALOG
#else
#define TCFG_LINEIN_INPUT_WAY LINEIN_INPUT_WAY_ADC//LINEIN_INPUT_WAY_ANALOG//LINEIN_INPUT_WAY_ANALOG
#endif
#define TCFG_LINEIN_MULTIPLEX_WITH_FM DISABLE // linein 脚与 FM 脚复用
#define TCFG_LINEIN_MULTIPLEX_WITH_SD 1 // linein 检测与 SD cmd 复用
//***** linein配置 *****//
    
```

32.GPIO 相关函数参数传参不正常会导致 USB 功能不正常 20200724 LZK

以下 GPIO 操作的函数传参不能传 NO_CONFIG_PORT,目前函数里面没有做限制传参超范围会设置到 USB IO 的寄存器。

```

gpio_set_pull_up(IO_PORTC_03, 0);
gpio_set_pull_down(IO_PORTC_03, 0);
gpio_set_direction(IO_PORTC_03, 1);
gpio_set_die(IO_PORTC_03, 0);
gpio_set_dieh(IO_PORTC_03, 0);
    
```

要注意 board 配置中的 NO CONFIG PORT 在实际使用中, 有没有被调用。

33.Linein 单声道模拟输入变双声道输出函数,(DAC 声道跟 linein 声道不一致也适用) 20200724 LZK

```
/*  
 *linein 单声道模拟输入的时候, 如果 dac 是两个声道的, 想要  
 *两个声道都有声音出来, 需要将 amux 通道合并一下, 即:  
 *call:audio_linein_ch_combine(1,1);  
 */  
void audio_linein_ch_combine(u8 LR_2_L, u8 LR_2_R);
```

34.SD_PG/PA0 脚无法正常控制 20200724 LZK

1, 如果使能 SD 卡功能, 检查 SD 的 power 变量是否配置了电源控制函数, 需要设置成 NULL

```
/****** SD config *****/  
#if TCFG_SD0_ENABLE  
SD0_PLATFORM_DATA_BEGIN(sd0_data)  
    .port = TCFG_SD0_PORTS, //sd0 support port 'A' and port 'B'  
    .data_width = TCFG_SD0_DAT_MODE,  
    .speed = TCFG_SD0_CLK,  
    .detect_mode = TCFG_SD0_DET_MODE,  
    .priority = 3,  
#if (TCFG_SD0_DET_MODE == SD_IO_DECT)  
    .detect_io = TCFG_SD0_DET_IO, //当检测模式为io口检测是有效  
    .detect_io_level = TCFG_SD0_DET_IO_LEVEL, //0: 低电平检测到卡。 1: 高电平检测到卡  
    .detect_func = sdmmc_0_io_detect, //库函数, 需要detect_io和detect_io_level两个元素  
    .power = sd_set_power, //库函数, 使用SDPG(PB8)引脚。可以自行重写其他的SD卡电源  
    /* .power = NULL, */  
#elif (TCFG_SD0_DET_MODE == SD_CLK_DECT)  
    .detect_io_level = TCFG_SD0_DET_IO_LEVEL, //0: 低电平检测到卡。 1: 高电平检测到卡  
    .detect_func = sdmmc_0_clk_detect, //库函数, 需要detect_io_level元素作为参数。可以  
    /* .power = sd_set_power, //库函数, 使用SDPG(PB8)引脚。可以自行重写其他的SD卡电  
    .power = NULL,  
#else  
    .detect_func = sdmmc_cmd_detect,  
    .power = NULL, //cmd检测需要全程供电, 建议用硬件固定电源。当然, 可以自行写其他的SD  
    /* .power = sd_set_power, */  
#endif  
SD0_PLATFORM_DATA_END()  
#endif
```

35.AC696 添加设备(优先)独立模式功能 20200724 LZK

修改文件简单说明

1.蓝牙字体为需要修改的文件

```
--- a/SDK/apps/soundbox/include/app_config.h  
+++ b/SDK/apps/soundbox/include/app_config.h
```

2.绿色字体为新增加内容

```
+  
+//分开U 盘跟 TF 模式  
+#define MUSIC_USB_TF_INDEPENDENT_ENABLE          ENABLE  
+
```

3.红色字体为删除内容

4.修改位置

```
@@ -206,6 +208,28 @@ static int _key_event_opr(struct sys_event *event)
```

代表修改内容在文件 -206 行附近，在 `_key_event_opr` 函数里面

修改示范：

示范 1：

```
diff --git a/SDK/apps/soundbox/include/app_config.h b/SDK/apps/soundbox/include/app_config.h  
index 1b8b4af..65db090 100644  
--- a/SDK/apps/soundbox/include/app_config.h  
+++ b/SDK/apps/soundbox/include/app_config.h  
@@ -276,4 +276,9 @@  
 #define TCFG_LOWPOWER_LOWPOWER_SEL          0          //开红外不进入低功耗  
 #endif /* #if TCFG_IRKEY_ENABLE */  
  
+  
+//分开U 盘跟 TF 模式  
+#define MUSIC_USB_TF_INDEPENDENT_ENABLE          ENABLE  
+#define MUSIC_PRIORITY_DEV          "null"          //进 music 模式优先设备 "null" "sd0" "sd1" "udisk"  
+  
 #endif
```

修改如下：

```
273 //*****  
274 #if TCFG_IRKEY_ENABLE  
275 #undef TCFG_LOWPOWER_LOWPOWER_SEL  
276 #define TCFG_LOWPOWER_LOWPOWER_SEL          0          //开红外不进入低功耗  
277 #endif /* #if TCFG_IRKEY_ENABLE */  
278  
+ 279  
+ 280 //分开U盘跟TF模式  
+ 281 #define MUSIC_USB_TF_INDEPENDENT_ENABLE          ENABLE  
+ 282 #define MUSIC_PRIORITY_DEV          "null"          //进music模式优先设备 "null" "sd0" "sd1" "udisk"  
+ 283  
284 #endif  
285
```

添加设备(优先)独立模式功能功能具体修改如下:

```
SDK/apps/soundbox/include/app_config.h | 5 ++++
SDK/apps/soundbox/include/tone_player.h | 5 +++-
SDK/apps/soundbox/music/music.c | 38 ++++++
SDK/apps/soundbox/music/music_player_api.c | 40 ++++++
SDK/cpu/br25/audio_dec/tone_player.c | 2 ++
5 files changed, 89 insertions(+), 1 deletion(-)

diff --git a/SDK/apps/soundbox/include/app_config.h b/SDK/apps/soundbox/include/app_config.h
index 1b8b4af..65db090 100644
--- a/SDK/apps/soundbox/include/app_config.h
+++ b/SDK/apps/soundbox/include/app_config.h
@@ -276,4 +276,9 @@
#define TCFG_LOWPOWER_LOWPOWER_SEL 0 //开红外不进入低功耗
#endif /* #if TCFG_IRKEY_ENABLE */

+
+//分开U盘跟TF模式
+#define MUSIC_USB_TF_INDEPENDENT_ENABLE ENABLE
+#define MUSIC_PRIORITY_DEV "null" //进music模式优先设备 "null" "sd0" "sd1" "udisk"
+
+
#endif

diff --git a/SDK/apps/soundbox/include/tone_player.h b/SDK/apps/soundbox/include/tone_player.h
index cf3ed0a..96b858a 100644
--- a/SDK/apps/soundbox/include/tone_player.h
+++ b/SDK/apps/soundbox/include/tone_player.h
@@ -61,7 +61,8 @@ enum {
#if (defined(TCFG_APP_RECORD_EN) && (TCFG_APP_RECORD_EN))
INDEX_TONE_RECORD,
#endif
-
+ INDEX_TONE_USB,
+ INDEX_TONE_TF,
INDEX_TONE_NONE = 0xFF,
};

@@ -92,6 +93,8 @@ enum {
#define TONE_RTC SDFILE_RES_ROOT_PATH"tone/rtc.*"
#define TONE_RECORD SDFILE_RES_ROOT_PATH"tone/record.*"
#define TONE_SPDIF SDFILE_RES_ROOT_PATH"tone/spdif.*"
```

```
+#define TONE_USB                SDFILE_RES_ROOT_PATH"tone/usb.*"
+#define TONE_TF                  SDFILE_RES_ROOT_PATH"tone/tf.*"

#ifdef CONFIG_CPU_BR18
#undef TONE_POWER_ON
diff --git a/SDK/apps/soundbox/music/music.c b/SDK/apps/soundbox/music/music.c
index 6362490..340a40c 100644
--- a/SDK/apps/soundbox/music/music.c
+++ b/SDK/apps/soundbox/music/music.c
@@ -40,6 +40,8 @@
#define LOG_CLI_ENABLE
#include "debug.h"

+char music_dev_change_flag = 0; //设备改变标志
+char is_need_play_dev_tone = 0; //设备提示音播放

#ifdef TCFG_SPEED_PITCH_ENABLE
static PITCH_SHIFT_PARM init_parm;
@@ -206,6 +208,28 @@ static int _key_event_opr(struct sys_event *event)
    }
    break;
#endif
+#if MUSIC_USB_TF_INDEPENDENT_ENABLE
+    case KEY_CHANGE_MODE:
+        //拦截切模式消息进行处理
+        printf(">>>>>>> music mode change mode\n");
+        is_need_play_dev_tone = 1;
+        if(file_opr_available_dev_total(>1) {
+            if(MUSIC_PRIORITY_DEV != "null") {
+                if(MUSIC_PRIORITY_DEV == music_play_get_cur_dev()) {
+                    music_play_change_dev_next();
+                    break;
+                }
+            } else {
+                if(music_dev_change_flag == 0) {
+                    music_dev_change_flag = 1;
+                    music_play_change_dev_next();
+                    break;
+                }
+            }
+        }
+        ret = false;
+        break;
+#endif
```

```
#if (defined(TCFG_LOUDSPEAKER_ENABLE) && (TCFG_LOUDSPEAKER_ENABLE))
    case KEY_SPEAKER_OPEN:
        if(speaker_if_working()) {
@@ -251,6 +275,8 @@ static int music_event_handler(struct application *app, struct sys_ev
ent *event)
        case DRIVER_EVENT_FROM_SD1:
        case DRIVER_EVENT_FROM_SD2:
        case DEVICE_EVENT_FROM_USB_HOST:
+           music_dev_change_flag = 0;
+           is_need_play_dev_tone = 1;
            music_play_device_event_deal((u32)event->arg, event->u.dev.event);
            return true;
        default://switch((u32)event->arg)
@@ -313,6 +339,18 @@ static int music_state_machine(struct application *app, enum app_sta
te state,
        switch (it->action) {
        case ACTION_APP_MAIN:
            log_info("ACTION_APP_MAIN\n");
+           music_dev_change_flag = 0;
+           is_need_play_dev_tone = 1;
+#if MUSIC_USB_TF_INDEPENDENT_ENABLE
+           if(it->exdata == NULL) {
+               //选择设备进行播放
+               if((file_opr_available_dev_total(>1) && (MUSIC_PRIORITY_DEV != "null"))
{
+                   y_printf("priority : %s\n",MUSIC_PRIORITY_DEV);
+                   music_app_init(MUSIC_PRIORITY_DEV);
+                   break;
+               }
+           }
+#endif
            music_app_init((void *)it->exdata);
            break;
        }
diff --git a/SDK/apps/soundbox/music/music_player_api.c b/SDK/apps/soundbox/music/music_p
layer_api.c
index 1bfcfb1..1032b8b 100644
--- a/SDK/apps/soundbox/music/music_player_api.c
+++ b/SDK/apps/soundbox/music/music_player_api.c
@@ -136,7 +136,13 @@ static const u8 MUSIC_SCAN_PARAM[] = "-t"
;
#endif//FILT_RECORD_FILE_EN
static u8 music_cycle_mde = FCYCLE_ALL;
+#if MUSIC_USB_TF_INDEPENDENT_ENABLE
```

```
+extern char is_need_play_dev_tone; //设备提示音播放
+//此处改为单设备循环,防止上下曲时切设备
+static u8 music_dev_cycle_mde = DEV_CYCLE_ONE;
+#else
+static u8 music_dev_cycle_mde = DEV_CYCLE_ALL;
+#endif
+static struct music_opr *__this = NULL;

@@ -211,6 +217,18 @@ static int _music_play_start(MUSIC_PLAYER *m_ply, struct audio_dec_b
reakpoint *b
+if TCFG_UI_ENABLE
+    u16 file_num = music_play_get_file_number();
+    ui_set_tmp_menu(MENU_FILENUM, 1000, file_num, NULL);
+#endif
+#if MUSIC_USB_TF_INDEPENDENT_ENABLE
+    //播放设备提示音
+    if(is_need_play_dev_tone) {
+        is_need_play_dev_tone = 0;
+        y_printf("play %s\n",__this->mplay->fopr->dev->logo);
+        if(!strcmp(__this->mplay->fopr->dev->logo,"udisk")) {
+            tone_play_index(IDEX_TONE_USB, 1);
+        } else {
+            tone_play_index(IDEX_TONE_TF, 1);
+        }
+    }
+#endif
+    }
+    return ret;
@@ -899,6 +917,14 @@ static int muisc_play_decode_event(u8 event, u8 value)
+    }
+    if (read_err) {
+        ///设备读错误应该是播放下一个设备
+#if MUSIC_USB_TF_INDEPENDENT_ENABLE
+        if(MUSIC_PRIORITY_DEV != "null") {
+            //U盘 拔出时 判断是否需要切模式
+            y_printf("priority dev%s\n",music_play_get_cur_dev());
+            if(music_play_get_cur_dev() != MUSIC_PRIORITY_DEV)
+                goto __change_mode;
+        }
+#endif
+        if (file_opr_available_dev_total() >= 2) {
+            music_ply_save_bp(__this->mplay);
+            err = music_play_by_dev_bp(DEV_SEL_NEXT, 0);

```

```
@@ -989,6 +1015,20 @@ int music_play_device_event_deal(u32 evt_src, u8 evt)
    && __this->mplay->fopr->dev
    && (!strcmp(logo, __this->mplay->fopr->dev->logo))) {
    //当前播放的设备拔出
+#if MUSIC_USB_TF_INDEPENDENT_ENABLE
+    if(MUSIC_PRIORITY_DEV != "null") {
+        //TF 拔出时 判断是否需要切模式
+        if(strcmp(__this->mplay->fopr->dev->logo,MUSIC_PRIORITY_DEV)) {
+            ///如果没有更多设备, 切换模式的时候保存断点, 此处不重复保存
+            printf("NO priority dev , switch next app\n");
+            app_task_next();
+            return 0;
+        } else {
+            printf("priority dev\n");
+        }
+    }
+    printf("play %s\n",__this->mplay->fopr->dev->logo);
+#endif
    if (file_opr_available_dev_total()) {
        music_ply_save_bp(__this->mplay);
    #if DEC_SWITCH_DEV_USE_BP
diff --git a/SDK/cpu/br25/audio_dec/tone_player.c b/SDK/cpu/br25/audio_dec/tone_player.c
index b2b9ae8..aeb42f 100644
--- a/SDK/cpu/br25/audio_dec/tone_player.c
+++ b/SDK/cpu/br25/audio_dec/tone_player.c
@@ -1687,6 +1687,8 @@ static const char *const tone_index[] = {
    #if (defined(TCFG_APP_RECORD_EN) && (TCFG_APP_RECORD_EN))
        TONE_RECORD,
    #endif
+    TONE_USB,
+    TONE_TF,
    };
    int tone_play_index_with_callback(u8 index, u8 preemption, void (*user_evt_handler)(void
    *priv), void *priv)
    {
```

36.AC696 音箱版本直接播放 MP3 提示音方法 20200801 YWP

新版本 696 的 V025 版本直接在工具里面替换即可。也可以把 V025 制作的 tone.cfg 文件直接拷贝替换其他 SDK 的 tone.cfg 文件。

```

tone_player.c Board_config.h Board_ac696x_demo_cfg.h Search Results Sdk.elf.resolution.txt Symbol
00416:     file_dec->idx = 0;
00417:     return 0;
00418: }
00419: struct tone_format {
00420:     const char *fmt;
00421:     u32 coding_type;
00422: };
00423:
00424: const struct tone_format tone_fmt_support_list[] = {
00425:     {"wtg", AUDIO_CODING_G729},
00426:     {"msbc", AUDIO_CODING_MSBC},
00427:     {"sbc", AUDIO_CODING_SBC},
00428:     {"mty", AUDIO_CODING_MTY},
00429:     {"aac", AUDIO_CODING_AAC},
00430:     {"mp3", AUDIO_CODING_MP3},
00431: };
00432:
00433: static u32 tone_file_format_match(char *fmt)
00434: {
00435:     int list_num = ARRAY_SIZE(tone_fmt_support_list);
00436:     int i = 0;
00437:
00438:     if (fmt == NULL) {
00439:         return AUDIO_CODING_UNKNOW;
00440:     }
00441:
00442:     for (i = 0; i < list_num; i++) {
00443:         if (ASCII_StrCmpNoCase(fmt, tone_fmt_support_list[i].fmt,

```

LuaConfig - 20200521.1 (1.0.18) - Version: AC696X-v0.01-cfg_tool-v0.08

打开 保存 加载 显示 language 关于

通用配置 蓝牙配置 提示音配置 状态配置 联合首里

提示音名字	操作	文件路径
4 数字3	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**6X_config_tool/conf/output/extra_tones/3.wtg
5 数字4	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**6X_config_tool/conf/output/extra_tones/4.wtg
6 数字5	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**6X_config_tool/conf/output/extra_tones/5.wtg
7 数字6	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**6X_config_tool/conf/output/extra_tones/6.wtg
8 数字7	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**6X_config_tool/conf/output/extra_tones/7.wtg
9 数字8	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**6X_config_tool/conf/output/extra_tones/8.wtg
10 数字9	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**6X_config_tool/conf/output/extra_tones/9.wtg
11 蓝牙模式	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**br25/tools/提示音/bt.mp3
12 连接成功	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**fig_tool/conf/output/extra_tones/bt_conn.wtg
13 断开连接	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**ig_tool/conf/output/extra_tones/bt_dconn.wtg
14 对箱连接成功	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**ig_tool/conf/output/extra_tones/tws_conn.wtg
15 对箱断开连接	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**g_tool/conf/output/extra_tones/tws_dconn.wtg
16 低电提示音	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**g_tool/conf/output/extra_tones/low_power.wtg
17 关机	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**g_tool/conf/output/extra_tones/power_off.wtg
18 开机	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**ig_tool/conf/output/extra_tones/power_on.wtg
19 来电	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**config_tool/conf/output/extra_tones/ring.wtg
20 配对模式	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**nfig_tool/conf/output/extra_tones/paired.wtg
21 linein模式	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**nfig_tool/conf/output/extra_tones/linein.wtg
22 music模式	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**onfig_tool/conf/output/extra_tones/music.wtg
23 fm模式	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**X_config_tool/conf/output/extra_tones/fm.wtg
24 record模式	打开 播放 删除	E:/AC696X/ac696n_soundbox_sdk_v0.2.5/SDK/cpu**nfig_tool/conf/output/extra_tones/record.wtg

加载 添加提示音 主要格式 保留原有格式 恢复默认提示音 保存提示音文件

37.696 系列 K 歌宝解决第一声延时长的问题 20200803 YWP

除了程序上我们看得到的干声和湿声的增益，其次还有 MIC 模拟打开出声音的优化。干声和湿声的增益大小范围是 0-4096.如下图。

```

p_common_key.c | Slidekey.c | Audio_reverb.c | Avctp_user.h | App_action.c | App_common.c | Bt.c | Key_event_deal.c | Search Results | Audio_dec.c | 全
00166:
00167: //*****混响接口*****//
00168: REVERB_API_STRUCT *open_reverb(REVERB_PARAM_SET *reverb_setting, u16 sample_rate)
00169: {
00170:     REVERB_API_STRUCT *reverb_api_obj;
00171:
00172:     int buf_len;
00173:
00174:     reverb_api_obj = zalloc(sizeof(REVERB_API_STRUCT));
00175:     if (!reverb_api_obj) {
00176:         return NULL;
00177:     }
00178:     reverb_api_obj->func_api = get_reverb_func_api();
00179:
00180:     //初始化混响参数
00181:     if (reverb_setting) {
00182:         memcpy(&reverb_api_obj->parm, reverb_setting, sizeof(REVERB_PARAM_SET));
00183:     } else {
00184:         reverb_api_obj->parm.decayval = 1500; //2000; //衰减系数 [0:4096]
00185:         reverb_api_obj->parm.deepval = 0; //4096; //调节混响深度, 影响pre-delay; [0-4096]. 4096 代表max_ms
00186:         reverb_api_obj->parm.filtsize = 0; // [0-4096], 如果要回声效果 置0; 混响效果建议4096
00187:         reverb_api_obj->parm.wetgain = 4096; //湿声增益: [0:4096]
00188:         reverb_api_obj->parm.drygain = 4096; //1000// 4096; //干声增益: [0:4096]
00189:         #if defined(TCFG_REVERB_SAMPLERATE_DEFAULT)
00190:         reverb_api_obj->parm.sr = TCFG_REVERB_SAMPLERATE_DEFAULT; //配置输入的采样率, 影响need_buf 大小
00191:         #else
00192:         reverb_api_obj->parm.sr = 16000; //配置输入的采样率, 影响need_buf 大小
00193:         #endif //TCFG_REVERB_SAMPLERATE_DEFAULT
00194:         reverb_api_obj->parm.max_ms = 200; //250; //所需要的最大延时, 影响 need_buf 大小
00195:         reverb_api_obj->parm.centerfreq_bandq = 0; //留声机音效, 不需要置0

```

干声是 MIC 完成 ADC 后得 PCM 数据，湿声是混响得后续处理数据。
在增加一个 MIC 模拟开启得函数。

```

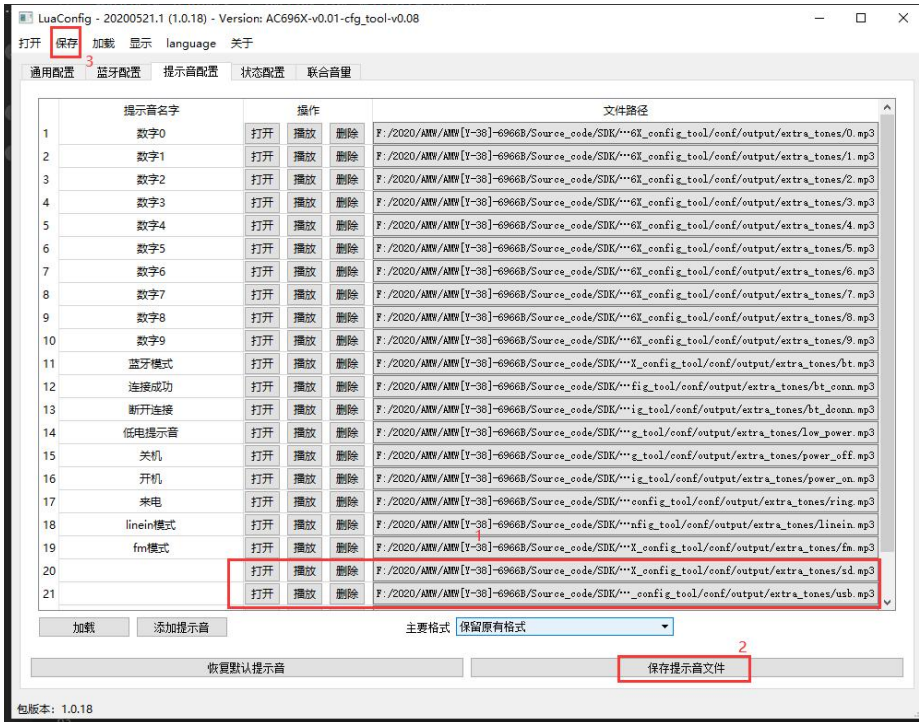
void mic_2_dac_rl(u8 r_en, u8 l_en)
{
    if(r_en) {
        SFR(JL_ANA->ADA_CON4, 19, 1, 1); //MIC_2_R
    } else {
        SFR(JL_ANA->ADA_CON4, 19, 1, 0); //MIC_2_R
    }
    if(l_en) {
        SFR(JL_ANA->ADA_CON4, 18, 1, 1); //MIC_2_L
    } else {
        SFR(JL_ANA->ADA_CON4, 18, 1, 0); //MIC_2_L
    }
    // printf("\n--func=%s\n", __FUNCTION__);
}

```

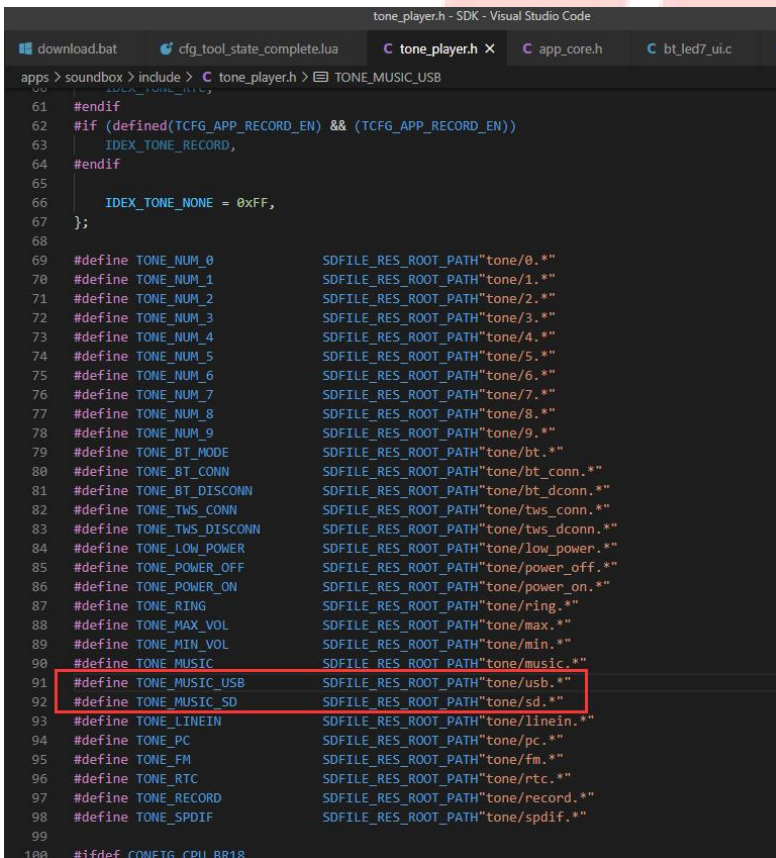
```
00648:     }  
00649:     }  
00650: #endif  
00651:  
00652: #if (defined(REVERB_EQ_ENABLE) && REVERB_EQ_ENABLE != 0)  
00653:     audio_eq_run(dec->p_eq, &dec->run_buf[REVERB_RUN_POINT_NUM * 2], REVERB_RUN_POINT_NUM * 2);  
00654: #endif  
00655:     if (dec->p_reverb_obj) {  
00656:         /* if(dec->first_tone >= 20) { */  
00657:         dec->p_reverb_obj->func_api->run(dec->p_reverb_obj->ptr, &dec->run_buf[REVERB_RUN_POINT_NUM * 2], REVERB_RUN_POINT_NUM * 2);  
00658:         /* }else{ */  
00659:         /* dec->first_tone++; */  
00660:         /* if(dec->first_tone == 1)*/  
00661:         mic_2_dac_rl(1,1);  
00662:         /* } */  
00663:     }  
00664: #if (defined(USER_AUDIO_PROCESS_ENABLE) && (USER_AUDIO_PROCESS_ENABLE != 0))  
00665:     if (dec->user_hdl) {  
00666:         u8 ch_num = 1;  
00667:         user_audio_process_handler_run(dec->user_hdl, &dec->run_buf[REVERB_RUN_POINT_NUM * 2], REVERB_RUN_POINT_NUM * 2, ch_num);  
00668:     }  
00669: #endif  
00670:  
00671:     dec->run_len <<= 1;
```

38. music 模式下 LED7 显示、播放对应设备名，并在播放完之后显示文件号 20200808 SQ

- 1) 添加提示音文件



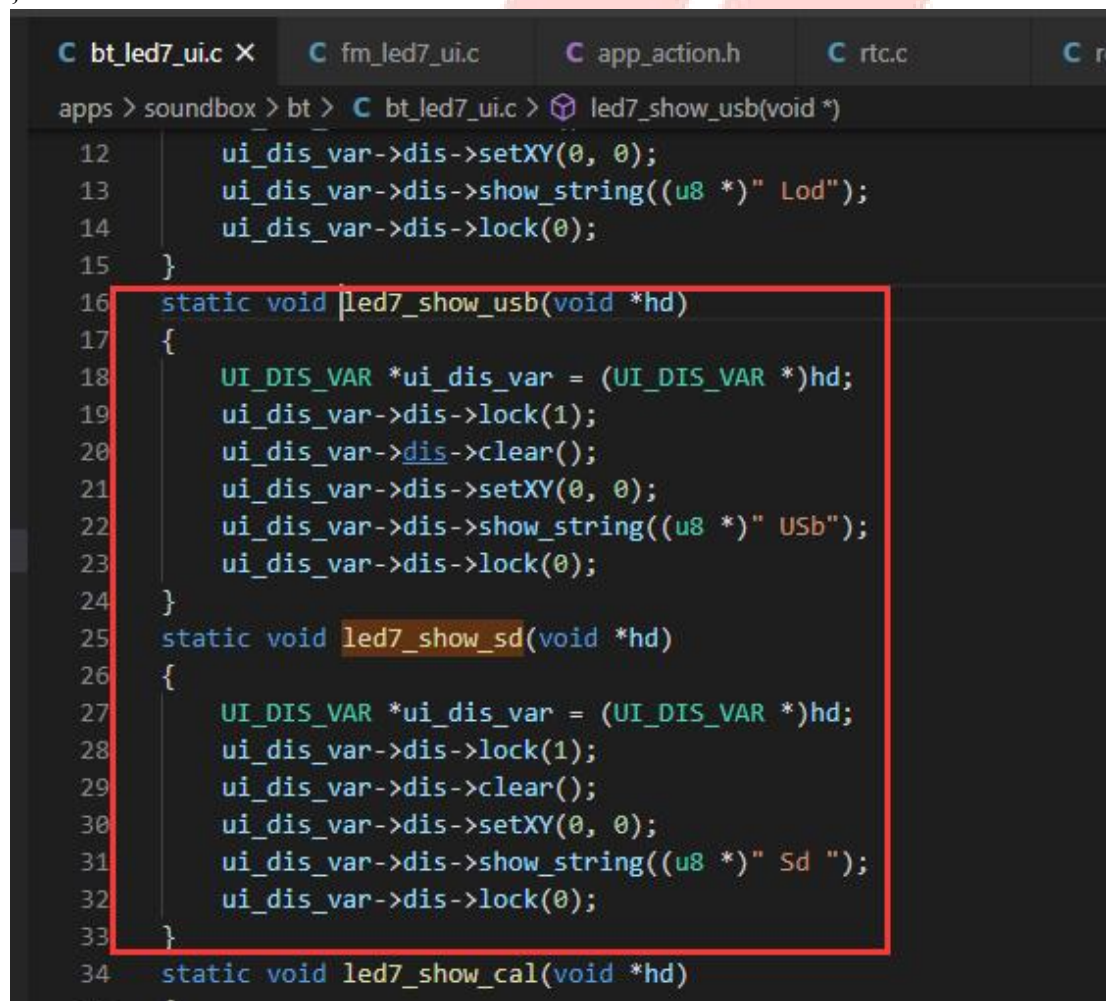
```
#define TONE_MUSIC_USB SDFILE_RES_ROOT_PATH"tone/usb.*"
#define TONE_MUSIC_SD SDFILE_RES_ROOT_PATH"tone/sd.*"
```



2) 添加显示设备名函数

```
static void led7_show_usb(void *hd)
{
```

```
    UI_DIS_VAR *ui_dis_var = (UI_DIS_VAR *)hd;
    ui_dis_var->dis->lock(1);
    ui_dis_var->dis->clear();
    ui_dis_var->dis->setXY(0, 0);
    ui_dis_var->dis->show_string((u8 *)" USB");
    ui_dis_var->dis->lock(0);
}
static void led7_show_sd(void *hd)
{
    UI_DIS_VAR *ui_dis_var = (UI_DIS_VAR *)hd;
    ui_dis_var->dis->lock(1);
    ui_dis_var->dis->clear();
    ui_dis_var->dis->setXY(0, 0);
    ui_dis_var->dis->show_string((u8 *)" Sd ");
    ui_dis_var->dis->lock(0);
}
```



The screenshot shows a code editor with several tabs: 'bt_led7_ui.c', 'fm_led7_ui.c', 'app_action.h', 'rtc.c', and 'rt'. The active file is 'bt_led7_ui.c'. The code in the editor is as follows:

```
apps > soundbox > bt > C bt_led7_ui.c > led7_show_usb(void *)
12     ui_dis_var->dis->setXY(0, 0);
13     ui_dis_var->dis->show_string((u8 *)" Lod");
14     ui_dis_var->dis->lock(0);
15 }
16 static void led7_show_usb(void *hd)
17 {
18     UI_DIS_VAR *ui_dis_var = (UI_DIS_VAR *)hd;
19     ui_dis_var->dis->lock(1);
20     ui_dis_var->dis->clear();
21     ui_dis_var->dis->setXY(0, 0);
22     ui_dis_var->dis->show_string((u8 *)" USB");
23     ui_dis_var->dis->lock(0);
24 }
25 static void led7_show_sd(void *hd)
26 {
27     UI_DIS_VAR *ui_dis_var = (UI_DIS_VAR *)hd;
28     ui_dis_var->dis->lock(1);
29     ui_dis_var->dis->clear();
30     ui_dis_var->dis->setXY(0, 0);
31     ui_dis_var->dis->show_string((u8 *)" Sd ");
32     ui_dis_var->dis->lock(0);
33 }
34 static void led7_show_cal(void *hd)
```

MENU_USER_USB,
MENU_USER_SD,

```
C ui_api.h X C board_ac696x_demo_cfg.h C
apps > soundbox > include > ui > C ui_api.h > _unnar
87 MENU_POWER,
88 MENU_LIST_DISPLAY,
89
90
91 MENU_LED0,
92 MENU_LED1,
93
94
95 MENU_RECORD,
96
97 MENU_USER_USB,
98 MENU_USER_SD,
99 MENU_USER_CAL,
100
```

- 3) 调用设备显示
在切换模式时不显示 lod

```
C app_action.c X download.bat cfg_tool_state_complete.lua C tone_player.h C a
apps > soundbox > C app_action.c > ...
428 #endif
429
430 }
431 #endif
432
433     _this->tone_busy = 1;
434     if (!mode_tone_play(app_reg->tone_name, app_mode_tone_play_end
435         if (app_reg->tone_prepare) {
436             app_reg->tone_prepare();
437         } else {
438             // ui_set_tmp_menu(MENU_WAIT, 0, 0, NULL);
439         }
440     }
441
```

```
case MENU_USER_USB:
    led7_show_usb(hd);
    break;
case MENU_USER_SD:
    led7_show_sd(hd);
    break;
```

```
C bt_led7_ui.c X C fm_led7_ui.c C app_action.h C rtc.c C
apps > soundbox > bt > C bt_led7_ui.c > ui_common(void *, void * private, u8, u32)
177
178 void ui_common(void *hd, void *private, u8 menu, u32 arg)//公共
179 {
180     u16 fre = 0;
181     UI_DIS_VAR *ui_dis_var = (UI_DIS_VAR *)hd;
182
183     if (!hd) {
184         return;
185     }
186
187     switch (menu) {
188     case MENU_POWER_UP:
189         led7_show_hi(hd);
190         break;
191     case MENU_MAIN_VOL:
192         led7_show_volume(hd, arg & 0xff);
193         break;
194     case MENU_USER_USB:
195         led7_show_usb(hd);
196         break;
197     case MENU_USER_SD:
198         led7_show_sd(hd);
199         break;
200     case MENU_USER_CAL:
```

4) 显示、播放设备名并在播放完提示音之后显示曲目

```
bool user_play_tone_dev_flag = 0;
static u16 user_file_num_flag = 0;
static void user_dely_play_file_number(u8 menu){
    ui_set_tmp_menu(MENU_FILENUM, 1000, user_file_num_flag, NULL);
}
```

```
201 }
202
203 bool user_play_tone_dev_flag = 0;
204 static u16 user_file_num_flag = 0;
205 static void user_dely_play_file_number(u8 menu){
206     ui_set_tmp_menu(MENU_FILENUM, 1000, user_file_num_flag, NULL);
207 }
208 static int _music_play_start(MUSIC_PLAYER *m_ply, struct audio_dec_breakpoint *bp)
209 {
210     /* printf("_music_play_start in\n"); */
211
212     int ret = music_ply_start(m_ply, bp);
213     printf("_music_play_start out %d, cur dev = %s\n", ret, m_ply->fopr->dev->logo);
214     if (!ret) {
215         __this->err_cnt = 0;
216 #if TCFG_UI_ENABLE
217         user_file_num_flag = music_play_get_file_number();
218         // ui_set_tmp_menu(MENU_FILENUM, 1000, user_file_num_flag, NULL);
219
220         extern int tone_play_by_path(const char *name, u8 preemption);
221         if(user_play_tone_dev_flag){
222             user_play_tone_dev_flag = 0;
223             if(!strcmp(__this->mplay->fopr->dev->logo,"udisk")) {
224                 tone_play_by_path(TONE_MUSIC_USB,1);
225                 ui_set_tmp_menu(MENU_USER_USB, 1000, 0, user_dely_play_file_number);
```

```
extern int tone_play_by_path(const char *name, u8 preemption);
if(user_play_tone_dev_flag){
    user_play_tone_dev_flag = 0;
    if(!strcmp(__this->mplay->fopr->dev->logo,"udisk")) {
        tone_play_by_path(TONE_MUSIC_USB,1);
        ui_set_tmp_menu(MENU_USER_USB, 1000, 0, user_dely_play_file_number);
    } else {
        tone_play_by_path(TONE_MUSIC_SD,1);
        ui_set_tmp_menu(MENU_USER_SD, 1000, 0, user_dely_play_file_number);
    }
} else {
    ui_set_tmp_menu(MENU_FILENUM, 1000, user_file_num_flag, NULL);
}
```



```
412         break;
413     case DEVICE_EVENT_FROM_USB_HOST:
414     #if TCFG_UDISK_ENABLE
415         if (event->u.dev.event == DEVICE_EVENT_IN) {
416             user_play_tone_dev_flag = 1;
417
418             if (!strcmp((char *)event->u.dev.value, "udisk", 5)) {
419                 ///if usb host mount ok, need check udisk fat mount, if mount err
420                 printf("udisk mount\n");
421                 #if (defined(CONFIG_FATFS_ENBALE) && CONFIG_FATFS_ENBALE)
422                 ret = file_opr_dev_add("udisk");
423                 #endif//CONFIG_FATFS_ENBALE
424                 if (ret) {
425                     log_error("file_opr_dev_add fail 11111 %x\n", ret - STORAGE_DE
426                     usb_mount_fail(g_usb_id);
427                     music_play_usb_host_mount_after();
428                 } else {
429                     usb_online_mount_after(g_usb_id);
430                     printf("usb_mount ok\n");
431                 }
432             } else {
433                 usb_online_mount_after(g_usb_id);
434             }
435
436     #if TCFG_USB_DM_MULTIPLEX_WITH_SD_DAT0
437         msg_notify_enable = 1;
438         if (flag_tp1) {
439             struct sys_event e = {0};
440             e.type = SYS_DEVICE_EVENT;
441             e.arg = (void *)flag_tp2;
442             if (flag_tp1 == 1) {
443                 e.u.dev.event = DEVICE_EVENT_IN;
444             } else if (flag_tp1 == 2) {
445                 e.u.dev.event = DEVICE_EVENT_OUT;
446             }
447             flag_tp1 = 0;
448             sys_event_notify(&e);
449         }
450     #endif
451
452     } else if (event->u.dev.event == DEVICE_EVENT_OUT) {
453         user_play_tone_dev_flag = 1;
454
455         if (file_opr_dev_check("udisk")) {
```



```
317
318 static int music_state_machine(struct application *app, enum app_state state,
319                               struct intent *it)
320 {
321     int ret;
322     switch (state) {
323     case APP_STA_CREATE:
324         break;
325     case APP_STA_START:
326         music_play_init();
327         if (!it) {
328             break;
329         }
330         switch (it->action) {
331         case ACTION_APP_MAIN:
332             log_info("ACTION_APP_MAIN\n");
333             u8 *logo_flag = (u8 *)it->exdata;
334
335             extern bool user_play_tone_dev_flag;
336             user_play_tone_dev_flag = 1;
337             if(logo_flag == JL_USER_USB || logo_flag == JL_USER_SD0 || logo_flag == JL_USER_SD1){
338                 // music_app_init((void *)it->exdata);
339             }else if(file_opr_available_dev_total(>1){
340                 logo_flag = JL_USER_USB_OR_SD;
341             }
342
343             music_app_init((void *)logo_flag);
344             break;
345         }
346         break;
347     case APP_STA_STOP:
```

切换设备时设置标志位。默认 sdk 不会再 music 模式下切换设备 需要自己添加。

```
C music.c X C file_api.c C music_player_api.c C music_player.c C fs.h
apps > soundbox > music > C music.c > _key_event_opr(sys_event *)
90 #endif
91 // key
92 extern u8 music_key_event_get(struct key_event *key);
93 static int _key_event_opr(struct sys_event *event)
94 {
95     int ret = true;
96     int err = 0;
97     u8 vol;
98     struct key_event *key = &event->u.key;
99
100 #if TCFG_APP_FM_EMITTER_EN
101 #if TCFG_UI_ENABLE
102     if (ui_get_app_menu(GRT_CUR_MENU) == MENU_FM_SET_FRE) {
103         return false;
104     }
105 #endif
106 #endif
107
108     u8 key_event = music_key_event_get(key);
109
110     switch (key_event) {
111     case KEY_CHANGE_MODE:
112         printf(">>>>>>>> music mode change mode\n");
113         if(file_opr_available_dev_total(>1){
114             if(JL_USER_USB_OR_SD == music_play_get_cur_dev()){
115                 user_play_tone_dev_flag = 1;
116                 music_play_change_dev_next();
117                 break;
118             }
119         }
120         ret = false;
121     break;
122
```

39. 报号前播放指定一个提示音 20200808 SQ

```
apps > soundbox > bt > C bt.c > number_to_play_list(char *, u32 *)
903
904     if (num) {
905         for (; i < strlen(num); i++) {
906             lst[i] = num0_9[num[i] - '0'] ;
907         }
908     }
909     lst[i++] = (u32)TONE_REPEAT_BEGIN(-1);
910     lst[i++] = (u32)TONE_RING;
911     lst[i++] = (u32)TONE_REPEAT_END();
912     lst[i++] = (u32)NULL;
913 }
914
915 static void number_to_play_list(char *num, u32 *lst)
916 {
917     u8 i = 0;
918     lst[i] = (u32)TONE_RING;
919     lst++;
920
921     if (num) {
922         for (; i < strlen(num); i++) {
923             lst[i] = num0_9[num[i] - '0'] ;
924         }
925     }
926 }
927     lst[i++] = (u32)TONE_REPEAT_BEGIN(-1);
928     lst[i++] = (u32)TONE_RING;
929     lst[i++] = (u32)TONE_REPEAT_END();
930     lst[i++] = (u32)NULL;
931 }
932
```

指定提示音

40. 固定音量播放提示音 20200808 SQ

```
audio_config.h - SDK - Visual Studio Code
file.h C audio_config.h X C ascii.h C audio_decoder.h C list.h C os_api.h
apps > soundbox > include > C audio_config.h > APP_AUDIO_STATE_MUSIC
63 #endif (SYS_VOL_TYPE == 1)
64 #define SYS_MAX_VOL MAX_ANA_VOL
65 #elif (SYS_VOL_TYPE == 2)
66 #define SYS_MAX_VOL MAX_COM_VOL
67 #endif
68
69 #define SYS_DEFAULT_VOL 0//((SYS_MAX_VOL/2)
70 #define SYS_DEFAULT_TONE_VOL (25)//(SYS_MAX_VOL)
71
72 // #define AUDIO_MIC_TEST
73
74
75
76 extern u8 audio_dma_buffer[AUDIO_FIXED_SIZE];
77 extern const u16 dig_vol_table[];
78
79 #define APP_AUDIO_STATE_WTONE_BY_MUSIC 0//使能宏, 提示音量使用music音量
80
81
82 #define APP_AUDIO_STATE_IDLE 0
83 #define APP_AUDIO_STATE_MUSIC 1
84 #define APP_AUDIO_STATE_CALL 2
85 #define APP_AUDIO_STATE_WTONE 3
86 #define APP_AUDIO_CURRENT_STATE 4
87
```

设置固定提示音量值

提示音量跟随music音量宏设为0

避免音量为0时播放最小提示音没有声音

```
C event.h C app_audio.c X C usb.h C storage_dev.h C file_operate.h
cpu > br25 > audio_common > C app_audio.c > app_audio_set_volume(u8, s8, u8)
449 void app_audio_set_volume(u8 state, s8 volume, u8 fade)
450 {
451     switch (state) {
452     case APP_AUDIO_STATE_IDLE:
453     case APP_AUDIO_STATE_MUSIC:
454         app_var.music_volume = volume;
455         if (app_var.music_volume > get_max_sys_vol()) {
456             app_var.music_volume = get_max_sys_vol();
457         }
458         volume = app_var.music_volume;
459         break;
460     case APP_AUDIO_STATE_CALL:
461         app_var.call_volume = volume;
462         if (app_var.call_volume > app_var.aec_dac_gain) {
463             app_var.call_volume = app_var.aec_dac_gain;
464         }
465         volume = app_var.call_volume;
466     #if TCFG_CALL_USE_DIGITAL_VOLUME
467         audio_digital_vol_set(volume);
468         return;
469     #endif
470     break;
471     case APP_AUDIO_STATE_WTONE:
472     #if (APP_AUDIO_STATE_WTONE_BY_MUSIC == 1)
473         app_var.wtone_volume = app_var.music_volume;
474         if (app_var.wtone_volume < 5) {
475             app_var.wtone_volume = 5;
476         }
477     #else
478         app_var.wtone_volume = volume;
479         app_var.wtone_volume = SYS_DEFAULT_TONE_VOL;
480     #endif
481     if (app_var.wtone_volume > get_max_sys_vol()) {
482         app_var.wtone_volume = get_max_sys_vol();

```

避免音量为0时播放提示音没声音

41.Linein 模式走模拟使用抢断方式播放提示音带有 Linein 声音 20200808 SQ

1) 确认 linein 通道为模拟通道

```

82
83 //*****
84 //                               linein配置
85 //*****
86 #define TCFG_LINEIN_ENABLE           ENABLE_THIS_MOUDLE // linein使能
87 // #define TCFG_LINEIN_LADC_IDX       0 // linein使用的ladc通道, 对应ladc_list
88 #define TCFG_LINEIN_LR_CH           AUDIO_LIN1_LR//AUDIO_LIN0L_CH
89 #define TCFG_LINEIN_CHECK_PORT       IO_PORTB_04 // linein检测IO
90 #define TCFG_LINEIN_PORT_UP_ENABLE   1 // 检测IO上拉使能
91 #define TCFG_LINEIN_PORT_DOWN_ENABLE 0 // 检测IO下拉使能
92 #define TCFG_LINEIN_AD_CHANNEL       NO_CONFIG_PORT // 检测IO是否使用AD检测
93 #define TCFG_LINEIN_VOLTAGE          0 // AD检测时的阈值
94 #if(TCFG_REVERB_ENABLE)
95 #define TCFG_LINEIN_INPUT_WAY        LINEIN_INPUT_WAY_ANALOG
96 #else
97 #define TCFG_LINEIN_INPUT_WAY        LINEIN_INPUT_WAY_ANALOG //LINEIN_INPUT_WAY_ANALOG//LINEIN_INPUT_WAY_ANALOG
98 #endif
99 #define TCFG_LINEIN_MULTIPLEX_WITH_FM DISABLE // linein 脚与 FM 脚复用
100 #define TCFG_LINEIN_MULTIPLEX_WITH_SD DISABLE // linein 检测与 SD cmd 复用
101

```

Linein走模拟

2) 播提示音时把 linein 通道 mute 掉，播放结束之后延时 umute。延时 umute 解决连续播提示音时提示音间会有 linein 声音。

```

linein.c - SDK - Visual Studio Code
C linein.c x C audio_linein.h C debug.h C timer.h C user_pa.h
apps > soundbox > linein > C linein.c > _key_event_opr(sys_event *)
258     user_manual_mute(!(__this->onoff));
259     break;
260     case KEY_VOL_UP:
261     #if (TCFG_LINEIN_INPUT_WAY != LINEIN_INPUT_WAY_ADC)
262     if (!__this->volume) {
263     audio_linein_mute(0);
264     }
265     #endif
266     if ((__this->volume+2) <= get_max_sys_vol()) {
267     __this->volume +=2;
268     linein_volume_set(__this->volume);
269     } else {
270     linein_volume_set(__this->volume);
271     if (tone_get_status() == 0) {
272     /* tone_play(TONE_MAX_VOL); */
273     #if TCFG_MAX_VOL_PROMPT
274     // STATUS *p_tone = get_tone_config();
275     // tone_play_index(p_tone->max_vol, 1);
276     #endif
277     if(!tone_get_status()){
278     audio_linein_mute(1);
279     tone_play_index(IDEX_TONE_MAX_VOL, 1);
280     }
281     #endif
282     #endif
283     }

```

```

u16 user_linein_umute_id = 0;
void user_linein_umute(void){

```

```

    if(__this->onoff && __this->volume && !tone_get_status()){
        audio_linein_mute(0);
    }

```


42.AC696 系列 V024V025 版本的 FM 背噪杂音的库优化 20200808 PYP

替换 cpu.a 库，然后如下截图文件对应修改。

```
m_inside.h *fm_inside.c
/* fm_inside_default_config(): */
/* #if CONFIG_CLOCK_BY_TYPE
u32 fm_clk = switching_code_type(AUDIO_ANOTHER_CLK_TYPE_FM, 1);
fm_inside_io_ctrl(SET_FM_INSIDE_SYS_CLK, fm_clk / 1000000L);
#else
fm_inside_io_ctrl(SET_FM_INSIDE_SYS_CLK, FM_CLOCK / 1000000L);
#endif */

fm_inside_mem_init();
fm_inside_io_ctrl(SET_FM_INSIDE_AGC_PARAM, FMSCAN_AGC, FMSCAN_ADD_DIFFER, FMSCAN_P_DIFFER, FMSCAN_N_DIFFER);
fm_inside_io_ctrl(SET_FM_INSIDE_COUNTRY_PARAM, REAL_FREQ_MIN, REAL_FREQ_MAX, FREQ_STEP);
fm_inside_io_ctrl(SET_FM_INSIDE_SCAN_ARG1, FMSCAN_SEEK_CNT_MIN, FMSCAN_SEEK_CNT_MAX, FMSCAN_CNR, FM_IF);
fm_inside_io_ctrl(SET_FM_INSIDE_SCAN_ARG2, FMSCAN_960_CNR, FMSCAN_1080_CNR);

fm_inside_on(); //fm analog init
fm_inside_set_stereo(0); //set[0,128], 0 mono, 1-128 stereo.

bool fm_inside_set_freq(void *priv, u16 freq)
{
    u8 ret;
    u32 freq;
    //fm_printf("[f] ", freq);
    freq = freq * 10;
    ret = fm_inside_freq_set(freq);
    return ret;
}

bool fm_inside_read_id(void *priv)
{
    return (bool)fm_inside_id_read();
}

4 #if(TCFG_FM_INSIDE_ENABLE == ENABLE)
5
6 #if DAC2IIS_EN
7 #define FM_DAC_OUT_SAMPLERATE 44100L
8 #else
9 #define FM_DAC_OUT_SAMPLERATE 44100L
10 #endif
11
12 /*****
13 * FM调试说明
14 * 真台少: 假台多: 叠台多:
15 * 减小 FMSCAN_CNR(主) 加大 FMSCAN_CNR(主) 减小: FMSCAN_AGC
16 * 减小 FMSCAN_P_DIFFER(主) 加大 FMSCAN_P_DIFFER(主)
17 * 加大 FMSCAN_N_DIFFER(次) 减小 FMSCAN_N_DIFFER(次)
18 *
19 * 注意: 不要插串口测试搜台数
20 *****/
21 #define FMSCAN_SEEK_CNT_MIN 400 //最小过零点数 400左右
22 #define FMSCAN_SEEK_CNT_MAX 600 //最大过零点数 600左右
23 #define FMSCAN_960_CNR 34 //指波96M的基础cnr 30~40
24 #define FMSCAN_1080_CNR 34 //指波108M的基础cnr 30~40
25 #define FMSCAN_AGC -95 //AGC阈值 -55左右
26 #define FMSCAN_ADD_DIFFER -67 //低于此值增加noise differ, -67左右
27
28 #define FMSCAN_CNR 2 //cnr 1以上
29 #define FMSCAN_P_DIFFER 2 //power differ 1以上
30 #define FMSCAN_N_DIFFER 8 //noise differ 8左右
31
32 #define FM_IF 3 //0,1.875; 1,2.143; 2,1.5; 3,cnr低的中频听台
33 /* struct FM_INSIDE_DAT {
34 AUDIO_STREAM *stream_io;
35 volatile u8 src_toggle;
36 }; */
37
38
39
40 #endif
```

资料如下从百度网盘提取:

链接: https://pan.baidu.com/s/1JDU0U9TSYzuSq_P1w69pxw

提取码: 933p

43.【音箱】AC696 系列 DAC 输出前后有 PO 声（DAC 设置高阻引起） 202008014 LZK

如果你方案的 DAC 输出方式是隔直方式，请按以下修改。

app_audio.c ——注释掉 DAC 设置高阻的函数

```
app_audio.c - SDK - Visual Studio Code
cpu > br25 > audio_common > app_audio.c > app_audio_output_ch_mute(u8, u8)
1783     return 0;
1784 }
1785
1786 void app_audio_output_ch_mute(u8 ch, u8 mute)
1787 {
1788     //audio_dac_ch_mute(&dac_hdl, ch, mute);
1789 }
1790
1791 int audio_output_sync_start(void)
```

44.【音箱】AC696 系列 AUX 走模拟时播叠加提示音时，音乐出现卡顿处理方法 202008015 WTS

AC696X 音箱 SDK 在 AUX 模式下走模拟通道，在播系统叠加提示音时（非外部提示音，外部提示音只能用打断方式），音乐出现卡顿处理(红色字体为增加的修改):

```
D:\...公版SDK\ac696n_soundbox_sdk_v0.2.4\SDK\apps\soundbox\linein\linein.c
2020/8/6 14:30:55 12,907 字节 C, C++, C#源代码 UTF-8 MIX

111 // linein stop
112 extern u8 linein_open_flag;
113
114 void linein_stop(void)
115 {
116     if (__this->onoff == 0) {
117         log_info("linein is already stop\n");
118         return;
119     }
120
121 #if (TCFG_LINEIN_INPUT_WAY == LINEIN_INPUT_WAY_ADC)
122     linein_dec_close();
123 #elif (TCFG_LINEIN_INPUT_WAY == LINEIN_INPUT_WAY_ANALOG)
124
125     if (TCFG_LINEIN_LR_CH & (BIT(0) | BIT(1))) {
126         audio_linein0_close(TCFG_LINEIN_LR_CH, 0);
127     } else if (TCFG_LINEIN_LR_CH & (BIT(2) | BIT(3))) {
128         audio_linein1_close(TCFG_LINEIN_LR_CH, 0);
129     } else if (TCFG_LINEIN_LR_CH & (BIT(4) | BIT(5))) {
130         audio_linein2_close(TCFG_LINEIN_LR_CH, 0);
131     }
132
133     linein_open_flag = 0;
134
135 #elif (TCFG_LINEIN_INPUT_WAY == LINEIN_INPUT_WAY_DAC)
136     audio_linein_via_dac_close(TCFG_LINEIN_LR_CH, 0);
137     linein_open_flag = 0;
138 #endif
139
140 app_audio_set_volume(APP_AUDIO_STATE_MUSIC, this->volume, 1);
141
142-51
```

```
D:\...\公版SDK\ac696n_soundbox_sdk_v0.2.4\SDK\apps\soundbox\linein\linein.c
2020/8/6 14:30:55 12,907 字节 C, C++, C#源代码 UTF-8 MIX
165 log_e("FM is not multiplexed with linein. channel selection err\n");
166 ASSERT(0, "err\n");
167 #else
168 linein_dec_open(TCFG_LINEIN_LR_CH, 44100);
169 #endif
170 #elif (TCFG_LINEIN_INPUT_WAY == LINEIN_INPUT_WAY_ANALOG)
171
172 linein_open_flag = 1;
173 if (TCFG_LINEIN_LR_CH & (BIT(0) | BIT(1))) {
174     audio_linein0_open(TCFG_LINEIN_LR_CH, 1);
175 } else if (TCFG_LINEIN_LR_CH & (BIT(2) | BIT(3))) {
176     audio_linein1_open(TCFG_LINEIN_LR_CH, 1);
177 } else if (TCFG_LINEIN_LR_CH & (BIT(4) | BIT(5))) {
178     audio_linein2_open(TCFG_LINEIN_LR_CH, 1);
179 }
180
181 audio_linein_gain(1); // high gain
182 #elif (TCFG_LINEIN_INPUT_WAY == LINEIN_INPUT_WAY_DAC)
183 linein_open_flag = 1;
184
185 if ((TCFG_LINEIN_LR_CH == AUDIO_LIN_DACL_CH) \
186     || (TCFG_LINEIN_LR_CH == AUDIO_LIN_DACR_CH)) {
187     audio_linein_via_dac_open(TCFG_LINEIN_LR_CH, 1);
188 } else {
189     ASSERT(0, "linein ch err\n");
190 }
191 linein_volume_set(app_audio_get_volume(APP_AUDIO_STATE_MUSIC));
192 #endif
193
```

```
D:\...公版SDK\ac696n_soundbox_sdk_v0.2.4\SDK\cpu\br25\audio_dec\audio_dec.c
2020/8/6 14:29:40 84,128 字节 C, C++, C#源代码 UTF-8 MIX

574 #endif
575
576     return wlen;
577 }
578
579 u8 linein_open_flag = 0;
580 u8 audio_output_flag = 0;
581
582 int audio_output_start(u32 sample_rate, u8 reset_rate)
583 {
584     if (reset_rate) {
585         app_audio_output_samplerate_set(sample_rate);
586     }
587
588     if(audio_output_flag)
589     {
590         return 0;
591     }
592
593     audio_output_flag = 1;
594
595     app_audio_output_start();
596
597 #if (defined(AUDIO_OUTPUT_AUTOMUTE) && (AUDIO_OUTPUT_AUTOMUTE == ENABLE))
598     audio_automute_onoff(automute, 1, 0);
599 #endif
600     return 0;
601 }
602
```

45.AC696 系列单独调节系统叠加提示音音量方法 202008015 WTS

直接对叠加音频的数据进行放大或缩小（红色是修改部分）：

```
D:\...公版SDK\ac696n_soundbox_sdk_v0.2.4\SDK\cpu\br25\audio_dec\tone_player.c
2020/8/6 15:29:00 51,040 字节 C, C++, C#源代码 UTF-8 MIX
1039 0x02, 0x00,
1040 0x00, 0x00,
1041 'f', 'a', 'c', 't', //f2id
1042 0x40, 0x00, 0x00, 0x00, //flen
1043 0xff, 0xff, 0xff, 0xff, //datalen
1044 'd', 'a', 't', 'a', //data
1045 0xff, 0xff, 0xff, 0xff, //sameple size
1046 };*/
1047
1048 static int sine_fread(struct audio_decoder *decoder, void *buf, u32 len)
1049 {
1050     int offset,i;
1051     // u8 *data = (u8 *)buf;
1052     s16 *data = (s16 *)buf;
1053
1054
1055     offset = sin_tone_make(sine_dec->sin_maker, data, len);
1056     sine_dec->sine_offset += offset;
1057
1058     for(i= 0;i<offset/2;i++)
1059     {
1060         data[i] = data[i]/5;
1061     }
1062
1063     return offset;
1064 }
1065
1066 static int sine fseek(struct audio decoder *decoder, u32 offset, int seek mode)
```

这里例子是除以5，音量调小五分之一。使用时自己根据情况调大或调小

46.AC696 系列直接在代码中修改叠加提示音方法 202008015 WTS

这里只拿其中一个提示音例子来讲：

```

board_config.h x tone_player.c x clock_manager.c x board_ac696x_demo_cfg.h x clock_cfg.h x audio_reverb.c x fm_ir
1095     default:
1096         return;
1097     }
1098 }
1099 }
1100
1101
1102 /*
1103  * 参数配置:
1104  * freq : 实际频率 * 512
1105  * points : 正弦波点数
1106  * win : 正弦窗
1107  * decay : 衰减系数(百分比), 正弦窗模式下为频率设置; 频率*512
1108  *
1109  */
1110 static const struct sin_param sine_16k_normal[] = {
1111     /*{0, 1000, 0, 100},*/
1112     {200 << 9 /*实际频率 * 512*/, 4000 /*正弦波点数*/, 0 /*正弦窗 */, 100 /*衰减系数(百分比)*/},
1113 };
1114
1115     频率           可以理解为播放时间长度
1116
1117
1118 #if CONFIG_USE_DEFAULT_SINE
1119 static const struct sin_param sine_tws_disconnect_16k[] = {
1120     /*
1121     {390 << 9, 4026, SINE_TOTAL_VOLUME / 4026},
1122     {262 << 9, 8000, SINE_TOTAL_VOLUME / 8000},
1123     */
1124     {262 << 9, 4026, 0, 100},
1125     {390 << 9, 8000, 0, 100},
1126 };
1127
1128 static const struct sin_param sine_tws_connect_16k[] = {

```

备注：基本修改的就只有频率和正弦波点数（播放时长），其他参数不要随意改。

如果想通过现有的一些提示音提取频率、持续点数信息，可以按以前“otp_AC691x_OTP 工具使用说明.pdf”文档操作：

4、提示音编辑

“提示音”选项主界面如下：

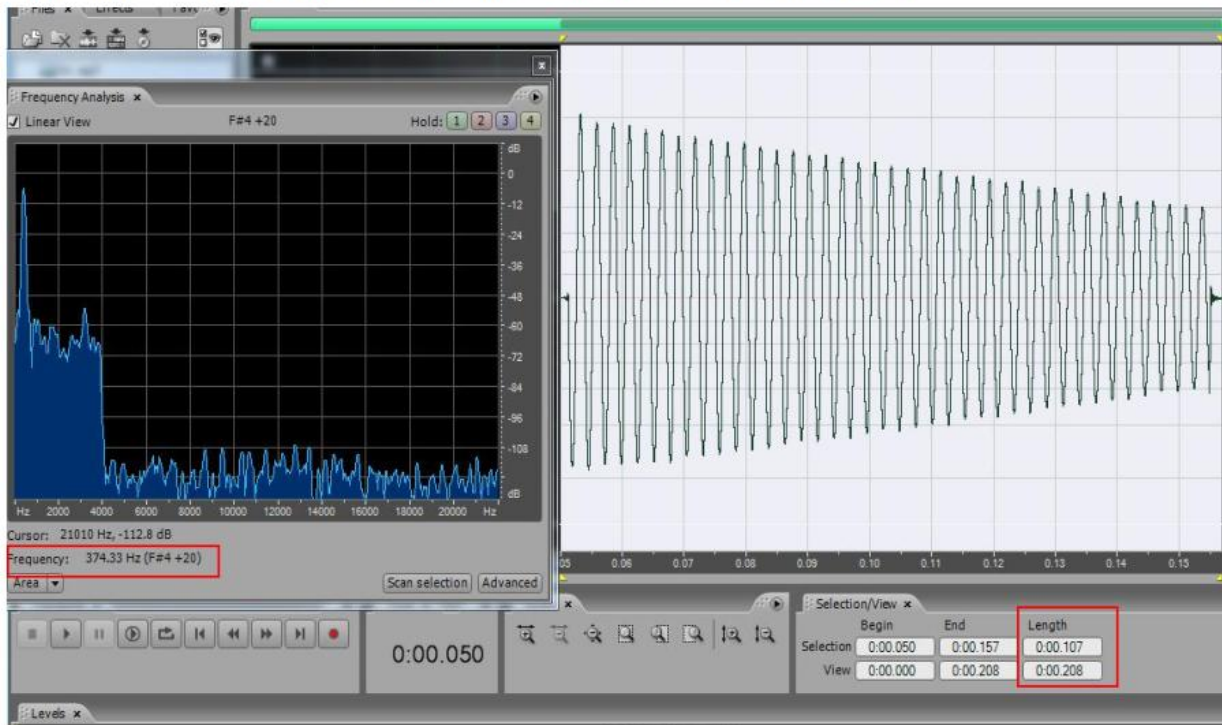
提示音支持两种修改方法：

1. 方法一 通过配置 配置提示音：需要的存储空间极小，这种方法制作提示音只能制作一些简单“中性提示音”
2. 方法二 WAV 提示音：提示音大小由 wav 音源决定，这种方法需要极大的资源

方法一配置提示音：

最多支持 8 个单音组成，每个单音可支持修改 频率、持续点数、类型、窗参数 这四种参数，如下图。

选择纯正的一枚自然数据，按快捷键 F4 就会弹出下图右边的窗口。



1) 频率：上图小窗口有个频率的参数 374, 374*512，这个就是需要填入第一个频率框的数据，512 是固定值。

2) 持续点数：右下角红色框的参数 Length:107 毫秒，107*48，这个参数就是持续点数，48 也是

“otp_AC691x_OTP 工具使用说明.pdf ” 可以在以下网盘路径获取：

链接：<https://pan.baidu.com/s/1E3aoxU0xKvZloOuk6mlivg>

提取码：bj0u

47.AC696 系列音箱 SDK2.5 判断录音录到哪个设备 202008015 WTS

可以通过 `rec->dev->dev_name` 判断：

```
key_event_deal.c x tone_player.h x audio_enc_recoder.c x audio_adc.h x audio_link.c x audio_de
552     }
553 }
554 //cut_head_time:单位 lms
555 //cut_tail_time:单位 lms
556 int record_player_encode_start(struct audio_fmt *f, struct storage_dev *dev, e
557 {
558     struct audio_fmt fmt;
559     u32 cut_tail_size = 0;
560
561     struct record_hdl *rec = NULL;
562
563     if (rec_hdl) {
564         record_player_encode_stop();
565     }
566
567     printf("struct record_hdl:%d ", sizeof(struct record_hdl));
568     rec = zalloc(sizeof(struct record_hdl));
569     ASSERT(rec);
570
571     rec->source = source;
572     rec->dev = dev;
573
574     printf("\n-----record dev = %s ----\n", rec->dev->dev_name);
575
576     if(!memcmp(rec->dev->dev_name, "sd0", 3))
577     {
578         puts("\n-----sd-----\n");
579     }
580     else
581     {
582         puts("\n-----usb-----\n");
583     }
584
585
586     if (f) {
587         memcpy((u8 *)&fmt, (u8 *)f, sizeof(struct audio_fmt));
588     } else {
589         printf("use default fmt====\n");
590         if (rec->source == ENCODE_SOURCE_MIC) {
591             fmt.channel = 1;
```

48.AC696 系列音箱 SDK2.5 快速按遥控有时不灵敏修改方法 202008015 WTS

可以通过适当减小 click_delay_time 参数。注意这个计数会影响双击、三击等多击的识别。如果遥控没有双击、三击等多击功能，调小这个参数没有影响。但如果有双击、三击等多击功能。调太小的话，会影响到多击功能识别（容易识别成单击），因此要跟进实际情况调节使用：

```
audio_enc_recoder.c x irkey.c x audio_adc.h x audio_link.c x audio_dec.c x audio_config.h x sine_make

1  #include "key_driver.h"
2  #include "irkey.h"
3  #include "gpio.h"
4  #include "asm/irflt.h"
5  #include "app_config.h"
6
7  #if TCFG_IRKEY_ENABLE
8
9  u8 ir_get_key_value(void);
10 //按键驱动扫描参数列表
11 struct key_driver_para irkey_scan_para = {
12     .scan_time      = 10,           //按键扫描频率, 单位: ms
13     .last_key       = NO_KEY,      //上一次get_value按键值, 初始化为NO_KEY;
14     .filter_time    = 2,           //按键消抖延时;
15     .long_time      = 75,          //按键判定长按数量
16     .hold_time      = (75 + 15),   //按键判定HOLD数量
17     .click_delay_time = 8, //20,   //按键被抬起后等待连击延时数量
18     .key_type       = KEY_DRIVER_TYPE_IR,
19     .get_value      = ir_get_key_value,
20 };
21
22
23
```

可以根据实际使用情况调小

49.AC696 系列如果误拿 ROM 中用到的 IO 口去做唤醒口，出现充电脚 5V 拔出后，耳机不会自动开机问题解决办法 202008020 WTS

开机时 ROM 中用到的 IO 口，在开机时电平会跳高或跳低，持续 50MS 左右。因此不要拿这些 IO 口去做唤醒脚或推灯。如何知道哪些 IO 口是开机 ROM 中用到的，请看 IC 用户手册中的 IO 口表：

(上拉)									
PB2 (高压)	PP2	ROM 后		SDOCMD_D		Q-decoder1_0			SPI1DIA
PB3 (超强驱动)	PP3	ROM 后	主要用于 FM 发射	SDODAT_D		Q-decoder1_1			
PB4	PF1	ROM 中		SDODAT_F	LVD	Q-decoder2_0	spi0_DAT2A(2)	SFC_DAT2A(2)	
PB5 (高压)	PP5	ROM 中	maskrom 串口升	SDODAT_B		Q-decoder2_1			SPI2DIA
PB6	PP6	ROM 后		SDOCMD_BF	AMUX1L	SDTAP_CLKD			SPI2CLKA
PB7	PP7	ROM 后		SDOCLK_BF	AMUX1R	SDTAP_DATD			SPI2DOA
PC1	PE1	ROM 中		PA_EN			spi0_CSB	SFC_CSB	SPI2DIB
PC2	PE2	ROM 中		LNA_EN			spi0_DIB(1)	SFC_DIB(1)	ALNK_MCLK
PC3	PE3	ROM 中		SDODAT_A			spi0_DAT2B(2)	SFC_DAT2B(2)	SPI1DIB
PC4	PF0	ROM 中		SDOCMD_A		SDTAP_CLKA	spi0_DAT3AB(3)	SFC_DAT3AB(3)	SPI1CLKB
PC5	PE5	ROM 后		SDOCLK_AE		SDTAP_DATA			SPI1DOB
PD0	PF4	ROM 中					spi0_CLKAB	SFC_CLKAB	
PD1	PF5	ROM 中					spi0_DOAB(0)	SFC_DOAB(0)	
PD2	PF3	ROM 中					spi0_DIA(1)	SFC_DIA(1)	

若误拿 ROM 中用到的 IO 口做了开机唤醒口，导致充电脚 5V 拔出后，耳机不会自动开机问题。可以参考以下修改处理：`(软件有开 “//是否支持拔出充电自动开机功能 #define TCFG_CHARGE_OFF_POWERON_NE ENABLE”)`

1、在 app_main.c 中添加函数：

```
AUDIO_DECODER_PROBE(sbc);  
AUDIO_DECODER_PROBE(msbc);  
AUDIO_DECODER_PROBE(cvsd);  
#ifndef CONFIG_LITE_DECODER  
    AUDIO_DECODER_PROBE(mty);  
#endif  
#if TCFG_BT_SUPPORT_AAC  
    AUDIO_DECODER_PROBE(aac);  
#endif  
  
/*编码器*/  
AUDIO_ENCODER_PROBE(msbc);  
AUDIO_ENCODER_PROBE(cvsd);  
}
```

新增函数

```
u8 is_ldo5v_wakeup_remap(void)  
{  
    ///1、 如果为检测的是拔出唤醒,且有对应唤醒源,认为是真的拔出唤醒  
    ///此时bug为不响应快速插入拔出充电线,认为是按键唤醒,走按键开机流程  
    if (get_charge_wkup_type() == WKUP_FAIL) {  
        if (get_wakeup_source() & BIT(3)) {  
            return 1;  
        }  
    }  
    return 0;  
}
```

```
void app_main()  
{
```

```
    int update = 0;  
    u32 addr = 0, size = 0;  
    struct intent it;
```

```
    log_info("app_main");
```

```
u8 is_ldo5v_wakeup_remap(void)  
{  
    ///1、 如果为检测的是拔出唤醒,且有对应唤醒源,认为是真的拔出唤醒  
    ///此时 bug 为不响应快速插入拔出充电线,认为是按键唤醒,走按键开机流程  
    if (get_charge_wkup_type() == WKUP_FAIL) {  
        if (get_wakeup_source() & BIT(3)) {  
            return 1;  
        }  
    }  
    return 0;  
}
```

2、在 app_main.c 文件的 app_main()函数中，将 is_ldo5v_wakeup()改为 is_ldo5v_wakeup_remap():

```
        if (get_charge_online_flag()) {
            set_charge_idle_flag(0);
#ifdef TCFG_SYS_LVD_EN == 1
            vbat_check_init();
#endif
            init_intent(&it);
            it.name = "idle";
            it.action = ACTION_IDLE_MAIN;
            start_app(&it);
        } else {
            check_power_on_voltage();

#ifdef TCFG_POWER_ON_NEED_KEY
            if (CONFIG_BT_MODE == BT_NORMAL)
                /*充电拔出,CPU软件复位, 不检测按键, 直接开机*/
#ifdef TCFG_CHARGE_OFF_POWERON_NE
            if ((!update && cpu_reset_by_soft()) || is_ldo5v_wakeup_remap()) {
                //printf("----11111-----update = %d  cpu_reset_by_soft = %d  is_ldo5v_wakeup_remap = %d\n",
                update, cpu_reset_by_soft, is_ldo5v_wakeup_remap);

            } else
                if (!update && cpu_reset_by_soft()) {
#ifdef TCFG_CHARGE_OFF_POWERON_NE
                if (!update && cpu_reset_by_soft()) {
                    app_var.play_poweron_tone = 0;
                } else {
                    //printf("----22222-----update = %d  cpu_reset_by_soft = %d  is_ldo5v_wakeup_remap = %d\n",
                    update, cpu_reset_by_soft, is_ldo5v_wakeup_remap);
                    check_power_on_key();
                }
            }
#endif
        }
#endif
#endif

        ui_manage_init();
        ui_update_status(STATUS_POWERON);

        init_intent(&it);
        it.name = "earphone";
        it.action = ACTION_EARPHONE_MAIN;
        start_app(&it);
    }
}
```

3、在 charge.c 中做以下修改:

```
#define LOG_ERROR_ENABLE
#define LOG_DEBUG_ENABLE
#include "debug.h"

#define IO_PORT_LDOIN 58//IO编号

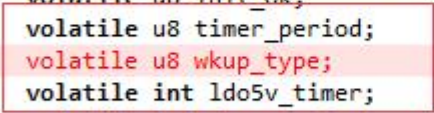
#define BIT_LD05V_IN BIT(0)
#define BIT_LD05V_OFF BIT(1)
#define BIT_LD05V_ERR BIT(2)

enum {
    CHARGE_TIMER_10_MS = 0,
    CHARGE_TIMER_10_S,
} CHARGE_CHECK_TIME;

typedef struct CHARGE_VAR {
    struct charge_platform_data *data;
    volatile u8 charge_online_flag;
    volatile u8 init_ok;
    volatile u8 timer_period;
    volatile u8 wkup_type;
    volatile int ldo5v_timer;
    volatile int charge_timer;
    volatile int charge_timer_pre;
} CHARGE_VAR;

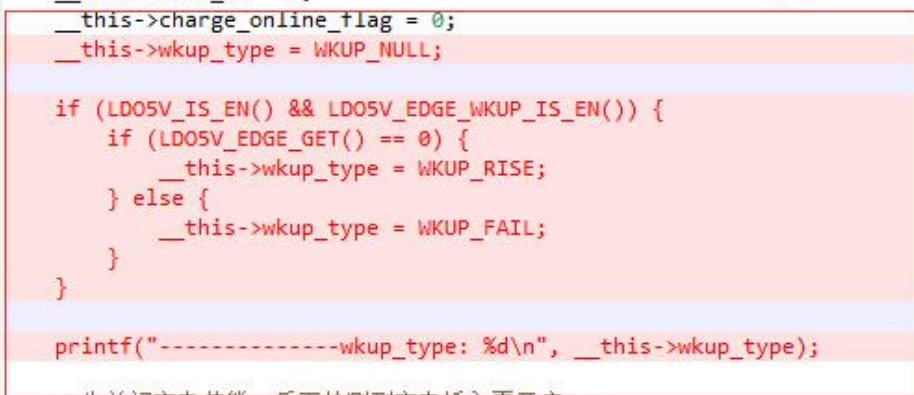
#define __this (&charge_var)
static CHARGE_VAR charge_var;
static u8 charge_flag;
extern void charge_set_callback(void (*wakup_callback)(void), void (*sub_callback)(
```

新增变量



```
}  
  
log_info("charge_full_v_val = %d\n", charge_full_v_val);  
  
CHARGE_FULL_V_SEL(charge_full_v_val);  
CHARGE_FULL_MA_SEL(__this->data->charge_full_ma);  
/* CHARGE_MA_SEL(__this->data->charge_ma); */  
CHARGE_MA_SEL(CHARGE_MA_20);  
}  
  
static int charge_init(const struct dev_node *node, void *arg)  
{  
    log_info("%s\n", __func__);  
  
    __this->data = (struct charge_platform_data *)arg;  
  
    ASSERT(__this->data);  
    __this->init_ok = 0;  
  
    __this->charge_online_flag = 0;  
    __this->wkup_type = WKUP_NULL;  
  
    if (LDO5V_IS_EN() && LDO5V_EDGE_WKUP_IS_EN()) {  
        if (LDO5V_EDGE_GET() == 0) {  
            __this->wkup_type = WKUP_RISE;  
        } else {  
            __this->wkup_type = WKUP_FAIL;  
        }  
    }  
  
    printf("-----wkup_type: %d\n", __this->wkup_type);  
  
    /*先关闭充电使能，后面检测到充电插入再开启*/  
    /* CHGBG_EN(0); */  
    /* CHARGE_EN(0); */  
  
    /*LDO5V的100K下拉电阻使能*/  
    L5V_LOAD_EN(__this->data->l5v_pulldown_en);  
  
    /*LDO5V,检测上升沿，用于检测ldoin插入*/  
    LDO5V_EN(1);  
    LDO5V_EDGE_SEL(0);  
}
```

增加以下内容



修改内容部分:

```
__this->wkup_type = WKUP_NULL;  
  
if (LDO5V_IS_EN() && LDO5V_EDGE_WKUP_IS_EN()) {  
    if (LDO5V_EDGE_GET() == 0) {  
        __this->wkup_type = WKUP_RISE;  
    } else {  
        __this->wkup_type = WKUP_FAIL;  
    }  
}  
}
```

```
if (check_charge_state()) {  
    if (__this->ldo5v_timer == 0) {  
        __this->ldo5v_timer = sys_hi_timer_add(0, ldo5v_detect, 2);  
    }  
} else {  
    CHARGE_WKUP_PND_CLR();  
    CHGBG_EN(0);  
    CHARGE_EN(0);  
}  
charge_set_callback(charge_wakeup_isr, sub_wakeup_isr);  
/* sys_hi_timer_add(0, test_func, 1); */  
  
__this->init_ok = 1;  
  
return 0;  
}
```

新增函数

```
u8 get_charge_wkup_type(void)  
{  
    return __this->wkup_type;  
}
```

```
const struct device_operations charge_dev_ops = {  
    .init = charge_init,  
};
```

```
u8 get_charge_wkup_type(void)  
{  
    return __this->wkup_type;  
}
```

4、在 charge.h 文件中，做以下修改：

```
CHARGE_EVENT_CHARGE_START,  
CHARGE_EVENT_CHARGE_CLOSE,  
CHARGE_EVENT_CHARGE_FULL,  
CHARGE_EVENT_CHARGE_ERR,  
CHARGE_EVENT_LD05V_IN,  
CHARGE_EVENT_LD05V_OFF,  
CHARGE_EVENT_USB_CHARGE_IN,  
CHARGE_EVENT_USB_CHARGE_OFF,  
};  
  
enum {  
    WKUP_NULL,  
    WKUP_RISE,  
    WKUP_FAIL,  
};  
  
extern void set_charge_online_flag(u8 flag);  
extern u8 get_charge_online_flag(void);  
extern u8 get_charge_poweron_en(void);  
extern void charge_start(void);  
extern void charge_close(void);  
extern u8 get_charge_mA_config(void);  
extern void set_charge_mA(u8 charge_mA);  
extern u8 get_ldo5v_pulldown_en(void);  
extern u8 get_ldo5v_online_hw(void);  
extern u8 get_lvcmp_det(void);  
extern void charge_check_and_set_pinr(u8 mode);  
extern u16 get_charge_full_value(void);  
extern const struct device_operations charge_dev_ops;  
extern u8 get_charge_wkup_type(void);  
  
#endif    // _CHARGE_H_
```

新增内容

```
enum {  
    WKUP_NULL,  
    WKUP_RISE,  
    WKUP_FAIL,  
};
```

```
extern u8 get_charge_wkup_type(void);
```

50.AC696X 音箱 SDK 在对箱时红外按键处理 2020 0822 LHY

当音箱中拥有红外按键控制时，对箱时会出现两种情况，假设现在有音箱 A 和 B

第一种是 A、B 同时接受到红外按键消息并同步给另外一边音箱：需要在两个音箱中过滤掉来自对箱消息。
第二种是 A 或者 B 只有一个接受到红外按键消息，这个时候没有接受到按键消息的需要执行来自对箱消息。
实现方法：

加入下图代码，主要是通过获取系统时间来判断是否需要执行对箱消息来实现。

```
C app_charge.c C key_event_deal.c X
SDK > apps > soundbox > bt > C key_event_deal.c > app_earphone_key_event_handler(sys_event *)
335
336 #if (TCFG_SPI_LCD_ENABLE)
337     extern int key_is_ui_takeover();
338     if (key_is_ui_takeover()) {
339         return false;
340     }
341 #endif
342 //r_printf("timer_get_ms == %d",timer_get_ms());
343 if(get_bt_tws_connect_status())&&(key->type == KEY_DRIVER_TYPE_IR)){
344
345     if((u32)event->arg == KEY_EVENT_FROM_TWS){
346         sys_time_tws = timer_get_ms();
347         if(sys_time_self < sys_time_tws){
348             if((sys_time_tws-sys_time_self) < 100){
349                 r_printf("IR MSG ignore\n");
350                 return ret;
351             }
352         }else{
353             if(((0xFFFFFFFF-sys_time_self)+sys_time_tws) < 100){
354                 r_printf("IR MSG ignore\n");
355                 return ret;
356             }
357         }
358     }else{
359         sys_time_self = timer_get_ms();
360     }
361 }
362 u8 key_event = bt_key_event_get(key);//key_table[key->value][key->event];
```

如果效果不佳可以调大或者调小这个值

51. AC696X 音箱 SDK 低电检测使用复位脚问题 LHY 2020 0822

在电量检测时候，一般都会使用分压电路(例如上下拉 9.1k 电阻)来采集电压，会导致引脚电压在 1.6-2.0V 之间，尤其在 1.6-1.8V 时候由于处于高低电平无法识别的区间，这时候如果使用复位脚作为检测脚会导致异常复位，建议关闭检测脚复位功能，或者提高电压变化区间。

52. AC696X 耳机 SDK016 面条耳机，把按键消息配置在工具里面，按键的消息会错位 XNW 20200824

处理方法：删掉下图的按键事件

```
45 #define SYS_NET_EVENT      0x08
46 #define SYS_BT_EVENT      0x10
47 #define SYS_IR_EVENT      0x20
48 #define SYS_PBG_EVENT     0x40
49
50 enum {
51     KEY_EVENT_CLICK,
52     KEY_EVENT_LONG,
53     KEY_EVENT_HOLD,
54     KEY_EVENT_UP,
55     KEY_EVENT_DOUBLE_CLICK,
56     KEY_EVENT_TRIPLE_CLICK,
57     //KEY_EVENT_SIX_CLICK,
58     KEY_EVENT_MAX,
59 };
60
61
62 enum {
63     DEVICE_EVENT_IN,
64     DEVICE_EVENT_OUT,
65     DEVICE_EVENT_ONLINE,
66     DEVICE_EVENT_OFFLINE,
67     DEVICE_EVENT_CHANGE,
68 };
```

53. AC696X 音箱 SDK025 使用外部 eq 配置文件后程序中的 6 个 eq 效果不起作用 SQ 20200825

- 主要是把配置文件的 eq 数据替换掉其中一个效果，以替换到 eq_tab_normal 为例
- 注意 EQTool V3.1.6 工具默认第十个频点配置为 20329 此频点需要改小不然程序会报错



```

download.bat X  C audio_eq.c  C audio_drc.c  C eq_config.c  C hw_eq_table.h  C rsp_bs_api.c  C init.h  C audio_eq_tab.h  C hw_eq.h  C spinloc
cpu > br25 > tools > download.bat
48 ::copy sbc_dec.bin .\res\
49 ::copy mp3_dec_lib.bin .\res\
50 ::copy aac_dec_dsp.bin .\res\
51 ::copy eq_bflt .\res\
52 ::copy commproc.bflt .\res\
53 ::copy config.cfg .\res\
54 ::copy convert.bflt .\res\
55
56
57
58
59 | isd_download.exe -tonorflash -dev br25 -boot 0x12000 -div8 -wait 300 -uboot uboot.boot -app app.bin cfg_tool.bin -res tone.cf; eq_cfg_hw.bin -uboot_compre
60 :: -format all
61 :: -reboot 2500
62
    
```

在download.bat中添加配置文件

```
C eq_config.c X C hw_eq_table.h C rcsp_bs_api.c C init.h C audio_eq_tab.h
cpu > br25 > audio_effect > C eq_config.c > eq_cfg_open(void)
1050 void eq_cfg_open(void)
1051 {
1052 > if (p_eq_cfg == NULL) {...
1058 memset(p_eq_cfg, 0, sizeof(EQ_CFG));
1059 > for (int ch = 0; ch < EQ_CH_NUM; ch++) {...
1062
1063 spin_lock_init(&p_eq_cfg->lock);
1064
1065 #if TCFG_EQ_FILE_ENABLE
1066 p_eq_cfg->eq_type = EQ_TYPE_FILE;
1067 for (int ch = 0; ch < EQ_CH_NUM; ch++) {
1068 p_eq_cfg->cur_sr[ch] = 44100;
1069 }
1070 eq_file_get_cfg(p_eq_cfg);
1071 // if (eq_file_get_cfg(p_eq_cfg)) //获取EQ文件失败
1072 #endif
1073 {
1074 p_eq_cfg->eq_type = EQ_TYPE_MODE_TAB;
1075 if (EQ_SECTION_MAX > 10) {
1076 p_eq_cfg->seg_num = 10;
1077 } else {
1078 p_eq_cfg->seg_num = EQ_SECTION_MAX;
1079 }
1080
1081 for (int i = 0; i < EQ_CH_NUM; i++) {
1082 p_eq_cfg->param[i].global_gain = 0;
1083 }
1084 }
```

memcpy((void *)eq_tab_normal, eq_cfg->param[eq_ch].seg, sizeof(EQ_CFG_SEG) * EQ_SECTION_MAX);

```
audio_eq_tab.h - SDK - Visual Studio Code
download.bat C audio_eq.c C audio_drc.c C eq_config.c C hw_eq_table.h C rcsp_bs_api.c C init.h C audio_eq_tab.h X C
cpu > br25 > audio_effect > C audio_eq_tab.h > eq_tab_normal
5 #include "app_config.h"
6 #include "audio_eq.h"
7
8 #include "user_fun_config.h"
9
10 #if (TCFG_EQ_ENABLE == 1)
11 // static const struct eq_seg_info eq_tab_normal[EQ_SECTION_MAX] = {
12 static struct eq_seg_info eq_tab_normal[EQ_SECTION_MAX] = {
13 {0, EQ_IIR_TYPE_BAND_PASS, 31, 0 << 20, (int)(0.7f * (1 << 24))},
14 {1, EQ_IIR_TYPE_BAND_PASS, 62, 0 << 20, (int)(0.7f * (1 << 24))},
15 {2, EQ_IIR_TYPE_BAND_PASS, 125, 0 << 20, (int)(0.7f * (1 << 24))},
16 {3, EQ_IIR_TYPE_BAND_PASS, 250, 0 << 20, (int)(0.7f * (1 << 24))},
17 {4, EQ_IIR_TYPE_BAND_PASS, 500, 0 << 20, (int)(0.7f * (1 << 24))},
18 {5, EQ_IIR_TYPE_BAND_PASS, 1000, 0 << 20, (int)(0.7f * (1 << 24))},
19 {6, EQ_IIR_TYPE_BAND_PASS, 2000, 0 << 20, (int)(0.7f * (1 << 24))},
20 {7, EQ_IIR_TYPE_BAND_PASS, 4000, 0 << 20, (int)(0.7f * (1 << 24))},
```

```

cpu > br25 > audio_effect > C eq_config.c > eq_file_get_cfg(EQ_CFG *)
291     if (file_data == NULL) {
292         ret = -ENOMEM;
293         break;
294     }
295
296     if (sizeof(EQ_FILE_SEG_HEAD) != fread(file, file_data, sizeof(EQ_FILE_SEG_HEAD))) {
297         ret = -EIO;
298         break;
299     }
300     eq_file_h = (EQ_FILE_SEG_HEAD *)file_data;
301     log_info("cfg_seg_num:%d\n", eq_file_h->seg_num);
302     log_info("cfg_global_gain:%d\n", (u32)(eq_file_h->global_gain));
303     log_info("cfg_enable_section:%x\n", eq_file_h->enable_section);
304
305     if (eq_file_h->seg_num > EQ_SECTION_MAX) {
306         ret = -EINVAL;
307         break;
308     }
309
310     read_size = eq_file_h->seg_num * sizeof(EQ_CFG_SEG);
311     if (read_size != fread(file, file_data + sizeof(EQ_FILE_SEG_HEAD), read_size)) {
312         ret = -EIO;
313         break;
314     }
315
316     if (eq_file_h->crc == crc_get_16bit(&eq_file_h->seg_num, read_size + sizeof(EQ_FILE_SEG_HEAD) - 2)) {
317         log_info("eq_cfg_file crc ok\n");
318         spin_lock(&eq_cfg->lock);
319         eq_cfg->seg_num = eq_file_h->seg_num;
320         eq_cfg->param[eq_ch].global_gain = eq_file_h->global_gain;
321         memcpy(eq_cfg->param[eq_ch].seg, file_data + sizeof(EQ_FILE_SEG_HEAD), sizeof(EQ_CFG_SEG) * eq_file_h->seg_num);
322         memcpy((void *)eq_tab_normal, eq_cfg->param[eq_ch].seg, sizeof(EQ_CFG_SEG) * EQ_SECTION_MAX);
323
324         eq_cfg->design_mask[eq_ch] = (u32) - 1;
325

```

54. AC696X 音箱 SDK025 程序中实时修改 eq 某频点增益 SQ 20200825

- 主要步骤：设置 eq mode 更新标志位->修改频点增益。
- 此方法适用修改程序内部 6 个 eq 效果的实时修改。
- 如果使用此方法修改外部配置文件频点增益，需要把外部数据替换到 eq 效果表中或者自行添加一个表，参照 53 点的修改。

以修改 1 和 10 频点为例



在触发更新 eq 数据的地方调用

```
void user_eq_updata(int bass,int terble){
    if(user_bass_vol == bass && user_terble_vol == terble){
        return;
    }

    user_bass_vol = bass;
    user_terble_vol = terble;

    user_eq_updata_flag = 1;
    if(p_eq_cfg){
        p_eq_cfg->mode_updata = 1;
    }
}
```

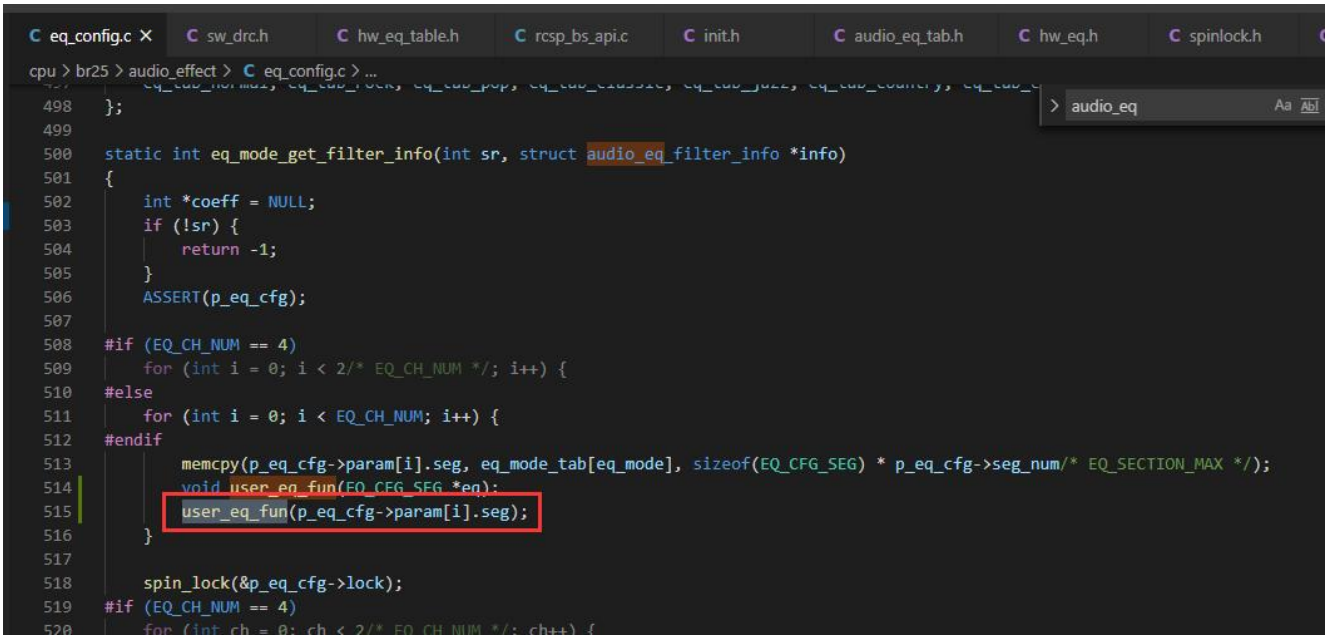
频点1增益

频点10增益

```
119
120 #define USER_BASS_POINT 0//eq配置文件的第x个频点
121 #define USER_TERBLE_POINT 9//eq配置文件的第x个频点
122
123 static int user_bass_vol = 0;
124 static int user_terble_vol = 0;
125 static bool user_eq_updata_flag = 0;
126 void user_eq_fun(EQ_CFG_SEG *eq){
127     if(!user_eq_updata_flag || !eq){
128         return;
129     }
130
131     printf("user eq updata run %d\n",eq_mode_get_cur());
132
133     #if (defined(USER_BASS_POINT) && (USER_BASS_POINT<EQ_SECTION_MAX))
134     if(user_eq_updata_flag){
135         eq[USER_BASS_POINT].gain = user_bass_vol<<20;
136         printf(" bass updata freq %d gain %d\n",eq[USER_BASS_POINT].freq,eq[USER_BASS_POINT].gain>>20);
137     }
138     #endif
139
140     #if (defined(USER_TERBLE_POINT) && (USER_TERBLE_POINT<EQ_SECTION_MAX))
141     if(user_eq_updata_flag){
142         eq[USER_TERBLE_POINT].gain = user_terble_vol<<20;
143         printf(" terble updata freq %d gain %d\n",eq[USER_TERBLE_POINT].freq,eq[USER_TERBLE_POINT].gain>>20);
144     }
145     #endif
146     user_eq_info_print(eq);
147     user_eq_updata_flag = 0;
148 }
149
```

配置工具里面第1个频点对应数据中的第0点
配置工具里面第10个频点对应数据中的第9点

在 eq mode 更新的回调函数中添加



```
cpu > br25 > audio_effect > C eq_config.c > ...
498 };
499
500 static int eq_mode_get_filter_info(int sr, struct audio_eq_filter_info *info)
501 {
502     int *coeff = NULL;
503     if (!sr) {
504         return -1;
505     }
506     ASSERT(p_eq_cfg);
507
508     #if (EQ_CH_NUM == 4)
509         for (int i = 0; i < 2/* EQ_CH_NUM */; i++) {
510     #else
511         for (int i = 0; i < EQ_CH_NUM; i++) {
512     #endif
513         memcpy(p_eq_cfg->param[i].seg, eq_mode_tab[eq_mode], sizeof(EQ_CFG_SEG) * p_eq_cfg->seg_num/* EQ_SECTION_MAX */);
514         void user_eq_fun(EQ_CFG_SEG *eq):
515         user_eq_fun(p_eq_cfg->param[i].seg);
516     }
517
518     spin_lock(&p_eq_cfg->lock);
519     #if (EQ_CH_NUM == 4)
520     for (int ch = 0; ch < 2/* EQ_CH_NUM */; ch++) {
```

以上截图所有程序修改代码如下：

#define USER_BASS_POINT 0//eq 配置文件的第 x 个频点

#define USER_TERBLE_POINT 9//eq 配置文件的第 x 个频点

```
static int user_bass_vol = 0;
```

```
static int user_terble_vol = 0;
```

```
static bool user_eq_updata_flag = 0;
```

```
void user_eq_fun(EQ_CFG_SEG *eq){
```

```
    if(!user_eq_updata_flag || !eq){
```

```
        return;
```

```
    }
```

```
    printf("user eq updata run %d\n",eq_mode_get_cur());
```

```
#if (defined(USER_BASS_POINT) && (USER_BASS_POINT<EQ_SECTION_MAX))
```

```
    if(user_eq_updata_flag){
```

```
        eq[USER_BASS_POINT].gain = user_bass_vol<<20;
```

```
        printf("                bass                updata                freq                %d
```

```
gain %d\n",eq[USER_BASS_POINT].freq,eq[USER_BASS_POINT].gain>>20);
```

```
    }
```

```
#endif
```

```
#if (defined(USER_TERBLE_POINT) && (USER_TERBLE_POINT<EQ_SECTION_MAX))
```

```
    if(user_eq_updata_flag){
```

```
        eq[USER_TERBLE_POINT].gain = user_terble_vol<<20;
```

```
        printf("                terble                updata                freq                %d
```

```
gain %d\n",eq[USER_TERBLE_POINT].freq,eq[USER_TERBLE_POINT].gain>>20);
```

```
    }
```

```
#endif
    user_eq_updata_flag = 0;
}

void user_eq_updata(int bass,int terble){
    if(user_bass_vol == bass && user_terble_vol == terble){
        return;
    }

    user_bass_vol = bass;
    user_terble_vol = terble;

    user_eq_updata_flag = 1;
    if(p_eq_cfg){
        p_eq_cfg->mode_updata = 1;
    }
}
```

55. AC696X ac696n_sdk_release_v0.1.6_2M 包加入来电铃声和来电报号的问题 YP

/* ***** 2M 包来电报号和铃声 ***** */

(wtg 提示音) 开机、关机、蓝牙连接、蓝牙断开、报号
(正弦波) 对耳连接、对耳断开、来电铃声

- 1、tone.cfg 文件小于等于 13K(当然越小越好)。
- 2、不支持 EQ。
- 3、不支持 ble。
- 4、格式只支持 wtg(flash 空间限制)。

参考的代码如下：链接：<https://pan.baidu.com/s/1035jB3MRmILYDoQ7YgbGig>

提取码：11bm

因为默认的 SDK 的来电号码和来电提示音要求是同样的提示音格式的，现在为了添加来电报号和来电铃声的功能，将来电铃声换成 wtg 格式，入参考代码的修改方式：

2020/8/20 11:41:43 56,278 字节 C、C++、C# 源代码 UTF-8 MIX

```

u8 phone_num_flag_test = 0;
static void number_to_play_list(char *num, u32 *lst)
{
    u8 i = 0;
    if (num) {
        for (i < strlen(num); i++) {
            lst[i] = num_9[num[i] - '0'];
        }
    }
    lst[i++] = (u32)NULL;
    phone_num_flag_test = 1;
    // lst[i++] = (u32)TONE_REPEAT_BEGIN(-1);
    // lst[i++] = (u32)TONE_REPEAT_BEGIN(-1);
    // lst[i++] = (u32)TONE_REPEAT_END();
    // lst[i++] = (u32)TONE_REPEAT_END();
    // lst[i++] = (u32)TONE_REPEAT_END();
}

static void ring_to_play_list(char *num, u32 *lst)
{
    u8 i = 0;
    lst[i++] = (u32)TONE_REPEAT_BEGIN(-1);
    lst[i++] = (u32)TONE_REPEAT_BEGIN(-1);
    lst[i++] = (u32)TONE_REPEAT_END();
    lst[i++] = (u32)TONE_REPEAT_END();
    lst[i++] = (u32)TONE_REPEAT_END();
}

int phone_ring_play_start_test(void)
{
    if (get_call_status() == BT_CALL_HANDUP) {
        log_info("hangup, --phone ring play return\n");
        return 0;
    }
    log_info("%s\n", __FUNCTION__);
    /* check if support inband ringtone */
    if (lbt_user_priv_var.inband_ringtone) {
        u32 *len_list = malloc(4 * 34);
        ring_to_play_list(NULL, len_list);
        tone_file_list_play((const char **)len_list);
        return 1;
    }
    return 0;
}

void phone_num_play_timer(void *priv)
{
    if (get_call_status() == BT_CALL_HANDUP) {

```

添加来电报号的标志记录

报号时将标志置位，同时不要将铃声放进数组内容中

添加一个播放来电铃声的函数

2020/7/20 8:51:04 55,489 字节 C、C++、C# 源代码 UTF-8 UNIX

```

static void number_to_play_list(char *num, u32 *lst)
{
    u8 i = 0;
    if (num) {
        for (i < strlen(num); i++) {
            lst[i] = num_9[num[i] - '0'];
        }
    }
}

lst[i++] = (u32)TONE_REPEAT_BEGIN(-1);
lst[i++] = (u32)TONE_REPEAT_BEGIN(-1);
lst[i++] = (u32)TONE_REPEAT_END();
lst[i++] = (u32)TONE_REPEAT_END();
lst[i++] = (u32)TONE_REPEAT_END();

void phone_num_play_timer(void *priv)
{
    if (get_call_status() == BT_CALL_HANDUP) {

```

然后在播放提示音结束的地方判断是否需要播放来电铃声

话(S) 文件(F) 编辑(E) 搜索(R) 视图(V) 工具(T) 帮助(H)

ac696n_sdk_release_v0.1.6 earphone.c tone_player.c

H:\...AC69\AC696X\功能需求\2M包来电报号和铃声\2M包来电报号和铃声\ac696n_sdk_release_v0.1.6\sd\apps\common\tone_player.c

2020/8/20 11:39:57 22,284 字节 C、C++、C# 源代码 UTF-8 MIX

```

switch (argv[0]) {
case AUDIO_PLAY_EVENT_END:
    log_info("tone player end\n");
    ASSERT(argv[1] == __this->file);
    tone_stop();
    break;
case AUDIO_PLAY_EVENT_ERR:
    log_error("tone player err %d\n", argv[1]);
    break;
default:
    break;
}
tone_event_to_user(argv[0]);
}
#endif
static void dec_event_tone_stop(u32 magic)
{
    if (__this->magic == magic) {
        tone_file_list_stop();
    }
}

extern u8 phone_num_flag_test;
extern int phone_ring_play_start_test(void);
if (phone_num_flag_test) {
    phone_num_flag_test = 0;
    phone_ring_play_start_test();
}

static void tone_list_dec_event_handler(void *priv, int argc, int *argv)
{
    switch (argv[0]) {
case AUDIO_PLAY_EVENT_END:
    log_info("tone list player end\n");
    dec event tone stop((u32)priv);

```

56. AC696X 程序上设置自定义蓝牙名【通用】 LZK

- 普通模式下蓝牙名（默认使用配置文件的蓝牙名）

```
const char *bt_get_local_name()
{
    return (const char *)(bt_cfg.edr_name);
}
```

- 独立自拍模式下的蓝牙名设置（该函数需要自行写入代码）

```
const char *bt_get_hid_name()
{
    return "JL_HID_DEFAULT";
}
```

- 动态修改蓝牙名

```
int bt_modify_name(u8 *new_name)
{
    u8 new_len = strlen(new_name);
    if (new_len >= LOCAL_NAME_LEN) {
        new_name[LOCAL_NAME_LEN - 1] = 0;
    }
    if (strcmp(new_name, bt_cfg.edr_name)) {
        syscfg_write(CFG_BT_NAME, new_name, LOCAL_NAME_LEN);
        memcpy(bt_cfg.edr_name, new_name, LOCAL_NAME_LEN);
        hci_vendor_update_name();
        log_info("mdy_name sucess\n");
        return 1;
    }
    return 0;
}
```

- BLE 广播名字

搜索程序上 `HCI_EIR_DATATYPE_COMPLETE_LOCAL_NAME` 字样，找到数据部分的传参就是实际的蓝牙名。根据实际需要修改指针指向的内容。

注意：包数据最长只有 31 字节

```
37
38 static int make_set_rsp_data(void)
39 {
40     u8 offset = 0;
41     u8 *buf = scan_rsp_data;
42
43     #if RCSP_BTMMATE_EN
44     u8 tag_len = sizeof(user_tag_string);
45     offset += make_eir_packet_data(&buf[offset], offset, HCI_EIR_DATATYPE_MANUFACTURER_SPECIFIC_DATA, (void *)user_tag_string, tag_len);
46     #endif
47
48     u8 name_len = gap_device_name_len;
49     u8 vaild_len = ADV_RSP_PACKET_MAX - (offset + 2);
50     if (name_len > vaild_len) {
51         name_len = vaild_len;
52     }
53     offset += make_eir_packet_data(&buf[offset], offset, HCI_EIR_DATATYPE_COMPLETE_LOCAL_NAME, (void *)gap_device_name, name_len);
54
55     if (offset > ADV_RSP_PACKET_MAX) {
56         puts("***rsp_data overflow!!!!\n");
57         return -1;
58     }
59
60     log_info("rsp_data(%d):", offset);
61     log_info_hexdump(buf, offset);
62     scan_rsp_data_len = offset;
63     ble_op_set_rsp_data(offset, buf);
64     return 0;
65 }
```

总数据长度不能大于31

名字

名字长度

57. 025 版本带混响功能有概率死机 20200910 xin

链接: <https://pan.baidu.com/s/1TJYHDZSh68HEv4ahxkkc-A> 提取码: pnsj

提取 media.a 库, rebuild 即可

58.AC696 的 V025 版本 SDK 进入蓝牙会自动退出蓝牙模式问题 20200911 YWP

bt_auto_exit();函数会计算到没有连接或者其他条件成立会自动退出蓝牙模式。

```

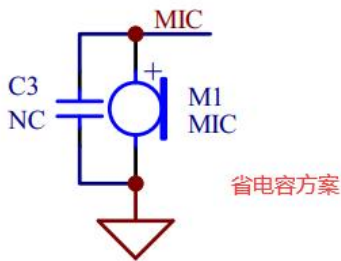
00374: }
00375:
00376: int bt_wait_connect_and_phone_connect_switch(void *p)
00377: {
00378:     int ret = 0;
00379:     int timeout = 0;
00380:     s32 wait_timeout;
00381:
00382:     if (__this->exiting) {
00383:         return 0;
00384:     }
00385:
00386:     if (get_remote_test_flag()) {
00387:         return 0;
00388:     }
00389:
00390:
00391:     log_info("connect_switch: %d, %d\n", (int)p, bt_user_priv_var.auto_connection_counter);
00392:
00393:     if (bt_user_priv_var.auto_connection_counter <= 0) {
00394:         bt_user_priv_var.auto_connection_timer = 0;
00395:         bt_user_priv_var.auto_connection_counter = 0;
00396:         user_send_cmd_prepare(USER_CTRL_PAGE_CANCEL, 0, NULL);
00397:         if (get_current_poweron_memory_search_index(NULL)) {
00398:             bt_init_ok_search_index();
00399:             return bt_wait_connect_and_phone_connect_switch(0);
00400:         } else {
00401:             bt_wait_phone_connect_control(1);
00402:             // bt_auto_exit();
00403:             return 0;
00404:         }
00405:     }
00406:     /* log_info(">>>phone_connect_switch=%d\n",bt_user_priv_var.auto_connection_counter ); */
00407:     if ((int)p == 0) {

```

这个函数注释掉

59.AC696 系列外插 MIC 做混响或扩音设计注意 20200915 YWP

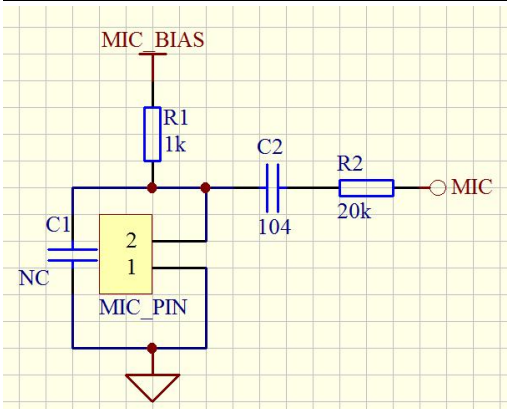
目前我们 AC696 系列给出的原理图，MIC 部分电路都是省电容的电路（如下图），如果客户用作带有混响或者扩音的功能，不建议使用省电容的电路，建议需加隔直电容，偏置用芯片的 MIC_BIAS（PA2）。



原因如下：

如果是省电容，芯片会执行内部的校准 MIC 偏置程序，由于做的是带混响或者扩音器，PCBA 板一般没有 MIC，那么我们内部的 MIC 校准程序校准不到值，就会导致开机慢和批量出现不一致的问题，或者产生其他由于 MIC 偏置导致的 MIC 声音的问题出来。

所以如果是 MIC 不在 PCB 板子上的带有混响或者扩音的机器，必须用加隔直电容 MIC 方案。参考电路如下。MIC 的偏置一定要选择芯片的 MIC_BIAS（PA2），如果用 VDDIO，一来底噪大，二来会跟随 VDDIO 变动出现杂音。

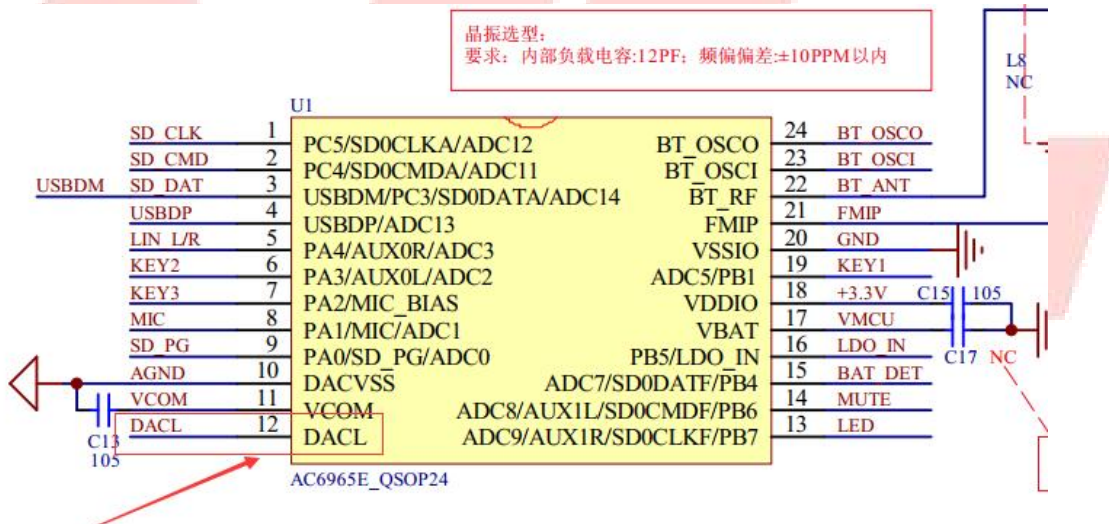


MIC_BIAS 的偏置请设置在 3.0V 以下，免得由于电池低电下跟随电压抖动出现杂音。

```
rd_ac696x_demo_cfg.h Board_ac696x_demo.c 全部关闭
00199: .mic_capless = 0,
00200: /*MIC免电容方案需要设置,影响MIC的偏置电压
00201: 21:1.18K 20:1.42K 19:1.55K 18:1.99K 17:2.2K 16:2.4K
00202: 10:3.91K 9:4.41K 8:5.0K 7:5.6K 6:6K 5:6.5K
00203: .mic_bias_res = 16,
00204: /*MIC LDO电压档位设置,也会影响MIC的偏置电压
00205: 0:2.3v 1:2.5v 2:2.7v 3:3.0v */
00206: .mic_ldo_vsel = 2,
00207: /*MIC电容隔直模式使用内部mic偏置(PA2)*/
00208: .mic_bias_inside = 1,
00209: /*保持内部mic偏置输出*/
00210: .mic_bias_keep = 0,
00211:
```

60.AC6965E 的 AUX/LINEIN 设计注意 20200915 YWP

AC6965E 是只绑定出 DACL 声道出来的 IC，如下图，所以在使用 AUX/LINEIN 上选用通道需注意。为了不需要程序多设置，AUX 的通道请选择对应的 AUXn_L 作为 LINEIN 的输入。如果选用 AUXn_R 进，LINEIN 下是没有声音输出的，因为声音从 DACR 出去了，没有从 DACL 出去。



扩展:

所以如果 AC696N 的芯片只绑定出 DACR，那么 AUX 的就选用 AUXn_R；只绑定出 DACL 那么 AUX 就选用 AUXn_L 进入。

同时请看下本文档的第 66 点！

61.AC696N 的 LADC 和蓝牙后台设计注意 20200915 YWP

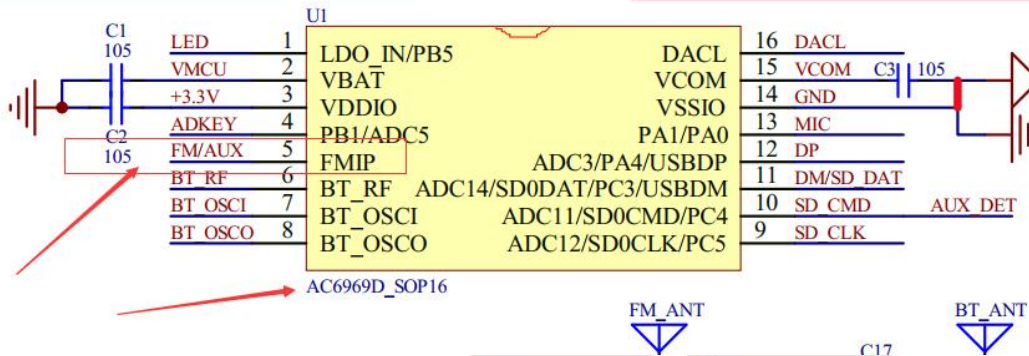
LADC:

AC696N 的 LADC 只有一路，跟 AC695N 的三路 LADC 不一样。所以在评估功能时需要注意。不能同时在 LINEIN 下开 ADC 和打开 MIC 的 ADC。例如混响，只能 LINEIN 走模拟，MIC 走 ADC。

同时例如 AC6969D 的第 5 脚 FMIP，有 LINEIN 和 FM 天线功能共用的，这其实 LINEIN 走的是 ADC，所以如果有 MIC 的 ADC，例如混响，那么 LINEIN 下是不支持 AUX 的 ADC 的。

- 、VDDIO, VBAT 必须使用原装电容，以防漏电。
- 、没有备注耐压值的电容，统一用耐压值 6.3V 的电容

AGND 接到功放 Bypass 处，再在芯片 VSSIO 处和 GND 连



蓝牙后台:

AC696N 的如果有蓝牙跑后，那么跟 AC692 一样，也是在 FM 模式下不支持蓝牙后台。

62.AC696N 的蓝牙连接成功自动播放 20200916 FSW

添加下面的代码

```
static int a2dp_auto_play_timer = 0;
void a2dp_auto_play_cmd_send(void *p)
{
    printf("a2dp_auto_play_cmd_send\n");
    if(get_total_connect_dev() && a2dp_get_status() != BT_MUSIC_STATUS_STARTING && get_call_status()
    == BT_CALL_HANGUP){
        user_send_cmd_prepare(USER_CTRL_AVCTP_OPID_PLAY, 0, NULL);
    }
    a2dp_auto_play_timer = 0;
}

void a2dp_auto_play_init(void)
{
    if(!get_remote_test_flag() && a2dp_auto_play_timer == 0){
        a2dp_auto_play_timer = sys_timeout_add(NULL, a2dp_auto_play_cmd_send, 1000);
    }
}
```

在 case BT_STATUS_CONN_A2DP_CH:处调用 a2dp_auto_play_init();即可。

63.AC696N 混响开发注意点 LZK

修改记录:

20200922: 添加常用函数 — LZK

(请先自行查看 SDK 自带文档《混响应用详细设计说明书》)

SDK 版本: 适用 AC696N_soundbox_sdk_release_1.0.0 及以后 sdk

AC696 混响延时测试数据(仅供参考)

混响功能选择	延时时间(ms)
MIC	10ms(理想状态)
MIC + echo	15±5ms
MIC + echo + PITCH + HOWLING	30±10ms
使能 noisegate 延时会根据配置的参数不同会有几ms差异	±n ms

混响 MIC 数据入口

```
static u32 mic_effect_effect_run(void *priv, void *in, void *out, u32 inlen, u32 outlen)
```

(回声)混响音量

1. mic_effect.c 文件中实现调节 wetgain 的函数(范围 0~4096)

```
void mic_effect_set_echo_wetgain(u32 gain)
{
    if (__this == NULL || __this->p_echo_hdl == NULL) {
        return ;
    }
    os_mutex_pend(&__this->mutex, 0);
    __this->fade.wet = gain;
    os_mutex_post(&__this->mutex);
}
```

3. void mic_effect_fade_run(struct __mic_effect *effect) 函数中添加以下内容

```
,  
}  
update = 0;  
if (effect->p_echo_hdl) {  
  
    if (effect->p_echo_hdl->echo_fix_parm.wetgain != effect->fade.wet) {  
        effect->p_echo_hdl->echo_fix_parm.wetgain = effect->fade.wet;  
        effect->p_echo_hdl->func_api->reset_wetdry(effect->p_echo_hdl->ptr, effect->fade.wet, effect->p_echo_hdl->echo_fix_parm.drygain);  
        return;  
    }  
  
    if (effect->p_echo_hdl->echo_parm_obj.delay != effect->fade.delay) {
```

(回声)混响延时

```
void mic_effect_set_echo_delay(u32 delay)  
u32 mic_effect_get_echo_delay(void)
```

(回声)混响衰减

```
void mic_effect_set_echo_decay(u32 decay)  
u32 mic_effect_get_echo_decay(void)
```

变调

方法 1: 在线调试调试好各个模式的变调参数, 使用下面的函数进行切换

```
void mic_effect_change_mode(u16 mode)  
  
u16 mic_effect_get_cur_mode(void)
```

若想添加/删除 音效模式, 自行参考 SDK 的“魔音”配置

混响在线调试

自行查看《混响音效在线调试说明文档.pdf》

MIC AUTOMUTE 检测

方法一: 使用 noisegate

混响添加 noisegate 处理 : MIC_EFFECT_CONFIG_NOISEGATE

修改 NOISEGATE 参数

没有外部音效文件, 使用默认参数

```
const NOISEGATE_PARM effect_noisegate_parm_default = {  
  
    .attackTime = 300,  
  
    .releaseTime = 5,  
  
    .threshold = -45000,  
  
    .low_th_gain = 0,
```

```
.sampleRate = MIC_EFFECT_SAMPLERATE,

.channel = 1,

};
```

有外部音效文件，使用音效文件的值
人声消除

1. 使能宏定义

```
/**
***//
//          人声消除使能
/**
***//
#define AUDIO_VOCAL_REMOVE_EN      0
```

2. 输出通道必须配成双声道 :DAC_OUTPUT_Ldac_outR

```
/*
DAC硬件上的连接方式,可选的配置:
    DAC_OUTPUT_MONO_L          左声道
    DAC_OUTPUT_MONO_R          右声道
    DAC_OUTPUT_Ldac_outR      立体声
    DAC_OUTPUT_MONO_LR_DIFF    单声道差分输出
*/
#define TCFG_AUDIO_DAC_CONNECT_MODE    DAC_OUTPUT_Ldac_outR
```

3. 开关人声消除

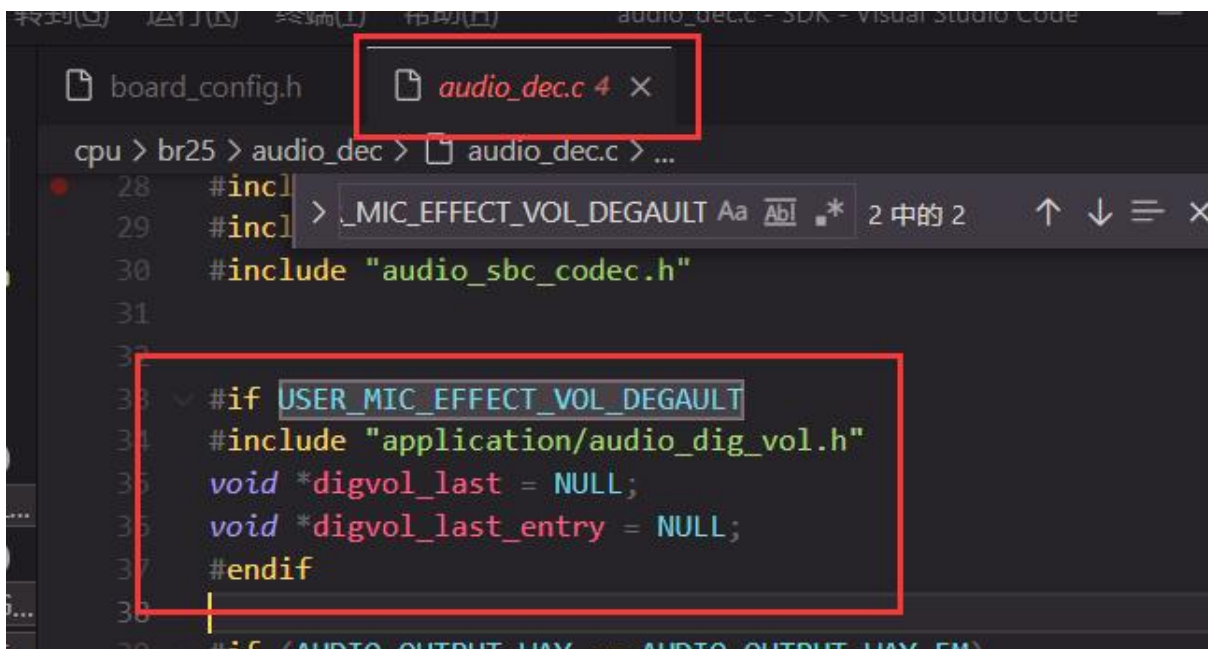
```
/*-----*/
/**@brief   audio_vocal_remove_sw 运行过程开关处理
  @param   _hdl:句柄
  @param   dis: 0:人声消除有效 1:人声消除无效
  @return  0:成功 -1:失败
  @note
*/
/*-----*/
int audio_vocal_remove_sw(vocal_remove_hdl *_hdl, u32 dis);
```

例子

```
void vocal_remove_sw(u8 sw)
{
    if (mix_vocal_remove_hdl) {
        audio_vocal_remove_sw(mix_vocal_remove_hdl,sw);
    }
}
```

64.AC696 1.1.0 SDK 媒体音量与麦克风音量独立实现方法 20200925 LHY- 20210817 WGH
修改

1. 在 mixer 数据流添加数字音量节点
- 1.1 在 cpu\br25\audio_dec\audio_dec.c 里添加



```
board_config.h audio_dec.c 4 x
cpu > br25 > audio_dec > audio_dec.c > ...
28 #incl
29 #incl > _MIC_EFFECT_VOL_DEGAULT Aa Abi .* 2 中的 2 ↑ ↓ ≡ ×
30 #include "audio_sbc_codec.h"
31
32
33 #if USER_MIC_EFFECT_VOL_DEGAULT
34 #include "application/audio_dig_vol.h"
35 void *digvol_last = NULL;
36 void *digvol_last_entry = NULL;
37 #endif
38
39
40 #if (AUDIO_OUTPUT_MAX == AUDIO_OUTPUT_MAX_EM)
```

```
#if USER_MIC_EFFECT_VOL_DEGAULT
#include "application/audio_dig_vol.h"
void *digvol_last = NULL;
void *digvol_last_entry = NULL;
#endif
```

```
board_config.h audio_dec.c 4 x
cpu > br25 > audio_dec > audio_dec.c > audio_dec_init()
680 #if A > _MIC_EFFECT_VOL_DEGAULT Aa Abi * 2 中的 2 ↑ ↓ ≡ ×
681 // 数据流串联。可以在mixer和last中间添加其他的数据流，比如
682 spec_hdl = spectrum_open_demo(sr, ch_num);
683 #endif
684
685 // 数据流串联。可以在mixer和last中间添加其他的数据流，比如
686 u8 entry_cnt = 0;
687 entries[entry_cnt++] = &mixer_entry;
688 #if TCFG_EQ_ENABLE && TCFG_AUDIO_OUT_EQ_ENABLE
689 if (mix_eq_drc) {
690     entries[entry_cnt++] = &mix_eq_drc->entry;
691 }
692 #endif
693
694 #if AUDIO_VOCAL_REMOVE_EN
695 if (mix_vocal_remove_hdl) {
696     entries[entry_cnt++] = &mix_vocal_remove_hdl->entry;
697 }
698 if (mix_ch_switch) {
699     entries[entry_cnt++] = &mix_ch_switch->entry;
700 }
701 #endif
702
703 #if AUDIO_OUTPUT_AUTOMUTE
704     entries[entry_cnt++] = mix_out_automute_entry;
705 #endif
706
707 #if USER_MIC_EFFECT_VOL_DEGAULT
708 audio_dig_vol_param digvol_last_param = {
709     .vol_start = 30,
710     .vol_max = 30,
711     .ch_total = 2,
712     .fade_en = 1,
713     .fade_points_step = 5,
714     .fade_gain_step = 10,
715     .vol_list = NULL,
716 };
717 digvol_last = audio_dig_vol_open(&digvol_last_param);
718 digvol_last_entry = audio_dig_vol_entry_get(digvol_last);
719 entries[entry_cnt++] = digvol_last_entry;
720
721 #endif
722
```

```
#if USER_MIC_EFFECT_VOL_DEGAULT
audio_dig_vol_param digvol_last_param = {
    .vol_start = 30,
    .vol_max = 30,
    .ch_total = 2,
    .fade_en = 1,
    .fade_points_step = 5,
    .fade_gain_step = 10,
    .vol_list = NULL,

```

```
};  
    digvol_last = audio_dig_vol_open(&digvol_last_param);  
    digvol_last_entry = audio_dig_vol_entry_get(digvol_last);  
    entries[entry_cnt++] = digvol_last_entry;  
  
#endif
```

2. 添加数字音量节点后，音量调节函数需要做修改

2.1 cpu\br25\audio_common\app_audio.c

```
board_config.h  app_audio.c 3 mic_effect.c 3 mic_effect.h soundcard.c  
cpu > br25 > audio_common > app_audio.c > audio_vol_set(u8, u8, u8)  
185 #endif  
186  
187 #if (AUDIO_OUTPUT_WAY == AUDIO_OUTPUT_WAY_BT)  
188     bt_emitter_set_vol(gain_l);  
189 #endif  
190     local_irq_disable();  
191     __this->fade_gain_l = gain_l;  
192     __this->fade_gain_r = gain_r;  
193  
194 #if (!USER_MIC_EFFECT_VOL_DEGAULT)  
195 #if (TCFG_AUDIO_DAC_CONNECT_MODE == DAC_OUTPUT_MONO_L)  
196     audio_dac_vol_set(TYPE_DAC_AGAIN, BIT(0), gain_l, fade);  
197     audio_dac_vol_set(TYPE_DAC_DGAIN, BIT(0), gain_l ? DEFAULT_DIGITAL_VOLUME : 0, fade);  
198 #elif (TCFG_AUDIO_DAC_CONNECT_MODE == DAC_OUTPUT_MONO_L)  
199     audio_dac_vol_set(TYPE_DAC_AGAIN, BIT(1), gain_r, fade);  
200     audio_dac_vol_set(TYPE_DAC_DGAIN, BIT(0), gain_l ? DEFAULT_DIGITAL_VOLUME : 0, fade);  
201 #else  
202     audio_dac_vol_set(TYPE_DAC_AGAIN, BIT(0), gain_l, fade);  
203     audio_dac_vol_set(TYPE_DAC_AGAIN, BIT(1), gain_r, fade);  
204     audio_dac_vol_set(TYPE_DAC_DGAIN, BIT(0), gain_l ? DEFAULT_DIGITAL_VOLUME : 0, fade);  
205     audio_dac_vol_set(TYPE_DAC_DGAIN, BIT(1), gain_r ? DEFAULT_DIGITAL_VOLUME : 0, fade);  
206 #endif  
207 #else  
208     extern int audio_dig_vol_set(void *hdl, u32 channel, u8 vol);  
209     extern int audio_dig_vol_fade(void *hdl, u32 channel, u8 fade_en);  
210     if(digvol_last_entry && digvol_last){  
211         r_printf("the digvol_last == %d, gain_r == %d \n", gain_l, gain_r);  
212         audio_dig_vol_set(digvol_last, BIT(0), gain_l);  
213         audio_dig_vol_set(digvol_last, BIT(1), gain_r);  
214     }  
215 #endif  
216     local_irq_enable();  
217     return 0;  
218  
219  
220  
#if USER_MIC_EFFECT_VOL_DEGAULT  
extern void *digvol_last;  
extern void *digvol_last_entry;  
#endif
```

```
#if (!USER_MIC_EFFECT_VOL_DEGAULT)  
#if (TCFG_AUDIO_DAC_CONNECT_MODE == DAC_OUTPUT_MONO_L)  
    audio_dac_vol_set(TYPE_DAC_AGAIN, BIT(0), gain_l, fade);  
    audio_dac_vol_set(TYPE_DAC_DGAIN, BIT(0), gain_l ? DEFAULT_DIGITAL_VOLUME : 0, fade);  
#endif
```

```
#elif (TCFG_AUDIO_DAC_CONNECT_MODE == DAC_OUTPUT_MONO_L)
    audio_dac_vol_set(TYPE_DAC_AGAIN, BIT(1), gain_r, fade);
    audio_dac_vol_set(TYPE_DAC_DGAIN, BIT(0), gain_l ? DEFAULT_DIGITAL_VOLUME : 0, fade);
#else
    audio_dac_vol_set(TYPE_DAC_AGAIN, BIT(0), gain_l, fade);
    audio_dac_vol_set(TYPE_DAC_AGAIN, BIT(1), gain_r, fade);
    audio_dac_vol_set(TYPE_DAC_DGAIN, BIT(0), gain_l ? DEFAULT_DIGITAL_VOLUME : 0, fade);
    audio_dac_vol_set(TYPE_DAC_DGAIN, BIT(1), gain_r ? DEFAULT_DIGITAL_VOLUME : 0, fade);
#endif
#else
    extern int audio_dig_vol_set(void *hdl, u32 channel, u8 vol);
    extern int audio_dig_vol_fade(void *hdl, u32 channel, u8 fade_en);
    if(digvol_last_entry && digvol_last){
        r_printf("the digvol_last == %d, gain_r == %d \n", gain_l, gain_r);
        audio_dig_vol_set(digvol_last, BIT(0), gain_l);
        audio_dig_vol_set(digvol_last, BIT(1), gain_r);
    }
#endif
    local_irq_enable();
    return 0;
}
```

3. MIC 音量调节通过以下函数接口调节
cpu\br25\audio_mic\mic_effect.c

```
/*-----*/  
/**@brief 数字音量调节接口  
@param  
@return  
@note  
*/  
-----*/  
void mic_effect_set_dvol(u8 vol)  
{  
    if (__this == NULL) {  
        return ;  
    }  
    audio_dig_vol_set(__this->d_vol, 3, vol);  
}  
u8 mic_effect_get_dvol(void)  
{  
    if (__this) {  
        return audio_dig_vol_get(__this->d_vol, 1);  
    }  
    return 0;  
}
```

65.AC696 1.1.0 SDK U 盘/SD 卡提示音独立实现方法 20200925 LHY

1.下面是定义盘符

```
37  
38 #define JL_USER_USB "udisk0"  
39 #define JL_USER_SD0 "sd0"  
40 #define JL_USER_SD1 "sd1"
```

2.U 盘 /SD 卡提示独立处理代码如下，无论是第一次上电还是后面插拔都会走到 KEY_MUSIC_PLAYER_START

```
456 case KEY_MUSIC_PLAYER_START:  
457     log_i("KEY_MUSIC_PLAYER_START !!\n");  
458     //断点播放活动设备  
459     logo = dev_manager_get_logo(dev_manager_find_active(1));  
460     r_printf("XXX %s\n",logo);  
461     if (music_player_get_play_status() == FILE_DEC_STATUS_PLAY) {  
462         if (music_player_get_dev_cur() && logo) {  
463             //播放的设备跟当前活动的设备是同一个设备，不处理  
464             if (0 == strcmp(logo, music_player_get_dev_cur())) {  
465                 log_w("the same dev!!\n");  
466                 break;  
467             }  
468         }  
469     }  
470     if(logo == NULL){  
471         r_printf("logo == NULL\n");  
472         break;  
473     }else{  
474         if(0 == strcmp(logo,JL_USER_USB)){  
475             r_printf("XXX IDEX_TONE_UDISK\n");  
476             tone_play_by_path(TONE_MUSIC_USB,1);  
477         }else{  
478             r_printf("XXX IDEX_TONE_TF\n");  
479             tone_play_by_path(TONE_MUSIC_SD,1);  
480         }  
481     }  
482     if (true == breakpoint_vm_read(breakpoint, logo)) {  
483         err = music_player_play_by_breakpoint(logo, breakpoint);  
484     } else {  
485         err = music_player_play_first_file(logo);  
486     }  
487     break;  
488     //播放器退出处理
```

66.AC696 系列 LINEIN 左右声道与 DACLR 输入出的注意 20201015 YWP

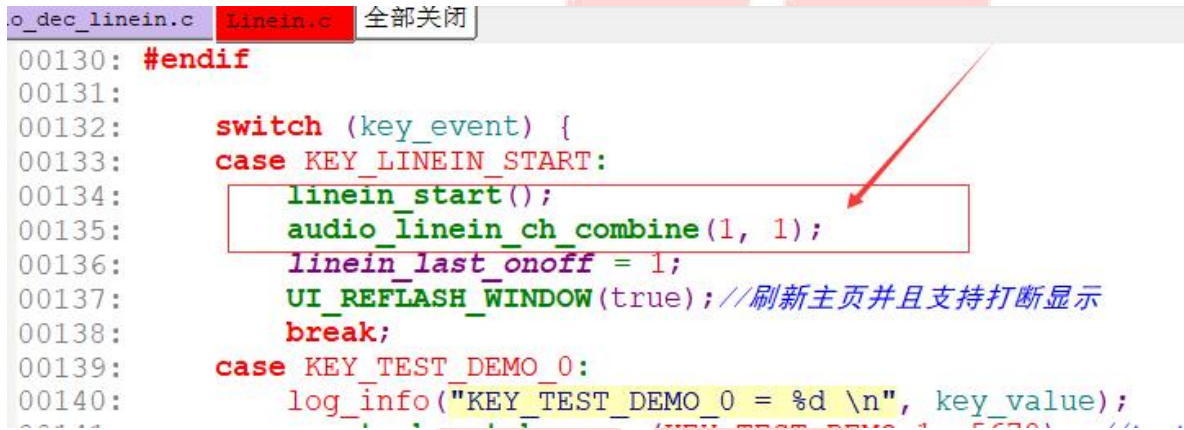
这部分请先读下本文档第 60 点。

如果设计时 LINEIN 的 R 声道进入，芯片是 DACL 输出，例如 AC6965E 是 DACL 输出的，那么在 LINEIN 下走模拟的是没有声音的，需要用如下函数调用。

```
/*  
*linein 单声道模拟输入的时候，如果 dac 是两个声道的，想要  
*两个声道都有声音出来，需要将 amux 通道合并一下，即：  
*call:audio_linein_ch_combine(1,1);  
*/
```

```
void audio_linein_ch_combine(u8 LR_2_L, u8 LR_2_R);
```

在 linein_start 后面调用。



```
o_dec_linein.c | Linein.c | 全部关闭 |  
00130: #endif  
00131:  
00132:     switch (key_event) {  
00133:     case KEY_LINEIN_START:  
00134:         linein_start();  
00135:         audio_linein_ch_combine(1, 1);  
00136:         linein_last_onoff = 1;  
00137:         UI_REFLASH_WINDOW(true); //刷新主页并且支持打断显示  
00138:         break;  
00139:     case KEY_TEST_DEMO_0:  
00140:         log_info("KEY_TEST_DEMO_0 = %d \n", key_value);  
00141:         UI_REFRESH_DEMO_0 = 1; //刷新demo0显示
```

67.AC696 系列 外挂 FLASH 使用方法与注意点 20201016 LHY

1、配置 SPI

```

board_ac696x_demo_cfg.h - SDK - Visual Studio Code
C mute.c C board_ac696x_demo.c C board_ac696x_demo_cfg.h X C spi_test.c
apps > soundbox > board > br25 > board_ac696x_demo > C board_ac696x_demo_cfg.h > TCFG_HW_SPI1_ENABLE
74 #define TCFG_HW_I2C0_PORTS 'B'
75 #define TCFG_HW_I2C0_CLK 100000 //硬件IIC波特率
76 //*****
77 //*****
78 //***** 硬件SPI 配置 ***** //
79 //*****
80 #define TCFG_HW_SPI1_ENABLE 0//ENABLE_THIS_MOUDLE
81 //A组IO: DI: PB2 DO: PB1 CLK: PB0
82 //B组IO: DI: PC3 DO: PC5 CLK: PC4
83 #define TCFG_HW_SPI1_PORT 'A'
84 #define TCFG_HW_SPI1_BAUD 400000L
85 #define TCFG_HW_SPI1_MODE SPI_MODE_BIDIR_1BIT
86 #define TCFG_HW_SPI1_ROLE SPI_ROLE_MASTER
87
88 #define TCFG_HW_SPI2_ENABLE ENABLE_THIS_MOUDLE
89 //A组IO: DI: PB8 DO: PB10 CLK: PB9
90 //B组IO: DI: PA13 DO: DM CLK: DP
91 #define TCFG_HW_SPI2_PORT 'A'
92 #define TCFG_HW_SPI2_BAUD 200000L
93 #define TCFG_HW_SPI2_MODE SPI_MODE_BIDIR_1BIT
94 #define TCFG_HW_SPI2_ROLE SPI_ROLE_MASTER
95

```

选择的板级文件

根据样机去选择对应IO组, 同时需要注意查看数据手册来设置最大波特率

2、配置 FLASH

```

96 //*****
97 //***** FLASH 配置 ***** //
98 //*****
99 #define TCFG_NORFLASH_DEV_ENABLE DISABLE_THIS_MOUDLE //需要关闭SD0
100 #define TCFG_FLASH_DEV_SPI_HW_NUM 1// 1: SPI1 2: SPI2
101 #define TCFG_FLASH_DEV_SPI_CS_PORT IO_PORTA_03
102
103

```

根据SPI配置来设置, 同时需要关闭SD功能

```

188 //*****
189 //***** fat_FLASH 配置 ***** //
190 //*****
191 #define TCFG_CODE_FLASH_ENABLE DISABLE_THIS_MOUDLE
192
193 #define FLASH_INSIDE_REC_ENABLE 0
194
195
196 #if TCFG_NORFLASH_DEV_ENABLE
197 #define TCFG_NOR_FAT 1//ENABLE
198 #define TCFG_NOR_FS 0//ENABLE
199 #define TCFG_NOR_REC 1//ENABLE
200 #else
201 #define TCFG_NOR_FAT 0//ENABLE
202 #define TCFG_NOR_FS 0//ENABLE
203 #define TCFG_NOR_REC 0//ENABLE
204 #endif
205
206

```

需要根据自己的需求来配置使能哪些, 如果只是播放歌曲只需要打开FAT或FS 注意flash大小来配置否则会导致复位

3.如何挂载设备并播放音乐

(1)、首先需要根据第二点打开的宏先找到对应的描述符, 比如 fat_nor、rec_nor 等

```

dev_manager.c - SDK - Visual Studio Code
C mute.c C board_ac696x_demo.c C board_ac696x_demo_cfg.h C dev_manager.c X C spi_test
apps > common > dev_manager > C dev_manager.c > dev_manager_task(void *)
1036     os_mutex_create(&__this->mutex);
1037     //设备初始化,
1038     devices_init();
1039
1040     #if SDFILE_STORAGE && TCFG_CODE_FLASH_ENABLE
1041     dev_manager_add(SDFILE_DEV);
1042     #endif
1043
1044     #if TCFG_NOR_REC
1045     dev_manager_add("rec_nor");
1046     #endif
1047
1048     #if TCFG_NOR_FAT
1049     dev_manager_add("fat_nor");
1050     #endif
1051
1052     #if TCFG_NOR_FS
1053     dev_manager_add("res_nor");
1054     #endif
1055
1056     #if TCFG_VIR_UDISK_ENABLE
1057     dev_manager_add("vir_udisk0");
1058     #endif
1059
1060     #if FLASH_INSIDE_REC_ENABLE
1061     set_rec_capacity(512*1024); //需要先设置容量,注意要小于Ini文件设置大小.
1062     _sdfile_rec_init();
1063     dev_manager_add("rec_sdfile");
1064     #endif
1065

```

(2)、挂载设备需要激活设备和切换模式

```

62 //-----*/
63 /**@brief  激活指定逻辑盘符的设备
64     @param
65     |      |      |      |
66     |      |      |      |      |
67     |      |      |      |      |
68     |      |      |      |      |
69     |      |      |      |      |
70 void dev_manager_set_active_by_logo(char *logo)
71 {
72     if (logo == NULL) {
73         return ;
74     }
75     struct __dev *dev = NULL;
76     os_mutex_pend(&__this->mutex, 0);
77     dev = dev_manager_check_by_logo(logo);
78     if (dev) {
79         dev->active_stamp = __dev_manager_get_time_stamp();//timer_get_ms();
80     }
81     os_mutex_post(&__this->mutex);
82 }
83 //-----*/

```

填入(1)中对应的盘符,可以先激活再切音乐模式,或者切音乐初始化后激活

```

3 //将选定的设备设置为活动设备
4 dev_manager_set_active_by_logo(dev_logo);
5 //切换模式
6 app_task_switch_to(APP_MUSIC_TASK);
7

```

参考这里

4、遇到问题需要的注意点

(1)、首先先保证能够读取到 ID 号，如果读取不到先从硬件排查是否接错 IO 口

```
[00:00:00.275]rec_nor_rec add start
[00:00:00.276]dev_manager_add can not find logo rec_nor_rec
[00:00:00.277]rec_nor add start
[00:00:00.278][Info]: [FLASH]norflash open

[00:00:00.279][Debug]: [SFI]spi clock source freq 24000000
[00:00:00.280][Info]: [FLASH]norflash_read_id: 0xf0014
[00:00:00.282][Info]: [FLASH]norflash_capacity: 0x100000

[00:00:00.317][Info]: [FLASH]norflash
open success !
```

先确保读取到 ID号

(2)、通过 flash 大小匹配设置项

```
1000 x 768
(1) 帮助(H) board_ac696x_demo.c - SDK - Visual Studio Code
C mute.c C event.c C board_ac696x_demo.c X C board_ac696x_demo
apps > soundbox > board > br25 > board_ac696x_demo > C board_ac696x_demo.c > ...
571     .role = TCFG_HW_SPI1_ROLE,
572 };
573 #endif
574
575 #if TCFG_HW_SPI2_ENABLE
576 const struct spi_platform_data spi2_p_data = {
577     .port = TCFG_HW_SPI2_PORT,
578     .mode = TCFG_HW_SPI2_MODE,
579     .clk = TCFG_HW_SPI2_BAUD,
580     .role = TCFG_HW_SPI2_ROLE,
581 };
582 #endif
583
584 #if TCFG_NOR_FAT
585 NORFLASH_DEV_PLATFORM_DATA_BEGIN(norflash_fat_dev_data)
586     .spi_hw_num = TCFG_FLASH_DEV_SPI_HW_NUM,
587     .spi_cs_port = TCFG_FLASH_DEV_SPI_CS_PORT,
588     #if (TCFG_FLASH_DEV_SPI_HW_NUM == 1)
589         .spi_pdata = &spi1_p_data,
590     #elif (TCFG_FLASH_DEV_SPI_HW_NUM == 2)
591         .spi_pdata = &spi2_p_data,
592     #endif
593     .start_addr = 0,
594     .size = 1*1024*1024,
595     NORFLASH_DEV_PLATFORM_DATA_END()
596 #endif
597
598 #if TCFG_NOR_FS
599 NORFLASH_DEV_PLATFORM_DATA_BEGIN(norflash_norfs_dev_data)
600     .spi_hw_num = TCFG_FLASH_DEV_SPI_HW_NUM,
601     .spi_cs_port = TCFG_FLASH_DEV_SPI_CS_PORT,
602     #if (TCFG_FLASH_DEV_SPI_HW_NUM == 1)
603         .spi_pdata = &spi1_p_data,
604     #elif (TCFG_FLASH_DEV_SPI_HW_NUM == 2)
605         .spi_pdata = &spi2_p_data,
606     #endif
607     .start_addr = 1*1024*1024,
608     .size = 1*1024*1024,
609     NORFLASH_DEV_PLATFORM_DATA_END()
610 #endif
611
612 #if TCFG_NOR_REC
613 NORFLASH_DEV_PLATFORM_DATA_BEGIN(norflash_norfs_rec_dev_data)
614     .spi_hw_num = TCFG_FLASH_DEV_SPI_HW_NUM,
615     .spi_cs_port = TCFG_FLASH_DEV_SPI_CS_PORT,
616     #if (TCFG_FLASH_DEV_SPI_HW_NUM == 1)
617         .spi_pdata = &spi1_p_data,
618     #elif (TCFG_FLASH_DEV_SPI_HW_NUM == 2)
619         .spi_pdata = &spi2_p_data,
620     #endif
621     .start_addr = 2*1024*1024,
622     .size = 2*1024*1024,
623     NORFLASH_DEV_PLATFORM_DATA_END()
624 #endif
625
```

需要自己去计算并分配flash,不能超过flash大小否则会导致复位

68.AC696 系列 SDK1.1.1 音乐模式频繁播放提示语导致播放异常 20201016 LHY

在频繁使用打断播放方式进行播放提示语时，音乐重新播放时会出现很一些杂音(突变式)，可以通过下面代码解决


```

903  static void  bt_tone_play_end_callback(void *priv, int flag)
904  {
905      u32 index = (u32)priv;
906
907      if (APP_BT_TASK != app_get_curr_task()) {
908          log_error("tone callback task out \n");
909          return;
910      }
911
912      switch (index) {
913      case IDEX_TONE_BT_MODE:
914          //提示音播放结束
915          r_printf(">>>>>>>> bt statr\n");
916          bt_task_start();
917          break;
918      default:
919          break;
920      }
921  }
922

```

播完提示语再初始化

70.AC696 系列 SDK1.1.1 指定拔出 U 盘切换到特定模式解决方法 20201016 LHY

看(V) 转到(G) 运行(R) 终端(T) 帮助(H) music.c - SDK - Visual Studio Code

C mute.c C board_ac696x_demo.c C music.c X C board_ac696x_demo_cfg.h C spi_test.c

apps > soundbox > task_manager > music > C music.c > music_key_event_opr(sys_event *)

```

473  }else{
474      if(0 == strcmp(logo,JL_USER_USB)){
475          r_printf("XXX IDEX_TONE_UDISK\n");
476          tone_play_by_path(TONE_MUSIC_USB,1);
477      }else{
478          r_printf("XXX IDEX_TONE_TF\n");
479          tone_play_by_path(TONE_MUSIC_SD,1);
480      }
481  }
482  if (true == breakpoint_vm_read(breakpoint, logo)) {
483      err = music_player_play_by_breakpoint(logo, breakpoint);
484  } else {
485      err = music_player_play_first_file(logo);
486  }
487  break;
488  //播放器退出处理
489  case KEY_MUSIC_PLAYER_QUIT:
490      r_printf("KEY_MUSIC_PLAYER_QUIT !!\n");
491      app_task_switch_to(APP_BT_TASK);
492      break;
493  //任务退出处理

```

修改这里模式来选择拔出U盘设备切换到哪个模式

71.AC696 系列 SDK1.1.1 开机优先设备(SD 卡/U 盘)上线实现方法 20201017 LHY

1、配置优先设备

```

C mute.c C board_ac696x_demo.c C music.c X C board_ac696x_demo_cfg.h C sp
apps > soundbox > task_manager > music > C music.c > JL_USER_USB
16 #include "u_manage.h"
17 #include "audio_dev.h"
18 #include "common/dev_status.h"
19
20 #include "led.h"
21 #include "common/power_off.h"
22 #include "fs.h"
23 #include "storage_dev/storage_dev.h"
24
25 /*****
26 此文件函数主要是music模式按键处理和事件处理
27
28 void app_music_task()
29 music模式主函数
30
31 static int music_sys_event_handler(struct sys_event *event)
32 music模式系统事件所有处理入口
33
34 static void music_task_close(void)
35 music模式退出
36
37 *****/
38
39 #define JL_USER_USB "udisk0"
40 #define JL_USER_SD0 "sd0"
41 #define JL_USER_SD1 "sd1"
42 #define JL_USER_USB_OR_SD JL_USER_USB //进music模式优先设备
43

```

2、音乐模式初始化激活优先设备

```

帮助(H) music.c - SDK - Visual Studio Code
driver.c C music.c X C bt.c C app_common.c C board_ac6965e_cfg.h C user_cfg.c
soundbox > task_manager > music > C music.c > app_music_task()
/*
/*-----*/
void app_music_task()
{
    int res;
    int msg[32];
    music_task_start();

    int user_dev_tone_number = IDEX_TONE_MUSIC;

    #ifdef USER_USB_OR_SD
    printf(">>>>> music dev onlie number %d\n",dev_manager_get_total(1));
    extern u8 switch_sd_mode_falg;
    if(dev_manager_get_total(1)>1){
        if(switch_sd_mode_falg == FALSE){
            if((USER_USB_OR_SD == USER_DEV_USB) || (USER_USB_OR_SD == USER_DEV_SD0) || (USER_USB_OR_SD == USER_DEV_SD1)){
                dev_manager_set_active_by_logo(/*dev_logo[0]*/USER_USB_OR_SD);
            }
        }else{
            switch_sd_mode_falg = FALSE;
            if(dev_manager_check_by_logo(USER_DEV_SD0) != NULL){
                dev_manager_set_active_by_logo(USER_DEV_SD0);
            }else{
                dev_manager_set_active_by_logo(USER_DEV_SD1);
            }
        }
    }
    #endif

    int err = tone_play_with_callback_by_name(tone_table[user_dev_tone_number], 1, music_tone_play_end_callback, (void *)IDEX_TON
    // if (err) {
    // // music_player_play_start();
    // }

```

3、由于不同 SD 卡和 U 盘上线时间不同所以需要去等待另一个设备上线

```

终端(T) 帮助(H) app_common.c - SDK - Visual Studio Code
C key_driver.c C music.c C bt.c C app_common.c X C board_ac6965e_cfg.h C user_cfg.c
apps > soundbox > task_manager > C app_common.c > wait_sd_usb_online_function(void)
609 }
610 }
611
612 static u8 power_on_falg_wait_sd = TRUE;
613 static u8 power_on_falg_wait_usb = TRUE;
614 u8 switch_sd_mode_falg = FALSE;
615 static void wait_sd_usb_online_function(void)
616 {
617     power_on_falg_wait_sd = FALSE;
618     power_on_falg_wait_usb = FALSE;
619     if(!lapp_check_curr_task(APP_MUSIC_TASK)){
620         r_printf("find_sd_no_online\n");
621         app_task_switch_to(APP_MUSIC_TASK);
622     }
623 }
624

```

```

C key_driver.c  C music.c  C btc  C app_common.c X  C board_ac6965e_cfg.h  C user_cfg.c
apps > soundbox > task_manager > C app_common.c > wait_sd_usb_online_function(void)
686     case DRIVER_EVENT_FROM_SD0:
687     case DRIVER_EVENT_FROM_SD1:
688     case DRIVER_EVENT_FROM_SD2:
689     #if TCFG_APP_MUSIC_EN
690         ret = dev_status_event_filter(event); //解码设备上下线, 设备挂载等处理
691         if (ret == true) {
692             if (event->u.dev.event == DEVICE_EVENT_IN) {
693                 //设备上线, 非解码模式切换到解码模式播放
694                 if (app_get_curr_task() != APP_MUSIC_TASK) {
695                     if(((u32)event->arg == DEVICE_EVENT_FROM_OTG)&&(power_on_falg_wait_sd == TRUE)){
696                         power_on_falg_wait_sd = FALSE;
697                         if(power_on_falg_wait_usb == TRUE){
698                             r_printf("wait_sd_online \n");
699                             sys_timeout_add(NULL, wait_sd_usb_online_function, 500);
700                         }else{
701                             app = APP_MUSIC_TASK;
702                         }
703                     }else if(((u32)event->arg != DEVICE_EVENT_FROM_OTG)&&(power_on_falg_wait_usb == TRUE)){
704                         power_on_falg_wait_usb = FALSE;
705                         if(power_on_falg_wait_sd == TRUE){
706                             r_printf("wait_usb_online \n");
707                             sys_timeout_add(NULL, wait_sd_usb_online_function, 500);
708                         }else{
709                             app = APP_MUSIC_TASK;
710                         }
711                     }else{
712                         if(((u32)event->arg != DEVICE_EVENT_FROM_OTG)&&(dev_manager_get_total(1)>1)){
713                             switch_sd_mode_falg = TRUE;
714                         }
715                         app = APP_MUSIC_TASK;
716                     }
717                 }

```

这两个值最好不要超过1000, 否则单设备上线会比较慢

72.AC696 系列 SDK1.1.1 关闭 TWS 对箱回连解决方法 20201017 LHY

```

行(R) 终端(T) 帮助(H)  bt_tws.c - SDK - Visual Studio Code
mute.h  C mute.c  C board_ac696x_demo_cfg.h  C app_common.c  C bt_tws.c X  ...
apps > soundbox > task_manager > bt > C bt_tws.c > bt_tws_poweron()
1291     #endif
1292
1293
1294     err = tws_get_sibling_addr(addr);
1295     if (err == 0) {
1296         /*
1297          * 获取到对方地址, 开始连接
1298          */
1299         log_info("\n -----have tws info-----\n");
1300         gtws.state |= BT_TWS_PAIRED;
1301
1302         BT_STATE_TWS_INIT(1);
1303
1304         tws_api_set_sibling_addr(addr);
1305         if (set_channel_by_code_or_res() == 0) {
1306             channel = bt_tws_get_local_channel();
1307             tws_api_set_local_channel(channel);
1308         }
1309
1310     #if TCFG_TEST_BOX_ENABLE
1311         if (chargestore_get_testbox_status()) {
1312         } else
1313     #endif
1314     {
1315     #ifdef CONFIG_NEW_QUICK_CONN
1316         if (bt_tws_get_local_channel() == 'L') {
1317             syscfg_read(CFG_TWS_LOCAL_ADDR, addr, 6);
1318         }
1319         tws_api_set_quick_connect_addr(addr);
1320     #endif
1321     }
1322     //<<有tws配对信息, 开启tws链接
1323     bt_tws_connect_sibling(CONFIG_TWS_CONNECT_SIBLING_TIMEOUT);
1324 }
1325 } else {
1326     log_info("\n -----no tws info-----\n");
1327
1328     BT_STATE_TWS_INIT(0);

```

在bt_tws_poweron函数中注释掉该段代

73.AC696 系列 SDK1.1.1 按键断开对箱后会回连的解决方法 20201017 LHY

主要是由于主动调用断开对箱时候, 另一个音箱已经断开连接而没有同步到断开对箱消息导致回连, 这时候需要用到延时同步消息。

```
C mute.c      C board_ac696x_demo_cfg.h      C app_common.c      C bt_tws.h
s > soundbox > include > task_manager > bt > C bt_tws.h > _unnamed_enum_0119_3
SYNC_CMD_LOW_LATENCY_ENABLE,
SYNC_CMD_LOW_LATENCY_DISABLE,
SYNC_CMD_EARPHONE_CHAREG_START,
SYNC_CMD_IRSENSOR_EVENT_NEAR,
SYNC_CMD_IRSENSOR_EVENT_FAR,

SYNC_CMD_BOX_ENTER_BT,
SYNC_CMD_BOX_EXIT_BT,
SYNC_CMD_BOX_INIT_EXIT_BT = 0x80 | SYNC_CMD_BOX_EXIT_BT,
#ifdef USE_DMA_TONE || GMA_EN
SYNC_CMD_CUT_TWS_TONE, //断开对耳提示音
SYNC_CMD_START_SPEECH_TONE, //AI键提示音
SYNC_CMD_DMA_CONNECTED_ALL_FINISH_TONE, //AI连接成功
SYNC_CMD_NEED_BT_TONE, //需要连接蓝牙
SYNC_CMD_PLEASE_OPEN_XIAODU_TONE, //请打开小度
#endif
#ifdef RCSP_ADV_EN
SYNC_CMD_SYNC_ADV_SETTING,
SYNC_CMD_ADV_COMMON_SETTING_SYNC,
#endif
SYNC_CMD_PHONE_PAIR_TONE,
SYNC_CMD_PHONE_ANSWER_TONE,
SYNC_CMD_PHONE_SIRI_TONE,
SYNC_CMD_MODE_CHANGE,
USER_SYNC_CMD_DEL_TWS_INFO,
};
```

定义一个同步消息

```
805 #if TCFG_USER_TWS_ENABLE
806 case KEY_TO_TWS_TRI:
807 {
808     if((u32)event->arg == KEY_EVENT_FROM_TWS){
809         break;
810     }
811     r_printf("KEY_TO_TWS_TRI\n");
812     if(!get_bt_tws_connect_status()){
813         if (bt_tws_start_search_and_pair()) {
814             tone_play_by_path(TONE_DING, 1);
815         }
816     }else{
817         tone_play_by_path(TONE_DING, 1);
818         bt_tws_api_push_cmd(USER_SYNC_CMD_DEL_TWS_INFO,200);
819     }
820     // tone_play_by_path(TONE_DING, 1);
821     // app_task_put_key_msg(KEY_TWS_SEARCH_REMOVE_PAIR, 0);
822 }
823 break;
```

参考这里，去发送这个消息，使得主从机在200ms后同时执行该消息处理（断开对箱）

```
ute.h | C mute.c | C board_ac696x_demo_cfg.h | C app_common.c | C bt_tws.c X
apps > soundbox > task_manager > bt > C bt_tws.c > tws_event_sync_fun_cmd(bt_event *)
2503 #endif
2504     break;
2505
2506     case SYNC_CMD_LOW_LATENCY_ENABLE:
2507         earphone_a2dp_codec_set_low_latency_mode(1);
2508         break;
2509     case SYNC_CMD_LOW_LATENCY_DISABLE:
2510         earphone_a2dp_codec_set_low_latency_mode(0);
2511         break;
2512     case SYNC_CMD_EARPHONE_CHARGE_START:
2513         tws_event_cmd_earphone_charge_start();
2514         break;
2515     case SYNC_CMD_IRSENSOR_EVENT_NEAR:
2516         tws_event_cmd_irsensor_event_near();
2517         break;
2518     case SYNC_CMD_IRSENSOR_EVENT_FAR:
2519         tws_event_cmd_irsensor_event_far();
2520         break;
2521
2522     case SYNC_CMD_MODE_CHANGE:
2523         tws_event_cmd_mode_change();
2524         break;
2525     case USER_SYNC_CMD_DEL_TWS_INFO:
2526         r_printf("USER_SYNC_CMD_DEL_TWS_INFO");
2527         extern u8 bt_tws_remove_tws_pair();
2528         if(bt_tws_remove_tws_pair()){
2529             r_printf("bt_tws_remove_pairs success\n");
2530         }
2531         break;
2532     default :
2533         break;
2534 }
2535 }
```

在这里加入消息处理，主要是断开对箱消息

74.AC696 系列 SDK1.1.1 对箱设置单声道时，播歌下关其中一个音箱后，另外一个有时会复位需要更换的库 -----WTS 20201022

请使用以下网盘的 media_app.a 库文件替换后，rebuild 工程处理：

链接：https://pan.baidu.com/s/15Z_7iM3Eq5OkUWIA_Tllig

提取码：jjel

75.AC696 系列 SDK1.1.1 播完 U 盘或卡后，不会自动播放下一个设备音乐的修改方法 -----WTS 20201022

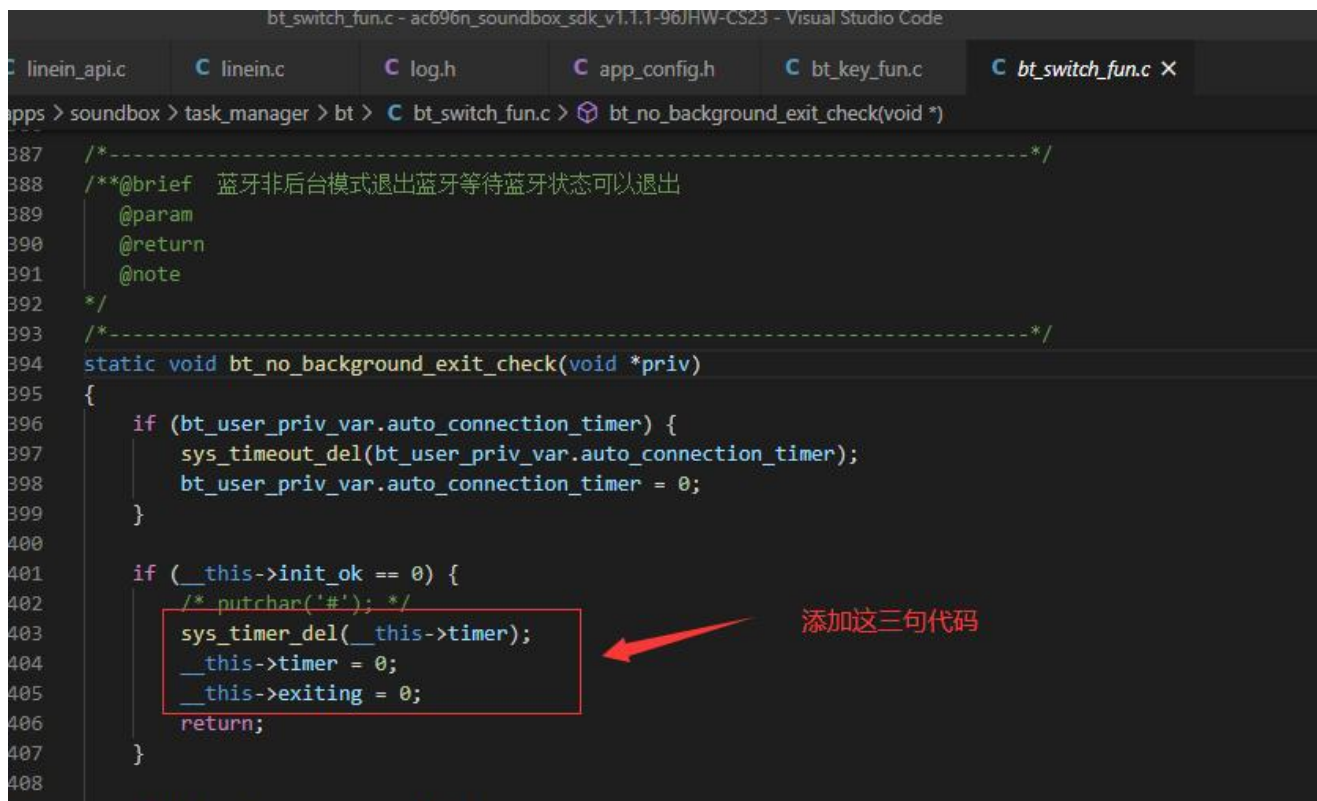
请使用以下网盘上的资料“AC696X SDK1.1.1 播完 U 盘或卡后，不会自动播放下一个设备音乐修改方法.zip”处理：

链接：<https://pan.baidu.com/s/1bAEP3mC5KUJoyYgWBGSCIA>

提取码：jjel

76.AC696 系列 SDK1.1.1 切蓝牙模式播提示音的时候（蓝牙初始化未完成）插入 TF 卡,无法切入 TF 卡模式，程序卡死问题处理方法 -----LAQ 20201022

按下图进行修改



```
bt_switch_fun.c - ac696n_soundbox_sdk_v1.1.1-96JHW-CS23 - Visual Studio Code
linein_api.c | linein.c | log.h | app_config.h | bt_key_fun.c | bt_switch_fun.c X
apps > soundbox > task_manager > bt > bt_switch_fun.c > bt_no_background_exit_check(void *)
387  /*-----*/
388  /**@brief  蓝牙非后台模式退出蓝牙等待蓝牙状态可以退出
389      @param
390      @return
391      @note
392  */
393  /*-----*/
394  static void bt_no_background_exit_check(void *priv)
395  {
396      if (bt_user_priv_var.auto_connection_timer) {
397          sys_timeout_del(bt_user_priv_var.auto_connection_timer);
398          bt_user_priv_var.auto_connection_timer = 0;
399      }
400
401      if (__this->init_ok == 0) {
402          /* putchar('#'); */
403          sys_timer_del(__this->timer);
404          __this->timer = 0;
405          __this->exiting = 0;
406          return;
407      }
408
409  }
```

77.AC696 系列 SDK1.1.1 数码管显示变化较快时，出现某一段亮一些，某一段暗一些的处理方法
-----LJW 20201023

```

535  /*-----*/
536  AT_LED_CODE
537  static void __led7_scan(void *param)
538  {
539      static u8 cnt = 0;
540      u8 seg, i;
541
542      if (__this->init == false) {
543          /* putchar('r'); */
544          return;
545      }
546
547      __ui_led7_clear_all();
548
549      /* if (get_vm_statu() && led7_skip_vm_flag()) { */
550      /*     return; */
551      /* } */
552
553      if (!cnt && !__this->lock) {
554          __ui_led7_update_bShowbuf1();
555      }
556
557      seg = __this->led7_var.bShowBuff1[cnt];
558
559      if (__this->user_data->pin_type == LED7_PIN7) {
560          //pin cnt output H
561          gpio_direction_output(__this->user_data->pin_cfg.pin7.pin[cnt], 1);
562          gpio_set_hd0(__this->user_data->pin_cfg.pin7.pin[cnt], 1);
563          gpio_set_hd( __this->user_data->pin_cfg.pin7.pin[cnt], 1);
564          for (i = 0; i < 7; i++) {
565              if (seg & BIT(i)) {
566                  //pin i output L
567                  gpio_set_hd0(__this->user_data->pin_cfg.pin7.pin[i], 0);
568                  gpio_set_hd( __this->user_data->pin_cfg.pin7.piin[i], 0);
569                  gpio_direction_output(__this->user_data->pin_cfg.pin7.pin[i], 0);
570              }
571          }
572      }
573  }
    
```

加上这几句

78.AC696 系列 SDK1.1.1 一边调回声延迟，一边用 MIC 说话会有沙沙声的处理方法
-----LJW 20201023

(1) 按如下修改

```

if (effect->p_echo_hdl) {
    if (effect->p_echo_hdl->echo_parm_obj.delay != effect->fade.delay) {
        update = 1;
        #if 0
        if (effect->p_echo_hdl->echo_parm_obj.delay > effect->fade.delay) {
            effect->p_echo_hdl->echo_parm_obj.delay -= 1;
        } else {
            effect->p_echo_hdl->echo_parm_obj.delay += 1;
        }
        #else
        effect->p_echo_hdl->echo_parm_obj.delay = effect->fade.delay;
        #endif
    }
}
    
```

修改此处

(2) 请使用以下网盘的 lib_reverb_cal.a 库文件替换后，rebuild 工程处理：

版权所有，侵权必究

链接: <https://pan.baidu.com/s/1fuMWdyV3DlzuKehWempfkg>
提取码: wf9g

79.AC696 系列 SDK1.1.1 插卡无法扫描到隐藏文件和带.的文件名的文件的处理方法 -----LJW 20201023

```
static void dev_manager_task(void *p)
{
    int res = 0;
    int msg[8] = {0};
    ///过滤隐藏 和 .开头名名的字文件
    hidden_file(0);
    ///初始化设备管理链表
    os_mutex_create(&__this->mutex);
    ///设备初始化,
    devices_init();
}
```

SDK默认打开,不过滤隐藏文件和.开头名字文件
请设为0

80.AC696 系列 SDK1.1.1 由于获取电量值改变的时间间隔太长导致电压检测不准的处理方法 -----LJW 20201027

```
C adc_apic X
SDK > cpu > br25 > _adc_apic > _adc_init(u32)
403 vbg_adc_value /= 10;
404 printf("vbg_adc_value = %d\n", vbg_adc_value);
405
406 u32 vbat_queue_ch = 0;
407 u32 vbg_queue_ch = 0;
408 if (vbat_vddio_tieup) {
409     vbg_queue_ch = adc_add_sample_ch(AD_CH_LDOREF);
410     adc_set_sample_freq(AD_CH_LDOREF, /*30000*/1000);
411     adc_queue[vbg_queue_ch].value = vbg_adc_value;
412 } else {
413     vbat_queue_ch = adc_add_sample_ch(AD_CH_VBAT);
414     adc_set_sample_freq(AD_CH_VBAT, /*30000*/1000);
415     adc_queue[vbat_queue_ch].value = vbat_adc_value;
416 }
```

改小该值

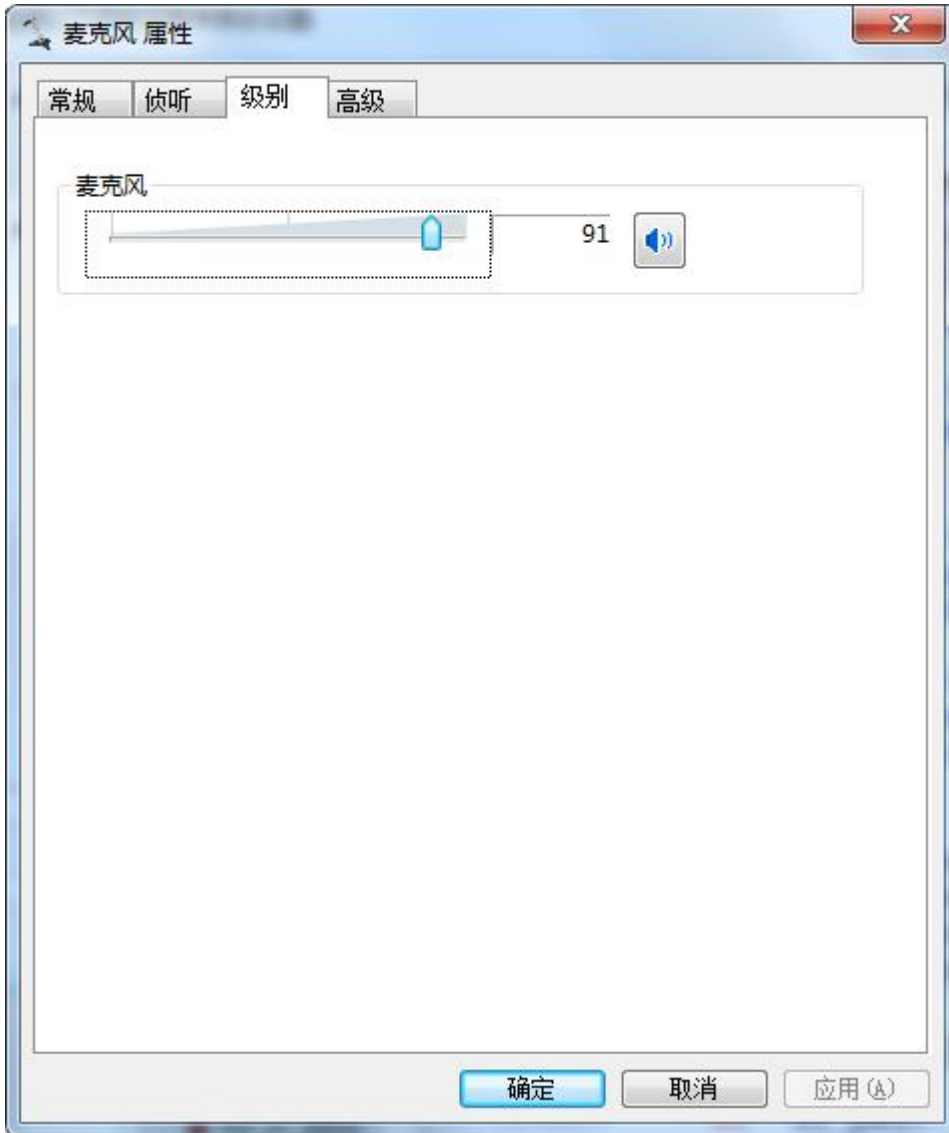
81.AC696 系列 SDK1.1.1 自动 mute 使用方法 -----LHY 20201028

1、打开自动 mute 的宏

84.AC696 系列 PC 模式插上电脑后默认 MIC 音量和系统音量值设置 -----WTS
20201103

```
1005
1006     i = tptr - ptr;
1007     return i;
1008 }
1009 u32 uac_register()
1010 {
1011     if (uac_info == NULL) {
1012 #if USB_MALLOC_ENABLE
1013         uac_info = (struct uac_info_t *)malloc(sizeof(struct uac_info_t));
1014         if (uac_info == NULL) {
1015             printf("uac_register err\n");
1016             return -1;
1017         }
1018 #else
1019         memset(&uac_info, 0, sizeof(struct uac_info_t));
1020         uac_info = &uac_info;
1021 #endif
1022     }
1023     uac_info->spk_left_vol = uac_get_spk_vol();
1024     uac_info->spk_right_vol = uac_get_spk_vol();
1025     uac_info->max_vol = 99;
1026     uac_info->min_vol = 0;
1027     uac_info->def_vol = 49;
1028     uac_info->vol_res = 1;
1029     //uac_info->mic_vol = 13 / 14.0 * 100;
1030     uac_info->mic_vol = 80;
1031     uac_info->spk_mute = 0;
1032     uac_info->mic_mute = 0;
1033     return 0;
1034 }
1035 void uac_release(const usb_dev usb_id)
1036 {
```

以下是电脑上 MIC 音量值查看的地方（参考）：



85.AC696 系列 SDK1.1.1 DACR 做 linein 输入时没声音的处理方法 -----LJW 20201103

请按如下修改:

```

C board_ac696x_demo_cfg.h X C audio_config.h
SDK > apps > soundbox > board > br25 > board_ac696x_demo > C board_ac696x_demo_cfg.h > ...
641 // linein配置 //
642 //*****//
643 #define TCFG_LINEIN_ENABLE TCFG_APP_LINEIN_EN // linein使能
644 // #define TCFG_LINEIN_LADC_IDX 0 // linein使用的ladc通道, 对应ladc_list
645 #if (RECORDER_MIX_EN)
646 #define TCFG_LINEIN_LR_CH AUDIO_LIN0L_CH//AUDIO_LIN0_LR
647 #else
648 #define TCFG_LINEIN_LR_CH AUDIO_LIN_DACR_CH ← 按如下修改
649 #endif/*RECORDER_MIX_EN*/
650 #define TCFG_LINEIN_CHECK_PORT IO_PORTB_01 // linein检测IO
651 #define TCFG_LINEIN_PORT_UP_ENABLE 1 // 检测IO上拉使能
652 #define TCFG_LINEIN_PORT_DOWN_ENABLE 0 // 检测IO下拉使能
653 #define TCFG_LINEIN_AD_CHANNEL NO_CONFIG_PORT // 检测IO是否使用AD检测
654 #define TCFG_LINEIN_VOLTAGE 0 // AD检测时的阈值
655 #if(TCFG_MIC_EFFECT_ENABLE)
656 #define TCFG_LINEIN_INPUT_WAY LINEIN_INPUT_WAY_ANALOG
657 #else
658 #if (RECORDER_MIX_EN)
659 #define TCFG_LINEIN_INPUT_WAY LINEIN_INPUT_WAY_ADC//LINEIN_INPUT_WAY_ANALOG
660 #else
661 #define TCFG_LINEIN_INPUT_WAY LINEIN_INPUT_WAY_DAC ← 按如下修改
662 #endif/*RECORDER_MIX_EN*/
663 #endif
664 #define TCFG_LINEIN_MULTIPLEX_WITH_FM DISABLE // linein 脚与 FM 脚复用
665 #define TCFG_LINEIN_MULTIPLEX_WITH_SD DISABLE // linein 检测与 SD cmd 复用
666 #define TCFG_LINEIN_SD_PORT 0// 0:sd0 1:sd1 //选择复用的sd
667

```

86.AC696 系列 SDK1.1.1 打开 PC 模式时某些 U 盘无法读取的处理方法 -----LJW 20201103

请按如下修改: (该方法同样适用 695 系列 SDK)

```

C board_ac696x_demo_cfg.h C usb_common_def.h
SDK > apps > common > usb > C usb_common_def.h > ...
58 #define TCFG_OTG_MODE_HOST 0
59 #endif
60
61 #define TCFG_OTG_SLAVE_ONLINE_CNT 50//2 ← 改为50
62 #define TCFG_OTG_SLAVE_OFFLINE_CNT 20//1 ← 改为20
63
64 #define TCFG_OTG_HOST_ONLINE_CNT 2

```

```
board_ac696x_demo_cfg.h  effect_reg.c  lib_media_config.c  usb_common_def.h
SDK > apps > soundbox > log_config > lib_media_config.c > config_mp3_dec_use_malloc
59 #endif
60
61 const int config_mixer_src_en = 1;
62
63
64 #if TCFG_BLUETOOTH_BACK_MODE
65 const int config_mp3_dec_use_malloc = 1;
66 const int config_mp3pick_dec_use_malloc = 1;
67 const int config_wma_dec_use_malloc = 1;
68 const int config_wmapick_dec_use_malloc = 1;
69 const int config_m4a_dec_use_malloc = 1;
70 const int config_m4apick_dec_use_malloc = 1;
71 const int config_wav_dec_use_malloc = 1;
72 const int config_alac_dec_use_malloc = 1;
73 const int config_dts_dec_use_malloc = 1;
74 const int config_amr_dec_use_malloc = 1;
75 const int config_flac_dec_use_malloc = 1;
76 const int config_ape_dec_use_malloc = 1;
77 const int config_aac_dec_use_malloc = 1;
78 const int config_aptx_dec_use_malloc = 1;
79 const int config_midi_dec_use_malloc = 1;
80 #else
81 const int config_mp3_dec_use_malloc = 1;
82 const int config_mp3pick_dec_use_malloc = 0;
83 const int config_wma_dec_use_malloc = 0;
```

改为1

88.AC696 系列 SDK1.1.1 单声道开启人声消除无效果的处理方法

公版 SDK 配置

```
#define TCFG_AUDIO_DAC_CONNECT_MODE DAC_OUTPUT_MONO_L
```

```
#define AUDIO_VOCAL_REMOVE_EN 1
```

后，人声消除无效果，可按以下修改处理。

```
SDK > cpu > br25 > audio_dec > audio_dec.c > audio_output_channel_num(void)
25 #include "application/audio_vocal_remove.h"
26 #include "audio_dongle_codec.h"
27 #include "channel_switch.h"
28 #include "audio_base.h"
29
30 #if TCFG_USER_TWS_ENABLE
31 #include "bt_tws.h"
```


新建头文件 SDK\include_lib\media\media_develop\media\channel_switch.h

```
#ifndef CHANNEL_SWITCH_H
#define CHANNEL_SWITCH_H

#include "media/audio_stream.h"

struct channel_switch {
    u8 out_ch_num;
    enum audio_channel out_ch_type;
    struct audio_stream_entry entry;
    u16 dat_len;
    u16 dat_total;
    u16 buf_len;
    u8 buf[0];
};

struct channel_switch *channel_switch_open(enum audio_channel out_ch_type, int buf_len);

void channel_switch_close(struct channel_switch **);

#endif
```

89.AC696 系列 SDK1.1.1 播放 wav 音频时 SD 卡复用 AUX 失败补丁 WTS 20201106

该问题是由于在播放 WAV 音频时，卡无法进入空闲引起。请使用以下网盘链接上的 cpu.a 库，替换后，rebuild 处理：

链接：<https://pan.baidu.com/s/19oK86jwxxyWq1Gaw8G4MjQ>

提取码：6faj

90.AC696 系列 脉冲型驱动功放使用(us 级别) LHY 20201110

```
180 static u16 abd_ctr_sw_timer = 60;
181 static void user_pa_P1ML_D(void)
182 {
183     u16 j = 0;
184     abd_ctr_sw_timer = (u16)((clk_get("sys")/1000000)*0.90)+0.5;
185     r_printf("abd_ctr_sw_timer == %d,clk_get = %d\n",abd_ctr_sw_timer,clk_get("sys"));
186     local_irq_disable();
187     JL_PORTB->OUT |= BIT(6);
188     j = abd_ctr_sw_timer;
189     while(j--){
190         __asm__ volatile ("nop");
191     }
192     JL_PORTB->OUT &= ~BIT(6);
193     j = abd_ctr_sw_timer;
194     while(j--){
195         __asm__ volatile ("nop");
196     }
197     JL_PORTB->OUT |= BIT(6);
198     local_irq_enable();
199 }
200
```

主要是修改这个系数来达到自己所需延时时间，主要是因为时钟是动态的

在变化波形时使用寄存器来操作

需要关中断，否则时间会波动较大

91.AC696 系列 优化插麦克风底噪问题 LHY 20201110

第一种：修改 threshold 的值，原来是-45000，修改时候不要大于-20000

```
cpu > br25 > audio_mic > C effect_reg.c > effect_noise_gate_parm_default
85     .effect_v = EFFECT_PITCH_SHIFT,
86     .formant_shift = 100,
87 };
88
89 const NOISEGATE_PARM effect_noise_gate_parm_default = {
90     .attackTime = 300,
91     .releaseTime = 5,
92     .threshold = -25000,
93     .low_th_gain = 0,
94     .sampleRate = MIC_EFFECT_SAMPLERATE,
95     .channel = 1,
96 };
97
```

第二种：直接从数据流入手去清空数据

主要是先打印下 in 指针中的数据,将 in 转换为 s16 类型指针, outlen 单位是字节不是 s16,所以需要自己去计算出数据个数,打印在插麦克风并没有播歌情况下所有数据大小,取最大的值(绝对值)作为阈值,在 mic_effect_effect_run 执行开始时候去检查数据,如果数据都在阈值范围,直接将 in 中所有数据清空。

```
运行(R) 终端(T) 帮助(H) mic_effect.c - SDK - Visual Studio Code
i5e_cfg.h board_ac6965e.c mic_effect.c x app_power_manage.c idle.c downl
cpu > br25 > audio_mic > C mic_effect.c > mic_effect_effect_run(void *, void *, void *, u32, u32)
215 /*-----*/
216 /**@brief mic数据流串接入口
217 @param
218 @return
219 @note
220 */
221 /*-----*/
222 /* #define TEST_PORT_IO_PORTA_00 */
223 static u32 mic_effect_effect_run(void *priv, void *in, void *out, u32 inlen, u32 outlen)
224 {
225     struct __mic_effect *effect = (struct __mic_effect *)priv;
226     if (effect == NULL) {
227         return 0;
228     }
229     struct audio_data_frame frame;
230     frame.channel = 1; //
231     /* frame.channel = 2; // */
232     frame.sample_rate = effect->parm.sample_rate;
233     frame.data_len = inlen;
234     frame.data = in;
235     effect->out_len = 0;
236     effect->process_len = inlen;
237     mic_effect_parm_update(effect); //更新参数
238     mic_effect_fade_run(effect); //淡入淡出
239     if (effect->pause_mark) {
240         return outlen;
241     }
242     /* gpio_direction_output(TEST_PORT, 1); */
243     while (1) {
244         /* putchar('A'); */
245         audio_stream_run(&effect->entry, &frame);
246         if (effect->out_len >= effect->process_len) {
247             /* putchar('B'); */
248             break;
249         }
250     }
251     /* gpio_direction_output(TEST_PORT, 0); */
252     return outlen;
253 }
```

92.AC696 系列 高关功放在软关机下的处理 LHY 20201110

由于高关功放需要引脚输出高电平来暂停功放芯片工作降低功耗，之前的做法是进入 idle 假关机，现在直接软关机后再配置相对应引脚设置为所需要的电平

```

board_ac6965e.c
void dac_power_off(void)
进软关机之前默认将IO口都设置成高阻状态, 需要保留原来状态的请修改该函数
board_set_soft_poweroff(void)
extern void dac_power_off(void);
extern void dac_sniff_power_off(void);
void board_set_soft_poweroff(void)
976
977
978
979
980 u32 porta_value = 0xffff;
981 u32 portb_value = 0xfffe;
982
983 key_wakeup_enable();
984 gpio_dir(GPIOA, 0, 16, porta_value, GPIO_OR);
985 gpio_set_pu(GPIOA, 0, 16, ~porta_value, GPIO_AND);
986 gpio_set_pd(GPIOA, 0, 16, ~porta_value, GPIO_AND);
987 gpio_die(GPIOA, 0, 16, ~porta_value, GPIO_AND);
988 gpio_dieh(GPIOA, 0, 16, ~porta_value, GPIO_AND);
989
990 //保留长按Reset Pin - PB1
991 gpio_dir(GPIOB, 1, 15, portb_value, GPIO_OR);
992 gpio_set_pu(GPIOB, 1, 15, ~portb_value, GPIO_AND);
993 gpio_set_pd(GPIOB, 1, 15, ~portb_value, GPIO_AND);
994 gpio_die(GPIOB, 1, 15, ~portb_value, GPIO_AND);
995 gpio_dieh(GPIOB, 1, 15, ~portb_value, GPIO_AND);
996
997
998 gpio_set_pull_up(IO_PORT_DP, 0);
999 gpio_set_pull_down(IO_PORT_DP, 0);
1000 gpio_set_direction(IO_PORT_DP, 1);
1001 gpio_set_die(IO_PORT_DP, 0);
1002 gpio_set_dieh(IO_PORT_DP, 0);
1003
1004 gpio_set_pull_up(IO_PORT_DM, 0);
1005 gpio_set_pull_down(IO_PORT_DM, 0);
1006 gpio_set_direction(IO_PORT_DM, 1);
1007 gpio_set_die(IO_PORT_DM, 0);
1008 gpio_set_dieh(IO_PORT_DM, 0);
1009
1010 extern u8 get_port_mute_ad_status(void);
1011 if(get_port_mute_ad_status() == USER_PA_P2MH){
1012     pa_ex_fun.str1(PA_POWER_OFF);
1013 }
1014
1015 VDDIOW_VOL_SEL(power_param.vddiow_lev);
1016 #if (TCFG_SD0_ENABLE || TCFG_SD1_ENABLE)
1017     sdpg_config(0);
1018 #endif
1019 #if (!USER_WAKEUP_EN)
1020     key_wakeup_disable();
1021 #endif
1022 //dac_power_off();

```

将所需要修改的引脚代码放在在这里

93.AC696 系列音箱 V1.1.1 获取音频数据的方法 FSW 20201110

在 your_data_process 中进行自己的处理

```

audio_dec.c
SDK > cpu > br25 > audio_dec > audio_dec.c > user_prob_handler(audio_stream_entry *, audio_data_frame *)
598
599 void your_data_process(s16 *data, u16 len)
600 {
601     //自己的处理
602 }
603
604 int user_prob_handler(struct audio_stream_entry *entry, struct audio_data_frame *in)
605 {
606     if(in->data_len - in->offset > 0){
607         your_data_process(in->data + in->offset / 2, in->data_len - in->offset);
608     }
609
610     return in->data_len;
611 }
612
613 /*-----*/

```

```

SDK > cpu > br25 > audio_dec > C audio_dec.c > audio_dec_init()
669
670 #if AUDIO_OUTPUT_AUTOMUTE
671     mix_out_automute_open();
672 #endif
673
674 // 数据流串联。可以在mixer和last中间添加其他的数据流，比如eq等
675 u8 entry_cnt = 0;
676 entries[entry_cnt++] = &mixer.entry;
677 mixer.entry.prog_handler = user_prog_handler;
678 #if TCFG_EQ_ENABLE && TCFG_AUDIO_OUT_EQ_ENABLE
679     if (mix_eq_drc) {
680         entries[entry_cnt++] = &mix_eq_drc.entry;
    
```

94.AC696 系列音箱 V1.1.1 添加自定义的数字音量控制的方法 FSW 20201110

可实现在不改变模拟音量的情况下，改变蓝牙、U 盘、SD 卡等数字音源的音量

```

SDK > cpu > br25 > audio_dec > C audio_dec.c > ...
592 void *user_dvol = NULL;
593 void *user_dvol_entry = NULL;
594 void user_digital_vol_set(u8 vol)
595 {
596     audio_dig_vol_set(user_dvol, AUDIO_DIG_VOL_CH(0), vol);
597 }
598
    
```

```

SDK > cpu > br25 > audio_dec > C audio_dec.c > audio_dec_init()
691 #endif
692
693 #if AUDIO_OUTPUT_AUTOMUTE
694     entries[entry_cnt++] = mix_out_automute_entry;
695 #endif
696
697 audio_dig_vol_param user_dig_vol;
698 user_dig_vol.vol_start = DIGITAL_VOL_MAX;
699 user_dig_vol.vol_max = DIGITAL_VOL_MAX;
700 user_dig_vol.ch_total = ch_num;
701 user_dig_vol.fade_en = 1;
702 user_dig_vol.fade_points_step = 5;
703 user_dig_vol.fade_gain_step = 50;
704 user_dig_vol.vol_list = user_dig_vol_table;
705 user_dvol = audio_dig_vol_open(&user_dig_vol);
706 user_dvol_entry = audio_dig_vol_entry_get(user_dvol);
707 entries[entry_cnt++] = user_dvol_entry;
708
709 entries[entry_cnt++] = &dac_hdl.entry;
710
    
```

这个值是表的大小减一

这个表可以自定义

SDK > cpu > br25 > audio_dec > C audio_dec.c > [e] user_dig_vol_table

```
553 struct channel_switch *ch_switch;//声道变换
554 #endif
555
556 #define DIGITAL_VOL_MAX 31
557 const u16 user_dig_vol_table[DIGITAL_VOL_MAX + 1] = {
558     0 , //0
559     93 , //1
560     111 , //2
561     132 , //3
562     158 , //4
563     189 , //5
564     226 , //6
565     270 , //7
566     323 , //8
567     386 , //9
568     462 , //10
569     552 , //11
570     660 , //12
571     789 , //13
572     943 , //14
573     1127, //15
574     1347, //16
575     1610, //17
576     1925, //18
577     2301, //19
578     2751, //20
579     3288, //21
580     3930, //22
581     4698, //23
582     5616, //24
583     6713, //25
584     8025, //26
585     9592, //27
586     11466, //28
587     15200, //29
588     16000, //30
589     16384 //31
590 };
591
592 void *user_dvol = NULL;
593 void *user_dvol_entry = NULL;
594 void user_digital_vol_set(u8 vol)
595 {
```

95.AC696 系列 SDK1.1.1 上下曲切歌有啞声问题需要更换的库 WTS 20201111

AC696 系列 SDK1.1.1 上下曲切歌有啞声问题需要更换的库:

链接: <https://pan.baidu.com/s/1kA6zg8KrYQcxbYBuNv2Yhg>
提取码: jiel

96.AC696 系列 SDK1.1.1 无法读取 MMC 卡的问题处理方法 LJW 20201116

该问题处理方法请参考第 89 问题点, 替换库

97.AC696 系列 SDK1.1.1 断开对箱后自动回链问题解决方法 LHY 20201117

```
apps > soundbox > task_manager > C app_common.c > app_common_key_msg_deal(sys_event *)
556     }
557     #endif
558     break;
559 #if TCFG_USER_TWS_ENABLE
560     case KEY_USER_TWS:
561         log_info("    KEY_USER_TWS \n");
562         int err = tone_play_with_callback_by_name(TONE_DI, 1, bt_tws_tone_play_end_callback, (void *)IDEX_TONE_DI);
563         if (err) {
564             printf("tws switch\n\n");
565             bt_tws_tone_play_end_callback((void *)IDEX_TONE_DI,0);
566         }
567     }
568     #endif
569 #endif
570     case USER_TWS_PLAY_TONE:
571         puts("USER_TWS_PLAY_TONE\n");
```

```
apps > soundbox > task_manager > C app_common.c > bt_tws_tone_play_end_callback(void *, int)
77     default:
78         break;
79     }
80 }
81 }
82 static void bt_tws_tone_play_end_callback(void *priv, int flag){
83     u32 index = (u32)priv;
84     if (APP_BT_TASK != app_get_curr_task()) {
85         log_error("tone callback task out \n");
86         return;
87     }
88 }
89 switch (index) {
90     case IDEX_TONE_DI:
91         //提示音播放结束
92         printf(" >>>>> tone \n");
93     }
94     if((tws_api_get_tws_state() & TWS_STA_ESCO_OPEN) ||
95        (tws_api_get_tws_state() & TWS_STA_ESCO_OPEN_LINK))
96     ){
97         printf("TWS_STA_OPEN\n");
98         break;
99     }
100 }
101 if(tws_api_get_tws_state() & TWS_STA_SIBLING_DISCONNECTED){
102     printf("    KEY_TWS_SEARCH_PAIR \n");
103     bt_tws_start_search_and_pair();
104 }else if(tws_api_get_tws_state() & TWS_STA_SIBLING_CONNECTED){
105     printf("    KEY_TWS_SEARCH_REMOVE_PAIR \n");
106     #if USER_IR_TWS_SYNC_DEL_INFO_EN
107     bt_tws_api_push_cmd(USER_SYNC_CMD_DEL_TWS_INFO, 200);
108     #endif
109 }else{
110     puts(">>>>> error\n");
111 }
112 break;
113 default:
114     break;
115 }
```

这里主要是在对箱前播提示音

对箱连接走正常的函数

断开对箱, 如果直接调用函数会出现从机配对信息未删除的情况, 所以需要走消息延时同步来实现主从机都删除配对信息


```

C bt_tws.c x
apps > soundbox > task_manager > bt > C bt_tws.c > bt_tws_remove_tws_pair()
802
803 /-----*/
804 u8 bt_tws_remove_tws_pair()
805 {
806     int state = get_bt_connect_status();
807
808     if ((get_call_status() == BT_CALL_ACTIVE) ||
809         (get_call_status() == BT_CALL_OUTGOING) ||
810         (get_call_status() == BT_CALL_ALERT) ||
811         (get_call_status() == BT_CALL_INCOMING)) {
812         return 0; //通话过程不允许
813     }
814
815 #if (CONFIG_TWS_USE_COMMON_ADDR && !USER_TWS_ADD_DELL_TWS_INFO)
816     if (tws_api_get_tws_state() & TWS_STA_PHONE_CONNECTED) {
817         return 0;
818     }
819 #endif

```

99.AC696/AC695 修改蓝牙配对码 PINCODE 20201118 YWP

/*提供接口外部设置配对方式*/

extern void __set_simple_pair_flag(u8 flag);

```

00277:         /* bt_set_tx_power(9); //ble txpwr level:0~9 */
00278:         memcpy(tmp_ble_addr, (void *)bt_get_mac_addr(), 6);
00279: #else
00280:         lib_make_ble_address(tmp_ble_addr, (void *)bt_get_mac_addr);
00281: #endif //
00282:         le_controller_set_mac((void *)tmp_ble_addr);
00283:         printf("\n-----edr + ble 's address-----");
00284:         printf_buf((void *)bt_get_mac_addr(), 6);
00285:         printf_buf((void *)tmp_ble_addr, 6);
00286:     }
00287: #endif // TCFG_USER_BLE_ENABLE
00288: /*
00289:     0 需要书输入PINCODE
00290:     1 简易配对
00291: */
00292:     __set_simple_pair_flag(0);
00293:
00294:
00295: } ? end bt_function_select_init ?
00296:
00297:
00298: /*

```

bt_function_select_init

```

Audio_dec.c | Music_player.c | Board_config.h | App_config.h | Sdk.elf.resolution.txt | Linei
fig.c | Bt.c | Update.c | User_cfg.c | 全部关闭
00112:     return (const char *) (bt_cfg.edr_name);
00113: }
00114:
00115: const char *bt_get_pin_code()
00116: {
00117:     return "6655";
00118: }
00119:
00120: extern STATUS_CONFIG status_config;
00121: extern struct charge_platform_data charge_data;
    
```

100.AC696X 音箱 SDK 自动调节 VDDIO 电压 20201119 YP

自动调节 vddio 电压等级，提供对应的接口，如下：有一点要注意，vddio 的电压会影响到电池电量检测，另外这个电压也要比 DACVDD 大最少 0.3V，否则会引发底噪。

```

42 u8 get_self_battery_level(void);
43 u8 get_cur_battery_level(void);
44 void vbat_check_init(void);
45 void vbat_timer_delete(void);
46 void vbat_check(void *priv);
47 bool vbat_is_low_power(void);
48 void check_power_on_voltage(void);
49 void app_reset_vddiom_lev(u8 lev);
50
51
52 #endif
53
<code>\ac696n_soundbox_sdk_v1.1.1\SDK\apps\soundbox\include\app_power_manage.h  cpp  utf-8[un
429     if(cnt >= 6){^M
430         cnt = 0;^M
431         if(!auto_sel_vddio_flag){^M
432             auto_sel_vddio_flag = 1;^M
433             r_printf("---%s--bat_lev = %d\n", __func__, bat_lev);^M
434             g_printf("---low pwr , auto sel vddio 3.2V\n");^M
435             app_reset_vddiom_lev(UDDIOM_VOL_32V);^M
436         }^M
437     }^M
438 }else{^M
439     cnt++;^M
440     if(cnt >= 6){^M
441         cnt = 0;^M
442         if(auto_sel_vddio_flag == 1){^M
443             auto_sel_vddio_flag = 0;^M
444             r_printf("---%s--bat_lev = %d\n", __func__, bat_lev);^M
445             g_printf("---charge online , auto sel vddio 3.4V\n");^M
446             app_reset_vddiom_lev(UDDIOM_VOL_34V);^M
447         }^M
448     }^M
449 }^M
450 ^M
451 #endif // M_AUTO_SET_UDDIO_LEVEL_EN^M
452 }^M
    
```

vddio电压设置的接口

使用的方法

101.AC696X 音箱 SDK 内置充电有时会处于很小(20mA)左右的处理方法 20201119 YP

版权所有，侵权必究

128

音箱 SDK 内置充电最大可以用 300mA 电流档位，但是有些时候比如在电池处于低电的时候，充电的电流可能只有 20mA 左右，充电比较慢，是因为 SDK 默认初始化的充电电流设置为 20mA，可以根据自身的需求更改这个初始值，**这个值设定为电池容量的 1/10C 的比例，根据电池标定来设置。**

```
1 charge.c
444
445 static void charge_config(void)
446 {
447     u8 charge_4202_trim_val = CHARGE_FULL_U_4202;
448     u8 offset = 0;
449     u8 charge_full_v_val = 0;
450
451     if (get_vbat_trim() == 0xf) {
452         log_info("vbat not trim, use default config!!!!!!");
453     } else {
454         charge_4202_trim_val = get_vbat_trim(); //4.2V对应的trim出来的实际档位
455     }
456
457     log_info("charge_4202_trim_val = %d\n", charge_4202_trim_val);
458
459     if (__this->data->charge_full_U >= CHARGE_FULL_U_4202) {
460         offset = __this->data->charge_full_U - CHARGE_FULL_U_4202;
461         charge_full_v_val = charge_4202_trim_val + offset;
462         if (charge_full_v_val > 0xf) {
463             charge_full_v_val = 0xf;
464         }
465     } else {
466         offset = CHARGE_FULL_U_4202 - __this->data->charge_full_U;
467         if (charge_4202_trim_val >= offset) {
468             charge_full_v_val = charge_4202_trim_val - offset;
469         } else {
470             charge_full_v_val = 0;
471         }
472     }
473
474     log_info("charge_full_v_val = %d\n", charge_full_v_val);
475
476     CHARGE_FULL_U_SEL(charge_full_v_val);
477     CHARGE_FULL_mA_SEL(__this->data->charge_full_mA);
478     /* CHARGE_mA_SEL(__this->data->charge_mA); */
479     // CHARGE_mA_SEL(CHARGE_mA_20);
480     CHARGE_mA_SEL(CHARGE_mA_100);
481 }
482
```

该值为电池容量只的1/10C

102.AC696X 音箱 SDK 1.1.1 硬件微妙延时 SQ 20201123

注：1、必需以晶振为时钟源 2、分频系数根据延时最小单位自己设置，分频设置参考用户手册中的寄存器说明

以定时器 3 做 10 微妙为单位的延时为例：

```
#include "debug.h"
```

```
#define USER_UDELAY_TIMER JL_TIMER3//微妙延时使用的定时器 地址
```

```
#define USER_UDELAY_TIMER_IRQ IRQ_TIME3_IDX//微妙延时使用的定时器 终端号
```

```
//只初始化一次
```

```
static void user_10us_delay_init(void){
```

```
    bit_clr_ie(USER_UDELAY_TIMER_IRQ);
}
//阻塞延时 10us
__attribute__((weak)) void user_udelay(u32 usec){
    USER_UDELAY_TIMER->CON = BIT(14); //0x4000;
    USER_UDELAY_TIMER->CNT = 0;
    USER_UDELAY_TIMER->PRD = (24/4) * usec; //延时时间=时钟源频率/分频系数*计数次数
    USER_UDELAY_TIMER->CON = BIT(0)|BIT(3)|BIT(4); //BIT(0) 定时计数模式 BIT(3): 晶振为时钟源
    BIT(4): 4 分频
    while ((USER_UDELAY_TIMER->CON & BIT(15)) == 0);
    USER_UDELAY_TIMER->CON = BIT(14);
}

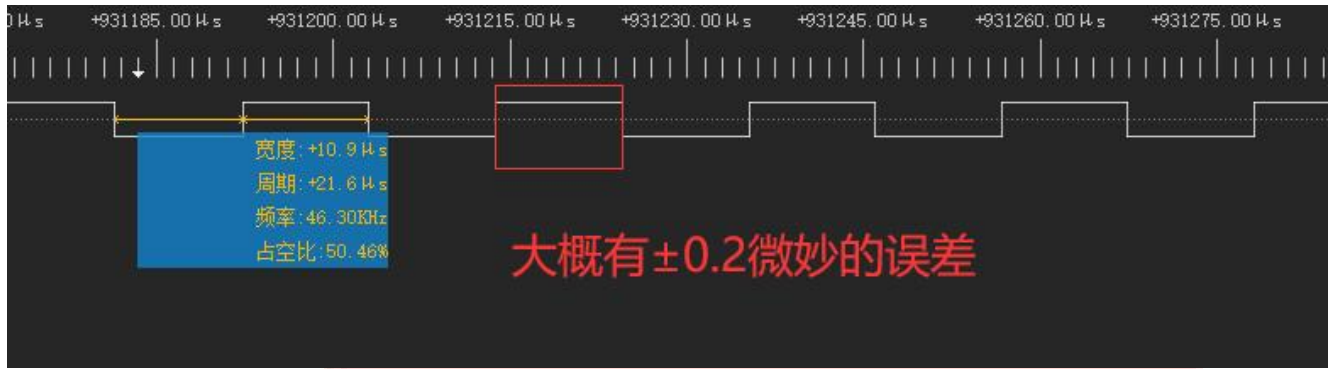
//使用 PB7 翻转电平 来测试延时是否正常
#define IO_DEBUG_OUTPUT(i,x)    {JL_PORT##i->DIR &= ~BIT(x), JL_PORT##i->DIE &= ~BIT(x);
JL_PORT##i->OUT ^= BIT(x);}
#define IO_DEBUG_TFLIP(i,x)    {JL_PORT##i->OUT ^= BIT(x);}
// #define IO_POR B,1

__attribute__((weak)) void user_100us_test(void){
    local_irq_disable();
    //IO_DEBUG_OUTPUT(B,7);
    //user_udelay(10);

    for(int i = 0; i < 10; i++){
        IO_DEBUG_TFLIP(B,7); //一定要使用寄存器的方式来控制 io 的输出高低
        user_udelay(10);
    }
    //user_udelay(10);
    //IO_DEBUG_TFLIP(B,7);

    local_irq_enable();
}

void user_fun(void){
    user_10us_delay_init();
    IO_DEBUG_OUTPUT(B,7);
    sys_hi_timer_add(NULL, user_100us_test, 200);
}
```



阻塞延时时间 这里为10微妙

执行这些代码所需时间 0.75微妙

一定要使用寄存器的方式来控制io的输出高低

宽度: +10.75 μs
周期: +21.45 μs
频率: 46.62KHz
占空比: 50.12%

103.耳机工程升级到 SDK020, 发现同样的参数, 回音比以前的版本明显, 调回音压制参数效果不明显的处理方法 XNW 2020112

按照一下图片参数处理:

```

app_config.h x board_ac6963a_tws_cfg.h x clock.h x key_event_deal.c x app_charge.c x aec_user.c x
88     .AGC_min_gain = AGC_MIN_GAIN,
89     .AGC_threshold = AGC_NEAREND_THR,
90     .AGC_fade = AGC_FADE_SPEED,
91     .ES_AggressFactor = -NLP_AGGRESS_FACTOR,
92     .ES_MinSuppress = NLP_SUPPRESS_FACTOR,
93     .ANS_AggressFactor = 1.25f, /*范围:.....1~2,动态调整,越大越强(1.25f)*/
94     .ANS_MinSuppress = 0.14f, /*范围:.....0~1,静态定死最小调整,越小越强(0.09f)*/
95     #if (defined CONFIG_ANS_V2 || defined CONFIG_AEC_MUX)
96     .hw_delay_offset = 75,
97     #else
98     .hw_delay_offset = 140,
99     #endif
100    #if TCFG_AEC_SIMPLEX
101    .wn_en = 1,
102    #else
103    .wn_en = 0,
104    #endif
105    .wn_gain = 331
    
```

104.696x 音箱 SDK 添加定时器扫描函数后有几率搜不到蓝牙问题 20201124 YP

AC696X 音箱的 SDK，在开机就需要跑一些定时器的扫描函数的，添加之后会出现有几率进入蓝牙模式之后无法搜到蓝牙名字的问题，原因是开机添加的定时器(sys_timer_add 等之类的)会触发消息，把消息池撑满，导致进入蓝牙的时候，造成蓝牙的一些事件消息无法接收处理，出现丢事件消息无法初始化蓝牙，从而搜不到蓝牙名字，对于此问题的处理是需要更换一下添加定时器扫描函数的地方，避免消息事件被频繁触发堆满消息池。对于以上的修改，可参考下图

```

994     if (0 == (timer_cnt % 10))
995     {
996     }
997     }
998     }
999     if (0 == (timer_cnt % 20)) //500ms
1000    {
1001    /**< 这里是自动mute功放的定时器轮训函数，500ms一次，为了能够自动mute和提示音的时候不自动mute的条件*/
1002    if ( (g_tPrj.var.ucFLAG_power_off_flag != E_FLAG_TRUE)&&(tone_get_status() == 0))
1003    { g_tPrj.Func.pa_auto_mute_det(M_PA_AUTO_MUTE_DET_MAX_CNT);
1004    }
1005    }
1006    // return;
1007    }
1008    }
1009    }
1010    //static^M
1011    void sys_user_timer_scan(void)
1012    {
1013    static u16 cnt = 0;
1014    cnt++;
1015    if(cnt%2==0){
1016    g_tPrj.func.timer_scan_func(cnt);
1017    }else if(cnt>1000){
1018    cnt = 0;
1019    }
1020    }
1021    }
1022    }
1023    }
1024    }
1025    /* ***** static void user_io_func_deal(u8 action) *****
1026    ***** */
1027    static void user_io_func_deal(u8 action)
1028    {
1029    r_printf("%s->%d\n", __func__, action);
1030    switch(action)
1031    {
1032    case E_PWR_IO_INIT:
1033    r_printf("E_PWR_IO_INIT\n");
1034    }
1035    }
1036    }
1037    }
1038    }
1039    }
1040    }
1041    }
1042    }
1043    }
1044    }
1045    }
1046    }
1047    }
1048    }
1049    }
1050    }
1051    }
1052    }
1053    }
1054    }
1055    }
1056    }
1057    }
1058    }
1059    }
1060    }
1061    }
1062    }
1063    }
1064    }
1065    extern void sys_user_timer_scan(void);
1066    sys_timer_add(NULL, sys_user_timer_scan, 20);
1067    extern void led_timer_scan(void);
1068    // sys_timer_add(NULL, led_timer_scan, 30);
1069    extern void user_rgb_init(void);
1070    user_rgb_init();
1071    os_sem_post(&__this->sem);
1072    }
1073    while (1) {
1074    res = os_task_pend("taskq", msg, ARRAY_SIZE(msg));
1075    switch (res) {
    
```

105.696x 音箱 SDK 蓝牙模式下打开 LINEIN 进行叠加 20201127 LHY

1、走数字通道(因为 696 只有一路 LADC,所以 linein 无法走数字立体音, 只能走单声道)

```
724: int linein_dec_open(u8 source, u32 sample_rate)
725: {
726:     int err;
727:     struct linein_dec_hdl *dec;
728:     dec = zalloc(sizeof(*dec));
729:     if (!dec) {
730:         return -ENOMEM;
731:     }
732:     linein_dec = dec;
733:     switch (source) {
734:     case BIT(0):
735:     case BIT(1):
736:         dec->channel = 1;
737:         break;
738:     case BIT(2):
739:     case BIT(3):
740:         dec->channel = 1;
741:         break;
742:     default:
743:         log_e("Linein can only select a single channel\n");
744:         ASSERT(0, "err\n");
745:         break;
746:     }
747:     dec->source = source;
748:
749:     dec->sample_rate = sample_rate;
750:
751:     dec->coding_type = AUDIO_CODING_PCM;
752:     dec->wait.priority = 2;
753:     dec->wait.preemption = 0;
754:     #if SOUND_CARD_ENABLE
755:     dec->wait.protect = 1;
756:     #endif
757:     dec->wait.handler = linein_wait_res_handler;
758:     clock_add(dec->coding_type);
759:     err = audio_decoder_task_add_wait(&decode_task, &dec->wait);
760:     return err;
761: } // end linein_dec_open
```

在linein解码函数加入该段代码,并在蓝牙模式下调用

2、走模拟通道, 在蓝牙模式下需要打开 linein 通道位置加入下面这段代码

```
100
101     if (TCFG_LINEIN_LR_CH & (BIT(0) | BIT(1))) {
102         audio_linein0_open(TCFG_LINEIN_LR_CH, 1);
103     } else if (TCFG_LINEIN_LR_CH & (BIT(2) | BIT(3))) {
104         audio_linein1_open(TCFG_LINEIN_LR_CH, 1);
105     } else if (TCFG_LINEIN_LR_CH & (BIT(4) | BIT(5))) {
106         audio_linein2_open(TCFG_LINEIN_LR_CH, 1);
107     }
```

106.696x 音箱 SDK 搜不到蓝牙问题（补充） 20201127 LZK

【具体现象】：第一次上电开机或者切到蓝牙模式下, 搜索不到蓝牙名。查看打印消息发现进入蓝牙模式后**没有 BT_STATUS_INIT_OK 的打印**出来

```
/* 蓝牙初始化完成、链接、通话播放等状态 */
static int bt_connection_status_event_handler(struct bt_event *bt)
{
    log_debug("-----bt_connection_status_event_handler %d"

    if (bt_status_event_filter(bt) == false) {
        return false;
    }

    switch (bt->event) {
    case BT_STATUS_EXIT_OK:
        log_info("BT_STATUS_EXIT_OK\n");
        break;
    case BT_STATUS_INIT_OK:
        log_info("BT_STATUS_INIT_OK\n");
        bt_status_init_ok(bt);
        break;
    case BT_STATUS_START_CONNECTED:
```

【修改方法】

1.优先看第 104 点 “696x 音箱 SDK 添加定时器扫描函数后有几率搜不到蓝牙问题”

2.如果根据第 104 点修改后，还是会出现。可尝试添加重发机制如下。（以下函数放在 bt.c 文件里面即可。）

3.添加完后，查看打印有 BT_STATUS_INIT_OK 打印出来即可。

```
//蓝牙初始化成功消息 1 秒重发
void bt_init_ok_msg_resend(void *msg)
{
    printf("bt_init_ok_msg_resend%d\n",get_bt_init_status());
    if(get_bt_init_status()==0){
        struct sys_event e;
        e.type = SYS_BT_EVENT;
        e.arg = (void *)SYS_BT_EVENT_TYPE_CON_STATUS;
        e.u.dev.event = BT_STATUS_INIT_OK;
        e.u.dev.value = (int)0;
        sys_event_notify(&e);
    }
}

int sys_event_recode(struct sys_event *event)
{
    printf("event->type:%d\n", event->type);
    if(event->type == SYS_BT_EVENT){
        struct bt_event *bt = &event->u.bt;
        printf("event:%d\n",bt->event);
        if(bt->event == BT_STATUS_INIT_OK){
            sys_timeout_add(NULL,bt_init_ok_msg_resend,1000);
        }
    }
    return 0;
}
```

107.AC696X 大音量播放歌曲声音会抖 20201201 HJY

【原因】默认的 SDK DAC 的 bise 电流档位选择为最低档位，大音量播歌的时候会因为驱动电流不够，导致声音不正常。

【修改方法】在各板级的.c 文件中修改.lpf_isel 值。

```
/****** DAC *****/
#if TCFG_AUDIO_DAC_ENABLE
struct dac_platform_data dac_data = {
    .ldo_volt = TCFG_AUDIO_DAC_LDO_VOLT, //DACVDD等级,需要根据具体硬件来设置(高低压)可选:2.4V/2.5V/2.6V/2.7V/2.8V/2.9V/3.0V/3.1V
    .vcmo_en = 0, //是否打开VCOMO
    .output = TCFG_AUDIO_DAC_CONNECT_MODE, //DAC输出配置,和具体硬件连接有关,需根据硬件来设置
    .ldo_isel = 0,
    .ldo_fb_isel = 0,
    .lpf_isel = 0x1, //将这个值改大
    .sys_vol_type = SYS_VOL_TYPE, //系统音量选择:模拟音量/数字音量,调节时调节对应的音量
    .max_ana_vol = MAX_ANA_VOL, //模拟音量最大等级
    /*.linein_sel = LADC_LINE1_MASK, */
};
#endif
/****** ADC *****/
```

```

C app_config.h C board_ac696x_demo_cfg.h C board_config.h C audio_platform.h X C board_ac6963a_tws_cfg.h
ac696n_earphone_v0.2.0 - Export > sdk > include_lib > media > cpu > br25 > asm > C audio_platform.h > dac_platform_data > ldo_fb_isel
35 e DAC_OUTPUT_DUAL_LR_DIFF 6 //双声道差分
36 e DAC_OUTPUT_FRONT_LR_REAR_L 7 //三声道单端输出 前L+前R+后L (不可设置vcmo公共端)
37 e DAC_OUTPUT_FRONT_LR_REAR_R 8 //三声道单端输出 前L+前R+后R (可设置vcmo公共端)
38 e DAC_OUTPUT_FRONT_LR_REAR_LR 9 //四声道单端输出
39
40 dac_platform_data
41 2 output : 4; //DAC输出模式
42 2 ldo_volt : 4; //DACVDD_LDO电压档选择
43 2 ldo_isel : 4; //LDO偏置电流选择档位, 0:5u, 1:10u, 2:15u, 3:20u, 4:25u, 5:30u, 6:35u, 7:40u 档位的选择范围
44 2 lpf_isel : 4; //LPF bias电流选择, 0:无, 1:0.3125u, 2:0.625u, 3:0.9375, 4:1.25u, 5:1.5625, 6:1.875u, 7:2.1875u, 8:2.5u, 9:2.8125u, 10:3.125u, 11:3.4375u, 12:3.75u
45 2 ldo_fb_isel : 2; //LDO负载电流选择, 0:15u, 1:48u, 2:81u, 3:114u
46 2 vcmo_en : 1; //VCMO直推使能
47 2 keep_vcmo : 1;
48

```

108.AC696 系列 SDK1.1.1 开启内置充电后红外遥控不起作用的处理方法 LJW 20201203

```

C board_ac696x_demo_cfg.h C irflt.c C log.h C adkey_table.c C app_config.h C board_ac6
SDK > cpu > br25 > C irflt.c > IR_TIME_REG
1 #include "asm/includes.h"
2 #include "asm/irflt.h"
3 #include "timer.h"
4 #include "generic/gpio.h"
5
6 #define ir_log log_d
7
8 //红外定时器定义
9 #define IR_TIMER TIMER1
10 #define IR_IRQ_TIME_IDX IRQ_TIME1_IDX
11 #define IR_TIME_REG JL_TIMER1
12

```

更换成空闲的定时器

109AC696 系列 SDK1.1.1 录音的同时 DAC 输出录音数据的处理方法 LJW 20201203

1.请按如下修改:

```

C board_ac696x_demo_cfg.h C mic_stream.c C record.c X C audio_enc_recorder.c C audio_enc_file.c
apps > soundbox > task_manager > record > C record.c > record_key_pp0
111 /*
112 /*-----
113 static void record_key_pp()
114 {
115     if (recorder_is_encoding()) {
116         log_i("mic record stop && replay\n");
117         record_mic_stop();
118         mic_effect_stop();
119         record_file_play();
120     } else {
121         record_file_close();
122         mic_effect_start();
123         record_mic_start();
124         log_i("mic record start\n");
125     }
126 }

```

添加这两句

2.修改录音的采样率

```

C board_ac696x_demo_cfg.h  C mic_effect.c  C record.c  X  C audio_enc_recoder.c  C audio_enc_file.c
apps > soundbox > task_manager > record > C record.c > record_mic_start(void)
61 static void record_mic_start(void)
62 {
63     struct record_file_fmt fmt = {0};
64     /* char logo[] = {"sd0"}; */ //可以指定设备
65     char folder[] = {REC_FOLDER_NAME}; //录音文件夹名称
66     char filename[] = {"AC69****"}; //录音文件名,不需要加后缀,录音接口会根据编码格式添加后缀
67
68 #if (TCFG_NOR_REC)
69     char logo[] = {"rec_nor"}; //外挂flash录音
70 #elif (FLASH_INSIDE_REC_ENABLE)
71     char logo[] = {"rec_sdfile"}; //内置flash录音
72 #else
73     char *logo = dev_manager_get_phy_logo(dev_manager_find_active(0)); //普通设备录音,获取最后活动
74 #endif
75
76     fmt.dev = logo;
77     fmt.folder = folder;
78     fmt.filename = filename;
79     fmt.coding_type = AUDIO_CODING_MP3; //编码格式: AUDIO_CODING_WAV, AUDIO_CODING_MP3
80     fmt.channel = 1; //声道数: 1: 单声道 2: 双声道
81     fmt.sample_rate = 44100; //采样率: 8000, 16000, 32000, 44100
82     fmt.cut_head_time = 300; //录音文件去头时间,单位ms
83     fmt.cut_tail_time = 300; //录音文件去尾时间,单位ms

```

3.

修改录音采样率与混响一致

4.3.配置 mic_effect 为 REVERB

```

C board_ac696x_demo_cfg.h  C mic_effect.c  C record.c  C audio_enc_recoder.c  C audio_enc_file.c
apps > soundbox > board > br25 > board_ac696x_demo > C board_ac696x_demo_cfg.h > TCFG_MIC_EFFECT_SEL
530 #define TCFG_ONLINE_ENABLE 1
537 #endif
538
539 #define MIC_EFFECT_REVERB 1
540 #define MIC_EFFECT_ECHO 0
541 #define TCFG_MIC_EFFECT_SEL MIC_EFFECT_REVERB
542 // #define TCFG_MIC_EFFECT_SEL MIC_EFFECT_ECHO
543
544 #if TCFG_MIC_EFFECT_ENABLE

```

5.

修改此处, 选用REVERB, 不要使用ECHO

6.4.关掉混响的效果

```

C adkey_table.c  C effect_reg.c  X  C loud_speaker.c  C key_event_deal.h  C board_ac696x_der
cpu > br25 > audio_mic > C effect_reg.c > MIC_EFFECT_CONFIG
6 #if (TCFG_MIC_EFFECT_SEL == MIC_EFFECT_REVERB)
7 /* #if (0) */
8 #define MIC_EFFECT_CONFIG 0 /*\
9 |
10 | BIT(MIC_EFFECT_CONFIG_REVERB) \
11 | BIT(MIC_EFFECT_CONFIG_PITCH) \
12 | BIT(MIC_EFFECT_CONFIG_NOISEGATE) \
13 | BIT(MIC_EFFECT_CONFIG_DVOL) \
14 | BIT(MIC_EFFECT_CONFIG_EQ) \
15 | BIT(MIC_EFFECT_CONFIG_FILT) */

```

7.

注释该部分内容

8.

110.AC696 系列 SDK1.1.1 在 music 模式下开混响，喊麦会卡音的处理方法 LJW 20201203

```
C app_audio.c X
SDK > cpu > br25 > audio_common > C app_audio.c > ...
60
61 #if AUDIO_OUTPUT_ONLY_DAC
62 s16 dac_buff[4 * 1024] SEC(.dac_buff);
63 #endif
64
```

尽量不要改小，默认4*1024

111.AC696 系列 SDK1.1.1 外部 EQ 文件替换 EQ 模式中的任何一种模式 HJY 20201203

【修改方法】在 eq_config.c 文件中，以替换 eq_tab_normal 模式为例。

```
C app_config.h C board_ac696x_demo_cfg.h C app_common.c C audio_config.h C adkey_table.c C lib_media_config.c C eq_config.c X C eq_config
SDK > cpu > br25 > audio_effect > C eq_config.c > eq_init(void)
parm.phone_eq_tab = phone_eq_tab_normal;
494 parm.phone_eq_tab_size = ARRAY_SIZE(phone_eq_tab_normal);
495 #endif
496 parm.ul_eq_tab = ul_eq_tab_normal;
497 parm.ul_eq_tab_size = ARRAY_SIZE(ul_eq_tab_normal);
498
499 parm.eq_tool_tab = eq_tool_tab;
500 parm.eq_mode_use_idx = eq_mode_use_idx;
501 parm.eq_type_tab = (void *)get_eq_mode_tab();
502 parm.type_num = EQ_MODE_MAX;
503 parm.section_max = EQ_SECTION_MAX;
504
505 parm.file_en = 1;
506
507 EQ_CFG *eq_cfg = eq_cfg_open(&parm);
508 user_eq_cfg = eq_cfg;
509 if (eq_cfg) {
510 eq_cfg->eq_type = EQ_TYPE_MODE_TAB;
511 memcpy(eq_tab_normal,eq_cfg->cfg_parm[song_eq_mode].song_eq_parm.parm.seg,sizeof(EQ_CFG_SEG)*SECTION_MAX);
512
513 for(int i = 0;i<SECTION_MAX;i++){
514 printf(">>>>> eq index %d freq %d gain %d\n",eq_tab_normal[i].index,eq_tab_normal[i].freq,eq_tab_normal[i].gain>>20);
515 }
516 #if APP_ONLINE_DEBUG
517 if (eq_cfg->app) {
518 app_online_db_register_handle(DB_PKT_TYPE_EQ, eq_app_online_parse);
519 }
520 #endif
521
522 for (int i = 0; i < eq_cfg->mode_num; i++) {
523 if (eq_cfg->eq_type == EQ_TYPE_MODE_TAB) {
524 set_global_gain(eq_cfg, i, 0);
525 drc_default_init(eq_cfg, i);
526 }
527 }
528 }
529 return 0;
530 }
531 __initcall(eq_init);
532
```

该参数为替换模式的数组名

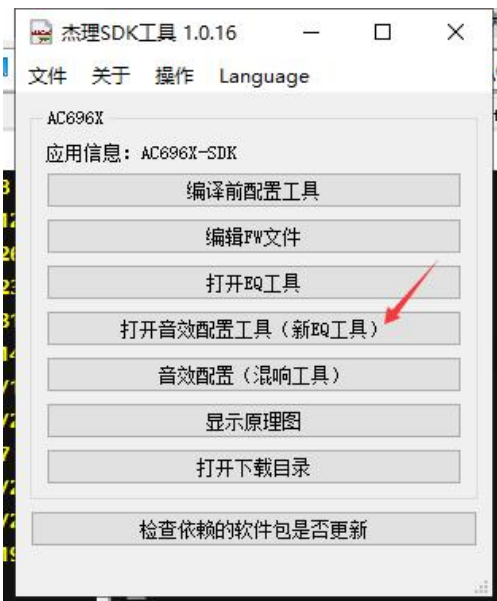
该打印为替换后外部EQ文件频点设定参数

```
#if !TCFG_USE_EQ_FILE
struct eq_seg_info eq_tab_normal[] 这里的const关键字要去掉
{
    {0, EQ_IIR_TYPE_BAND_PASS, 31, 0 << 20, (int)(0.7f * (1 << 24))},
    {1, EQ_IIR_TYPE_BAND_PASS, 62, 0 << 20, (int)(0.7f * (1 << 24))},
    {2, EQ_IIR_TYPE_BAND_PASS, 125, 0 << 20, (int)(0.7f * (1 << 24))},
    {3, EQ_IIR_TYPE_BAND_PASS, 250, 0 << 20, (int)(0.7f * (1 << 24))},
    {4, EQ_IIR_TYPE_BAND_PASS, 500, 0 << 20, (int)(0.7f * (1 << 24))},
    {5, EQ_IIR_TYPE_BAND_PASS, 1000, 0 << 20, (int)(0.7f * (1 << 24))},
    {6, EQ_IIR_TYPE_BAND_PASS, 2000, 0 << 20, (int)(0.7f * (1 << 24))},
    {7, EQ_IIR_TYPE_BAND_PASS, 4000, 0 << 20, (int)(0.7f * (1 << 24))},
    {8, EQ_IIR_TYPE_BAND_PASS, 8000, 0 << 20, (int)(0.7f * (1 << 24))},
    {9, EQ_IIR_TYPE_BAND_PASS, 16000, 0 << 20, (int)(0.7f * (1 << 24))},
}
```

112.AC696 系列 SDK1.1.1 外部添加双 EQ 文件的方法

HJY 20210419

【修改方法】使用新 EQ 工具



```
运行(R) 终端(T) 帮助(H) eq_config.c - ac696n_soundbox_sdk_v1.1.1 - Visual Studio Code
media_config.c 2 C eq_config.c 3 X C board_ac696x_demo.c 3 C app_common.c 5 C
SDK > cpu > br25 > audio_effect > C eq_config.c > drc_default_init(EQ_CFG *, u8)
446 }
447
...
448 EQ_CFG *user_eq_cfg =NULL;
449
450
451 int eq_init(void)
452 {
453     audio_eq_init();
454     eq_adjust_parm parm = {0};
455     #if TCFG_EQ_ONLINE_ENABLE
456     parm.online_en = 1;
```

```

//*****
//                               EQ配置                               //
//*****
//EQ配置，使用在线EQ时，EQ文件和EQ模式无效。有EQ文件时，默认不用EQ模式切换功能
#define TCFG_EQ_ENABLE           1 //支持EQ功能
#define TCFG_EQ_ONLINE_ENABLE   0 //支持在线EQ模式
#define TCFG_BT_MUSIC_EQ_ENABLE 1 //支持蓝牙音乐EQ
#define TCFG_PHONE_EQ_ENABLE    1 //支持电话近端EQ
#define TCFG_MUSIC_MODE_EQ_ENABLE 1 //支持音乐模式EQ
#define TCFG_LINEIN_MODE_EQ_ENABLE 1 //支持linein近端EQ
#define TCFG_FM_MODE_EQ_ENABLE  0 //支持fm模式EQ
#define TCFG_SPDIF_MODE_EQ_ENABLE 0 //支持SPDIF模式EQ
#define TCFG_PC_MODE_EQ_ENABLE  1 //支持pc模式EQ
#define TCFG_AUDIO_OUT_EQ_ENABLE 0 //mix_out后高低音EQ

#define TCFG_USE_EQ_FILE         1 //离线eq使用配置文件还是默认系数表 1: 使用文件 0 使用默认系数表
#if TCFG_EQ_ONLINE_ENABLE

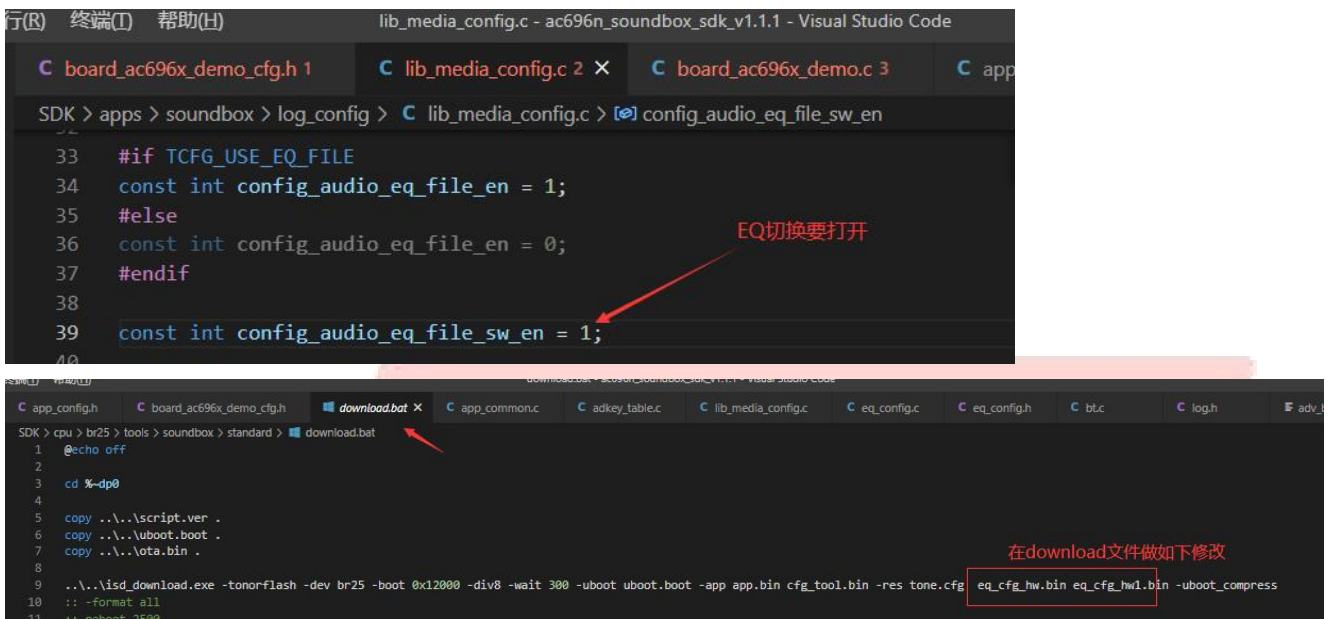
```

板级文件中选择“使用文件”



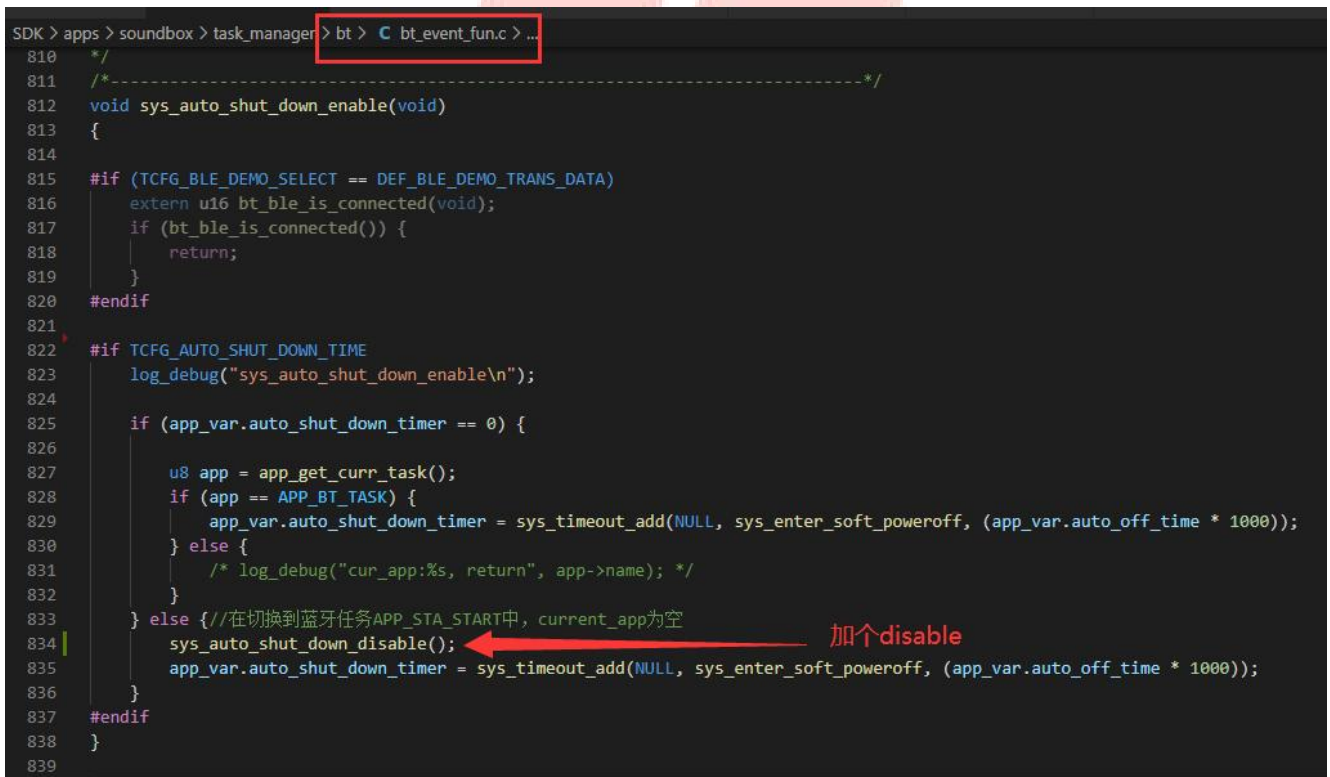
```
终端(T) 帮助(H) eq_config.c - ac696h_soundbox_sdk_V1.1.1 - Visual Studio Code
nfig.c 2  C eq_config.c 3 X  C board_ac696x_demo.c 3  C app_common.c 5  C adkey_table.c ...\board_ac6
SDK > cpu > br25 > audio_effect > C eq_config.c > user_eq_change(void)
498     parm.ul_eq_tab = ul_eq_tab_normal;
499     parm.ul_eq_tab_size = ARRAY_SIZE(ul_eq_tab_normal);
500
501     parm.eq_tool_tab = eq_tool_tab;
502     parm.eq_mode_use_idx = eq_mode_use_idx;
503     parm.eq_type_tab = (void *)get_eq_mode_tab();
504     parm.type_num = EQ_MODE_MAX;
505     parm.section_max = EQ_SECTION_MAX;
506
507     EQ_CFG *eq_cfg = eq_cfg_open(&parm);
508     user_eq_cfg = eq_cfg;
509     if (eq_cfg) {
510 #if APP_ONLINE_DEBUG
511         if (eq_cfg->app) {
512             app_online_db_register_handle(DB_PKT_TYPE_EQ, eq_app_online_parse);
513         }
514 #endif
515
516         for (int i = 0; i < eq_cfg->mode_num; i++) {
517             if (eq_cfg->eq_type == EQ_TYPE_MODE_TAB) {
518                 set_global_gain(eq_cfg, i, 0);
519                 drc_default_init(eq_cfg, i);
520             }
521         }
522     }
523     return 0;
524 }
525 __initcall(eq_init);
526
527 extern s32 eq_file_get_cfg(EQ_CFG *eq_cfg, u8 *path);
528
529 #define EQ_FILE_NAME        SDFILE_RES_ROOT_PATH"eq_cfg_hw.bin"
530 #define EQ_FILE1_NAME      SDFILE_RES_ROOT_PATH"eq_cfg_hw1.bin"
531
532 void user_eq_change(void)
533 {
534     s32 ret = 0;
535     static u8 flag = 0;
536     r_printf("user_eq_change:%d", flag);
537     if (user_eq_cfg) {
538         if (flag) {
539             flag = 0;
540             ret = eq_file_get_cfg(user_eq_cfg, EQ_FILE_NAME);
541         } else {
542             flag = 1;
543             ret = eq_file_get_cfg(user_eq_cfg, EQ_FILE1_NAME);
544         }
545     }
546     printf("eq_file_get_cfg ret=%d\n", ret);
547 }
548
```

添加代码



113.AC696 系列 SDK1.1.1 如果出现莫名其妙的自动关机（非复位死机），可以试下按下图修改 20201204 FSW

AC696 系列 SDK1.1.1 如果出现莫名其妙的自动关机(非复位死机),需要修改 sys_auto_shut_down_enable 函数（原因是不同的流程可能导致重复执行 sys_auto_shut_down_enable），修改如下



114.AC696X 111 SDK 版本使用 U 盘或者 SD 卡不能进行跳转升级的问题 20201204 YP

AC696X 音箱 111 版本的 SDK 使用跳转无法升级的问题，或者编译不过的问题，需要做如下修改：编译不过需要如下更改：

```
board_config.h x board_ac696x_demo_cfg.h x update.c x
415     break;
416 #endif
417
418     default:
419         updata_parm_set(up_type, NULL, 0);
420         break;
421     }
422
423
424 #ifdef DEV_UPDATE_SUPPORT_JUMP
425 #if 1//TCFG_BLUETOOTH_BACK_MODE //后台模式需要把蓝牙
426     if (BT_MODULES_IS_SUPPORT(BT_MODULE_LE)) {
427         ll_hci_destory();
428     }
429     hci_controller_destory();
430 #endif
431     switch (up_type) {
432 #ifdef CONFIG_SD_UPDATE_ENABLE
433     case SD0_UPDATA:
434     case SD1_UPDATA:
435         //sd1_unmount();
436         ;extern void sdx_hw_close(JL_SD_TypeDef *JL_SDx);
437         sdx_hw_close(JL_SD0);
438         break;
439 #endif //CONFIG_SD_UPDATE_ENABLE
440 #ifdef CONFIG_USB_UPDATE_ENABLE
441     case USB_UPDATA:
442         usb_sie_close_all();
443         break;
```

不能跳转升级的问题，需要做如下修改，系插入设备升级的时候，蓝牙模块没有关闭导致升级不了，需要如下修改，关闭蓝牙模块，防止 ram 被修改导致无法升级：

```
473
474     case SPP_APP_UPDATA:
475         updata_parm_set(BLE_APP_UPDATA, (u8 *)&addr, sizeof(addr)); //暂时
不支持SPP
476         break;
477 #endif
478
479     default:
480         updata_parm_set(up_type, NULL, 0);
481         break;
482 }
483
484
485 #ifdef DEV_UPDATE_SUPPORT_JUMP
486 #if 1//TCFG_BLUETOOTH_BACK_MODE //后台模式需要把蓝牙关掉
487     if (BT_MODULES_IS_SUPPORT(BT_MODULE_LE)) {
488         ll_hci_destory();
489     }
490     // hci_controller_destory();^M
491     extern void bredr_bd_close();^M
492     bredr bd close();
493 #endif^M
494 ^M
495     audio_disable_all();^M
496 ^M
497
498     extern void fm_inside_off(void); //关闭内置收音
499     fm_inside_off();
500     switch (up_type) {
501 #ifdef CONFIG_SD_UPDATE_ENABLE
502     case SD0_UPDATA:
503     case SD1_UPDATA:
504         //sd1_unmount();^M
补充一点：关于 AC696X
< v1.1.1\SDK\apps\common\update\update.c c utf-8[unix] 78% 行 458: 5
```

补充一点：关于 AC696X_V111 版本跳转升级的问题，先插入有升级文件的设备开机会导致升级不了的问题，系因为有一些 audio dac 模块没有关闭，按以下修改即可：

```

break;
}

#ifdef DEV_UPDATE_SUPPORT_JUMP
#if 1//TCFG_BLUETOOTH_BACK_MODE //后台模式需要把蓝牙关掉
if (BT_MODULES_IS_SUPPORT(BT_MODULE_LE)) {
    ll_hci_destory();
}
// hci_controller_destory();^H
extern void bredr_bd_close();^H
bredr_bd_close();
#endif^H
audio_disable_all();^H

extern void fm_inside_off(void); //关闭内置收音
fm_inside_off();
switch (up_type) {
#ifdef CONFIG_SD_UPDATE_ENABLE
case SD0_UPDATA:
case SD1_UPDATA:
    //sd1_unmount();^H
    ;extern void sdx_hw_close(JL_SD_TypeDef *ll_SDX);^H
}

356 _WEAK
357 void ran_protect_close(void)
358 {
359     printf("ERROR:%s is no implementation!\n", FUNCTION);
360 }
361 ^H
362 ^H
363 static void audio_disable_all(void)^H
364 {^H
365     //DAC:DACEN^H
366     JL_AUDIO->DAC_CON &= ~BIT(4);^H //加入关闭audio dac 的模块
367     //ADC:ADGEN^H
368     JL_AUDIO->ADC_CON &= ~BIT(4);^H
369     //EQ:^H
370     JL_EQ->CON0 &= ~BIT(0);^H
371     //FFT:^H
372     JL_FFT->CON = BIT(1); //置1强制关闭模块, 不管是否已经运算完成
373 }^H
374
375 extern void ll_hci_destory(void);
376 extern void hci_controller_destory(void);
377 extern const int support_norflash_update_en;
378
379 void update_mode_api(UPDATA_TYPE up_type, ...)
380 {

```

115.AC69 系列 sdk V111 版本无缝循环播放的处理 20201207 YWP

ac696n_soundbox_sdk_v1.1.1 版本实现无缝循环播放，如下截图宏 FILE_DEC_REPEAT_EN 置 1。

```

Audio_dec_file.h Search Results Audio_dec_file.c 全部关闭
00010: #include "app_config.h"
00011: #include "music/music_decrypt.h"
00012: #include "music/music_id3.h"
00013: |
00014: #define FILE_DEC_REPEAT_EN
00015: |
00016: enum {
00017:     FILE_DEC_STREAM_CLOSE = 0,
00018:     FILE_DEC_STREAM_OPEN,
00019: };
00020: |
00021: struct file_dec_hdl {
00022:     struct audio_stream *stream; // 音频流
00023:     struct file_decoder file_dec; // file解码句柄
00024:     struct audio_res_wait wait; // 资源等待句柄
00025:     struct audio_mixer_ch mix_ch; // 叠加句柄
00026:     struct audio_eq_drc *eq_drc; // eq drc句柄
00027:     struct audio_dec_breakpoint *dec_bp; // 断点

```



```

.c | Tone_player.h | Music.h | Music_decrypt.h | Music_player.h | Tone_player.c | Audio_decoder.h
Search Results | Audio_dec_file.c | 全部关闭
00485:      dec->stream = audio_stream_open(dec, file_dec_out_strea
00486:          audio_stream_add_list(dec->stream, entries, entry_cnt);
00487:
00488:      stream set end:
00489:          log_i("total_time : %d \n", dec->file_dec.dec_total_tim
00490:
00491:      #if FILE_DEC_REPEAT_EN
00492:          file_dec_repeat_set(3);
00493:      #endif
00494:
00495:          audio_output_set_start_volume(APP_AUDIO_STATE_MUSIC);
00496:
00497:      // 文件打开就暂停
    
```

file_dec_start函数这里的3改为-1, 实现无限次数播放

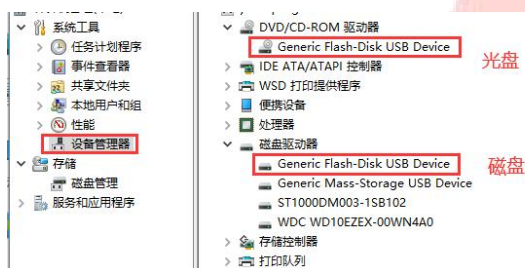
同时循环播放的音乐文件首尾的静音资源要剪切掉，以免干扰无缝播放。

116.AC69 系列 sdk V111 插两驱动类型的(第一驱动类型为光盘模式)U 盘死机 20201207 SQ

1) 目前小机只支持:

- a) MBR 分区表引导的磁盘
- b) 分区表中只有一个磁盘分区。
- c) 磁盘格式为: FAT32、exFAT

2) 插入 windos 电脑可以看到 U 盘的两个驱动类型



2) 代码修改如下

```

E:\Project\SVN\696X\SDK\ac696n_soundbox_sdk_v1.1.1-cd-usb\SDK\apps\common\usb\host\usb_storage.h
2020/12/7 17:54:36 1,160 字节 C,C++,C#,ObjC 源代码 UTF-8 UNIX
25      u8 target_epin;
26      #if HUSB_MODE
27          u16 rxmaxp;
28          u16 txmaxp;
29      #endif
30  };
31  #define MULTI_DISK 1  定义该宏
32  struct mass_storage {
33      OS_MUTEX mutex;
34
35      struct usb_scsi_cbw cbw;
36      struct usb_scsi_csw csw;
37      struct request_sense_data sense;
38
39      #if MULTI_DISK
40          char *name;
    
```

#define MULTI_DISK 1

<p>2020/9/29 14:08:41 41,230 字节 C,C++,C#,ObjC 源代码</p> <pre> 24 /* #define LOG_DUMP_ENABLE */ 25 #define LOG_CLI_ENABLE 26 #include "debug.h" 27 28 #if TCFG_UDISK_ENABLE 29 static int set_stor_power(struct usb_host_de 30 static int get_stor_power(struct usb_host_de 31 32 const struct interface_ctrl udisk_ops = { 33 .interface_class = USB_CLASS_MASS_STORAC 34 .set_power = set_stor_power, 35 .get_power = get_stor_power, 36 .ioctl = NULL, 37 }; 38 39 40 static struct mass storagee mass stor ://SEC </pre> <p style="text-align: right;">修改前</p>	<p>2020/12/7 17:56:35 42,709 字节 C,C++,C#,ObjC 源代码</p> <pre> 25 /* #define LOG_DUMP_ENABLE */ 26 #define LOG_CLI_ENABLE 27 #include "debug.h" 28 29 30 static int set_stor_power(struct usb_host_d 31 static int get_stor_power(struct usb_host_d 32 33 const struct interface_ctrl udisk_ops = { 34 .interface_class = USB_CLASS_MASS_STORA 35 .set_power = set_stor_power, 36 .get_power = get_stor_power, 37 .ioctl = NULL, 38 }; 39 40 static struct mass storagee mass stor ://SEC </pre> <p style="text-align: right;">修改后</p>
--	--

<p>2020/9/29 14:08:41 41,230 字节 C,C++,C#,ObjC 源代码</p> <pre> 1431 .read = usb_stor_read, 1432 .write = usb_stor_write, 1433 .ioctl = usb_stor_ioctl, 1434 .close = usb_stor_close, 1435 }; 1436 1437 static u8 usb_stor_idle_query(void) 1438 { 1439 struct device *device = &udisk_device; 1440 struct mass_storage *disk = host_device; 1441 if (disk) { 1442 return (disk->dev_status == DEV_IDLE 1443 } 1444 return true; 1445 } 1446 1447 REGISTER_LP_TARGET(usb_stor_lp_target) = { 1448 .name = "udisk", 1449 .is_idle = usb_stor_idle_query, </pre> <p style="text-align: right;">改前</p>	<p>2020/12/7 20:14:56 41,448 字节 C,C++,C#,ObjC 源代码</p> <pre> 1435 .read = usb_stor_read, 1436 .write = usb_stor_write, 1437 .ioctl = usb_stor_ioctl, 1438 .close = usb_stor_close, 1439 }; 1440 #if USB_HOST_SUPPORT_MSD 1441 1442 static u8 usb_stor_idle_query(void) 1443 { 1444 struct device *device = &udisk_device; 1445 struct mass_storage *disk = host_device; 1446 if (disk) { 1447 return (disk->dev_status == DEV_IDL 1448 } 1449 return true; 1450 } 1451 1452 REGISTER_LP_TARGET(usb_stor_lp_target) = { 1453 .name = "udisk", 1454 .is_idle = usb_stor_idle_query, </pre> <p style="text-align: right;">改后</p>
---	--

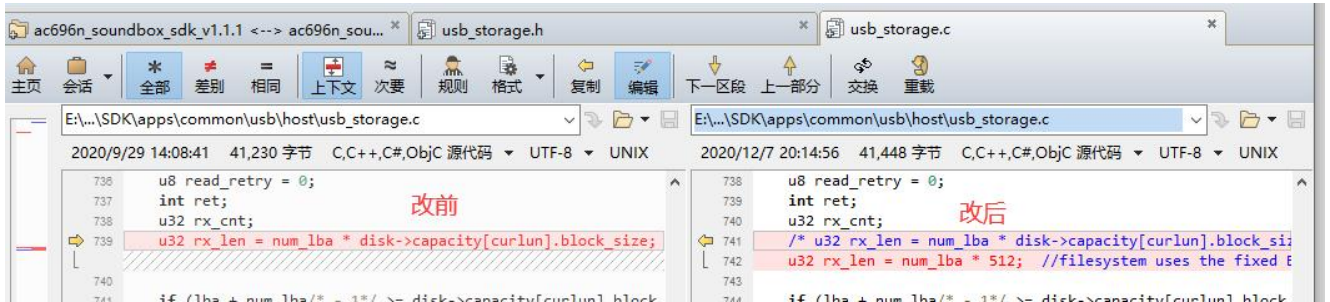
#if USB_HOST_SUPPORT_MSD

<p>2020/9/29 14:08:41 41,230 字节 C,C++,C#,ObjC 源代码</p> <pre> 255 const u32 txmaxp = usb_stor_txmaxp(disk); 256 const u32 rxmaxp = usb_stor_rxmaxp(disk); 257 258 int ret = DEV_ERR_NONE; 259 struct inquiry_data inquiry; 260 261 log_info("usb_stor_inquiry disk =%x", (u 262 263 u32 total_lun = 1; //disk->lun; 264 265 for (int i = 0; i <= total_lun; i++) { </pre> <p style="text-align: right;">改前</p>	<p>2020/12/7 20:14:56 41,448 字节 C,C++,C#,ObjC 源代码</p> <pre> 254 const u32 txmaxp = usb_stor_txmaxp(disk); 255 const u32 rxmaxp = usb_stor_rxmaxp(disk); 256 257 int ret = DEV_ERR_NONE; 258 struct inquiry_data inquiry; 259 260 log_info("usb_stor_inquiry disk =%x", (261 262 #if MULTI_DISK 263 u32 total_lun = disk->lun; 264 #else 265 u32 total_lun = 1; 266 #endif 267 268 for (int i = 0; i <= total_lun; i++) { </pre> <p style="text-align: right;">改后</p>
---	--

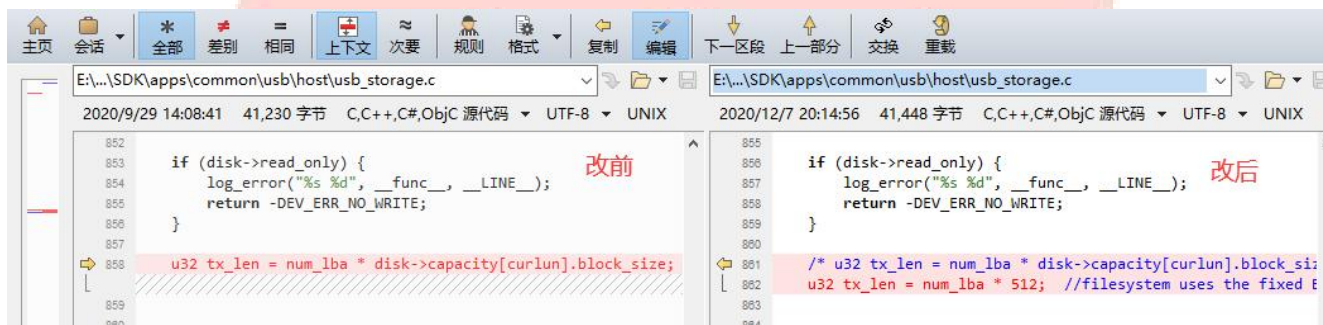
```

#if MULTI_DISK
    u32 total_lun = disk->lun;
#else
                
```

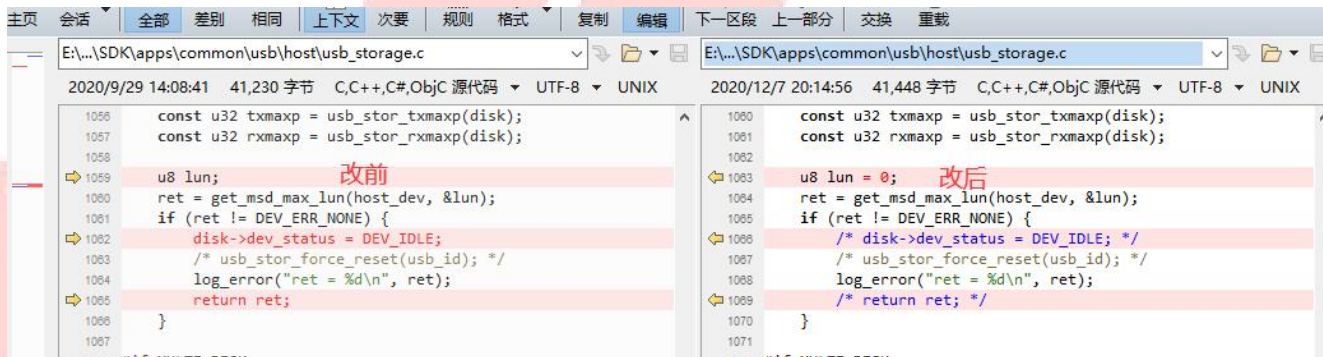
```
u32 total_lun = 1;
#endif
```



```
/* u32 rx_len = num_lba * disk->capacity[curlun].block_size; */
u32 rx_len = num_lba * 512; //filesystem uses the fixed BLOCK_SIZE
```



```
/* u32 tx_len = num_lba * disk->capacity[curlun].block_size; */
u32 tx_len = num_lba * 512; //filesystem uses the fixed BLOCK_SIZE
```



```
u8 lun = 0;
ret = get_msd_max_lun(host_dev, &lun);
if (ret != DEV_ERR_NONE) {
    /* disk->dev_status = DEV_IDLE; */
    /* usb_stor_force_reset(usb_id); */
    log_error("ret = %d\n", ret);
    /* return ret; */
}
```

<pre> 1095 case 1: 1096 log_info("--- test unit ready ---"); 1097 ret = usb_stor_test_unit_ready(device); 1098 if (ret < DEV_ERR_NONE) { 1099 os_time_dly(50); 1100 } else if (disk->csw.bCSWStatus) { 1101 os_time_dly(50); 1102 ret = usb_stor_request_sense(device); 1103 if ((disk->sense.SenseKey & 0x0f) != NO_SENSE 1104 if (disk->sense.SenseKey == NOT_READY) { 1105 os_time_dly(30); 1106 if (disk->cur_available_lun++ == lun) 1107 disk->cur_available_lun = 0; 1108 } 1109 } 1110 } 1111 } </pre> <p style="text-align: right; color: red;">改前</p>	<pre> 1099 case 1: 1100 log_info("--- test unit ready ---"); 1101 ret = usb_stor_test_unit_ready(device); 1102 if (ret < DEV_ERR_NONE) { 1103 os_time_dly(50); 1104 } else if (disk->csw.bCSWStatus) { 1105 os_time_dly(50); 1106 ret = usb_stor_request_sense(device); 1107 if ((disk->sense.SenseKey & 0x0f) != NO_SENSE 1108 if (disk->sense.SenseKey == NOT_READY) { 1109 os_time_dly(30); 1110 if (disk->curlun++ == lun) { 1111 disk->curlun = 0; 1112 } 1113 } </pre> <p style="text-align: right; color: red;">改后</p>
---	---

```

if (disk->curlun++ == lun) {
    disk->curlun = 0;
}
    
```

<pre> 1137 } 1138 state = 0; 1139 1140 if (ret != DEV_ERR_NONE) { 1141 disk->dev_status = DEV_IDLE; 1142 usb_stor_force_reset(usb_id); 1143 log_error("ret = %d\n", ret); 1144 } else { 1145 /* disk->test_unit_ready_tick = sys_timer_add(device,u 1146 #if ENABLE_DISK_HOTPLUG 1147 disk->media_sta_cur = 1; 1148 disk->media_sta_prev = 1; 1149 #endif 1150 } 1151 disk->suspend_cnt = 0; 1152 return ret; 1153 } 1154 } </pre> <p style="text-align: right; color: red;">改前</p>	<pre> 1141 } 1142 state = 0; 1143 1144 if (ret != DEV_ERR_NONE) { 1145 disk->dev_status = DEV_IDLE; 1146 /* usb_stor_force_reset(usb_id); */ 1147 log_error("ret = %d\n", ret); 1148 } else { 1149 /* disk->test_unit_ready_tick = sys_timer_add(device,u 1150 #if ENABLE_DISK_HOTPLUG 1151 disk->media_sta_cur = 1; 1152 disk->media_sta_prev = 1; 1153 #endif 1154 } 1155 disk->suspend_cnt = 0; 1156 return ret; 1157 } 1158 } </pre> <p style="text-align: right; color: red;">改后</p>
--	--

<pre> 1214 case IOCTL_SET_CUR_LUN: 1215 #if MULTI_DISK 1216 if (arg > disk->lun) { 1217 return -EINVAL; 1218 } 1219 } 1220 usb_stor_set_curlun(arg); 1221 ret = usb_stor_read_capacity(device); 1222 if (ret < 0) { </pre> <p style="text-align: right; color: red;">改前</p>	<pre> 1218 case IOCTL_SET_CUR_LUN: 1219 #if MULTI_DISK 1220 if (arg > disk->lun) { 1221 return -EINVAL; 1222 } 1223 } 1224 usb_stor_set_curlun(disk,*(u32*)arg); 1225 ret = usb_stor_read_capacity(device); 1226 if (ret < 0) { </pre> <p style="text-align: right; color: red;">改后</p>
--	--

```
usb_stor_set_curlun(disk,*(u32*)arg);
```

117.AC696 系列 sdk 1.1.1 麦克风（混响）无法调数字音量解决方法 20201211 LHY

1、调用 mic_effect_set_dvol 函数（大小 0~30）无法调节数字音量的主要是没有初始化数字音量，导致在设置时候直接 return

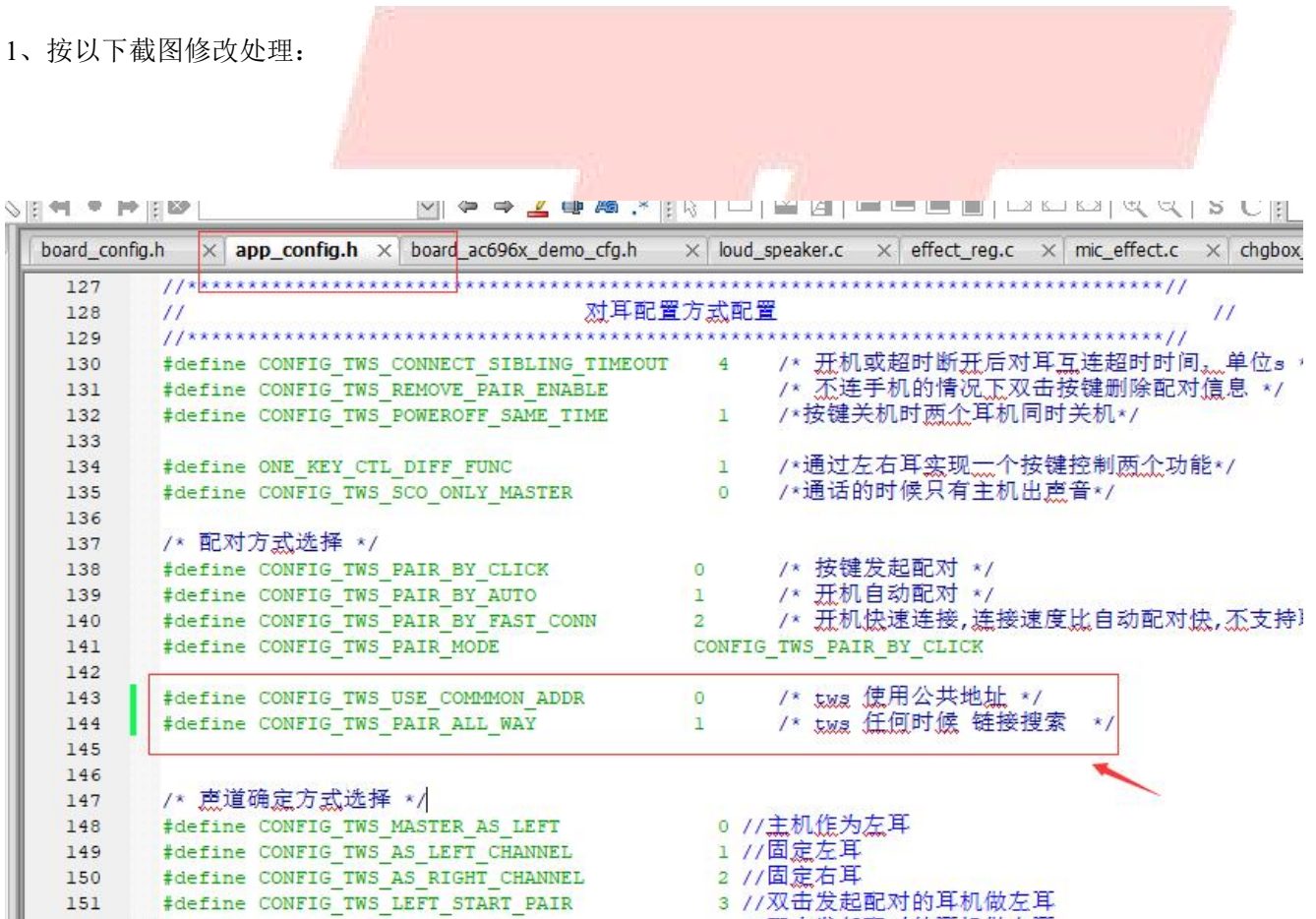
```

///初始化数字音量
if (effect->parm.effect_config & BIT(MIC_EFFECT_CONFIG_DVOL)) {
    /* effect->dvol = audio_dig_vol_open(&effect_dvol_default_parm); */
}
    
```

打开这个注释

118.AC696 系列 sdk 1.1.1 主机连上手机后，副机发起配对无法连接主机处理方法 20201211 WTS

1、按以下截图修改处理：



119.AC696 系列 TWS 音箱同时按左右两边配对：一边连手机的情况下，配对慢 20201211 LZK

【具体现象】 连接手机的样机相对较难搜到对箱，从而导致配对慢。

【修改方法】 !! 注意以下修改：只适用于同时按左右两边配对的配对方式

1，同时按左右两边配对程序配置如下

```
/* 配对方式选择 */  
#define CONFIG_TWS_PAIR_BY_CLICK 0 /* 按键发起配对 */  
#define CONFIG_TWS_PAIR_BY_AUTO 1 /* 开机自动配对 */  
#define CONFIG_TWS_PAIR_BY_FAST_CONN 2 /* 开机快速连接,连接速度比自动配对快,  
不支持取消配对操作 */  
#define CONFIG_TWS_PAIR_MODE CONFIG_TWS_PAIR_BY_CLICK
```

```
#define CONFIG_TWS_USE_COMMON_ADDR 0 /* tws 使用公共地址 */
#define CONFIG_TWS_PAIR_ALL_WAY 1 /* tws 任何时候 链接搜索 */

#define CONFIG_TWS_PAIR_BY_BOTH_SIDES 1
```

2, 搜索流程修改如下

SDK/apps/soundbox/task_manager/bt/bt_tws.c: 2ef09dd9	SDK/apps/soundbox/task_manager/bt/bt_tws.c: Working Tree
<pre>541 /* 542 /**@brief... 蓝牙tws 开启tws的主机搜索链接 543 ...@param... 无 544 ...@return... 无 545 ...@note... tws 在 BT_TWS_UNPAIRED 状态开启搜索到tws即可链接 546 547 ... 根据配对码搜索TWS设备 548 ... 搜索超时会收到事件: TWS_EVENT_SEARCH_TIMEOUT 549 ... 搜索到连接超时会收到事件: TWS_EVENT_CONNECTION_TIMEOUT 550 ... 搜索到并连接成功会收到事件: TWS_EVENT_CONNECTED 551 */ 552 553 static void bt_tws_search_and_pair() 554 { 555 u8 mac_addr[6]; 556 y_printf("tws.state %x\n", tws.state); 557 if (tws.state & BT_TWS_UNPAIRED) { 558 559 bt_tws_delete_pair_timer(); 560 tws_api_get_local_addr(gtw.s.addr); 561 #if CONFIG_TWS_USE_COMMON_ADDR 562 #if CONFIG_TWS_PAIR_MODE == CONFIG_TWS_PAIR_BY_CLICK 563 get_random_number(mac_addr, 6); 564 lmp_hci_write_local_address(mac_addr); 565 #endif 566 #endif 567 568 #if CONFIG_TWS_PAIR_BY_BOTH_SIDES 569 bt_user_priv_var.search_counter = 5; 570 bt_set_pair_code_en(1); 571 search_and_connectable_switch(0); 572 #else 573 tws_api_search_sibling_by_code(bt_get_tws_device_indicate(NULL), 574 #endif 575 } 576 } 577 578</pre> <p style="color: red; text-align: center;">修改前</p>	<pre>541 /* 542 /**@brief... 蓝牙tws 开启tws的主机搜索链接 543 ...@param... 无 544 ...@return... 无 545 ...@note... tws 在 BT_TWS_UNPAIRED 状态开启搜索到tws即可链接 546 547 ... 根据配对码搜索TWS设备 548 ... 搜索超时会收到事件: TWS_EVENT_SEARCH_TIMEOUT 549 ... 搜索到连接超时会收到事件: TWS_EVENT_CONNECTION_TIMEOUT 550 ... 搜索到并连接成功会收到事件: TWS_EVENT_CONNECTED 551 */ 552 553 static void bt_tws_search_and_pair() 554 { 555 u8 mac_addr[6]; 556 y_printf("tws.state %x\n", tws.state); 557 if (tws.state & BT_TWS_UNPAIRED) { 558 559 bt_tws_delete_pair_timer(); 560 tws_api_get_local_addr(gtw.s.addr); 561 #if CONFIG_TWS_PAIR_MODE == CONFIG_TWS_PAIR_BY_CLICK 562 get_random_number(mac_addr, 6); 563 lmp_hci_write_local_address(mac_addr); 564 #endif 565 #endif 566 567 #if CONFIG_TWS_PAIR_BY_BOTH_SIDES 568 bt_user_priv_var.search_counter = 5; 569 bt_set_pair_code_en(1); 570 search_and_connectable_switch((void *)1); 571 #else 572 tws_api_search_sibling_by_code(bt_get_tws_device_indicate(NULL), 573 #endif 574 #endif 575 } 576 } 577</pre> <p style="color: red; text-align: center;">修改后</p>
<pre>1200 1201 static void search_and_connectable_switch(void *sw) 1202 { 1203 int timeout = 0; 1204 int sw = (int)sw; 1205 1206 bt_tws_close_search_and_pair(); 1207 1208 if (sw == 0) { 1209 timeout = tws_wait_pair_and_wait_conn(2); 1210 } else if (sw == 1) { 1211 printf("search cnt %d\n", bt_user_priv_var.search_counter); 1212 if ((tws_wait_pair_time) && (bt_user_priv_var.search_counter > 0)) { 1213 printf("here11\n"); 1214 timeout = 5000; 1215 tws_wait_pair_and_wait_conn(2); 1216 } else { 1217 if (bt_user_priv_var.search_counter > 0) { 1218 printf("here22\n"); 1219 tws_api_search_sibling_by_code(bt_get_tws_device_indicate(NULL), 15000); 1220 timeout = 1000 + (rand32() % 4 + 1) * 500; 1221 } else { 1222 printf("here33\n"); 1223 tws_api_search_sibling_by_code(bt_get_tws_device_indicate(NULL), 2000); 1224 return; 1225 } 1226 } 1227 } 1228 } 1229 1230 bt_user_priv_var.search_counter--;</pre> <p style="color: red; text-align: center;">修改前</p>	<pre>1198 1199 static void search_and_connectable_switch(void *sw) 1200 { 1201 int timeout = 0; 1202 int sw = (int)sw; 1203 1204 bt_tws_close_search_and_pair(); 1205 1206 if (sw == 0) { 1207 timeout = tws_wait_pair_and_wait_conn(2); 1208 } else if (sw == 1) { 1209 printf("search cnt %d\n", bt_user_priv_var.search_counter); 1210 if ((tws_wait_pair_time) && (bt_user_priv_var.search_counter > 0)) { 1211 printf("here11\n"); 1212 timeout = 5000; 1213 tws_wait_pair_and_wait_conn(2); 1214 } else { 1215 if (!((tws_api_get_tws_state() & TWS_STA_PHONE_CONNECTED))) { 1216 if (bt_user_priv_var.search_counter > 0) { 1217 printf("here22\n"); 1218 tws_api_search_sibling_by_code(bt_get_tws_device_indicate(NULL), 15000); 1219 timeout = 2000 + (rand32() % 4 + 1) * 500; 1220 } else { 1221 printf("here33\n"); 1222 tws_api_search_sibling_by_code(bt_get_tws_device_indicate(NULL), 2000); 1223 return; 1224 } 1225 } else { 1226 if (bt_user_priv_var.search_counter > 0) { 1227 printf("here44\n"); 1228 timeout = tws_wait_pair_and_wait_conn(2); 1229 } else { 1230 printf("here55\n"); 1231 tws_api_search_sibling_by_code(bt_get_tws_device_indicate(NULL), 2000); 1232 return; 1233 } 1234 } 1235 } 1236 } 1237 } 1238 bt_user_priv_var.search_counter--;</pre> <p style="color: red; text-align: center;">修改后</p> <p style="color: red; text-align: center;">连手机不搜TWS</p>

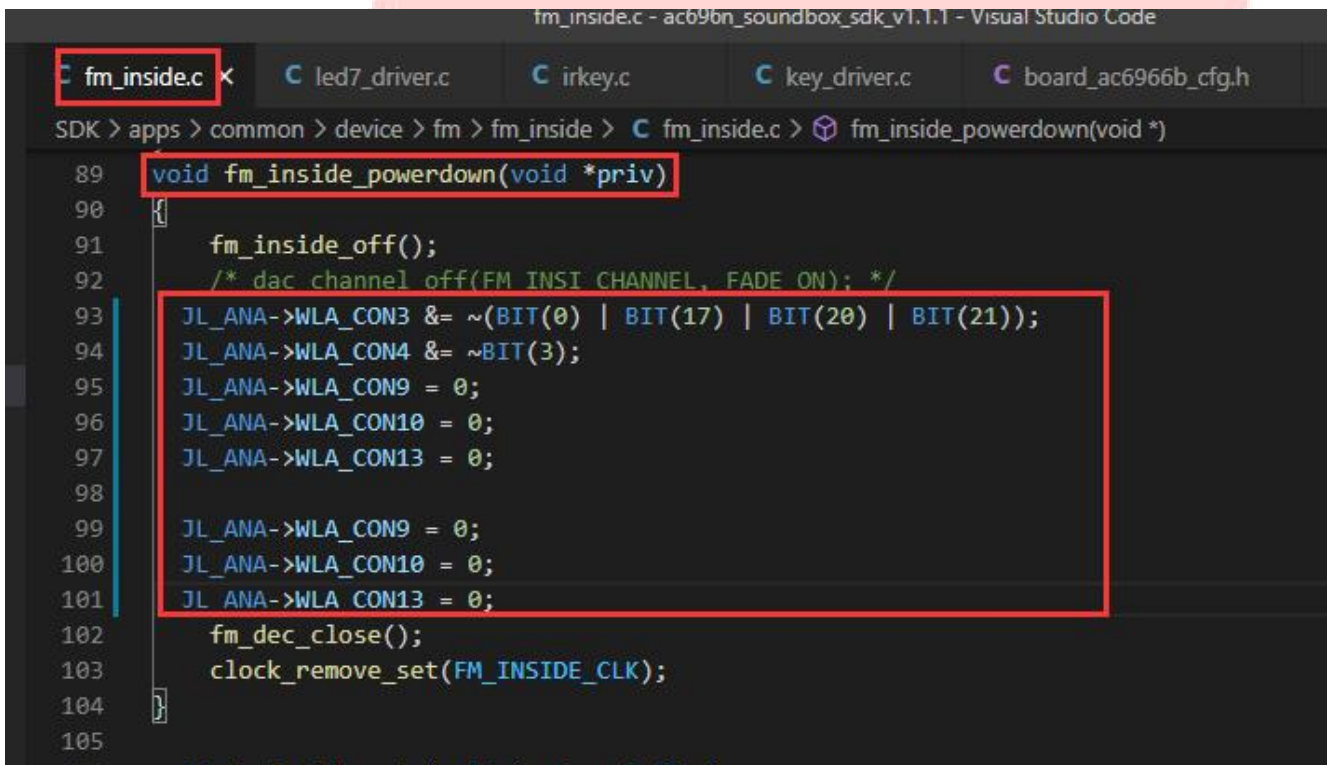
120.AC696N SDK1.1.1 录音模式播放录音文件时插入 U 盘播歌，再回录音模式播放录音文件时无法播放问题 20201211 LAQ

更新库文件：

链接：https://pan.baidu.com/s/1wpWLxx1jR7IA0gN1_cC0lw

提取码：JLKJ

121.AC696N SDK1.1.1 退出 FM 之后功耗大 20201214 SQ



```
fm_inside.c - ac696n_soundbox_sdk_v1.1.1 - Visual Studio Code
fm_inside.c x led7_driver.c irkey.c key_driver.c board_ac6966b_cfg.h
SDK > apps > common > device > fm > fm_inside > C fm_inside.c > fm_inside_powerdown(void *)
89 void fm_inside_powerdown(void *priv)
90 {
91     fm_inside_off();
92     /* dac channel off(FM INSI CHANNEL, FADE ON); */
93     JL_ANA->WLA_CON3 &= ~(BIT(0) | BIT(17) | BIT(20) | BIT(21));
94     JL_ANA->WLA_CON4 &= ~BIT(3);
95     JL_ANA->WLA_CON9 = 0;
96     JL_ANA->WLA_CON10 = 0;
97     JL_ANA->WLA_CON13 = 0;
98
99     JL_ANA->WLA_CON9 = 0;
100    JL_ANA->WLA_CON10 = 0;
101    JL ANA->WLA CON13 = 0;
102    fm_dec_close();
103    clock_remove_set(FM_INSIDE_CLK);
104 }
105
```

```
JL_ANA->WLA_CON3 &= ~(BIT(0) | BIT(17) | BIT(20) | BIT(21));
JL_ANA->WLA_CON4 &= ~BIT(3);
JL_ANA->WLA_CON9 = 0;
JL_ANA->WLA_CON10 = 0;
JL_ANA->WLA_CON13 = 0;
```

```
JL_ANA->WLA_CON9 = 0;
JL_ANA->WLA_CON10 = 0;
JL_ANA->WLA_CON13 = 0;
```

121.AC696N SDK1.1.1 利用定时器保存音乐断点 20201214 LJW

```

// music.c
void one_minute_auto_save_breakpoint(void)
{
    char *logo = music_player_get_dev_cur();
    if (music_player_get_playing_breakpoint(breakpoint, 1) == true) {
        breakpoint_vm_write(breakpoint, logo);
    }
}

void app_music_task()
{
    int res;
    int msg[32];
    music_task_start();

    int err = tone_play_with_callback_by_name(tone_table[INDEX_TONE_MUSIC]);
    if (err) {
        music_player_play_start();
    }

    sbp_id = sys_timer_add(NULL, one_minute_auto_save_breakpoint, 30000);

    while (1) {
        app_task_get_msg(msg, ARRAY_SIZE(msg), 1);
        switch (msg[0]) {

```

```

// timer.h
u16 sbp_id = 0;

static void music_task_close()
{
    UI_HIDE_CURR_WINDOW();
    tone_play_stop_by_path(tone_table[INDEX_TONE_MUSIC]); // 停止播放提示音
    if (sbp_id) {
        sys_timer_del(sbp_id);
    }
    char *logo = music_player_get_dev_cur();
    if (music_player_get_playing_breakpoint(breakpoint, 1) == true) {
        breakpoint_vm_write(breakpoint, logo);
    }
    breakpoint_handle_destroy(&breakpoint);
    music_player_destroy();
    memset(_this, 0, sizeof(struct __music));
}

```

请添加方框中的内容

122.AC696N SDK1.1.1 音乐模式下获取播放设备的音乐文件数目 20201214 LJW

```

// fs.h
#define VFS_ENABLE 1

// music_player.c
int get_music_file_num(void)
{
    char *logo = dev_manager_get_logo(dev_manager_find_active(1));
    if (logo == NULL) {
        r_printf("logo is null\n");
        return -1;
    }
    _this->dev = dev_manager_find_spec(logo, 1);
    if (_this->dev == NULL) {
        r_printf("dev_manager_find_spec error\n");
        return -1;
    }
    _this->fsn = dev_manager_scan_disk(_this->dev, NULL, scan_parm, cycle_mode,
    _this->parm.cb->fsn_break);

    if (_this->fsn == NULL) {
        r_printf("dev_manager_scan_disk error\n");
        return -1;
    }
    return _this->fsn->file_number;
}

```

123.配置多个唤醒脚时开机唤醒时判断哪个 IO 唤醒 20201214 xin

```
log_i("\n~~~~~");
log_i("      setup_arch %s %s", __DATE__, __TIME__);
log_i("~~~~~\n");

/* clock_dump(); */
power_sanity_check();

/* log_info("resour est: %d", get_boot_flag()); */
//set_boot_flag(99);
/* log_info("resour est: %d", get_boot_flag()); */

reset_source_dump();

power_reset_src = power_reset_source_dump();

r_printf("R3 WKUP_PND:0x%x", P33_CON_GET(R3_WKUP_PND));
```

```
const struct wakeup_param wk_param = {
    .port[1] = &port0,
    .port[2] = &port1,
    .sub = &sub_wkup,
    .charge = &charge_wkup,
```

bit(1) == 2
bit(2) == 4

124.SDK1.1.1 开 TWS 同时开混响有一边音乐断续问题 20201218 WTS

更新库文件:

链接: <https://pan.baidu.com/s/1hZZrlezY0Z7ybZsAPi6gSg>

提取码: p5an

```
1157
1158 static void board_power_wakeup_init(void)
1159 {
1160     power_wakeup_init(&wk_param);
1161     return;
1162     key_wakeup_disable();
1163     #if TCFG_POWER_ON_NEED_KEY
1164         extern u8 power_reset_src;
1165         if ((power_reset_src & BIT(0)) || (power_reset_src & BIT(1))) {
1166             #if TCFG_CHARGE_ENABLE
1167                 log_info("is ldo5v wakeup:%d\n", is_ldo5v_wakeup());
1168                 if (is_ldo5v_wakeup()) {
1169                     return;
1170                 }
1171                 if (get_ldo5v_online_hw()) {
1172                     return;
1173                 }
1174                 /*LD05V,检测上升沿,用于检测ldoin插入*/
1175                 LD05V_EN(1);
1176                 LD05V_EDGE_SEL(0);
1177                 LD05V_PND_CLR();
1178                 LD05V_EDGE_WKUP_EN(1);
1179             #endif
1180             power_set_callback(TCFG_LOWPOWER_LOWPOWER_SEL, sleep_enter_callback, sleep_exit_call
1181             power_set_soft_poweroff();
1182         }
1183     #endif
1184 }
```

在这里加return;

126. 低电检测报低电提示音后，电压拉高还会继续报提示音解决办法 20201221 LHY

```
C app_power_manage.c X C board_ac6082_demo.c
apps > soundbox > power_manage > C app_power_manage.c > lowpower_timer
304
305     }
306 }
307
308 void vbat_timer_delete(void)
309 {
310     if (vbat_timer) {
311         sys_timer_del(vbat_timer);
312         vbat_timer = 0;
313         vbat_check_idle = 1;
314     }
315 }
316
317 static u8 cur_bat_st = VBAT_NORMAL;
318 static u8 lowpower_timer;
319 void vbat_check(void *priv)
320 {
321     static u8 cur_timer_period = VBAT_TIMER_2_MS;
322     static u8 unit_cnt = 0;
323     static u8 low_warn_cnt = 0;
324     static u8 low_off_cnt = 0;
325     static u8 low_voice_cnt = 0;
326     static u8 low_power_cnt = 0;
327     static u8 power_normal_cnt = 0;
328     static u8 charge_cvolt_v_cnt = 0;
```

```
C app_power_manage.c X C board_ac6082_demo.c
apps > soundbox > power_manage > C app_power_manage.c > vbat_check(void *)
377     log_info("\n*****Low Power,enter sortpowerott*****\n");
378     low_power_cnt = 0;
379     sys_timer_del(vbat_timer);
380     if (lowpower_timer) {
381         sys_timer_del(lowpower_timer);
382         lowpower_timer = 0 ;
383     }
384     power_event_to_user(POWER_EVENT_POWER_LOW);
385 }
386 } else if (low_warn_cnt > (detect_cnt / 2)) { //低电提醒
387     low_voice_cnt ++;
388     low_power_cnt = 0;
389     power_normal_cnt = 0;
390     cur_bat_st = VBAT_WARNING;
391     if ((low_voice_first_flag && low_voice_cnt > 1) || //第一次进低电10s后报一次
392         (!low_voice_first_flag && low_voice_cnt >= 5)) {
393         low_voice_first_flag = 0;
394         low_voice_cnt = 0;
395         if (!lowpower_timer) {
396             log_info("\n**Low Power,Please Charge Soon!!!*\n");
397             power_event_to_user(POWER_EVENT_POWER_WARNING);
398             lowpower_timer = sys_timer_add((void *)POWER_EVENT_POWER_WARNING, (void (*)(void *))power_event_to_use
399         }
400     }
401 } else {
402     if (lowpower_timer) {
403         sys_timer_del(lowpower_timer);
404         lowpower_timer = 0 ;
405     }
406     power_normal_cnt++;
407     low_voice_cnt = 0;
408     low_power_cnt = 0;
409     if (power_normal_cnt > 2) {
410         if (cur_bat_st != VBAT_NORMAL) {
411             log_info("[Normal power]\n");
412             cur_bat_st = VBAT_NORMAL;
413             power_event_to_user(POWER_EVENT_POWER_NORMAL);
```

u8改成u16

在这里加入这段代码

127.开机判断是哪个唤醒口唤醒 20201222 xin

```

power_reset_src = power_reset_source_dump();

r_printf("R3 WKUP PND:0x%x", P33_CON_GET(R3 WKUP PND)); ← 注意放的位置, 不能乱放
r_printf("get_wakeup_source:0x%x", get_wakeup_source());
request_irq(1, 2, exception_irq_handler, 0);

- 4.8k setup.c c/1

/***** PWR config *****/
struct port_wakeup port0 = {
    .pullup_down_enable = ENABLE, //配置I/O 内部上下拉
    .edge = FALLING_EDGE, //唤醒方式选择, 可选:
    .attribute = BLUETOOTH_RESUME, //保留参数
    .iomap = IO_PORTA_06, //唤醒口选择
};

struct port_wakeup port1 = {
    .pullup_down_enable = ENABLE, //配置I/O 内部上下拉
    .edge = FALLING_EDGE, //唤醒方式选择, 可选:
    .attribute = BLUETOOTH_RESUME, //保留参数
    .iomap = IO_PORTB_01, //唤醒口选择
};

const struct sub_wakeup sub_wkup = {
    .attribute = BLUETOOTH_RESUME,
};

const struct charge_wakeup charge_wkup = {
    .attribute = BLUETOOTH_RESUME,
};

const struct wakeup_param wk_param = {
    .port[1] = &port0, bit(1) == 2
    .port[2] = &port1, bit(2) == 4
    .sub = &sub_wkup,
    .charge = &charge_wkup,
};
    
```

128.SDK1.1.1 实时修改自定义 EQ 效果的指定频点增益 20201224 SQ

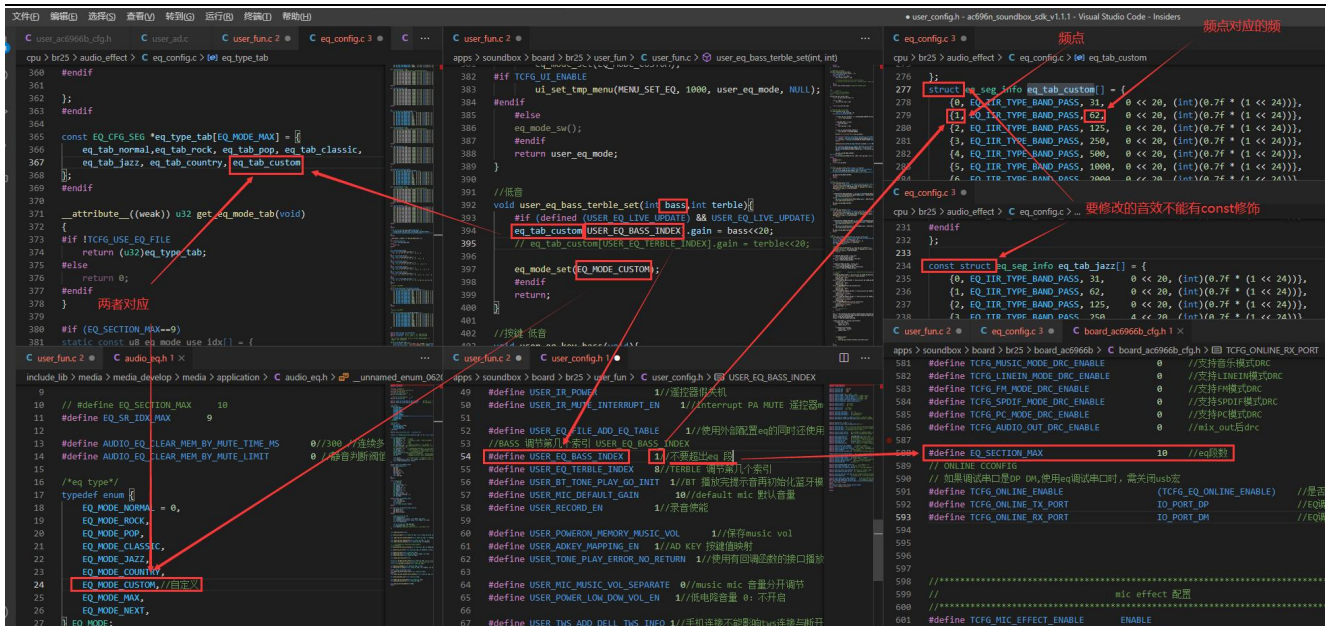
1.确认自定义 eq 效果注意点

```

audio_eq.h
14 #define AUDIO_EQ_CLEAR_MEM_BY_MUTE_LIM
15
16 /*eq type*/
17 typedef enum {
18     EQ_MODE_NORMAL = 0,
19     EQ_MODE_ROCK,
20     EQ_MODE_POP,
21     EQ_MODE_CLASSIC,
22     EQ_MODE_JAZZ,
23     EQ_MODE_COUNTRY,
24     EQ_MODE_CUSTOM, //自定义
25     EQ_MODE_MAX,
26     EQ_MODE_NEXT,
27 } EQ_MODE;
28
29 /*eq type*/
30 typedef enum {
31     EQ_IIR_TYPE_BAND_PASS, 96000, 0 << 20, (int)(0.7f * (1 << 24)),
32     {26, EQ_IIR_TYPE_BAND_PASS, 96000, 0 << 20, (int)(0.7f * (1 << 24))},
33     {27, EQ_IIR_TYPE_BAND_PASS, 96000, 0 << 20, (int)(0.7f * (1 << 24))},
34     {28, EQ_IIR_TYPE_BAND_PASS, 96000, 0 << 20, (int)(0.7f * (1 << 24))},
35     {29, EQ_IIR_TYPE_BAND_PASS, 96000, 0 << 20, (int)(0.7f * (1 << 24))},
36     {30, EQ_IIR_TYPE_BAND_PASS, 96000, 0 << 20, (int)(0.7f * (1 << 24))},
37     {31, EQ_IIR_TYPE_BAND_PASS, 96000, 0 << 20, (int)(0.7f * (1 << 24))},
38 } #endif
39
40 };
41 #endif
42
43 const EQ_CFG_SEG *eq_type_tab[EQ_MODE_MAX] = {
44     eq_tab_normal, eq_tab_rock, eq_tab_pop, eq_tab_classic, eq_tab_jazz, eq_tab_country, eq_tab_custom
45 };
46 #endif
47
48 #if (EQ_SECTION_MAX > 10)
49 //10段之后频率值设置96k,目的是让10段之后的eq走直
50 {10, EQ_IIR_TYPE_BAND_PASS, 96000, 0 << 20, (
51
52 #endif
53
54 #endif
55
56 #endif
57
58 #endif
59
60 #endif
61
62 #endif
63 #endif
64
65 const EQ_CFG_SEG *eq_type_tab[EQ_MODE_MAX] = {
66     eq_tab_normal, eq_tab_rock, eq_tab_pop,
67     eq_tab_classic, eq_tab_jazz,
68     eq_tab_country, eq_tab_custom
69 };
70 #endif
71
72 #if (EQ_SECTION_MAX > 10)
73 //10段之后频率值设置96k,目的是让10段之后的eq走直
74 {10, EQ_IIR_TYPE_BAND_PASS, 96000, 0 << 20, (
75
76 #endif
77
78 #endif
79
80 #endif
81
82 #endif
83
84 #endif
85
86 #endif
87
88 #endif
89
90 #endif
91
92 #endif
93
94 #endif
95
96 #endif
97
98 #endif
99
100 #endif
101
102 #endif
103
104 #endif
105
106 #endif
107
108 #endif
109
110 #endif
111
112 #endif
113
114 #endif
115
116 #endif
117
118 #endif
119
120 #endif
121
122 #endif
123
124 #endif
125
126 #endif
127
128 #endif
129
130 #endif
131
132 #endif
133
134 #endif
135
136 #endif
137
138 #endif
139
140 #endif
141
142 #endif
143
144 #endif
145
146 #endif
147
148 #endif
149
150 #endif
151
152 #endif
153
154 #endif
155
156 #endif
157
158 #endif
159
160 #endif
161
162 #endif
163
164 #endif
165
166 #endif
167
168 #endif
169
170 #endif
171
172 #endif
173
174 #endif
175
176 #endif
177
178 #endif
179
180 #endif
181
182 #endif
183
184 #endif
185
186 #endif
187
188 #endif
189
190 #endif
191
192 #endif
193
194 #endif
195
196 #endif
197
198 #endif
199
200 #endif
201
202 #endif
203
204 #endif
205
206 #endif
207
208 #endif
209
210 #endif
211
212 #endif
213
214 #endif
215
216 #endif
217
218 #endif
219
220 #endif
221
222 #endif
223
224 #endif
225
226 #endif
227
228 #endif
229
230 #endif
231
232 #endif
233
234 #endif
235
236 #endif
237
238 #endif
239
240 #endif
241
242 #endif
243
244 #endif
245
246 #endif
247
248 #endif
249
250 #endif
251
252 #endif
253
254 #endif
255
256 #endif
257
258 #endif
259
260 #endif
261
262 #endif
263
264 #endif
265
266 #endif
267
268 #endif
269
270 #endif
271
272 #endif
273
274 #endif
275
276 #endif
277
278 #endif
279
280 #endif
281
282 #endif
283
284 #endif
285
286 #endif
287
288 #endif
289
290 #endif
291
292 #endif
293
294 #endif
295
296 #endif
297
298 #endif
299
300 #endif
301
302 #endif
303
304 #endif
305
306 #endif
307
308 #endif
309
310 #endif
311
312 #endif
313
314 #endif
315
316 #endif
317
318 #endif
319
320 #endif
321
322 #endif
323
324 #endif
325
326 #endif
327
328 #endif
329
330 #endif
331
332 #endif
333
334 #endif
335
336 #endif
337
338 #endif
339
340 #endif
341
342 #endif
343
344 #endif
345
346 #endif
347
348 #endif
349
350 #endif
351
352 #endif
353
354 #endif
355
356 #endif
357
358 #endif
359
360 #endif
361
362 #endif
363
364 #endif
365
366 #endif
367
368 #endif
369
370 #endif
371
372 #endif
373
374 #endif
375
376 #endif
377
378 #endif
379
380 #endif
381
382 #endif
383
384 #endif
385
386 #endif
387
388 #endif
389
390 #endif
391
392 #endif
393
394 #endif
395
396 #endif
397
398 #endif
399
400 #endif
401
402 #endif
403
404 #endif
405
406 #endif
407
408 #endif
409
410 #endif
411
412 #endif
413
414 #endif
415
416 #endif
417
418 #endif
419
420 #endif
421
422 #endif
423
424 #endif
425
426 #endif
427
428 #endif
429
430 #endif
431
432 #endif
433
434 #endif
435
436 #endif
437
438 #endif
439
440 #endif
441
442 #endif
443
444 #endif
445
446 #endif
447
448 #endif
449
450 #endif
451
452 #endif
453
454 #endif
455
456 #endif
457
458 #endif
459
460 #endif
461
462 #endif
463
464 #endif
465
466 #endif
467
468 #endif
469
470 #endif
471
472 #endif
473
474 #endif
475
476 #endif
477
478 #endif
479
480 #endif
481
482 #endif
483
484 #endif
485
486 #endif
487
488 #endif
489
490 #endif
491
492 #endif
493
494 #endif
495
496 #endif
497
498 #endif
499
500 #endif
501
502 #endif
503
504 #endif
505
506 #endif
507
508 #endif
509
510 #endif
511
512 #endif
513
514 #endif
515
516 #endif
517
518 #endif
519
520 #endif
521
522 #endif
523
524 #endif
525
526 #endif
527
528 #endif
529
530 #endif
531
532 #endif
533
534 #endif
535
536 #endif
537
538 #endif
539
540 #endif
541
542 #endif
543
544 #endif
545
546 #endif
547
548 #endif
549
550 #endif
551
552 #endif
553
554 #endif
555
556 #endif
557
558 #endif
559
560 #endif
561
562 #endif
563
564 #endif
565
566 #endif
567
568 #endif
569
570 #endif
571
572 #endif
573
574 #endif
575
576 #endif
577
578 #endif
579
580 #endif
581
582 #endif
583
584 #endif
585
586 #endif
587
588 #endif
589
590 #endif
591
592 #endif
593
594 #endif
595
596 #endif
597
598 #endif
599
600 #endif
601
602 #endif
603
604 #endif
605
606 #endif
607
608 #endif
609
610 #endif
611
612 #endif
613
614 #endif
615
616 #endif
617
618 #endif
619
620 #endif
621
622 #endif
623
624 #endif
625
626 #endif
627
628 #endif
629
630 #endif
631
632 #endif
633
634 #endif
635
636 #endif
637
638 #endif
639
640 #endif
641
642 #endif
643
644 #endif
645
646 #endif
647
648 #endif
649
650 #endif
651
652 #endif
653
654 #endif
655
656 #endif
657
658 #endif
659
660 #endif
661
662 #endif
663
664 #endif
665
666 #endif
667
668 #endif
669
670 #endif
671
672 #endif
673
674 #endif
675
676 #endif
677
678 #endif
679
680 #endif
681
682 #endif
683
684 #endif
685
686 #endif
687
688 #endif
689
690 #endif
691
692 #endif
693
694 #endif
695
696 #endif
697
698 #endif
699
700 #endif
701
702 #endif
703
704 #endif
705
706 #endif
707
708 #endif
709
710 #endif
711
712 #endif
713
714 #endif
715
716 #endif
717
718 #endif
719
720 #endif
721
722 #endif
723
724 #endif
725
726 #endif
727
728 #endif
729
730 #endif
731
732 #endif
733
734 #endif
735
736 #endif
737
738 #endif
739
740 #endif
741
742 #endif
743
744 #endif
745
746 #endif
747
748 #endif
749
750 #endif
751
752 #endif
753
754 #endif
755
756 #endif
757
758 #endif
759
760 #endif
761
762 #endif
763
764 #endif
765
766 #endif
767
768 #endif
769
770 #endif
771
772 #endif
773
774 #endif
775
776 #endif
777
778 #endif
779
780 #endif
781
782 #endif
783
784 #endif
785
786 #endif
787
788 #endif
789
790 #endif
791
792 #endif
793
794 #endif
795
796 #endif
797
798 #endif
799
800 #endif
801
802 #endif
803
804 #endif
805
806 #endif
807
808 #endif
809
810 #endif
811
812 #endif
813
814 #endif
815
816 #endif
817
818 #endif
819
820 #endif
821
822 #endif
823
824 #endif
825
826 #endif
827
828 #endif
829
830 #endif
831
832 #endif
833
834 #endif
835
836 #endif
837
838 #endif
839
840 #endif
841
842 #endif
843
844 #endif
845
846 #endif
847
848 #endif
849
850 #endif
851
852 #endif
853
854 #endif
855
856 #endif
857
858 #endif
859
860 #endif
861
862 #endif
863
864 #endif
865
866 #endif
867
868 #endif
869
870 #endif
871
872 #endif
873
874 #endif
875
876 #endif
877
878 #endif
879
880 #endif
881
882 #endif
883
884 #endif
885
886 #endif
887
888 #endif
889
890 #endif
891
892 #endif
893
894 #endif
895
896 #endif
897
898 #endif
899
900 #endif
901
902 #endif
903
904 #endif
905
906 #endif
907
908 #endif
909
910 #endif
911
912 #endif
913
914 #endif
915
916 #endif
917
918 #endif
919
920 #endif
921
922 #endif
923
924 #endif
925
926 #endif
927
928 #endif
929
930 #endif
931
932 #endif
933
934 #endif
935
936 #endif
937
938 #endif
939
940 #endif
941
942 #endif
943
944 #endif
945
946 #endif
947
948 #endif
949
950 #endif
951
952 #endif
953
954 #endif
955
956 #endif
957
958 #endif
959
960 #endif
961
962 #endif
963
964 #endif
965
966 #endif
967
968 #endif
969
970 #endif
971
972 #endif
973
974 #endif
975
976 #endif
977
978 #endif
979
980 #endif
981
982 #endif
983
984 #endif
985
986 #endif
987
988 #endif
989
990 #endif
991
992 #endif
993
994 #endif
995
996 #endif
997
998 #endif
999
1000 #endif
    
```

2.以修改 eq_tab_custom 的第二个频点为例

版权所有，侵权必究



```
#define USER_EQ_LIVE_UPDATE          1//EQ 旋钮实时更新
//BASS 调节第几个索引 USER_EQ_BASS_INDEX
#define USER_EQ_BASS_INDEX          1//不要超出 eq 段
//低音
void user_eq_bass_terble_set(int bass,int terble){
    #if (defined (USER_EQ_LIVE_UPDATE) && USER_EQ_LIVE_UPDATE)
    eq_tab_custom[USER_EQ_BASS_INDEX].gain = bass<<20;
    // eq_tab_custom[USER_EQ_TERBLE_INDEX].gain = terble<<20;

    eq_mode_set(EQ_MODE_CUSTOM);
    #endif
    return;
}
```

9.修改任意 eq 效果的某个频点增益

1) 把需要修改的 eq 效果对应的数组复制到 eq_tab_custom 中然后调用 eq_mode_set

```
Eg:
memcpy(eq_tab_custom,eq_tab_jazz,sizeof(struct eq_seg_info));
eq_mode_set(EQ_MODE_CUSTOM);
```

2) 结合使用外部 eq 可以实时设置程序内置 6 种 eq 效果+外部 eq 的任意频点增益

129.SDK1.1.1 对箱功能需要与 692x 一样 20201224 SQ

692 对箱：按键配对、发起配对为主机、非主从切换、主机为 L 耳

```

    帮助(H) app_config.h - ac696n_soundbox_sdk_v1.1.1 - Visual Studio Code
    ... :fg.h C app_config.h X C user_ac6969d_cfg.h C user_fun.c C power_interface.h C
    SDK > apps > soundbox > include > C app_config.h > CONFIG_TWS_MASTER_AS_LEFT
    122 #endif //(CONFIG_BT_MODE != BT_NORMAL)
    123
    124
    125 #if TCFG_USER_TWS_ENABLE
    126
    127 //*****
    128 //                                对耳配置方式配置
    129 //*****
    130 #define CONFIG_TWS_CONNECT_SIBLING_TIMEOUT    4    /* 开机或超时断开后对耳互连超
    131 #define CONFIG_TWS_REMOVE_PAIR_ENABLE        /* 不连手机的情况下双击按键删
    132 #define CONFIG_TWS_POWEROFF_SAME_TIME       1    /*按键关机时两个耳机同时关机*
    133
    134 #define ONE_KEY_CTL_DIFF_FUNC                1    /*通过左右耳实现一个按键控制两
    135 #define CONFIG_TWS_SCO_ONLY_MASTER         0    /*通话的时候只有主机出声音*/
    136
    137 /* 配对方式选择 */
    138 #define CONFIG_TWS_PAIR_BY_CLICK            0    /* 按键发起配对 */
    139 #define CONFIG_TWS_PAIR_BY_AUTO            1    /* 开机自动配对 */
    140 #define CONFIG_TWS_PAIR_BY_FAST_CONN      2    /* 开机快速连接,连接速度比自动
    141 #define CONFIG_TWS_PAIR_MODE              CONFIG_TWS_PAIR_BY_CLICK
    142
    143 #define CONFIG_TWS_USE_COMMON_ADDR         0    /* tws 使用公共地址 */
    144 #define CONFIG_TWS_PAIR_ALL_WAY          1    /* tws 任何时候 链接搜索 */
    145
    146
    147 /* 声道确定方式选择 */
    148 #define CONFIG_TWS_MASTER_AS_LEFT         0 //主机作为左耳
    149 #define CONFIG_TWS_AS_LEFT_CHANNEL        1 //固定左耳
    150 #define CONFIG_TWS_AS_RIGHT_CHANNEL       2 //固定右耳
    151 #define CONFIG_TWS_LEFT_START_PAIR        3 //双击发起配对的耳机做左耳
    152 #define CONFIG_TWS_RIGHT_START_PAIR       4 //双击发起配对的耳机做右耳
    153 #define CONFIG_TWS_EXTERN_UP_AS_LEFT     5 //外部有上拉电阻作为左耳
    154 #define CONFIG_TWS_EXTERN_DOWN_AS_LEFT    6 //外部有下拉电阻作为左耳
    155 #define CONFIG_TWS_SECECT_BY_CHARGESTORE  7 //充电仓决定左右耳
    156 #define CONFIG_TWS_CHANNEL_SELECT        CONFIG_TWS_MASTER_AS_LEFT //配对
    157
    158 #define CONFIG_TWS_CHANNEL_CHECK_IO       IO_PORTA_07 //
    159
    160
  
```

130.SDK1.1.1 打开混响 TCFG_MIC_EFFECT_ENABLE 这个宏后，使用测试盒进行通话测试会有啸叫的处理方法 20201226 LJW

1.请按如下进行修改，但是有个缺点是通话时不能开混响：

```
board_ac696x_demo_cfg.h X mic_effect.h mic_effect.c audio_aec.c audio_config.h
ac696n_soundbox_sdk_v1.1.1 > SDK > apps > soundbox > board > br25 > board_ac696x_demo > C board_ac696x_demo_cfg.h >
529 // mic effect 配置 //
530 //*****
531 #define TCFG_MIC_EFFECT_ENABLE 1//DISABLE
532 #define TCFG_MIC_EFFECT_DEBUG 0//调试打印
533 #define TCFG_MIC_EFFECT_ONLINE_ENABLE 0//混响音效在线调试使能
534 #if ((TCFG_ONLINE_ENABLE == 0) && TCFG_MIC_EFFECT_ONLINE_ENABLE)
535 #undef TCFG_ONLINE_ENABLE
536 #define TCFG_ONLINE_ENABLE 1
537 #endif
538
539 #define MIC_EFFECT_REVERB 0
540 #define MIC_EFFECT_ECHO 1
541 // #define TCFG_MIC_EFFECT_SEL MIC_EFFECT_REVERB
542 #define TCFG_MIC_EFFECT_SEL MIC_EFFECT_ECHO
543
544 #if TCFG_MIC_EFFECT_ENABLE
545 #define TCFG_CALLING_EN_REVERB 0//ENABLE
546 #else
547 #define TCFG_CALLING_EN_REVERB DISABLE
548 #endif

C audio_enc.c C mic_effect.h C mic_effect.c X C btc C audio_enc.c C mic_effect.h
ac696n_soundbox_sdk_v1.1.1 > SDK > cpu > br25 > audio_mic > C mic_effect.c > mic_effect_start(void)
360 /*-----
361
362 volatile u8 effect_flag = 0;
363
364 bool mic_effect_start(void)
365 {
366     effect_flag = 1;
367     bool ret = false;
368     printf("\n--func=%s\n", __FUNCTION__);
369     if (__this) {
370         log_e("reverb is already start \n");
371         return ret;
372     }
}

ac696n_soundbox_sdk_v1.1.1 > SDK > cpu > br25 > audio_n
498
499 @return
500 @note
501 */
502 /*-----
503 void mic_effect_stop(void)
504 {
505     effect_flag = 0;
506     mic_effect_destroy(&__this);
507 }
508 /**@brief (mic数据流)混响暂停接口
509 @param
510 @return
```

```

extern volatile u8 effect_flag;
int audio_mic_open(struct adc_mic_ch *mic, u16 sample_rate, u8 gain)
{
    r_printf("audio_mic_open\n");
    if (mic_var.init_flag) {
        log_i("\n mic init_flag \n\n\n");
        mic_var.init_flag = 0;
        atomic_set(&mic_var.used, 0);
        os_mutex_create(&mic_var.mutex);
        // mic_var.adc_buf = (s16 *)zalloc(MIC_ADC_BUFS_SIZE * 2);
        if(effect_flag){
            mic_var.adc_buf = (s16 *)zalloc(2 * 64 * 2);
        }
        else{
            mic_var.adc_buf = (s16 *)zalloc(2 * 256 * 2);
        }

        mic_var.states = 0;
    }

    os_mutex_pend(&mic_var.mutex, 0);
    if (!mic_var.adc_buf) {
        log_i("\n mic malloc \n\n\n");
        //mic_var.adc_buf = (s16 *)zalloc(MIC_ADC_BUFS_SIZE * 2);
        if(effect_flag){
            mic_var.adc_buf = (s16 *)zalloc(2 * 64 * 2);
        }
        else{
            mic_var.adc_buf = (s16 *)zalloc(2 * 256 * 2);
        }
    }

    if (mic_var.states == 0) {
        atomic_inc_return(&mic_var.used);
        audio_adc_mic_open(mic, AUDIO_ADC_MIC_CH, &adc_hdl);
        audio_adc_mic_set_sample_rate(mic, sample_rate);
        audio_adc_mic_set_gain(mic, gain);
        // audio_adc_mic_set_buffs(mic, mic_var.adc_buf, MIC_ADC_IRQ_POINTS);
        if(effect_flag){
            audio_adc_mic_set_buffs(mic, mic_var.adc_buf, 64 * 2, 2);
        }
        else{
            audio_adc_mic_set_buffs(mic, mic_var.adc_buf, 256 * 2, 2);
        }
        mic_var.sample_rate = sample_rate;
        mic_var.states = 1;
    }
}

// In esco_enc.c:
if (!esco_enc) {
    esco_enc = zalloc(sizeof(*esco_enc));
    //os_sem_create(&esco_enc->pcm_frame_sem, 0);
    audio_encoder_open(&esco_enc->encoder, &esco_enc_input, encode
    audio_encoder_set_handler(&esco_enc->encoder, &esco_enc_handle
    audio_encoder_set_fmt(&esco_enc->encoder, &fmt);
    audio_encoder_set_event_handler(&esco_enc->encoder, esco_enc_e
    audio_encoder_set_output_buffs(&esco_enc->encoder, esco_enc->
    sizeof(esco_enc->output_frame),
    audio_encoder_start(&esco_enc->encoder);
    printf("esco sample_rate: %d,mic_gain:%d\n", fmt.sample_rate,
    #if 0
    audio_adc_mic_open(&esco_enc->mic_ch, AUDIO_ADC_MIC_CH, &adc_h
    audio_adc_mic_set_sample_rate(&esco_enc->mic_ch, fmt.sample_ra
    audio_adc_mic_set_gain(&esco_enc->mic_ch, app_var.aec_mic_gain
    audio_adc_mic_set_buffs(&esco_enc->mic_ch, esco_enc->adc_buf,
    ESCO_ADC_IRQ_POINTS * 2, ESCO_ADC_BUF
    esco_enc->adc_output.handler = adc_mic_output_handler;
    audio_adc_add_output_handler(&adc_hdl, &esco_enc->adc_output);
    audio_adc_mic_start(&esco_enc->mic_ch);
    #else
    // In esco_enc_open function, comment out this macro
    #if 0//TCFG_MIC_EFFECT_ENABLE
    IF (fmt.sample_rate != MIC_EFFECT_SAMPLERATE) {
        esco_enc->adc_output.handler = adc_mic_output_handler_down
    } else {
        esco_enc->adc_output.handler = adc_mic_output_handler;
    }
    audio_mic_open(&esco_enc->mic_ch, MIC_EFFECT_SAMPLERATE, app_v
    #else
    esco_enc->adc_output.handler = adc_mic_output_handler;
    audio_mic_open(&esco_enc->mic_ch, fmt.sample_rate, app_var.aec
    #endif
    audio_mic_add_output(&esco_enc->adc_output);
    audio_mic_start(&esco_enc->mic_ch);
    #endif
    clock_set_cur();
}
    
```

```

break;
case BT_STATUS_LAST_CALL_TYPE_CHANGE:
    log_info("BT STATUS LAST CALL TYPE CHANGE\n");
    extern void mic_effect_stop(void);
    if(effect_flag){
        mic_effect_stop();
    }
}
    
```

2.如果通话结束后要自动打开混响, 请按如下修改:

添加方框内的内容

```

    case BT_STATUS_PHONE_ACTIVE:
        log_info("BT_STATUS_PHONE_ACTIVE\n");
        bt_status_phone_active(bt);
        break;
    case BT_STATUS_PHONE_HANGUP:
        log_info(" BT_STATUS_PHONE_HANGUP\n");
        bt_status_phone_hangup(bt);
        call_flag_a = 1;
        break;
    case BT_STATUS_PHONE_NUMBER:
        log_info("BT_STATUS_PHONE_NUMBER\n");
    
```

```

    volatile u8 call_flag_a = 0;
    static int bt_connction_status_event_handler(struct bt
    {
        log_debug("-----bt_connction_st
    if (bt_status_event_filter(bt) == false) {
        return false;
    }
    }
    
```

esco_audio_res_close函数中, 添加方框中的内容

```

    #endif /*TCFG_ESCO_USE_SPEC_MIX_LEN*/
    app_audio_state_exit(APP_AUDIO_STATE_CALL);
    bt_esco_dec->dec.start = 0;
    if(call_flag_a)
    {
        extern bool mic_effect_start(void);
        mic_effect_start();
        call_flag_a = 0;
    }
    }
    
```

131.SDK1.1.1 打开 linein 录音，播放的声音不正常的处理方法

20210104

LJW

```

    }
    }
    u16 sr = audio_mixer_get_cur_sample_rate(&recorder_mixer);
    fmt.sample_rate = RECORDER_MIX_DEFAULT_SR_LIMIT(sr);
    printf("[%s], fmt.sample_rate = %d\n", __FUNCTION__, fmt.sample_rate);
    fmt.cut_head_time = 300; //录音文件去头时间,单位ms
    fmt.cut_tail_time = 300; //录音文件去尾时间,单位ms
    fmt.limit_size = 3000; //录音文件大小最小限制, 单位byte
    fmt.source = ENCODE_SOURCE_MIX; //录音输入源
    fmt.err_callback = recorder_mix_err_callback;
    
```

改成linein的采样率: 44100

132.SDK1.1.1 麦克风能量获取方法

20210105

LHY

链接: https://pan.baidu.com/s/1n5jmjU4CS6r_2PMCOcx5qA

提取码: 8888

1、下载上面代码，对比 SDK1.1.1 公版程序，修改到自己工程中

2、通过下面函数去获取麦克风能量值

```

511 int audio_mic_energy_get(void)
512 {
513     #if AUDIO_OUTPUT_AUTOMUTE
514     int audio_energy_detect_energy_get(void *_hdl, u8 ch);
515     if (user_mic_energy_hdl) {
516         r_printf("audio_mic_energy_get = %d\n", audio_energy_detect_energy_get(user_mic_energy_hdl, BIT(0)));
517         return audio_energy_detect_energy_get(user_mic_energy_hdl, BIT(0));
518     }
519     return (-1);
520 #else
521     return 0;
522 #endif
523 }
524

```

133.SDK1.1.1 的 SD 选择 C 组、CMD 检测，录音设备为 SD 卡时，录音时 SD 卡会掉线的处理方法 20210108 LJW

有以下三种处理方法：

- (1) 将 CMD 检测改为 CLK 检测；

```

//C组IO: CMD:PA4 CLK:PA2 DAT0:PA3 //F组IO: CMD:PB6 CLK:PB7 DAT0:PB4
#define TCFG_SD0_ENABLE ENABLE_THIS_MODULE
#define TCFG_SD0_PORTS 'C'
#define TCFG_SD0_DAT_MODE 1//AC696x不支持4线模式
#define TCFG_SD0_DET_MODE SD_CLK_DECT //改为clk检测
#define TCFG_SD0_DET_IO IO_PORT_DM//当SD_DET_MODE为2时有效
#define TCFG_SD0_DET_IO_LEVEL 0//IO检查, 0: 低电平检测到卡。 1: 高电平(外部电源)检测到卡。 2: 高电平(SD卡电源)检测到卡。
#define TCFG_SD0_CLK (3000000*2L)

```

- (2) 关掉 MIC 的内部偏置 PA2，使用外部偏置；
- (3) MIC 使用省电容的方式；

134.SDK 1.2.0 linein 为 DAC 进使用打断方式播放提示音，一直有 linein 背景音 20210112 SQ

```

509
510 //linein PP 状态记录 与查询返回
511 bool user_comm_linein_status(u8 cmd){
512     static bool _status_ = 1;
513
514     if(1 == cmd){
515         _status_ = cmd;
516     }else if(!cmd){
517         _status_ = 0;
518     }
519
520     return _status_;
521 }
522

```

135.

136.//linein PP 状态记录 与查询返回

137.bool user_comm_linein_status(u8 cmd){

138. static bool _status_ = 1;

139.

版权所有，侵权必究

```
140.   if(1 == cmd){
141.       _status_ = cmd;
142.   }else if(!cmd){
143.       _status_ = 0;
144.   }
145.
146.   return _status_;
147.}
```

```
case KEY_MUSIC_PP:
    log_info("KEY_MUSIC_PP\n");
    /* app_task_put_key_msg(KEY_TEST_DEMO_0,1234); //test demo
    linein_last_onoff = linein_volume_pp();
    linein_last_onoff ? ui_update_status(STATUS_LINEIN_PLAY)\
    : ui_update_status(STATUS_LINEIN_PAUSE);
    user_comm_linein_status(linein_get_status());
    user_pa_linein(!linein_last_onoff);
148.   break;
```

```
149.user_comm_linein_status(linein_get_status());
```

```
95  /*-----*/
96  void app_linein_task()
97  {
98      int res;
99      int err = 0;
100     int msg[32];
101
102     user_lienin_tone_play_check_en(1);
103
104     #if TCFG_APP_BT_EN
105         linein_bt_back_flag = get_bt_back_flag();//从蓝牙后台返回标志
106         set_bt_back_flag(0);
107     #else
108         linein_bt_back_flag = 0;
109     #endif
110     log_info("linein_bt_back_flag == %d linein_last_onoff = %d\n", \
111             linein_bt_back_flag, linein_last_onoff);
112
113     linein_app_init();//初始化时钟和开启ui
114
115     err = tone_play_with_callback_by_name(tone_table[IDEX_TONE_LINEIN], 1,
116                                           line_tone_play_end_callback, (void *)IDEX_TONE_LINEIN);
117     // if (err) { //
118     //     ///提示音播放失败, 直接推送KEY_MUSIC_PLAYER_START启动播放
119     //     app_task_put_key_msg(KEY_LINEIN_START, 0);
120     // }
121     user_linein_status(1);
122     while (1) {
123         app_task_get_msg(msg, ARRAY_SIZE(msg), 1);
124
125         switch (msg[0]) {
126             case APP_MSG_SYS_EVENT:
127                 if (linein_sys_event_handler((struct sys_event *)&msg[1]) == false) {
128                     app_default_event_deal((struct sys_event *)&msg[1]); //由common统一处理
129                 }
130                 break;
131             default:
132                 break;
133         }
134
135         if (app_task_exitting()) {
136             user_lienin_tone_play_check_en(0);
```

```
void user_lienin_tone_play_check(void *priv){
#if 1//(defined(USER_TONE_PLAY_MODE) && USER_TONE_PLAY_MODE)

    u16 *id = (u16 *)priv;
    bool tone_status_cur = 0;
    static bool tone_status_old = 0;
    static u32 timer_flag = 0;

    if(!app_check_curr_task(APP_LINEIN_TASK) || !priv){
        if(priv){
            *id = 0;
        }
        return;
    }

    tone_status_cur = (TONE_START == tone_get_status())?1:0;

    if(tone_status_old != tone_status_cur){
        if(!tone_status_old && tone_status_cur){//开始播放提示音
            if(linein_get_status()/*user_linein_status(0xff)*/){
                linein_stop();
            }
            r_printf("tone status=%x old=%x",tone_status_cur,tone_status_old);
            tone_status_old = tone_status_cur;
        }else if(tone_status_old && !tone_status_cur){//提示音播放结束
            if(!timer_flag){
                timer_flag = timer_get_ms();
            }

            if(timer_get_ms()-timer_flag>=900){//延时一段时间后打开
                if(user_linein_status(0xff) && !linein_get_status()){
                    r_printf("user linein start");
                    linein_start();
                }

                r_printf("1111 tone status=%x old=%x",tone_status_cur,tone_status_old);
                tone_status_old = tone_status_cur;
                timer_flag = 0;
            }
        }
    }
}

    *id = sys_hi_timeout_add(id,user_lienin_tone_play_check,tone_status_cur?10:50);
```

```
#endif
}

void user_lienin_tone_play_check_en(u8 cmd){
    #if 1//(defined(USER_TONE_PLAY_MODE) && USER_TONE_PLAY_MODE)
    bool check_en = 0;
    static u16 linein_check_id = 0;

    if(1 == cmd){
        check_en = cmd;
    }else if(0 == cmd){
        check_en = cmd;
    }else{
        return ;
    }

    if(check_en){
        if(!linein_check_id){
            linein_check_id = sys_hi_timeout_add(&linein_check_id,user_lienin_tone_play_check,10);
        }
    }else{
        if(linein_check_id){
            sys_s_hi_timeout_del(linein_check_id);
            linein_check_id = 0;
        }
    }
}
#endif
}
```

135.AC696X V111 版本的 SDK 对箱配对后来电接听近端无声音的问题 20210121 YP

关于 AC696X 音箱 SDK V111 版本，在关闭打印后，对箱配对后，来电接听电话，会出现近端没有声音的情况，系因为接听的时候，dac 这端没有数据流，导致 audio dac 挂起了，没有恢复的地方，导致近端没有声音，可以按照如下修改：

The image shows a C code editor with multiple panes of code for an audio decoder. Red annotations and arrows highlight specific parts of the code:

- Line 772:** A comment in Chinese: "写一个恢复audio handler的函数" (Write a function to restore the audio handler).
- Line 777:** A function call: `if(bt_esco_dec && bt_esco_dec->dec.start){ audio_decoder_resume(&bt_esco_dec->dec.decoder); }`. A red arrow points to this line with the note: "在esco_dec_start函数里面, 添加恢复audio handler的定时器, 用来唤醒音频" (Add a timer to restore the audio handler in the esco_dec_start function to wake up the audio).
- Line 809:** A function call: `if(dec->start_resume_time){ sys_hi_timer_del(dec->start_resume_time); }`. A red arrow points to this line with the note: "恢复了audio handler后要delete掉启动的定时器" (After restoring the audio handler, delete the start timer).
- Line 809:** Another function call: `if(dec->start_resume_time){ sys_hi_timer_add(NULL, &esco_start_resume_func, 10); }`. A red arrow points to this line with the note: "首先添加一个变量" (First add a variable).
- Line 809:** A comment: "如果先没有buf复用, 则没有先后顺序要求" (If there is no buf reuse first, there are no order requirements).
- Line 809:** A comment: "esco关闭的时候也要delete定时器的" (When esco closes, the timer must also be deleted).

136.AC696X V111 版本的 SDK 连接手机后, 无法控制手机拍照的处理方法 20210126 LJW

The image shows a terminal window with the following code snippet:

```

SDK > apps > soundbox > task_manager > bt > bt_key_func.c > bt_key_hid_control()
571 @return
572 @note
573 */
574 /*-----*/
575 void bt_key_hid_control()
576 {
577     /* log_info("get_curr_channel_state:%x\n", get_curr_channel_state()); */
578     if (get_curr_channel_state() & HID_CH) {
579         log_info("KEY_HID_CONTROL\n");
580         user_send_cmd_prepare(USER_CTRL_HID_VOL_UP, 0, NULL);
581     }
582 }
    
```

A yellow box highlights the line `user_send_cmd_prepare(USER_CTRL_HID_VOL_UP, 0, NULL);` and a yellow arrow points to it from the left.

```
C board_ac696x_demo_cfg.h C bt_key_fun.c C bt_profile_config.c C board_config.h
SDK > apps > common > config > C bt_profile_config.c > ...
120 /*注意hid conn depend on dev company器1之后, 安卓手机会默认断开HID连接 */
121 const u8 hid_conn_depend_on_dev_company = 0;
122 const u8 sdp_get_remote_pnp_info = 0;
```

136.AC696X SDK121 暂停播放提示音 20210309 XNW

```
void tone_dec_list_pp(struct tone_dec_handle *dec)
{
    if(dec)
    {
        if(dec->dec_file)
        {
            audio_dec_app_pp(dec->dec_file->dec);
        }
        if(dec->dec_sin)
        {
            audio_dec_app_pp(dec->dec_sin->dec);
        }
    }
}

void tone_play_pp(void)
{
    tone_dec_list_pp(tone_dec);
}
```

```

tone_player.c | audio_dec_app.h | app_config.h | board_config.h | board_ac696x_demo_cfg.h | adv_key_setting.c | bt.c | board_ac696x_demo.c | adv_led_setting.c
132
133
134 ////////////////////////////////////////////////////
135
136 void tone_dec_list_pp(struct tone_dec_handle *dec)
137 {
138     if(dec)
139     {
140         if(dec->dec_file)
141         {
142             audio_dec_app_pp(dec->dec_file->dec);
143         }
144         if(dec->dec_sin)
145         {
146             audio_dec_app_pp(dec->dec_sin->dec);
147         }
148     }
149 }
150
151 void tone_play_pp(void)
152 {
153     tone_dec_list_pp(tone_dec);
154 }
155
156

```

137.AC696X V121 版本的 SDK 遥控功能和收音功能同时开异常处理方法 20210311 lzk

解决办法：由于新 SDK 优化了 FM：当打开数码管 UI 宏后是把显示的代码放在 timer5 里面跑，此时跟红外使用的 timer5 冲突引起。

处理方法：把红外的 timer5 改为 timer3

<pre> music.c board_config.h board_ac696x_demo_cfg.h irflt.c (工作物) irkey.c irflt.c led7_timer.c SDK > cpu > br25 > C irflt.c > IR_TIMER 1 #include "asm/includes.h" 2 #include "asm/irflt.h" 3 #include "timer.h" 4 #include "generic/gpio.h" 5 6 #define ir_log log_d 7 8 //红外定时器定义 9- #define IR_TIMER TIMER5 10- #define IR_IRQ_TIME_IDX IRQ_TIMES5_IDX 11- #define IR_TIME_REG JL_TIMERS5 12 13 #define IRFLT_OUTPUT_TIMER_SEL(x) SFR(JL_IOMAP->CON0, 5, 3, x) 14 15 16 IR_CODE ir_code; ///<红外遥控码 17 </pre>	<pre> 1 #include "asm/includes.h" 2 #include "asm/irflt.h" 3 #include "timer.h" 4 #include "generic/gpio.h" 5 6 #define ir_log log_d 7 8 //红外定时器定义 9+ #define IR_TIMER TIMER3 10+ #define IR_IRQ_TIME_IDX IRQ_TIME3_IDX 11+ #define IR_TIME_REG JL_TIMERS3 12 13 #define IRFLT_OUTPUT_TIMER_SEL(x) SFR(JL_IOMAP->CON0, 5, 3, x) 14 15 16 IR_CODE ir_code; ///<红外遥控码 17 </pre>
---	--

```
91 /*
92 /* #define APP_TIMER_CLK          clk_get("timer")//选osc时钟源 */
93 #define APP_TIMER_CLK          12000000L //pll12m
94 #define TIMER_UNIT_MS          1
95 #define MAX_TIME_CNT 0x07ff //分频准确范围, 更具实际情况调整
96 #define MIN_TIME_CNT 0x0030
97 void set_ir_clk(void)
98 {
99     u32 prd_cnt;
100     u8 index;
101     for (index = 0; index < (sizeof(timer_div) / sizeof(timer_div[0]));
102         prd_cnt = TIMER_UNIT_MS * (APP_TIMER_CLK / 1000) / timer_div[index];
103         if (prd_cnt > MIN_TIME_CNT && prd_cnt < MAX_TIME_CNT) {
104             break;
105         }
106     }
107     ir_code.timer_pad = prd_cnt;
108     cmp_start = 0;
109     request_irq(IR_IRQ_TIME_IDX, 5, timer_ir_isr, 0);
110     /* IR_TIME_REG->CON = ((index << 4) | BIT(3) | BIT(1) | BIT(0));//选
111     JL_IOMAP->CON0 |= BIT(21); //这里已选了timer5, 时钟源选10信号源的pll_12m, 不
112     IR_TIME_REG->CON = ((index << 4) | BIT(2) | BIT(1) | BIT(0));
113 }
114
```

138.AC696X V020 版本的 SDK 普通包关 TWS，设置立体声输出，通话结束后会复位的处理方法 20210317 LJW

更新库文件：
链接：<https://pan.baidu.com/s/14MUby8KwRVD4HnN14dm8xw>
提取码：8888

139.AC696X V020 版本的 SDK 设置立体声输出,播左右声道歌曲后 DAC 没有声音或声音变小的处理方法 20210317 LJW

```
log.h | bt_decode.c X | audio_config.h | user_cfg.c | earphone.c | app_audio.c | tone...
k > apps > earphone > bt_decode.c > music_auto_mute_handler(u8, u8)
670 static u8 mute_R = 0;
671
672 __BANK_TONE_ENTRY
673 static void music_auto_mute_handler(u8 event, u8 channel)
674 {
675     switch (event) {
676     case DEC_EVENT_AUTO_MUTE:
677         r_printf("DEC_EVENT_AUTO_MUTE");
678 #if (AUDIO_OUTPUT_MODE == AUDIO_OUTPUT_STEREO)
679         if (channel == 0) {
680             r_printf("channel == 0");
681             if (mute_R) {
682                 /*break;*/
683             }
684             // app_audio_mute(AUDIO_MUTE_L_CH);
685             mute_L = 1;
686         } else if (channel == 1) {
687             r_printf("channel == 1");
688             if (mute_L) {
689                 /*break;*/
690             }
691             // app_audio_mute(AUDIO_MUTE_R_CH);
692             mute_R = 1;
693         } else if (channel == 2) {
694             r_printf("channel == 2");
695             mute_L = 1;
696             mute_R = 1;
697             /*声道分离：两个声道能量都变低的情况下不用mute全部声道,修复iphone按键音的问题*/
698             app_audio_mute(AUDIO_MUTE_DEFAULT);
699         }
700     #else
701         app_audio_mute(AUDIO_MUTE_DEFAULT);
702     #endif
703     /*log_info(">>>auto mute %d\n", channel);*/
704     break;

```

141.AC6965 系列设置 DAC 管脚高阻态 20210326 YWP

```
/*
ch: FL:BIT(0) FR:BIT(1) RL:BIT(2) RR:BIT(3)
例如: BIT(0)是 FL 左声道, BIT(1)是 FR 右声道, FLR BIT(0)|bit(1)是立体声
MUTE:1:MUTE 0:UNMUTE
*/
void audio_dac_mute_ch(u8 ch, u8 mute)
{
    if (mute) {

```

版权所有, 侵权必究

```
JL_ANA->DAA_CON1 &= ~((ch & 0x3) << 10);
JL_ANA->DAA_CON1 &= ~((ch & 0x3) << 13);
} else {
    JL_ANA->DAA_CON1 |= ((ch & 0x3) << 10);
    JL_ANA->DAA_CON1 |= ((ch & 0x3) << 13);
}
}
```

142.AC696N 系列的左右调反实现 20210402 YWP

本文档介绍有两种方式，针对 2Mbit 的应用和非 2Mbit 的 DAC 左右声道的调反应用。

第一类：针对非 2Mbit 包的 DAC 数据处理

这里适用于所有解码的左右声道调反。

```
int ops_lr(void *priv, struct audio_data_frame *in)
{
    s16 *f_lr;
    s16 tmp_l,tmp_r;
    u16 rlen;
    u32 len=0;
    putchar('h');
    // if((in->data_len- in->offset)>0)
    {
        f_lr = in->data;// + in->offset/2;
        len = in->data_len;// - in->offset;

        rlen = len>>2;  ////u8*4
        for(u16 i=0; i<rlen; i++) ///lrlrlr.....
        {
            tmp_l = f_lr[i*2];
            tmp_r = f_lr[i*2+1];
            f_lr[i*2] = tmp_r;
            f_lr[i*2+1] = tmp_l;
        }
    }
    return len;
}
```

```
C audio_dec.c 4 x C audio_stream.h 1 C stream_entry.c 2 C app_config.h
cpu:br25 > audio_dec > C audio_dec.c > ops_lr(void *, audio_data_frame *)
584 #else
585 int ops_lr(void *priv, struct audio_data_frame *in)
586 {
587     s16 *f_lr;
588     s16 tmp_l,tmp_r;
589     u16 rlen;
590     u32 len=0;
591     putchar('h');
592 // if((in->data_len- in->offset)>0)
593 {
594     f_lr = in->data;// + in->offset/2;
595     len = in->data_len;// - in->offset;
596
597     rlen = len>>2; ////u8*4
598     for(u16 i=0; i<rlen; i++) ///lrlrlr.....
599     {
600         tmp_l = f_lr[i*2];
601         tmp_r = f_lr[i*2+1];
602         f_lr[i*2] = tmp_r;
603         f_lr[i*2+1] = tmp_l;
604     }
605 }
606
607     return len;
608 }
609 #endif
610 /*-----*/
611 /**@brief 音频解码初始化
612 @param
613 @return
614 @note
615 */
616 /*-----*/
617 extern struct audio_dac_hdl dac_hdl;
618 struct audio_dac_channel default_dac;
619 int audio_dec_init()
620 {
```

```
struct __stream_entry *enage_ll;
enage_ll = stream_entry_open(&mixer, ops_lr, 0);
entries[entry_cnt++] = &enage_ll->entry;
```

```

719 #if AUDIO_VOCAL_REMOVE_EN
720     if (mix_vocal_remove_hdl) {
721         |   entries[entry_cnt++] = &mix_vocal_remove_hdl->entry;
722     }
723     if (mix_ch_switch) {
724         |   entries[entry_cnt++] = &mix_ch_switch->entry;
725     }
726 #endif
727
728 #if AUDIO_OUTPUT_AUTOMUTE
729     entries[entry_cnt++] = mix_out_automute_entry;
730 #endif
731
732     struct __stream_entry *enage_ll;
733     enage_ll = stream_entry_open(&mixer, ops_lr, 0);
734     entries[entry_cnt++] = &enage_ll->entry;
735
736     /* #if (AUDIO_OUTPUT_WAY == AUDIO_OUTPUT_WAY_DAC) */
737 #if AUDIO_OUTPUT_INCLUDE_DAC
738     audio_dac_new_channel(&dac_hdl, &default_dac);
739     struct audio_dac_channel_attr attr;
740     audio_dac_channel_get_attr(&default_dac, &attr);
741     attr.delay_time = 50;
742     attr.protect_time = 5;

```

audio_dec_init

C audio_dec.c 4 C audio_stream.h 1 C stream_entry.c 2 X C app_config.h

apps > common > audio > stream > C stream_entry.c > stream_entry_data_handler(audio_stream_entry *, audio_data_frame *, audio_data_frame *)

```

4 static int stream_entry_data_handler(struct audio_stream_entry *entry,
5                                     struct audio_data_frame *in,
6                                     struct audio_data_frame *out)
7 {
8     struct __stream_entry *hdl = container_of(entry, struct __stream_entry, entry);
9     if (in->data_len == 0) {
10         |   return 0;
11     }
12
13     if (hdl->is_end == 0) {
14         |   //中间节点才需要往后传数据, 终节点不用
15         |   out->data = in->data;
16         |   out->data_len = in->data_len;
17         |   out->channel = in->channel;
18         |   out->sample_rate = in->sample_rate;
19     }
20
21     if (hdl->data_callback) {
22         |   // return hdl->data_callback(hdl->data_priv, in);
23         |   // if((in->data_len - in->offset)>0)
24         |   if((in->offset)==0)
25         |   {
26         |       |   return hdl->data_callback(hdl->data_priv, in);//(hdl->data_priv, in->data + in->offset/2, in->data_len - in->offset);
27         |   }
28     }
29
30     return in->data_len;
31 }

```

```

if (hdl->data_callback) {
    // return hdl->data_callback(hdl->data_priv, in);
    if((in->offset)==0)
    {
        return hdl->data_callback(hdl->data_priv, in);//(hdl->data_priv, in->data
+ in->offset/2, in->data_len - in->offset);
    }
}

```

第二类：针对 2M 包应用的解码数据左右通道的调反

这里介绍 a2dp 蓝牙解码的处理案例。

```
int ops_lr_bt(void *priv, struct audio_data_frame *in)
{
    s16 *f_lr;
    s16 tmp_l,tmp_r;
    u16 rlen;
    u32 len=0;
    // putchar('h');
    {
        f_lr = in->data;///in->offset/2;
        len = in->data_len;///- in->offset;

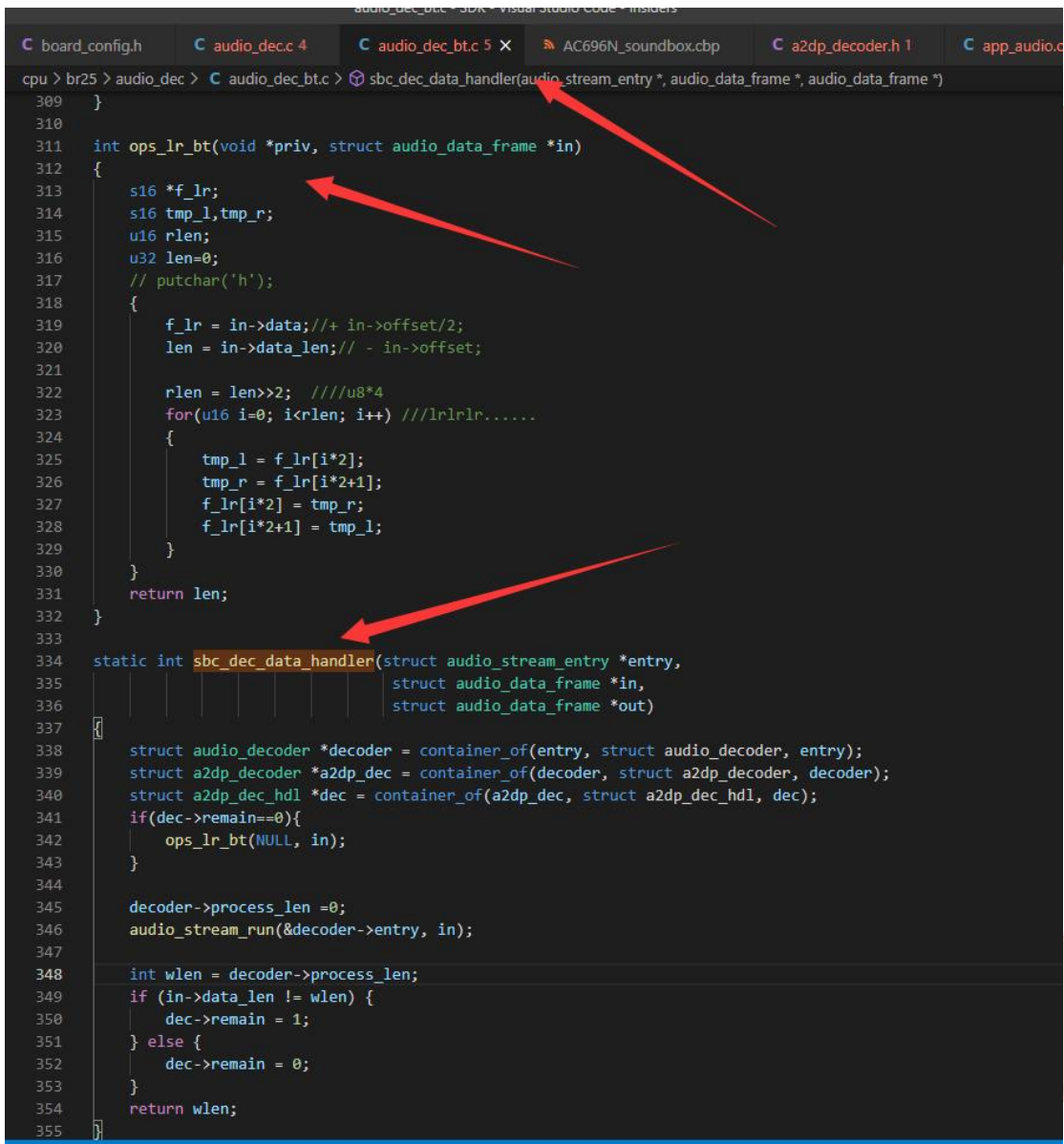
        rlen = len>>2; ////u8*4
        for(u16 i=0; i<rlen; i++) ///lrlrlr.....
        {
            tmp_l = f_lr[i*2];
            tmp_r = f_lr[i*2+1];
            f_lr[i*2] = tmp_r;
            f_lr[i*2+1] = tmp_l;
        }
    }
    return len;
}

static int sbc_dec_data_handler(struct audio_stream_entry *entry,
                                struct audio_data_frame *in,
                                struct audio_data_frame *out)
{
    struct audio_decoder *decoder = container_of(entry, struct audio_decoder, entry);
    struct a2dp_decoder *a2dp_dec = container_of(decoder, struct a2dp_decoder, decoder);
    struct a2dp_dec_hdl *dec = container_of(a2dp_dec, struct a2dp_dec_hdl, dec);
    if(dec->remain==0){
        ops_lr_bt(NULL, in);
    }

    decoder->process_len =0;
    audio_stream_run(&decoder->entry, in);

    int wlen = decoder->process_len;
    if (in->data_len != wlen) {
        dec->remain = 1;
    }
}
```

```
} else {  
    dec->remain = 0;  
}  
return wlen;  
}
```

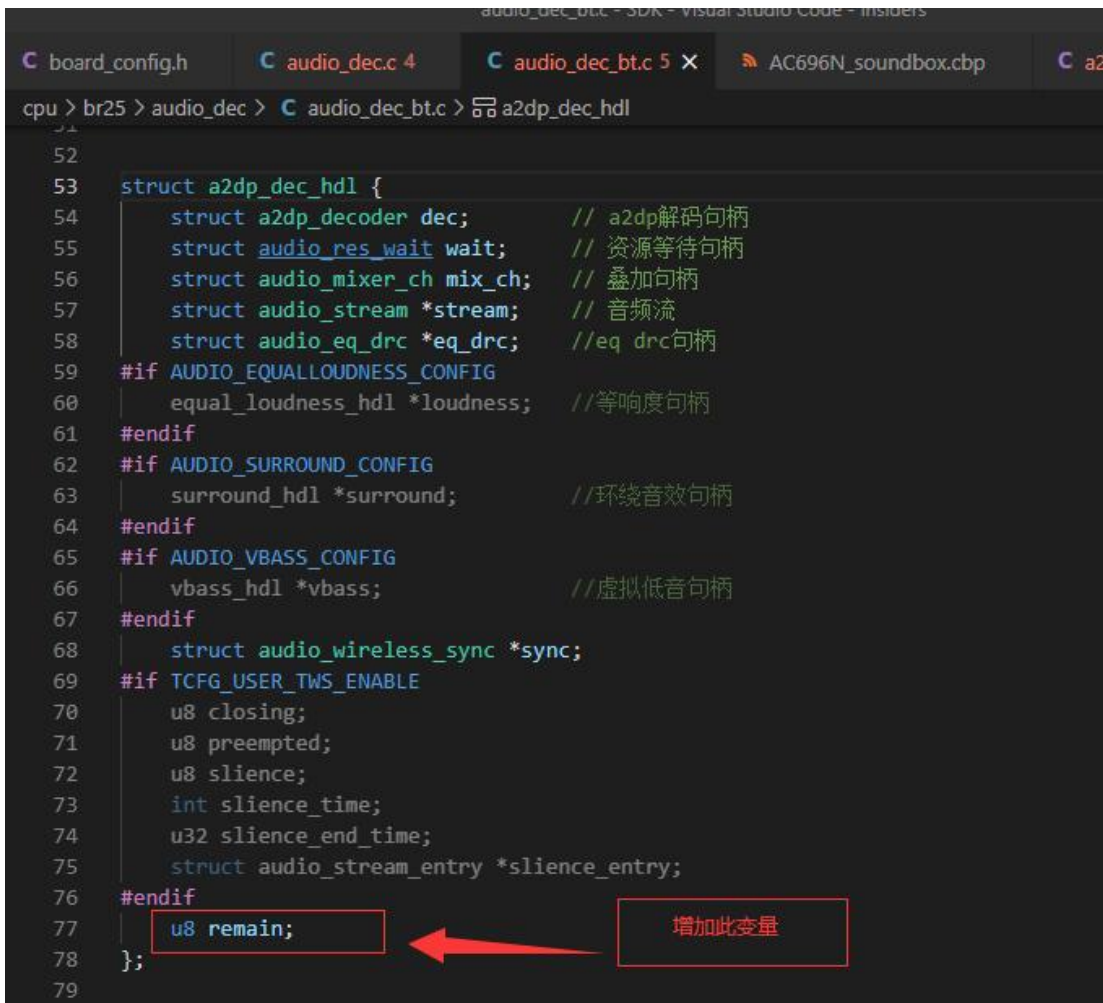


```
cpu > br25 > audio_dec > C audio_dec_bt.c > sbc_dec_data_handler(audio_stream_entry *, audio_data_frame *, audio_data_frame *)  
309 }  
310  
311 int ops_lr_bt(void *priv, struct audio_data_frame *in)  
312 {  
313     s16 *f_lr;  
314     s16 tmp_l,tmp_r;  
315     u16 rlen;  
316     u32 len=0;  
317     // putchar('h');  
318     {  
319         f_lr = in->data;///  
320         len = in->data_len;///  
321  
322         rlen = len>>2; ///  
323         for(u16 i=0; i<rlen; i++) ///  
324         {  
325             tmp_l = f_lr[i*2];  
326             tmp_r = f_lr[i*2+1];  
327             f_lr[i*2] = tmp_r;  
328             f_lr[i*2+1] = tmp_l;  
329         }  
330     }  
331     return len;  
332 }  
333  
334 static int sbc_dec_data_handler(struct audio_stream_entry *entry,  
335                               struct audio_data_frame *in,  
336                               struct audio_data_frame *out)  
337 {  
338     struct audio_decoder *decoder = container_of(entry, struct audio_decoder, entry);  
339     struct a2dp_decoder *a2dp_dec = container_of(decoder, struct a2dp_decoder, decoder);  
340     struct a2dp_dec_hdl *dec = container_of(a2dp_dec, struct a2dp_dec_hdl, dec);  
341     if(dec->remain==0){  
342         ops_lr_bt(NULL, in);  
343     }  
344  
345     decoder->process_len =0;  
346     audio_stream_run(&decoder->entry, in);  
347  
348     int wlen = decoder->process_len;  
349     if (in->data_len != wlen) {  
350         dec->remain = 1;  
351     } else {  
352         dec->remain = 0;  
353     }  
354     return wlen;  
355 }
```

dec->dec.decoder.entry.data_handler = sbc_dec_data_handler;

```
audio_dec_bt.c - SDK - Visual Studio Code - Insiders
board_config.h audio_dec.c 4 audio_dec_bt.c 5 x AC696N_soundbox.cbp a2dp_decoder.h 1
cpu > br25 > audio_dec > C audio_dec_bt.c > a2dp_dec_start(void)
401     return -EINVAL;
402 }
403
404     log_i("a2dp_dec_start: in\n");
405
406     err = a2dp_decoder_open(&dec->dec, &decode_task);
407     if (err) {
408         goto __err1;
409     }
410     audio_decoder_set_event_handler(&dec->dec.decoder, a2dp_dec_event_handler, 0);
411
412     err = audio_decoder_get_fmt(&dec->dec.decoder, &fmt);
413     if (err) {
414         goto __err2;
415     }
416
417     ch_num = audio_output_channel_num();
418     ch_type = audio_output_channel_type();
419     a2dp_decoder_set_output_channel(&dec->dec, ch_num, ch_type);
420
421     if (AUDIO_DEC_BT_MIXER_EN) {
422         audio_mode_main_dec_open(AUDIO_MODE_MAIN_STATE_DEC_A2DP);
423         audio_mixer_ch_open_head(&dec->mix_ch, &mixer); // 挂载到mixer最前面
424         audio_mixer_ch_set_src(&dec->mix_ch, 1, 0);
425         audio_mixer_ch_set_no_wait(&dec->mix_ch, 1, 20); // 超时自动丢数
426         audio_mixer_ch_sample_sync_enable(&dec->mix_ch, 1);
427         audio_mixer_ch_set_sample_rate(&dec->mix_ch, fmt->sample_rate);
428     }
429
430     dec->eq_drc = a2dp_eq_drc_open(fmt->sample_rate, ch_num);
431
432     dec->dec.decoder.entry.data_handler = sbc_dec_data_handler;
433
434     #if AUDIO_SURROUND_CONFIG
435         dec->surround = surround_open_demo(ch_num);
436     #endif
437     #if AUDIO_VBASS_CONFIG
438         dec->vbass = vbass_open_demo(fmt->sample_rate, ch_num);
439     #endif
```

在a2dp_dec_start函数



```
cpu > br25 > audio_dec > C audio_dec_bt.c > a2dp_dec_hdl
52
53 struct a2dp_dec_hdl {
54     struct a2dp_decoder dec;           // a2dp解码句柄
55     struct audio_res_wait wait;       // 资源等待句柄
56     struct audio_mixer_ch mix_ch;     // 叠加句柄
57     struct audio_stream *stream;     // 音频流
58     struct audio_eq_drc *eq_drc;     //eq drc句柄
59 #if AUDIO_EQUALLOUDNESS_CONFIG
60     equal_loudness_hdl *loudness;    //等响度句柄
61 #endif
62 #if AUDIO_SURROUND_CONFIG
63     surround_hdl *surround;         //环绕音效句柄
64 #endif
65 #if AUDIO_VBASS_CONFIG
66     vbass_hdl *vbass;              //虚拟低音句柄
67 #endif
68     struct audio_wireless_sync *sync;
69 #if TCFG_USER_TWS_ENABLE
70     u8 closing;
71     u8 preempted;
72     u8 slience;
73     int slience_time;
74     u32 slience_end_time;
75     struct audio_stream_entry *slience_entry;
76 #endif
77     u8 remain;
78 };
79
```

```
struct a2dp_dec_hdl {
    struct a2dp_decoder dec;           // a2dp 解码句柄
    struct audio_res_wait wait;       // 资源等待句柄
    struct audio_mixer_ch mix_ch;     // 叠加句柄
    struct audio_stream *stream;     // 音频流
    struct audio_eq_drc *eq_drc;     //eq drc 句柄
#if AUDIO_EQUALLOUDNESS_CONFIG
    equal_loudness_hdl *loudness;    //等响度句柄
#endif
#if AUDIO_SURROUND_CONFIG
    surround_hdl *surround;         //环绕音效句柄
#endif
#if AUDIO_VBASS_CONFIG
    vbass_hdl *vbass;              //虚拟低音句柄
#endif
    struct audio_wireless_sync *sync;
#if TCFG_USER_TWS_ENABLE
    u8 closing;

```

```
u8 preempted;
u8 slience;
int slience_time;
u32 slience_end_time;
struct audio_stream_entry *slience_entry;
#endif
u8 remain;
};
```

143.AC696Nmusic 模式下文件解码成功后保存断点时没有清除“时间”信息 20210412 SQ

背景:

- a) 切出 music 模式保存“时间”信息
- b) 切换歌曲保存“歌曲”信息

现象:

- a)A 文件播放到 N 秒拔出卡切换到其他模式
- b)插卡进 music 之后上下曲进行切歌 此时播放到 B 文件的第 X 秒，样机断电
- c)样机重新上电，插卡进 music 播歌，此时会从 B 文件的第 N 秒播放

正常情况 会从 B 文件的第 0 秒播放

修改方法:

在 music_player.c 中的 music_player_get_playing_breakpoint 中添加红色部分代码

```
bool music_player_get_playing_breakpoint(struct __breakpoint *bp, u8 flag)
{
    if (__this == NULL || bp == NULL) {
        return false;
    }

    int data_len = bp->dbp.data_len;
    memset(bp, 0, sizeof(struct __breakpoint) + data_len);
    bp->dbp.data_len = data_len;
    .....
    .....
    .....
}
```

144. ac696n_soundbox_sdk_v1.2.0 只能获取到 DAC_L 能量的问题

20210413 YP

ac696n_soundbox_sdk_v1.2.0 在使用获取能量的时候只能获取到 DAC_L 的能量值，不能获取到 DAC_R

的能量值，系 SDK 存在的 bug，SDK 默认只获取了 DAC_L 的能量值，可以做如下修改：

```
134 /*-----*/
135 int audio_dac_energy_get(void)
136 {
137 #if AUDIO_OUTPUT_AUTOMUTE
138 int audio_energy_detect_energy_get(void *_hdl, u8 ch);
139 if (mix_out_automute_hdl) {
140 | return audio_energy_detect_energy_get(mix_out_automute_hdl, BIT(0));
141 | }
142
143 return (-1);
144 #else
145 return 0;
146 #endif
147 }^M
148 ^M
149
150 int audio_dac_L_energy_get(void)
151 {
152 #if AUDIO_OUTPUT_AUTOMUTE
153 int audio_energy_detect_energy_get(void *_hdl, u8 ch);
154 if (mix_out_automute_hdl) {
155 | return audio_energy_detect_energy_get(mix_out_automute_hdl, BIT(0));
156 | }
157
158 return (-1);
159 #else
160 return 0;
161 #endif
162 }^M
163 ^M
164 int audio_dac_R_energy_get(void)
165 {
166 #if AUDIO_OUTPUT_AUTOMUTE
167 int audio_energy_detect_energy_get(void *_hdl, u8 ch);
168 if (mix_out_automute_hdl) {
169 | return audio_energy_detect_energy_get(mix_out_automute_hdl, BIT(1));
170 | }
171
172 return (-1);
173 #else
174 return 0;
175 #endif
176 }^M
177 ^M
178 int ave_dac_energy_get(void)^M
179 {^M
180 int r_eny = audio_dac_R_energy_get();^M
181 int l_eny = audio_dac_L_energy_get();^M
182 ^M
183 printf("----r_eny = %d,l_eny = %d---\n",r_eny,l_eny);^M
184 return (r_eny > l_eny ? r_eny:l_eny);^M
185 }
186 ////////////////////////////////////////////////////////////////////
187 /*-----*/
```

调用这个ave_dac_energy_get()函数获取，即可获取到左右声道的能量

145. 696n_sdk_v122 控制 PA1 脚(MIC 脚)电平状态, AUX 输入没有声音 20210419 WGH

使用 gpio_direction_output 的接口控制 PA1 脚电平状态, 底层会调用 audio_adc_mic_ctl(u8 mode)接口。调用这个接口会影响芯片 ADC 功能, 目前仅测试会影响 AUX 的数字功能, 其他的没有经过测试。若想要控制 PA1 脚的电平, 可以使用操作寄存器的方法控制。



146. 使用 MIC 脚做普通 IO 的注意事项

WGH

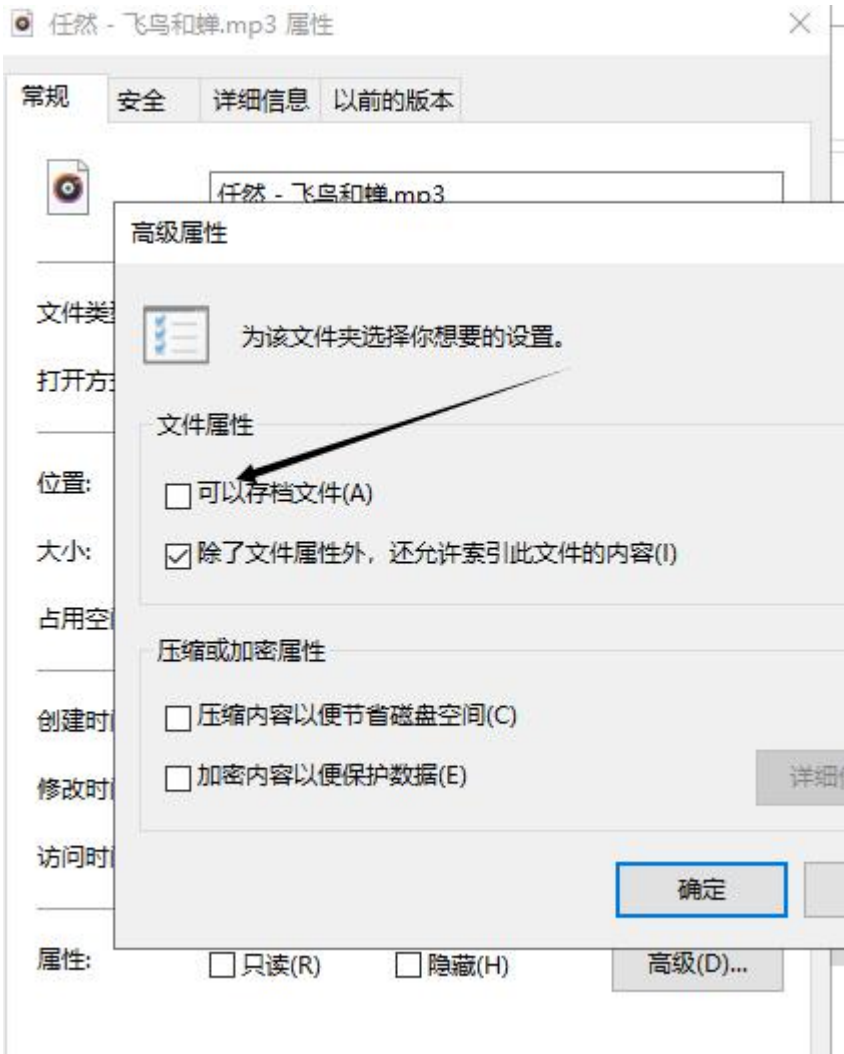
通常使用 MIC 脚做普通 IO 会根据 AC696N MIC 脚当普通 IO 口注意事项.pdf 来设置, 不过这个文档里面会调用 audio_adc_mic_ctl(u8 mode)接口, 会导致出 145 点问题。

利用打印 (JL_SYSTEM->CHIP_ID & 0xF) 可以查看晶圆版本, E 版本一下 (数值<4) 的芯片, 不建议客户使用 MIC 脚做普通 IO 使用, 存在漏电风险。E 版本及以上 (数值=>4) 的芯片, 可以将 PA1 做为普通 IO 使用。

不建议参考这个文档，直接操作寄存器做普通IO功能

AC696N MIC 脚当普通 IO 口注意事项

- 147. AC696X V122 版本的 SDK 音频文件属性不勾选“可以存档文件”的不能播放的问题
- 148. 20210430 LJW



请替换库

链接：<https://pan.baidu.com/s/1wzX7sVVFiy1KgFqzA4T5Xg>

提取码：8888

149. AC696X V122 版本的 SDK 打开设备提示音，插 SD 卡再插 U 盘有啧啧声的问题

150. 20210430 LJW

请替换文件：

链接：<https://pan.baidu.com/s/1XCh9OISz6iIQasheuSwRAQ>

提取码：8888

```
#define TCFG_PREVENT_TASK_FILL 0 // 防止task占满cpu
```

```
#define TCFG_AUDIO_DECODER_OCCUPY_TRACE 1
```

151. AC696X V122 版本的 SDK 插 U 盘播歌速度变快的问题处理

20210430

LJW

替换文件

版权所有，侵权必究

链接: https://pan.baidu.com/s/1U1sA9qunals1888QqMS2_A

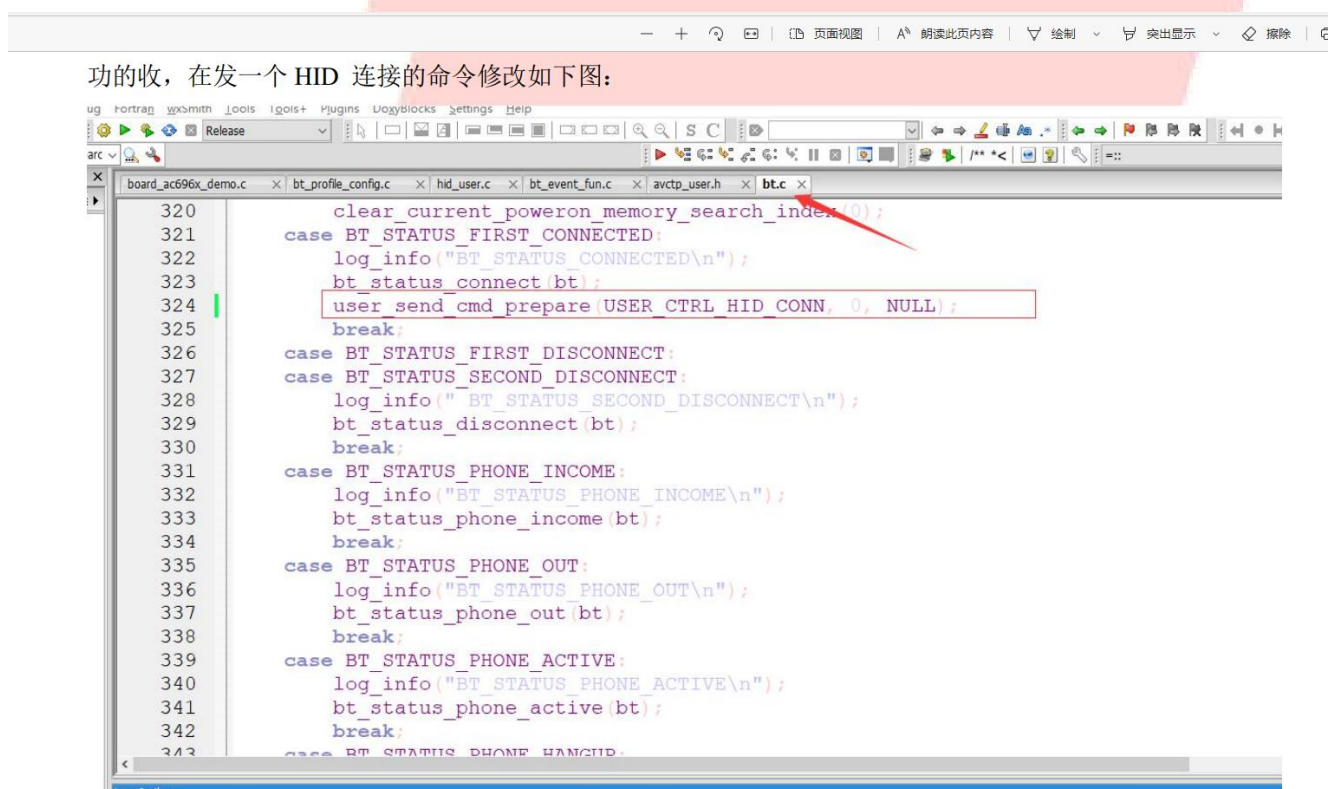
提取码: 8888

152. AC696X 音箱工程 V123 版本安卓手机不能拍照或者手机回连后不能拍照 20210430 XNW

问题的原因: 安卓版本的 HID 协议没连上

修改方法: 将 `hid_conn_depend_on_dev_company = 1;` 改从 `hid_conn_depend_on_dev_company = 0;`

按照这个改了后, 如果出现关安卓手机蓝牙, 再开安卓手机蓝牙, 手机回连后, 拍照无作用, 在蓝牙连接成功的收, 在发一个 HID 连接的命令修改如下图:



153. AC696X 音箱工程 V0.2.6、1.1.1 有概率出现软关机功耗大 20210506 LHY

解决方法: 替换 cpu.a 库

链接: https://pan.baidu.com/s/1B0acGhrt_R0a_k4fPPcLHQ

提取码: 8888

154. AC696X 音箱工程 V1.2.3 部分手机不能拍照--红米 K30 20210513 XNW

解决方法: 替换一个库文件

连接: <https://pan.baidu.com/s/1NbimZQqdKnpPWozxvUJs-A>

提起吗: 8888

AC696X 音箱工程 V1.2.3U 盘和 TF 卡复用，一直切设备，某些 U 盘会掉线的处理方法 20210513 LJW

```
u32 usb_host_remount(const usb_dev id, u32 retry, u32 delay, u32 ot, u8 notify)
{
#if USB_MAX_HW_NUM > 1
    const usb_dev usb_id = id;
#else
    const usb_dev usb_id = 0;
#endif
    u32 ret;
    struct sys_event event;

    ret = _usb_host_unmount(usb_id);
    if (ret) {
        goto __exit_fail;
    }

    struct usb_host_device *host_dev = &host_devices[usb_id];
    os_sem_set(host_dev->sem,0);

    ret = _usb_host_mount(usb_id, retry, delay, ot);
    if (ret) {
        goto __exit_fail;
    }

    if (notify) {
        struct usb_host_device *host_dev = &host_devices[usb_id];
        usb_event_notify(host_dev, 1);
    }
}
```

添加这两句

155. 在主控的 flash 中存放 N 个文件，读取这些文件需要的时间有很大的差异 SQ 20210514

原因：与文件系统有关。

处理方法：把文件放在不同的文件夹中

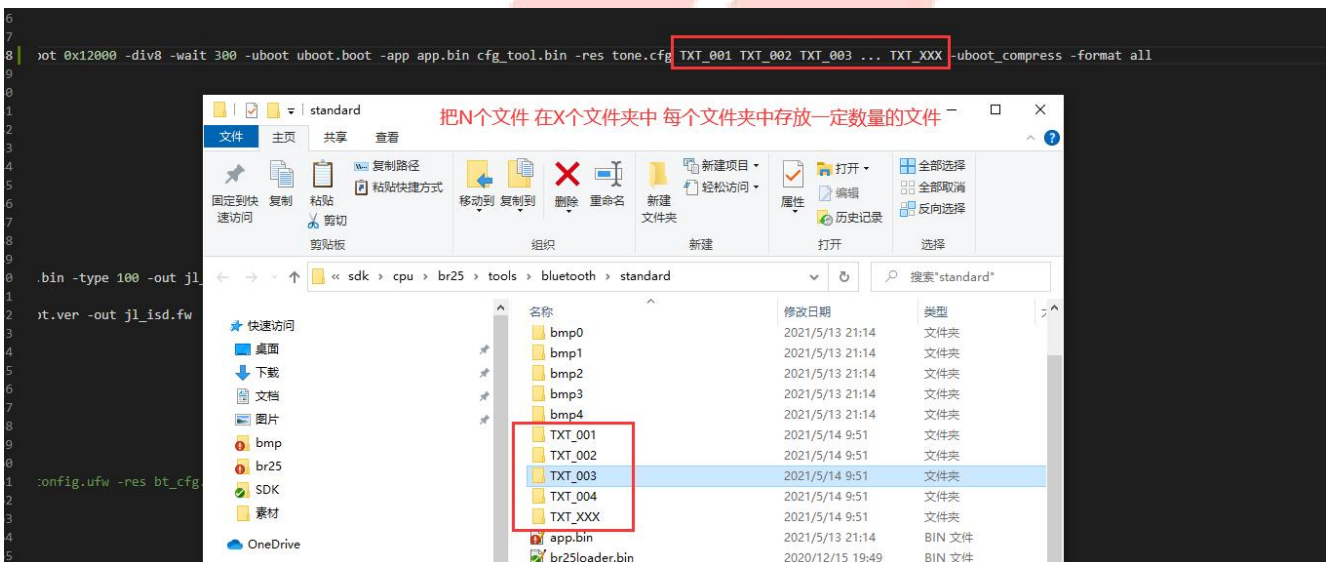
问题：

```
Flash -dev br25 -boot 0x12000 -div8 -wait 300 -uboot uboot.boot -app app.bin cfg_tool.bin -res tone.cfg file001.txt file002.txt file003.txt ... filexxx.txt
```

```
FILE *fp;
// user_rgb_p_timer(0);
u8 path[256]={0};
int number = 0;
u32 timer_cnt = timer_get_ms();
for(;number<500;number++){
    sprintf(path,"mnt/sdfile/res/file%03d.txt",number++);
    timer_cnt = timer_get_ms();
    fp = fopen(path, "r");
    r_printf("open timer=%d",timer_get_ms()-timer_cnt);
    timer_cnt = timer_get_ms();
    fclose(fp);
}
```

open的时间随着number的变大而变大，基本上number变大100 open的时间加大10ms

程序中修改：



```
// r_printf("path = %s",info->path);

FILE *fp;
// user_rgb_p_timer(0);
u8 path[256]={0};
int number = 0;
u32 timer_cnt = timer_get_ms();
for(;number<500;number++){
    sprintf(path,"mnt/sdfile/res/TXT_%03d/%04d.txt",number/100,number++);
    // sprintf(path,"mnt/sdfile/res/file%03d.txt",number++);
    timer_cnt = timer_get_ms();
    fp = fopen(path, "r");
    r_printf("open timer=%d",timer_get_ms()-timer_cnt);
    timer_cnt = timer_get_ms();
    fclose(fp);
}
```

文件夹名称 文件名称 这里以每个文件夹中存放100个文件为例

156. AC696x 1.2.2sdk 串口通信接收数据时产生一次中断之后，接下来接收数据异常 20210514 SQ

原因：产生 rx 中断之后没有清除对应的 rx pnd 导致

程序修改:

```
31 static void UT0_isr_fun(void)
32 {
33     u32 rx_len = 0;
34     if ((JL_UART0->CON0 & BIT(2)) && (JL_UART0->CON0 & BIT(15))) {
35         JL_UART0->CON0 |= BIT(13);
36         UT_OSSemPost(&uart0.sem_tx);
37         if (uart0.isr_cbfun) {
38             uart0.isr_cbfun(&uart0, UT_TX);
39         }
40     }
41     if ((JL_UART0->CON0 & BIT(3)) && (JL_UART0->CON0 & BIT(14))) {
42         JL_UART0->CON0 |= BIT(7); //DMA模式
43         JL_UART0->CON0 |= BIT(10); //清OTCNT PND
44         JL_UART0->CON0 |= BIT(12); //清RX PND(这里的顺序不能改变, 这里要清一次)
45         rx_len = JL_UART0->HRXCNT; //读当前串口接收数据的个数
46         uart0.kfifo.buf_in += uart0.frame_length; //每满frame_length字节则产生一次中断
47         UT_OSSemPost(&uart0.sem_rx);
48         if (uart0.isr_cbfun) {
49             uart0.isr_cbfun(&uart0, UT_RX);
50         }
51     }
}
```

```
30 static void UT1_isr_fun(void)
31 {
32     u32 rx_len = 0;
33     if ((JL_UART1->CON0 & BIT(2)) && (JL_UART1->CON0 & BIT(15))) {
34         JL_UART1->CON0 |= BIT(13);
35         UT_OSSemPost(&uart1.sem_tx);
36         if (uart1.isr_cbfun) {
37             uart1.isr_cbfun(&uart1, UT_TX);
38         }
39     }
40     if ((JL_UART1->CON0 & BIT(3)) && (JL_UART1->CON0 & BIT(14))) {
41         JL_UART1->CON0 |= BIT(7); //DMA模式
42         JL_UART1->CON0 |= BIT(10); //清OTCNT PND
43         JL_UART1->CON0 |= BIT(12); //清RX PND(这里的顺序不能改变, 这里要清一次)
44         rx_len = JL_UART1->HRXCNT; //读当前串口接收数据的个数
45         uart1.kfifo.buf_in += uart1.frame_length; //每满32字节则产生一次中断
46         UT_OSSemPost(&uart1.sem_rx);
47         if (uart1.isr_cbfun) {
48             uart1.isr_cbfun(&uart1, UT_RX);
49         }
50     }
}
```

```
0 static void UT2_isr_fun(void)
1 {
2     u32 rx_len = 0;
3     if ((JL_UART2->CON0 & BIT(2)) && (JL_UART2->CON0 & BIT(15))) {
4         JL_UART2->CON0 |= BIT(13);
5         UT_OSSemPost(&uart2.sem_tx);
6         if (uart2.isr_cbfun) {
7             uart2.isr_cbfun(&uart2, UT_TX);
8         }
9     }
10    if ((JL_UART2->CON0 & BIT(3)) && (JL_UART2->CON0 & BIT(14))) {
11        JL_UART2->CON0 |= BIT(7); //DMA模式
12        JL_UART2->CON0 |= BIT(10); //清OTCNT PND
13        JL_UART2->CON0 |= BIT(12); //清RX PND(这里的顺序不能改变, 这里要清一次)
14        rx_len = JL_UART2->HRXCNT; //读当前串口接收数据的个数
15        uart2.kfifo.buf_in += uart2.frame_length; //每满32字节则产生一次中断
16        UT_OSSemPost(&uart2.sem_rx);
17        if (uart2.isr_cbfun) {
18            uart2.isr_cbfun(&uart2, UT_RX);
19        }
20    }
}
```

```
151
152 void uart_dev_test_main()
153 {
154     const uart_bus_t *uart_bus;
155     struct uart_platform_data_t u_arg = {0};
156     u_arg.tx_pin = IO_PORTA_01;
157     u_arg.rx_pin = IO_PORTA_02;
158     u_arg.rx_cbuf = uart_cbuf;
159     u_arg.rx_cbuf_size = 512;
160     u_arg.frame_length = 32;
161     u_arg.rx_timeout = 100;
162     u_arg.isr_cbfun = uart_isr_hook;
163     u_arg.baud = 9600;
164     u_arg.is_9bit = 0;
165
166     uart_bus = uart_dev_open(&u_arg);
167     if (uart_bus != NULL) {
168         printf("uart_dev_open() success\n");
169     }
170     #if (UART_DEV_USAGE_TEST_SEL == 2)
171     os_task_create(uart_u_task, (void *)uart_bus, 31, 512, 0, "uart_u_task");
172     #endif
173 }
```

产生异常rx中断的长度
这个参数直接给 循环buf的大小，意思是不让它产生rx中断

157. ac696n_soundbox_sdk_v1.3.0 开启串口 test 功能会复位问题 20210611 YP

ac696n_soundbox_sdk_v1.3.0 开启串口 test 功能会复位问题系串口有注册事件处理事件，公用别人的事件使用，在开机的时候有冲突，会导致一直复位问题。可以如下修改：

```
7 //设备事件响应 demo
8 static void uart_event_handler(struct sys_event *e)
9 {
10     const uart_bus_t *uart_bus;
11     u32 uart_rxcnt = 0;
12     struct device_event *usr_dev = &e->u.dev;
13
14     printf("----%s,usr_dev->event = %d    value = %d\n",__func__,usr_dev->event,usr_dev->value);
15     switch(usr_dev->event){
16
17     case DEVICE_EVENT_CHANGE:
18         r_printf("-----%s,DEVICE_EVENT_CHANGE\n",__func__);
19         if (!strcmp(e->arg, "uart_rx_overflow"))
20
21             uart_bus = (const uart_bus_t *)e->u.dev.value;
22             uart_rxcnt = uart_bus->read(uart_rxbuf, sizeof(uart_rxbuf), 0);
23             if (uart_rxcnt) {
24                 printf("get_buffer:\n");
25                 for (int i = 0; i < uart_rxcnt; i++) {
26                     my_put_u8hex(uart_rxbuf[i]);
27                     if (i % 16 == 15) {
28                         putchar('\n');
29                     }
30                 }
31                 if (uart_rxcnt % 16) {
32                     putchar('\n');
33                 }
34                 uart_bus->write(uart_rxbuf, uart_rxcnt);
35             }
36         if (!strcmp(e->arg, "uart_rx_outtime")) {
37             uart_bus = (const uart_bus_t *)e->u.dev.value;
38             uart_rxcnt = uart_bus->read(uart_rxbuf, sizeof(uart_rxbuf), 0);
39             if (uart_rxcnt) {
40                 printf("get_buffer:\n");
41                 for (int i = 0; i < uart_rxcnt; i++) {
42                     my_put_u8hex(uart_rxbuf[i]);
43                     if (i % 16 == 15) {
44                         putchar('\n');
45                     }
46                 }
47                 if (uart_rxcnt % 16) {
48                     putchar('\n');
49                 }
50                 uart_bus->write(uart_rxbuf, uart_rxcnt);
51             }
52         }
53         break;
54
55     default:
56         g_printf("----other event ----\n");
57         break;
58     }
59 }
```

```
//设备事件响应 demo
static void uart_event_handler(struct sys_event *e)
{
    const uart_bus_t *uart_bus;
    u32 uart_rxcnt = 0;
    struct device_event *usr_dev = &e->u.dev;
```

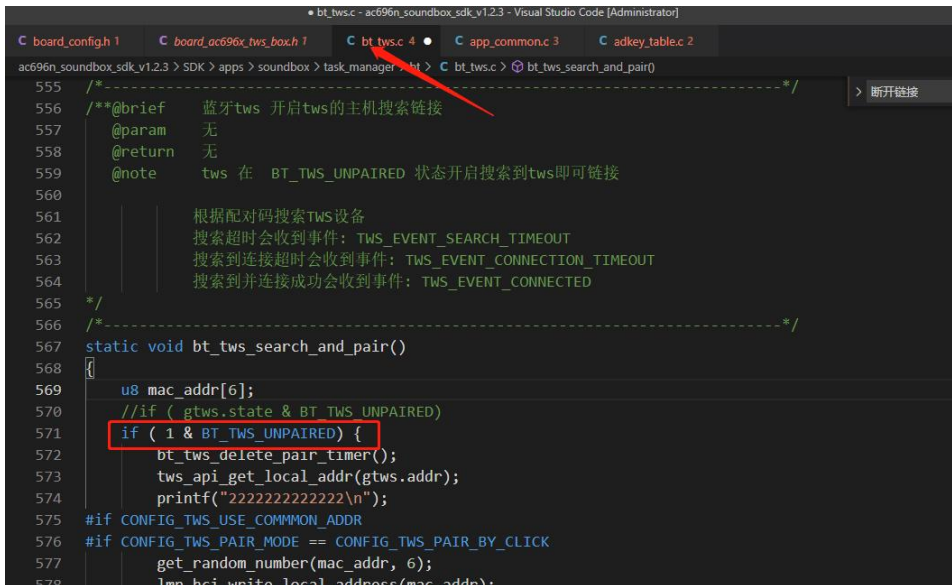
```
printf("---%s,usr_dev->event = %d      value = %d\n",__func__,usr_dev->event,usr_d
ev->value);
switch(usr_dev->event){

case DEVICE_EVENT_CHANGE:
    r_printf("-----%s,DEVICE_EVENT_CHANGE\n",__func__);
    if (!strcmp(e->arg, "uart_rx_overflow")) {

        uart_bus = (const uart_bus_t *)e->u.dev.value;
        uart_rxcnt = uart_bus->read(uart_rxbuf, sizeof(uart_rxbuf), 0);
        if (uart_rxcnt) {
            printf("get_buffer:\n");
            for (int i = 0; i < uart_rxcnt; i++) {
                my_put_u8hex(uart_rxbuf[i]);
                if (i % 16 == 15) {
                    putchar('\n');
                }
            }
            if (uart_rxcnt % 16) {
                putchar('\n');
            }
            uart_bus->write(uart_rxbuf, uart_rxcnt);
        }
    }
    if (!strcmp(e->arg, "uart_rx_outtime")) {
        uart_bus = (const uart_bus_t *)e->u.dev.value;
        uart_rxcnt = uart_bus->read(uart_rxbuf, sizeof(uart_rxbuf), 0);
        if (uart_rxcnt) {
            printf("get_buffer:\n");
            for (int i = 0; i < uart_rxcnt; i++) {
                my_put_u8hex(uart_rxbuf[i]);
                if (i % 16 == 15) {
                    putchar('\n');
                }
            }
            if (uart_rxcnt % 16) {
                putchar('\n');
            }
            uart_bus->write(uart_rxbuf, uart_rxcnt);
        }
    }
    break;
default:
    g_printf("---other event ----\n");
    break;
}
}
```

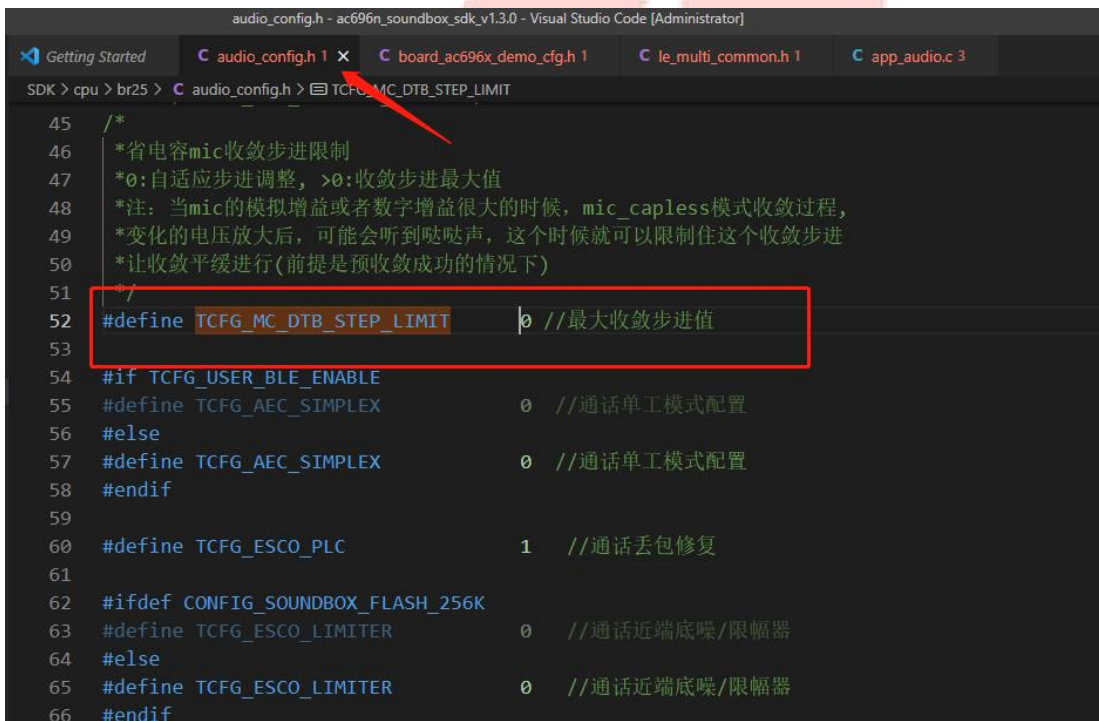
158. ac696n_soundbox_sdk_v1.2.3 对箱配对后无法连接其他对箱解决方法
LCP

20210623



```
555 /*-----*/
556 /**@brief  蓝牙tws 开启tws的主机搜索链接
557     @param  无
558     @return 无
559     @note   tws 在 BT_TWS_UNPAIRED 状态开启搜索到tws即可链接
560
561     根据配对码搜索TWS设备
562     搜索超时会收到事件: TWS_EVENT_SEARCH_TIMEOUT
563     搜索到连接超时会收到事件: TWS_EVENT_CONNECTION_TIMEOUT
564     搜索到并连接成功会收到事件: TWS_EVENT_CONNECTED
565 */
566 /*-----*/
567 static void bt_tws_search_and_pair()
568 {
569     u8 mac_addr[6];
570     //if (gtws.state & BT_TWS_UNPAIRED)
571     if ( 1 & BT_TWS_UNPAIRED) {
572         bt_tws_delete_pair_timer();
573         tws_api_get_local_addr(gtws.addr);
574         printf("22222222222222\n");
575 #if CONFIG_TWS_USE_COMMON_ADDR
576 #if CONFIG_TWS_PAIR_MODE == CONFIG_TWS_PAIR_BY_CLICK
577         get_random_number(mac_addr, 6);
578         lmp_hci_write_local_address(mac_addr);
```

157.ac696n_soundbox_sdk_v1.3.0 linein 走 ADC,并复用 fm 引脚,使用省电容接法时,进 linein 出来后,麦声音变得很小解决方法
20210623 LCP



```
45 /*
46 *省电容mic收敛步进限制
47 *0:自适应步进调整, >0:收敛步进最大值
48 *注: 当mic的模拟增益或者数字增益很大的时候, mic_capless模式收敛过程,
49 *变化的电压放大后, 可能会听到哒哒声, 这个时候就可以限制住这个收敛步进
50 *让收敛平缓进行(前提是预收敛成功的情况下)
51 */
52 #define TCFG_MC_DTB_STEP_LIMIT 0 //最大收敛步进值
53
54 #if TCFG_USER_BLE_ENABLE
55 #define TCFG_AEC_SIMPLEX 0 //通话单工模式配置
56 #else
57 #define TCFG_AEC_SIMPLEX 0 //通话单工模式配置
58 #endif
59
60 #define TCFG_ESCO_PLC 1 //通话丢包修复
61
62 #ifdef CONFIG_SOUNDBOX_FLASH_256K
63 #define TCFG_ESCO_LIMITER 0 //通话近端底噪/限幅器
64 #else
65 #define TCFG_ESCO_LIMITER 0 //通话近端底噪/限幅器
66 #endif
```

158. ac696n_soundbox_sdk_v1.1.1 TWS 非公共 mac 地址，去除配对限制 20210726 SQ

现象：tws 配对最后 如果想哪其中一个与其他机子配对 出现不能配对，需要删除 tws 配对信息之后才能配对

程序修改

```
#define USER_TWS_DISCONNECTED_REMOVE 1//解除 tws 配对限制 公版 sdk tws 没有解除配对  
就不能重新配对
```

```
}  
u8 bt_tws_remove_tws_pair()  
{  
    int state = get_bt_connect_status();  
  
    if ((get_call_status() == BT_CALL_ACTIVE) ||  
        (get_call_status() == BT_CALL_OUTGOING) ||  
        (get_call_status() == BT_CALL_ALERT) ||  
        (get_call_status() == BT_CALL_INCOMING)) {  
        return 0;//通话过程不允许  
    }  
  
    #if (CONFIG_TWS_USE_COMMON_ADDR && !USER_TWS_ADD_DELL_TWS_INFO)  
        if (tws_api_get_tws_state() & TWS_STA_PHONE_CONNECTED) {  
            return 0;  
        }  
    #endif  
  
    if (USER_TWS_DISCONNECTED_REMOVE || (gtws.state & BT_TWS_SIBLING_CONNECTED)) {  
        tws_remove_tws_pairs();  
        return 1;  
    }  
    return 0;  
}
```

```

94 }
95
96 #define USER_TWS_DISCONNECTED_REMOVE 1//解除tws配对限制 公版sdk tws 没有解除配对就不能重新配对
97 static void bt_tws_search_and_pair()
98 {
99     u8 mac_addr[6];
100
101     if ((USER_TWS_DISCONNECTED_REMOVE && (tws_api_get_tws_state() & TWS_STA_SIBLING_DISCONNECTED)) ||
102         (gtws.state & BT_TWS_UNPAIRED)) {
103         bt_tws_delete_pair_timer();
104         #if (USER_TWS_DISCONNECTED_REMOVE)
105         tws_cancle_all_noconn();
106         #endif
107         tws_api_get_local_addr(gtws.addr);
108     #if CONFIG_TWS_USE_COMMON_ADDR
109     #if CONFIG_TWS_PAIR_MODE == CONFIG_TWS_PAIR_BY_CLICK
110         get_random_number(mac_addr, 6);
111         lmp_hci_write_local_address(mac_addr);
112     #endif
113     #endif
114         tws_api_search_sibling_by_code(bt_get_tws_device_indicate(NULL), 15000);
115     }
116 }
117
118 */
119 /*-----*/
120 static void tws_event_esco_add_connect(struct bt_event *evt)
121 {
122     int role = evt->args[0];
123     if (role == TWS_ROLE_MASTER) {
124         bt_tws_sync_volume();
125     }
126 }
127
128 #if (CONFIG_TWS_USE_COMMON_ADDR == 0)
129 int tws_host_get_local_role()
130 {
131     int role = 0;
132
133     if (syscfg_read(VM_TWS_ROLE, &role, 1) == 1) {
134         if (role == TWS_ROLE_SLAVE) {
135             #if (defined(USER_TWS_DISCONNECTED_REMOVE) && USER_TWS_DISCONNECTED_REMOVE)
136             if (tws_api_get_tws_state() & TWS_STA_PHONE_CONNECTED)
137             {
138                 return TWS_ROLE_MASTER;
139             }else{
140                 return TWS_ROLE_SLAVE;
141             }
142             #endif
143
144             printf("\n\nrole is slave\n\n");
145
146             return TWS_ROLE_SLAVE;
147         }
148     }
149
150     r_printf("\n\n-----cccccc-----\n\n");
151     return TWS_ROLE_MASTER;
152 }
153 #endif
154
155
156

```

159. 公版 V1.3.0sdk2M 板级配置关掉 eq 之后打入电话长时间响铃死机 20210702 SQ

复现配置:

```
#define CONFIG_BOARD_AC696X_BTBOX
```

版权所有，侵权必究

#define TCFG_EQ_ENABLE 0 //支持 EQ 功能,EQ 总使能

修改方法：修改函数的实现、修改时钟

1) 修改函数的实现

//修改文件 audio_dec_bt.c

//修改前

```
static void esco_dec_out_stream_resume(void *p){
    struct esco_dec_hdl *dec = (struct esco_dec_hdl *)p;
    audio_decoder_resume(&dec->dec.decoder);
}
void esco_rx_notice_to_decode(void){
    if (bt_esco_dec && bt_esco_dec->dec.start) {
        /* audio_decoder_resume(&bt_esco_dec->dec.decoder); */
        if (bt_esco_dec->dec.wait_resume) {
            bt_esco_dec->dec.wait_resume = 0;
            audio_decoder_resume(&bt_esco_dec->dec.decoder);
        }
    }
}
```

//修改后

```
static void esco_dec_out_stream_resume(void *p){
    struct esco_dec_hdl *dec = (struct esco_dec_hdl *)p;
}
void esco_rx_notice_to_decode(void){
    if (bt_esco_dec && bt_esco_dec->dec.start) {
        audio_decoder_resume(&bt_esco_dec->dec.decoder);
    }
}
```

2) 修改时钟

//修改文件 clock_manager.c

//修改前

```
{ AEC16K_ADV_CLK, (50), "AEC16K_ADV_CLK " },
```

//修改后

```
{ AEC16K_ADV_CLK, (80), "AEC16K_ADV_CLK " },
```

160. 1.2.2 SDK 定时器使用注意点 20210702 SQ

sys_hi_timeout_add、sys_s_hi_timerout_add 区别

sys_hi_timeout_add: usr_timeout_add(a, b, c, 1) 在中断中执行

sys_s_hi_timerout_add: usr_timeout_add(a, b, c, 0)在任务中执行

161. sd 卡 sd_io_suspend、sd_io_resume 接口参数 20210702 SQ
u8 sd_io_suspend(u8 sdx, u8 sd_io);
u8 sd_io_resume(u8 sdx, u8 sd_io);
sdx : 0: sd0 1: sd1 sd_io : 0: CMD 脚 1: CLK 脚 2: DAT 脚

162. 1.2.2SDK 读 U 盘 txt 文件案例 20210702 SQ

```
#define USER_READ_DEV "udisk0"/"sd0" "sd1" 其他设备名参考 dev_reg.h
#define USER_READ_FILE_NAME "1.TXT"
#define USER_READ_FILE_BUF_SIZE (512)

char user_file_path[48] = {0};
const char user_file_name[] = USER_READ_FILE_NAME;

struct __dev *dev = dev_manager_find_spec(USER_READ_DEV, 0);
if(!dev){
    r_printf("dev_manager_find_spec error");
    return;
}
sprintf(user_file_path, "%s%s", dev_manager_get_root_path(dev), user_file_name);

r_printf("read file path=%s",user_file_path);
FILE *fd = fopen(user_file_path, "r");
if (!fd) {
    r_printf("open %s file err!!!\n",user_file_path);
    return ;
}

u8 buf[USER_READ_FILE_BUF_SIZE]={0};
u8 len;
len = fread(fd, buf, sizeof(buf));
r_printf("read file len %d",len);

r_printf("read file data %s",buf);

fclose(fd);
//sys_hi_timeout_add
sys_s_hi_timerout_add(NULL,user_file_read_test,1000);
}
```

163. 1.2.3 SDK 打开 TWS 跳转升级无法维持 IO 电平 20210702 lzk

替换 btctrlr.a 文件，下载地址如下

<https://audiodoc.notion.site/IO-8208aad74df045a0b172f3cf5ce18021>

跳转升级具体修改

1. 打开跳转升级的宏定义 DEV_UPDATE_SUPPORT_JUMP

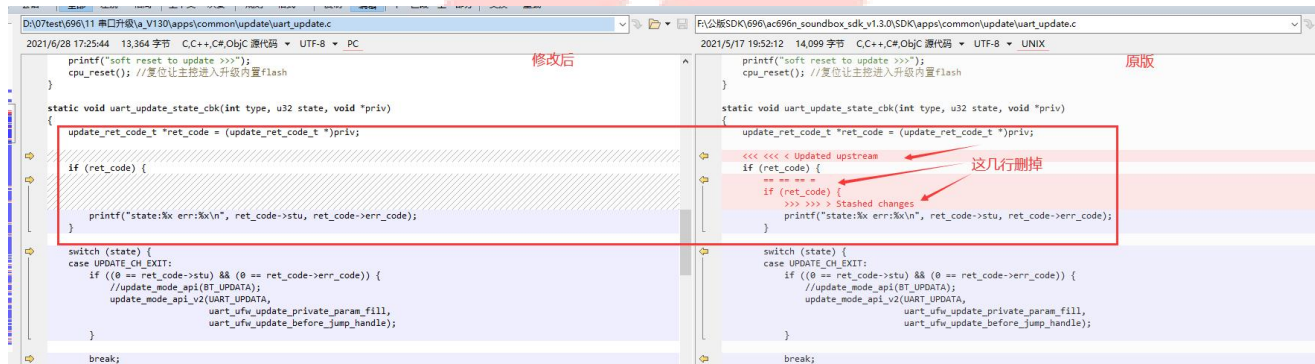
```
#if (defined(CONFIG_CPU_BR23) || defined(CONFIG_CPU_BR25))
//升级IO保持使能
#define DEV_UPDATE_SUPPORT_JUMP //目前只有br23\br25支持
#endif
```

2. isd_config.ini文件中打开跳转升级

```
UPDATE_JUMP=1;
```

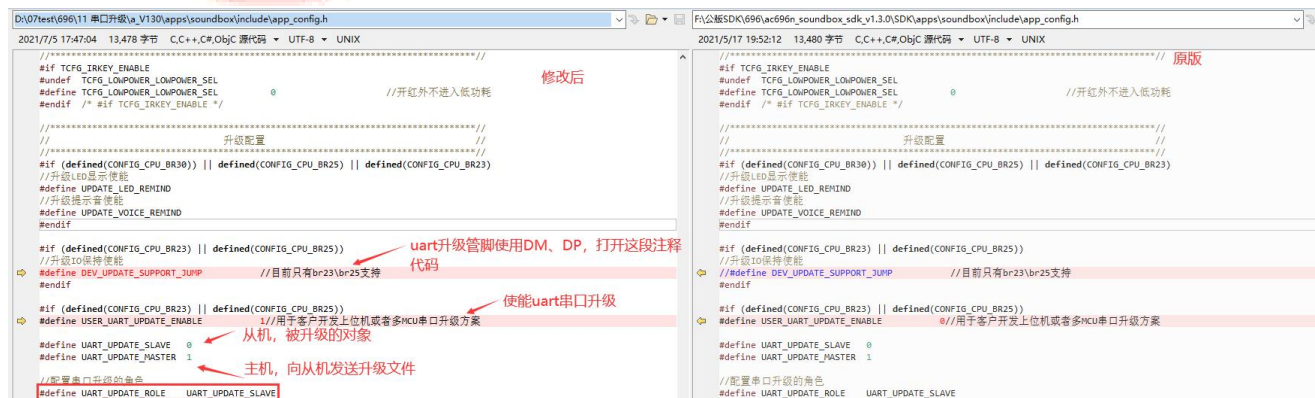
164. 1.3.0 MCU uart 串口升级注意事项 20210705 HJY

uart_update.c 文件中删除多余代码。



The image shows a side-by-side comparison of the `uart_update.c` file. The left window shows the modified version, and the right window shows the original version. Red boxes and arrows highlight the code that was removed in the modified version. The removed code includes the `update_ret_code_t` variable definition and the `if (ret_code)` block that prints error messages. A red arrow points to the text '这几行删掉' (Delete these lines).

配置文件修改:



The image shows a side-by-side comparison of the `app_config.h` file. The left window shows the modified version, and the right window shows the original version. Red boxes and arrows highlight the configuration changes. The changes include:
1. Defining `DEV_UPDATE_SUPPORT_JUMP` as 1.
2. Defining `USER_UART_UPDATE_ENABLE` as 1.
3. Defining `UART_UPDATE_SLAVE` as 0.
4. Defining `UART_UPDATE_MASTER` as 1.
5. Defining `UART_UPDATE_ROLE` as `UART_UPDATE_SLAVE`.
Red arrows point to the text 'uart升级管脚使用DM、DP，打开这段注释代码' (UART upgrade pins use DM, DP, open this comment code), '使能uart串口升级' (Enable UART serial upgrade), '从机，被升级的对象' (Slave, upgrade target), and '主机，向从机发送升级文件' (Master, send upgrade file to slave).

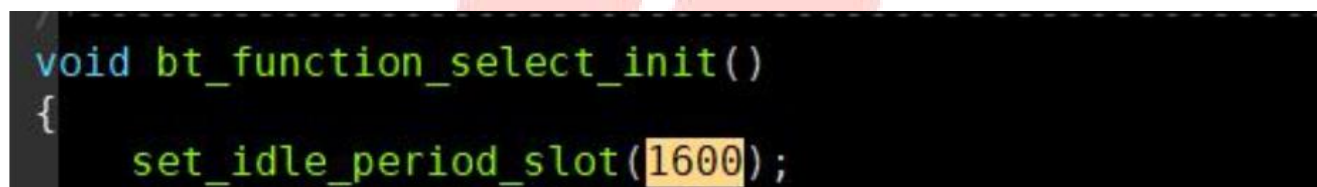
对应的板级.c 文件中可修改升级管脚



```
/* power_wakeup_init(&wk_param); */  
  
#if (!TCFG_IOKEY_ENABLE && !TCFG_ADKEY_ENABLE)  
    charge_check_and_set_pinnr(0);  
#endif  
  
#if USER_UART_UPDATE_ENABLE  
    {  
#include "uart_update.h"  
    uart_update_cfg update_cfg = {  
        .rx = IO_PORTA_02, //IO_PORT_DM, //IO_PORTA_02,  
        .tx = IO_PORTA_03, //IO_PORT_DP, //IO_PORTA_03,  
        .output_channel = CH1_UT1_TX,  
        .input_channel = INPUT_CH0  
    };  
    uart_update_init(&update_cfg);  
    }  
#endif  
}
```

165. AC696 音箱 TWS 配对慢优化方法 20210705 LZK

可以减少蓝牙空闲状态时间。
方法如下：



```
void bt_function_select_init()  
{  
    set_idle_period_slot(1600);  
}
```

SDK 默认值为 1600(时间实际为 1600*0.625ms)，建议修改为 200。该值最小为 100，注意不能使用低功耗适用于音箱对功耗要求不高的产品使用。

166. ac696n_soundbox_sdk_v1.2.1-v1.3.0 蓝牙开混合录音 WAV 格式，录音音频出现卡音问题 20210707 WGH

```
dx_len:0x5
[00:00:31.097]2
[00:00:32.097]3
[00:00:33.098]4
[00:00:34.098]5
[00:00:35.098]6
[00:00:36.098]7
[00:00:37.098]8
[00:00:38.099]9
[00:00:39.098]10
[00:00:40.100]11
[00:00:41.098]12
[00:00:42.099]13
[00:00:43.099]14
[00:00:44.098]15
[00:00:45.100]16
[00:00:46.099]17
[00:00:47.100]18
[00:00:48.101]19
~~~~~
~~~~~
~~~~~
[00:00:49.099]20
[00:00:50.099]21
[00:00:51.099]22
[00:00:52.100]24
[00:00:53.099]25
[00:00:54.100]26
[00:00:55.100]27
[00:00:56.100]28
[00:00:57.100]29
[00:00:58.099]30
~~~~~
~~~~~
~~~~~
[00:00:59.100]31
[00:01:00.100]32
[00:01:01.100]33
[00:01:02.102]34
[00:01:03.100]35
~~~~~
~~~~~
~~~~~
[00:01:04.102]36
[00:01:05.101]37
[00:01:06.101]38
[00:01:07.100]39
```

从打印上看会打印~，是写 PCM cbuf 时候返回 0。


```
// 写pcm数据
int pcm2file_enc_write_pcm(void *priv, s16 *data, int len)
{
```

修改点:

Audio_enc_file.c

```
35
36 struct pcm2file_enc_hdl
37     struct audio_encoder encoder;
38     s16 output_frame[1152 / 2];           //align 4Bytes
39
40     int pcm_frame[64];                   //align 4Bytes
41     u8 file_head_frame[128];
42     u8 file_head_len;
43     /* u8 pcm_buf[PCM_ENC2FILE_PCM_LEN]; */
44     u8 *pcm_buf;
45     cbuffer_t pcm_cbuf;
46
47     int out_file_frame[512 / 4];
48     /* u8 out_file_buf[PCM_ENC2FILE_FILE_LEN]; */
49     u8 *out_file_buf;
50     cbuffer_t out_file_cbuf;
51
52     void *whdl;
53     OS_SEM sem_wfile;
54     OS_MUTEX mutex;
55
56     volatile u32 status : 1;
57     volatile u32 enc_err : 1;
58     volatile u32 encoding : 1;
59     volatile u32 write_head_busy : 1;
60     volatile u32 input_busy : 1;
61     u32 lost;
62
63 #if PCM2FILE_ENC_BUF_COUNT
64     u16 pcm_buf_max;
65     u16 out_file_max;
66 #endif
67
68     u32 head_update_tick;
69
70 ;
71
```

添加



```
// 写pcm数据
int pcm2file_enc_write_pcm(void *priv, s16 *data, int len)

struct pcm2file_enc_hdl *enc = (struct pcm2file_enc_hdl *)priv;
if (!enc || enc->enc_err || enc->status==0) {
    return 0;
}

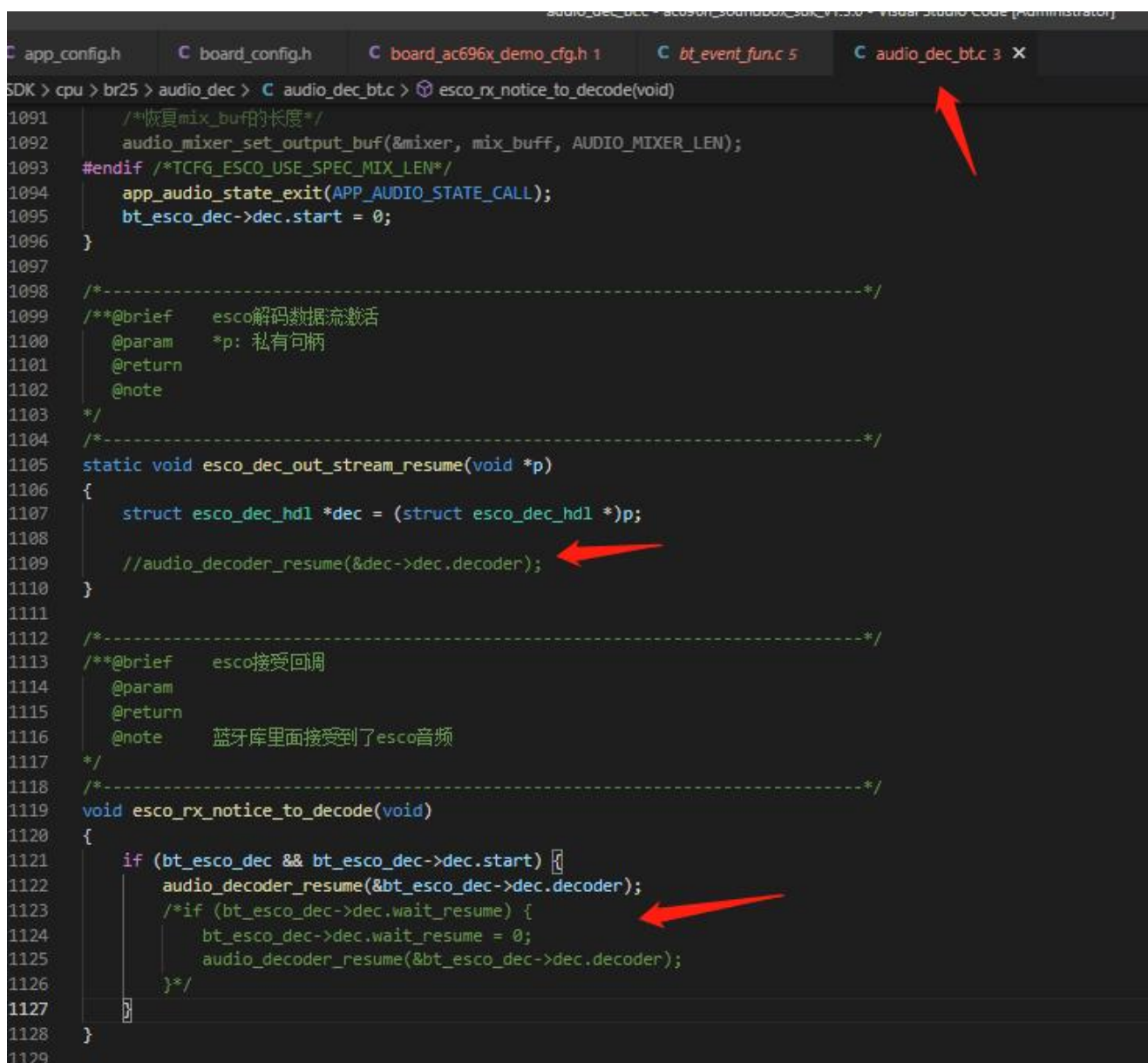
u16 wlen = 0;
// os_mutex_pend(&enc->mutex, 0);
enc->input_busy=1;
if (enc->status==1) {
    wlen = cbuf_write(&enc->pcm_cbuf, data, len);
    if (!wlen) {
        enc->lost++;
        putchar('~');
    } else {
        /* putchar('G');//_G_ */
    }
}
#if PCM2FILE_ENC_BUF_COUNT
    if (enc->pcm_buf_max < enc->pcm_cbuf.data_len) {
        enc->pcm_buf_max = enc->pcm_cbuf.data_len;
    }
#endif
/* printf("wl:%d ", wlen); */
// 激活录音编码器
pcm2file_enc_resume(enc);
enc->input_busy=0;
// os_mutex_post(&enc->mutex);
return wlen;
```

```
void pcm2file_enc_close(void **hdl)
{
    struct pcm2file_enc_hdl *pcm2file = *hdl;
    if (!pcm2file) {
        return;
    }
    os_mutex_pend(&pcm2file->mutex, 0);
    pcm2file->status = 0;
    os_mutex_post(&pcm2file->mutex);
    while(pcm2file->input_busy) {
        os_time_dly(1);
    }
    audio_encoder_close(&pcm2file->encoder);
    enc_write_file_stop(pcm2file->whdl, 1000);
    enc_write_file_close(pcm2file->whdl);

    printf("pcm2file_enc_close, lost:%d ", pcm2file->lost);

#ifdef PCM2FILE_ENC_BUF_COUNT
    printf("pcm_buf_max:%d,%d/100; out_file_max:%d,%d/100 ",
        pcm2file->pcm_buf_max, pcm2file->pcm_buf_max * 100 / PCM_ENC2FILE_PCM_LEN,
        pcm2file->out_file_max, pcm2file->out_file_max * 100 / PCM_ENC2FILE_FILE_LEN
    );
#endif
    if (pcm2file_used_overlay == 0) {
```

167. ac696n_soundbox_sdk_v1.2.2-v1.3.0 对箱 TWS 通话有回音，噗噗声，卡顿问题
20210805 WGH



```
SDK > cpu > br25 > audio_dec > C audio_dec_bt.c > esco_rx_notice_to_decode(void)
1091     /*恢复mix_buf的长度*/
1092     audio_mixer_set_output_buf(&mixer, mix_buff, AUDIO_MIXER_LEN);
1093     #endif /*TCFG_ESCO_USE_SPEC_MIX_LEN*/
1094     app_audio_state_exit(APP_AUDIO_STATE_CALL);
1095     bt_esco_dec->dec.start = 0;
1096 }
1097
1098 /*-----*/
1099 /**@brief     esco解码数据流激活
1100     @param     *p: 私有句柄
1101     @return
1102     @note
1103     */
1104 /*-----*/
1105 static void esco_dec_out_stream_resume(void *p)
1106 {
1107     struct esco_dec_hdl *dec = (struct esco_dec_hdl *)p;
1108
1109     //audio_decoder_resume(&dec->dec.decoder);
1110 }
1111
1112 /*-----*/
1113 /**@brief     esco接受回调
1114     @param
1115     @return
1116     @note     蓝牙库里面接受到了esco音频
1117     */
1118 /*-----*/
1119 void esco_rx_notice_to_decode(void)
1120 {
1121     if (bt_esco_dec && bt_esco_dec->dec.start) {
1122         audio_decoder_resume(&bt_esco_dec->dec.decoder);
1123         /*if (bt_esco_dec->dec.wait_resume) {
1124             bt_esco_dec->dec.wait_resume = 0;
1125             audio_decoder_resume(&bt_esco_dec->dec.decoder);
1126         }*/
1127     }
1128 }
1129
```

168. ac696n_soundbox_sdk_v1.2.2-v1.2.3 系统音量选择独立通道数字音量，播放提示音低概率复位问题 20210805 WGH

V130 版本已修复该问题

```
board_config.h  usb_bulk_transfer.c  audio_dec_tone.c x  board_ac696x_demo_cfg.h  audio_dec_bt.c 4
> br25 > audio_dec > audio_dec_tone.c > tone_dec_file_app_evt_cb(void *, audio_dec_app_event event, int *)
    audio_stream_close(file_dec->dec->stream);
    file_dec->dec->stream = NULL;
}
dec->cur_list->stream_handler(dec->cur_list->stream_priv, event, file_dec->dec);
}
clock_add_set(DEC_TONE_CLK);
audio_dec_file_app_init_ok(file_dec);
break;
case AUDIO_DEC_APP_EVENT_DEC_CLOSE:
    if (!audio_dec_app_mix_en) {
        tone_dec_stream_run_stop(file_dec->dec);
    }
    if (dec->cur_list->stream_handler) {
        dec->cur_list->stream_handler(dec->cur_list->stream_priv, event, file_dec->dec);
    }

#if SYS_DIGVOL_GROUP_EN
    sys_digvol_group_ch_close("tone_tone");
#endif // SYS_DIGVOL_GROUP_EN
```

169. ac696n 系列耳机工程，如果用外置触摸，触摸通过主控的 IO 口供电，测试盒串口升级，升级到 70%，又重新升级问题修改方法 20210810 XNW

问题的原因：在串口升级过程中，给触摸供电的 IO 口输出低，导致触摸一直给主控的 PB1 输出一个低电平，导致主控 8 秒复位

修改方法：在 INI 文件中设置触摸的供电脚在升级过程中给触摸正常供电

```
#UTRX=PB01;串口升级[PB00 PB05 PA05]
RESET=PB01 08 0; //port口_长按时间_有效电平 (长按时间有00、01、0
POWER_PIN=PC03_1;

#0:disable
#1:PA9 PA10
#2:USB
```

注意：这样改后，触摸的供电口任然有 1.6 秒的输出低，但不会触发 8 秒复位功能。

170. AC696X AUX 输入信号没有大于 1.2V，但 DAC 输出会失真问题处理 20210811 WTS

公版 SDK 把 AUX 增益默认配置为放大 6db，可以适当调小（1 级为 2db）：

```
4
5 if (source != 0xff) {
6     audio_adc_linein_open(&linein->linein_ch, (source << 2), &adc_hdl);
7 }
8
9 cbuf_init(&linein->cbuf, linein->store_pcm_buf, ADC_STORE_PCM_SIZE);
10 linein->sample_rate = sample_rate;
11 linein->audio_track = audio_local_sample_track_open(linein->channel_num
12 os_sem_create(&linein->sem, 0);
13 if (source == 0xff) {
14     return linein;
15 }
16 linein->channel_num = adc_hdl.channel;
17 audio_adc_linein_set_sample_rate(&linein->linein_ch, sample_rate);
18 audio_adc_linein_set_gain(&linein->linein_ch, 3);
19
20 audio_adc_set_buffs(&linein->linein_ch, linein->adc_buf, ADC_CH_NUM * A
21
22 linein->sample_output.handler = linein_sample_output_handler;
23 linein->sample_output.priv = linein;
24
25 audio_adc_add_output_handler(&adc_hdl, &linein->sample_output);
26
27 audio_adc_linein_start(&linein->linein_ch);
28
29 return linein;
30 }
```

171. AC6969D 芯片使用内置 flash 录音修改文件 20210823 LCP

网盘文件对比 608_1.2.1_SDK 修改

链接：<https://pan.baidu.com/s/1QvCiSvV6dQgVt7ESEBjCng>

提取码：6666

172. AC696X SDK130 对箱配对上后想保持音箱都输出立体声修改方法

20210824 WTS

在 audio_dec_bt.c 中重定义以下函数：

```
369 static int demo_new_data_handler(struct audio_stream_entry *entry,
370                                 struct audio_data_frame *in,
371                                 struct audio_data_frame *out)
372 {
373     // 对数据进行处理
374     put_buf(in->data, 8);
375     // 调用原来的接口输出,这里就可以保存一下是否输出完,就可以做比较多的处理
376     int wlen = ((int (*)(struct audio_stream_entry *, struct audio_data_fr
377     return wlen;
378 }
379 #endif
380
381
382 extern const int audio_tws_auto_channel ;
383 extern const int CONFIG_BTCTLER_TWS_ENABLE ;
384
385
386 void a2dp_dec_set_output_channel(struct a2dp_decoder *dec)
387 {
388     int state;
389     enum audio_channel ch_type = AUDIO_CH_DIFF;
390
391     u8 channel_num = dec->output_ch_num;//audio_output_channel_num();
392
393
394     // printf("\n---channel_num = %d ---\n",channel_num);
395
396     if ((channel_num == 2) || (channel_num == 4)) {
397         ch_type = AUDIO_CH_LR;
398         if (!AUDIO_CH_IS_MONO(dec->output_ch_type)) {
399             ch_type = dec->output_ch_type;
400             // printf("\n--ll--ch_type = %d---\n",ch_type);
401         }
402     } else {
403         if (AUDIO_CH_IS_MONO(dec->output_ch_type)) {
404             ch_type = dec->output_ch_type;
405         }
406     }
```

引用这两个变量

重定义这个函数

代码内容:

```
extern const int audio_tws_auto_channel ;
extern const int CONFIG_BTCTLER_TWS_ENABLE ;

void a2dp_dec_set_output_channel(struct a2dp_decoder *dec)
{
    int state;
    enum audio_channel ch_type = AUDIO_CH_DIFF;

    u8 channel_num = dec->output_ch_num;//audio_output_channel_num();

    // printf("\n---channel_num = %d  --\n",channel_num);

    if((channel_num == 2) || (channel_num == 4)) {
        ch_type = AUDIO_CH_LR;
        if(!AUDIO_CH_IS_MONO(dec->output_ch_type)) {
            ch_type = dec->output_ch_type;
            // printf("\n-11--ch_type = %d---\n",ch_type);
        }
    } else {
        if(AUDIO_CH_IS_MONO(dec->output_ch_type)) {
            ch_type = dec->output_ch_type;
        }
    }

    if(CONFIG_BTCTLER_TWS_ENABLE && audio_tws_auto_channel) {
        state = tws_api_get_tws_state();
        if(state & TWS_STA_SIBLING_CONNECTED) {
            if(channel_num == 1) {
                ch_type = tws_api_get_local_channel() == 'L' ? AUDIO_CH_L : AUDIO_CH_R;
            } else if(channel_num == 2) {

                // ch_type = tws_api_get_local_channel() == 'L' ? AUDIO_CH_DUAL_L :
                AUDIO_CH_DUAL_R;
                ch_type = tws_api_get_local_channel() == 'L' ? AUDIO_CH_LR : AUDIO_CH_LR;

                //printf("\n----22-----ch_type = %d -----\n",ch_type);
            }
        }
    }
}
```

```
}  
}  
  
if (ch_type != dec->ch_type) {  
    log_i("Set decode channel: %d\n", ch_type);  
    int ret = audio_decoder_set_output_channel(&dec->decoder, ch_type);  
    dec->ch = AUDIO_CH_IS_MONO(ch_type) ? 1 : 2;  
  
    //printf("\n-----dec->ch = %d-----\n",dec->ch);  
  
    if (ret == 0) {  
        dec->ch_type = ch_type;  
    }  
}  
}
```

173. 音箱工程 V1.3.1 版本 SDK 在连接上 TWS 后播歌时不断拨打挂断几次后有概率出现从箱无声音 A2DP 链路切换不成功的问题 20210830 MJW

问题描述:

在开启 TWS 功能, 对箱完成并连接手机, 手机一直播歌, 然后打微信语音电话或者拨号通话, 只拨打不接听, 反复拨打挂断拨打挂断 7~10 次, 会有概率出现从箱没有音乐声出来, 只有主箱有声音, 但是此时确实是在 tws 状态。

解决方法: 替换 btctrler.a 库

链接: <https://pan.baidu.com/s/134nVtdwFkV3cNiqcjSDMwA>

提取码: 8888

174. 注意!! 注意!! 量产程序需要关掉异常中断!! 注意!! 注意 20210907 LHY

```
lib_system_config.c - SDK - Visual Studio Code  
C lib_system_config.c X  
apps > soundbox > log_config > C lib_system_config.c  
9  
10 * Last modified : 2019-03-18 15:22  
11  
12 * Copyright:(c)JIELI 2011-2019 @ , All Rights Reserved.  
13 *****  
14  
15 #include "app_config.h"  
16 #include "system/includes.h"  
17  
18  
19 //打印是否时间打印信息  
20 const int config_printf_time = 0;  
21  
22 //看门狗中断, asser打印开启  
23 #ifdef CONFIG_RELEASE_ENABLE  
24 const int config_asser = 0; ← 在实际量产程序中需要注意关掉, 这个功能不受debug打印控制  
25 #else 如果没关掉, 看门狗复位不会软复位而是会进入异常中断  
26 const int config_asser = 1;  
27 #endif  
28  
29 const int config_system_info = 0;  
30  
31 //-----  
32 // SDFILE 精简使能  
33 //-----  
34 #ifdef CONFIG_SOUNDBOX_FLASH_256K  
35 const int SDFILE_VFS_REDUCE_ENABLE = 1;  
36 #else  
37 const int SDFILE_VFS_REDUCE_ENABLE = 0;  
38 #endif  
39  
40 //-----  
41 // FLASH FAT管理使能  
42 //-----  
43 #ifdef TCFG_VIRFAT_FLASH_SIMPLE  
44 const int VIRFAT_FLASH_ENABLE = 1;
```

175. AC696 音箱蓝牙发射器和接收器之间进行对讲的 demo 20211028 FSW

注意 user_send_cmd_prepare（接收器用）和 user_emitter_cmd_prepare（发射器用）!!!!!!!!!!!!!!

打开 AG

```
SDK > apps > soundbox > board > br25 > board_ac696x_btemitter > C board_ac696x_btemitter_cfg.h > ...
649 #define USER_SUPPORT_PROFILE_HFP 1
650 #define USER_SUPPORT_PROFILE_A2DP 1
651 #define USER_SUPPORT_PROFILE_AVCTP 1
652 #define USER_SUPPORT_PROFILE_HID 0
653 #define USER_SUPPORT_PROFILE_PNP 1
654 #define USER_SUPPORT_PROFILE_PBAP 0
655 #define USER_SUPPORT_PROFILE_HFP_AG 1
656
```

启动和结束

```
SDK > apps > soundbox > task_manager > bt > C bt.c > bt_key_event_handler(sys_event *)
934
935 switch (key_event) {
936
937 case KEY_MUSIC_PP:
938     log_info(" KEY_MUSIC_PP \n");
939     if(bt_emitter_role_get() == BT_EMITTER_EN){
940         if(get_call_status() == BT_CALL_HANGUP){
941             user_emitter_cmd_prepare(USER_CTRL_HFP_CALL_LAST_NO, 0, NULL);
942         }
943         else{
944             user_emitter_cmd_prepare(USER_CTRL_HFP_CALL_HANGUP, 0, NULL);
945         }
946     }
947     else{
948         if(get_call_status() == BT_CALL_HANGUP){
949             user_send_cmd_prepare(USER_CTRL_HFP_CALL_LAST_NO, 0, NULL);
950         }
951         else{
952             user_send_cmd_prepare(USER_CTRL_HFP_CALL_HANGUP, 0, NULL);
953         }
954     }
955     // bt_key_music_pp();
956     break;
```

自动接听

```

SDK > apps > soundbox > task_manager > bt > C bt_event_fun.c > bt_status_phone_income(bt_event *)
1498
1499     }
1500     user_send_cmd_prepare(USER_CTRL_HFP_CALL_CURRENT, 0, NULL); //发命令获取电话号码
1501 } else {
1502     /* log_debug("SCO busy now:%d,%d\n", check_esco_state_via_addr(tmp_bd_addr), bt_user_priv_var.ph
1503 }
1504 if(bt_emitter_role_get() == BT_EMITTER_EN){
1505     user_emitter_cmd_prepare(USER_CTRL_HFP_CALL_ANSWER, 0, NULL);
1506 }
1507 else{
1508     user_send_cmd_prepare(USER_CTRL_HFP_CALL_ANSWER, 0, NULL);
1509 }
1510 }
1511
1512
    
```

176. AC696 0.2.0 耳机 SDK 版本 使用联合音量对箱后出现左右耳音量不同步解决方法 20211109 LHY

1.在case TWS_EVENT_CONNECTED:设一次提示音的音量

```

#if (SYS_VOL_TYPE == VOL_TYPE_AD)
    app_audio_set_volume(APP_AUDIO_STATE_WTONE, app_audio_get_volume(APP_AUDIO_STATE_WTONE));
#endif
    
```

```

1243
1244 //ui_update_status(STATUS_BT_TWS_CONN);
1245 pbg_user_set_tws_state(1);
1246 if (gtws.tws_dly_discon_time) {
1247     r_printf("gtws.tws_dly_discon_time_del=0x%x\n", gtws.tws_dly_discon_time);
1248     sys_timeout_del(gtws.tws_dly_discon_time);
1249     gtws.tws_dly_discon_time = 0;
1250 }
1251
1252 #if (SYS_VOL_TYPE == VOL_TYPE_AD)
1253     app_audio_set_volume(APP_AUDIO_STATE_WTONE, app_audio_get_volume(APP_AUDIO_STATE_WTONE));
1254 #endif
1255     break;
1256
1257 case TWS_EVENT_SEARCH_TIMEOUT:
1258
    
```

2.在case BT_STATUS_A2DP_MEDIA_START:设一次MUSIC的音量

```

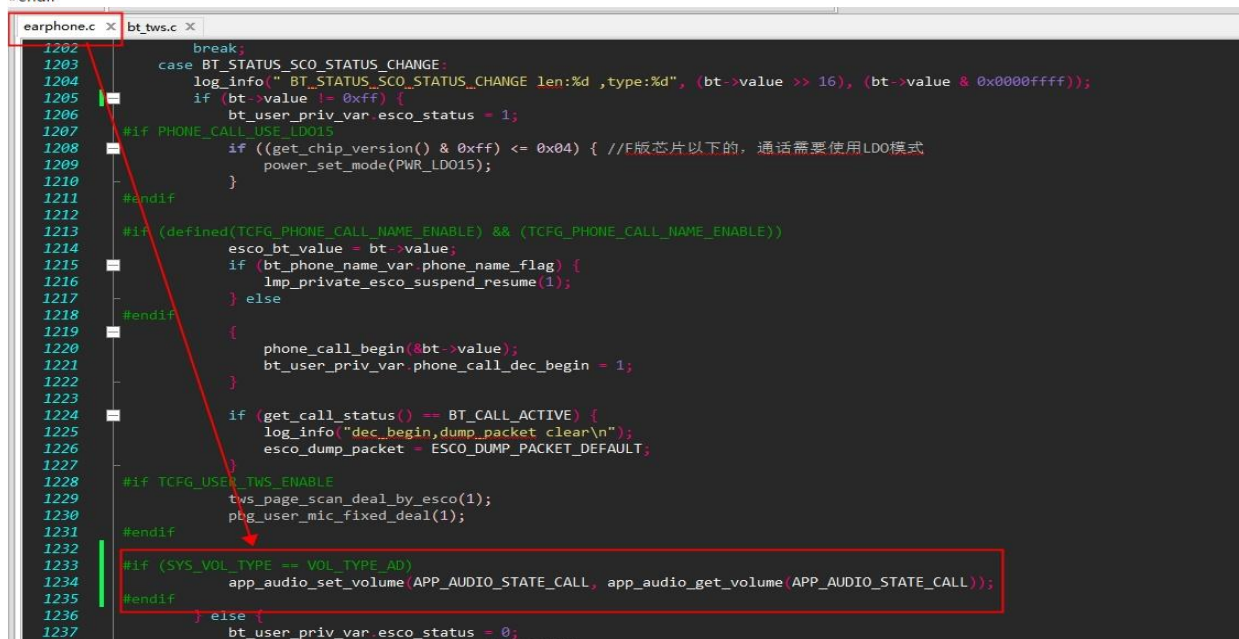
#if (SYS_VOL_TYPE == VOL_TYPE_AD)
    app_audio_set_volume(APP_AUDIO_STATE_MUSIC, app_audio_get_volume(APP_AUDIO_STATE_MUSIC));
#endif
    
```

```

1175
1176 case BT_STATUS_BEGIN_AUTO_CON:
1177     log_info("BT_STATUS_BEGIN_AUTO_CON\n");
1178     break;
1179 case BT_STATUS_A2DP_MEDIA_START:
1180     log_info(" BT_STATUS_A2DP_MEDIA_START");
1181     clk_set("sys", BT_A2DP_HZ);
1182     earphone_a2dp_audio_codec_open(bt->value);
1183     #if TCFG_USER_TWS_ENABLE
1184     if (tws_api_get_role() == TWS_ROLE_MASTER) {
1185         bt_tws_sync_volume();
1186     }
1187     #if (SYS_VOL_TYPE == VOL_TYPE_AD)
1188         app_audio_set_volume(APP_AUDIO_STATE_MUSIC, app_audio_get_volume(APP_AUDIO_STATE_MUSIC));
1189     #endif
1190     //bt_tws_sync_led_status();
1191 #endif
1192     break;
1193
1194 case BT_STATUS_A2DP_MEDIA_STOP:
    
```

3.在case BT_STATUS_SCO_STATUS_CHANGE:设一次CALL的音量

```
#if (SYS_VOL_TYPE == VOL_TYPE_AD)  
    app_audio_set_volume(APP_AUDIO_STATE_CALL, app_audio_get_volume(APP_AUDIO_STATE_CALL));  
#endif
```



```
1202         break;  
1203     case BT_STATUS_SCO_STATUS_CHANGE:  
1204         log_info(" BT_STATUS_SCO_STATUS_CHANGE len:%d ,type:%d", (bt->value >> 16), (bt->value & 0x0000ffff));  
1205         if (bt->value != 0xff) {  
1206             bt_user_priv_var esco_status = 1;  
1207             #if PHONE_CALL_USE_LDO15  
1208                 if ((get_chip_version() & 0xff) <= 0x04) { //F版芯片以下的，通话需要使用LDO模式  
1209                     power_set_mode(PWR_LDO15);  
1210                 }  
1211             #endif  
1212  
1213             #if (defined(TCFG_PHONE_CALL_NAME_ENABLE) && (TCFG_PHONE_CALL_NAME_ENABLE))  
1214                 esco_bt_value = bt->value;  
1215                 if (bt_phone_name_var phone_name_flag) {  
1216                     lmp_private_esco_suspend_resume(1);  
1217                 } else  
1218                 #endif  
1219                 {  
1220                     phone_call_begin(&bt->value);  
1221                     bt_user_priv_var phone_call_dec_begin = 1;  
1222                 }  
1223  
1224                 if (get_call_status() == BT_CALL_ACTIVE) {  
1225                     log_info("dec_begin,dump_packet clear\n");  
1226                     esco_dump_packet = ESCO_DUMP_PACKET_DEFAULT;  
1227                 }  
1228             #if TCFG_USER_TWS_ENABLE  
1229                 tws_page_scan_deal_by_esco(1);  
1230                 plg_user_mic_fixed_deal(1);  
1231             #endif  
1232  
1233             #if (SYS_VOL_TYPE == VOL_TYPE_AD)  
1234                 app_audio_set_volume(APP_AUDIO_STATE_CALL, app_audio_get_volume(APP_AUDIO_STATE_CALL));  
1235             #endif  
1236         } else  
1237             bt_user_priv_var esco_status = 0;
```

177. AC696 耳机 SDK 版本 使用 mic_rec_play_start()测试 mic 无声的解决方法 20211110 FSW

按照下图修改

```

apps > earphone > C app_audio.c > mic_play_fseek(void *, u32, int)
662 }
663
664 static int mic_play_fread(void *file, void *buf, u32 len)
665 {
666 #if 0
667     u8 head_len = 0;
668     if (mic_play.file_ptr < sizeof(pcm_wav_header)) {
669         head_len = sizeof(pcm_wav_header);
670         head_len -= mic_play.file_ptr;
671         if (len < head_len) {
672             head_len = len;
673         }
674         memcpy(buf, &pcm_wav_header[mic_play.file_ptr], head_len);
675         mic_play.file_ptr += head_len;
676         buf += head_len;
677     }
678 #endif
679     int rlen = 0;
680     int error_cnt = 0;
681     while(1){
682         rlen = dev_read(file, buf, len);
683         if(rlen || ++error_cnt >= 3){
684             break;
685         }
686         os_time_dly(2);
687     }
688     return rlen;
689 }
690

```

178. AC696 音箱 1.5.0 SDK 版本 Linein 走数字通道 Automute 不起作用 20220211 SQ

<https://kdocs.cn/l/ceB5MK3Mz7Hz>

代码修改:

```

audio_config.h > .audio_dec.c > audio_dec_linein.c > audio_enc_recorder.c > board_config.h ... buffers
291
292 /* linein = &g_linein_sample_hdl; // zalloc(sizeof(struct linein_sample_hdl)); */
293 linein = zalloc(sizeof(struct linein_sample_hdl));
294 if (!linein) {
295     return NULL;
296 }
297
298 memset(linein, 0x0, sizeof(struct linein_sample_hdl));
299
300 linein->store_pcm_buf = malloc(ADC_STORE_PCM_SIZE);
301 if (!linein->store_pcm_buf) {
302     return NULL;
303 }
304
305 if (source != 0xff) {
306     audio_adc_linein_open(&linein->linein_ch, (source << 2), &adc_hdl);
307     SFR(JL_ANA->ADA_CON1, 19, 1, 0); //23khz 打开linein后加多这句设置 adc 抖动的频率
308 }
309
310 }
311
312 cbuf_init(&linein->cbuf, linein->store_pcm_buf, ADC_STORE_PCM_SIZE);
313 linein->sample_rate = sample_rate;
314 linein->audio_track = audio_local_sample_track_open(linein->channel_num, sample_rate, 1000);
315 os_sem_create(&linein->sem, 0);
316 if (source == 0xff) {
317     return linein;
318 }
319 linein->channel_num = adc_hdl.channel;
320
321 printf("linein sample rate=%d\n", sample_rate, __LINE__);
322 audio_adc_linein_set_sample_rate(&linein->linein_ch, sample_rate);
323 audio_adc_linein_set_gain(&linein->linein_ch, 3);
324
325 audio_adc_set_buffs(&linein->linein_ch, linein->adc_buf, ADC_CH_NUM * ADC_IRQ_POINTS * 2, ADC_BUF_NUM);
326
327 linein->sample_output_handler = linein_sample_output_handler;
328
329 NORMAL br25_150 cpu/br25/audio_enc/audio_enc_recorder.c c 39% 307/773 : 1
audio_recorder.c [85] <<audio_adc_linein_demo>> audio_adc_linein_open(&adc_linein->ch, AUDIO_ADC_LINE0_L, &adc_hdl);

```

```
emo.c > .board_ac696x_demo_cfg.h > .audio_energy_detect.c > ^lib_driver_config.c > ^lib_driver.c
226 /*MIC内部上拉电阻档位配置,影响MIC的偏置电压
227 (G版后的芯片省电容方式这个配置有效,隔直模式配置这个参数无效,固定为3k,使用 mic b
ias 供电硬件最好预留电阻位)
228 21:1.18K 20:1.42K 19:1.55K 18:1.99K 17:2.2K 16:2.4K 15:2.6K
14:2.91K 13:3.05K 12:3.5K 11:3.73K
229 10:3.91K 9:4.41K 8:5.0K 7:5.6K 6:6K 5:6.5K 4:7K
3:7.6K 2:8.0K 1:8.5K */
230 .mic_bias_res = 16,
231 /*MIC LDO电压档位设置,也会影响MIC的偏置电压
232 0:2.3v 1:2.5v 2:2.7v 3:3.0v */
233 .mic_ldo_vsel = 2,
234 /*MIC电容隔直模式使用内部mic偏置(PA2)*/
235 .mic_bias_inside = 1,
236 /*保持内部mic偏置输出*/
237 .mic_bias_keep = 0,
238
239 // ladc 通道
/ 240 .ladc_num = ARRAY_SIZE(ladc_list),
/ 241 .ladc = ladc_list,
tte 242 .dither_amplitude = 3, //3是最小db档位的抖动
243 },
244 #endif
o.c
o_c 245
er/ 246 /* struct audio_pf_data audio_pf_d = { */
box 247 /* #if TCFG_AUDIO_DAC_ENABLE */
248 /* .adc_pf_data = &adc_data, */
249 /* #endif */
250
ox/ 251 /* #if TCFG_AUDIO_ADC_ENABLE */
252 /* .dac_pf_data = &dac_data, */
253 /* #endif */
254 /* }; */
255
256 /* struct audio_platform_data audio_data = { */
257 /* .private_data = (void *) &audio_pf_d, */
258 /* }; */
259
NORMAL | br25_150 <ard_ac696x_demo/board_ac696x_demo.c c 22% 242/1093 : 24
```

加多这句设置抖动的幅度

3. 替换 media_app.a 库

- 1) 修复 automute_linein 模式下获取能量值 值不对的问题
- 2) 添加更新 automute 参数的接口

```
void *audio_energy_detect_modify(void *hdl_arg, audio_energy_detect_param *param)
```

- 3) 新增接口使用案例:

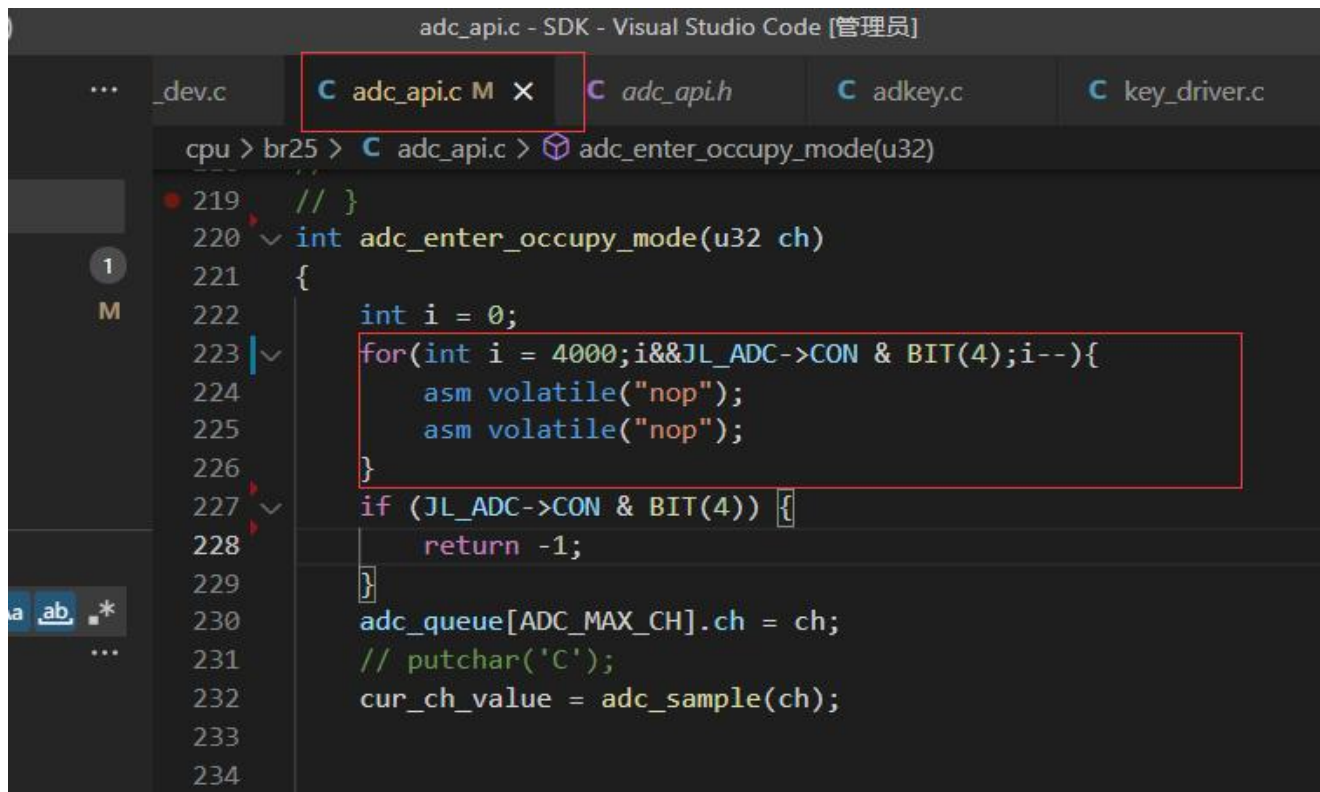
```
audio_enc.c  audio_eq.h  audio_energy_detect.h  audio_dec.c 3 X  app_audio.c
SDK > cpu > br25 > audio_dec > C audio_dec.c > ...
1073 {
1074     u8 mute = !skip;
1075     if (mix_out_automute_hdl) {
1076         audio_energy_detect_skip(mix_out_automute_hdl, 0xFFFF, skip);
1077         audio_mix_out_automute_mute(mute);
1078     }
1079 }
1080
1081 void in_linein_automute_update(void){
1082     audio_energy_detect_param e_det_param = {0};
1083     if (mix_out_automute_hdl) {
1084         audio_energy_detect_param e_det_param = {0};
1085         e_det_param.mute_energy = 4;
1086         e_det_param.unmute_energy = 10;
1087         e_det_param.mute_time_ms = 100;
1088         e_det_param.unmute_time_ms = 50;
1089         e_det_param.count_cycle_ms = 10;
1090         e_det_param.sample_rate = 44100;
1091         e_det_param.event_handler = mix_out_automute_handler;
1092         e_det_param.ch_total = app_audio_output_channel_get();
1093         e_det_param.dcc = 1;
1094         audio_energy_detect_modify(mix_out_automute_hdl,&e_det_param);
1095     }
1096 }
1097
1098 void exit_linein_automute_update(void){
1099     if (mix_out_automute_hdl) {
1100         audio_energy_detect_param e_det_param = {0};
1101         e_det_param.mute_energy = 5;
1102         e_det_param.unmute_energy = 10;
1103         e_det_param.mute_time_ms = 1000;
1104         e_det_param.unmute_time_ms = 50;
1105         e_det_param.count_cycle_ms = 10;
1106         e_det_param.sample_rate = 44100;
1107         e_det_param.event_handler = mix_out_automute_handler;
1108         e_det_param.ch_total = app_audio_output_channel_get();
1109         e_det_param.dcc = 1;
1110         audio_energy_detect_modify(mix_out_automute_hdl,&e_det_param);
1111     }
1112 }
```

```
92 /*-----*/
93 void user_get_dac_energy(void);
94 extern void in_linein_automute_update(void);
95 extern void exit_linein_automute_update(void);
96 void app_linein_task()
97 {
98     int res;
99     int err = 0;
100     int msg[32];
101     in_linein_automute_update();//测试
102     user_get_dac_energy();
103 #if TCFG_APP_BT_EN
104     linein_bt_back_flag = get_bt_back_flag();//从蓝牙后台返回标志
105     set_bt_back_flag(0);
106 #else
107     linein_bt_back_flag = 0;
108 #endif
109     log_info("linein_bt_back_flag == %d linein_last_onoff = %d\n", \
110             linein_bt_back_flag, linein_last_onoff);
111
112     linein_app_init();//初始化时钟和开启ui
113
114 #if TCFG_DEC2TWS_ENABLE
115     extern void set_tws_background_connected_flag(u8 flag);
116     extern u8 get_tws_background_connected_flag();
117     if (get_tws_background_connected_flag()) { //不播放提示音
118         app_task_put_key_msg(KEY_LINEIN_START, 0);
119         set_tws_background_connected_flag(0);
120     } else
121 #endif
122     {
123         err = tone_play_with_callback_by_name(tone_table[INDEX_TONE_LINEIN
124             line_tone_play_end_callback
125         }
126 //     if (err) { //
127 //         ///提示音播放失败，直接推送KEY_MUSIC_PLAYER_START启动播放
128 //         app_task_put_key_msg(KEY_LINEIN_START, 0);
129 //     }
130
131     while (1) {
132         app_task_get_msg(msg, ARRAY_SIZE(msg), 1);
133
134         switch (msg[0]) {
135             case APP_MSG_SYS_EVENT:
136                 if (linein_sys_event_handler((struct sys_event *)&msg[1])) =
137                     app_default_event_deal((struct sys_event *)&msg[1]);
138                 }
139             break;
140             default:
141                 break;
142         }
143
144         if (app_task_exitting()) {
145             exit_linein_automute_update();//测试
146             linein_task_close();
147             return;
148         }
149     }
150 }
151
```

4.

179. AC696 音箱 1.5.0 或者之前的 SDK 版本包括 695 的音箱 SDK 出现 Linein 检测有概率出现检测不了拔出 20220114 MJW

解决办法增加如下截图代码:



```
adc_api.c - SDK - Visual Studio Code [管理员]
... _dev.c C adc_api.c M X C adc_api.h C adkey.c C key_driver.c
cpu > br25 > C adc_api.c > adc_enter_occupy_mode(u32)
219 // }
220 int adc_enter_occupy_mode(u32 ch)
221 {
222     int i = 0;
223     for(int i = 4000; i && JL_ADC->CON & BIT(4); i--){
224         asm volatile("nop");
225         asm volatile("nop");
226     }
227     if (JL_ADC->CON & BIT(4)) {
228         return -1;
229     }
230     adc_queue[ADC_MAX_CH].ch = ch;
231     // putchar('C');
232     cur_ch_value = adc_sample(ch);
233
234
```

180. AC696N_soundbox_sdk_release_1.4.0 SDK 配成单声道会出现声音破音，波形失真 LZK

单声道需要把以下变量置 1

const int config_pcm_dual_to_dual_or_single_half_en = 1; // 左右声道叠加除 2 使能

181. AC696 耳机 SDK020 开立体声时，通话前 20S 左右远端听不到声音处理方法 20201201 WTS

请按以下网盘资料处理:

链接: https://pan.baidu.com/s/1w_KbBfa-yFGcjV_BFRPDA

提取码: jiel

182、AC696_earphone_v030 版本 SDK 添加连接确认功能 ---YP 20220506

AC696_earphone_v030 版本 SDK 添加连接确认功能，需要更新蓝牙库，然后上层修改如下:

182. AC696 音箱 SDK 使用 ANCS 功能

```
le_trans_data.c - SDK - visual studio code
le_trans_data.c X tuya_demo.c task_key.c app_task_switch.c event.h spi.c
apps > common > third_party_profile > jieli > trans_data_demo > le_trans_data.c > att_write_callback(hci_con_handle_t, uint16_t, uint16_t, uint16_t, uint8_t *, uint16_t)
37
38 #include "le_trans_data.h"
39 #include "le_common.h"
40
41 #include "rcsp_bluetooth.h"
42 #include "JL_rcsp_api.h"
43 #include "custom_cfg.h"
44
45
46 #if (TCFG_BLE_DEMO_SELECT == DEF_BLE_DEMO_TRANS_DATA)
47
48 //TRANS ANCS
49 #define TRANS_ANCS_EN 0
50 #if TRANS_ANCS_EN
51 #include "btstack/btstack_event.h"
52 #endif
53
54 #define TEST_SEND_DATA_RATE 0 //测试上行发送数据
55 #define TEST_SEND_HANDLE_VAL ATT_CHARACTERISTIC_ae02_01_VALUE_HANDLE
56 /* #define TEST_SEND_HANDLE_VAL ATT_CHARACTERISTIC_ae05_01_VALUE_HANDLE */
57 #define EXT_ADV_MODE_EN 0
58
59 #define TEST_AUDIO_DATA_UPLOAD 0 //测试文件上传
60
```

打开ANCS功能



```
终端(1) 帮助(H) lib_btstack_config.c - SDK - Visual Studio Code
td.c  C app_config.h  C lib_btstack_config.c X  C tuya_demo.c  C task_key.c  C app_task_switch.c  C event.h  C spi.c
pps > soundbox > log_config > C lib_btstack_config.c > config_le_sm_support_enable
30 // 可以按任意键退出SHIT
37 const int config_btstask_auto_exit_sniff = 1;
38
39
40
41 #if SMART_BOX_EN
42 const int config_rcsp_stack_enable = 1;
43 #else
44 const int config_rcsp_stack_enable = 0;
45 #endif
46
47 #if TRANS_MULTI_BLE_EN
48 //le 配置,可以优化代码和RAM
49 const int config_le_hci_connection_num = 2;//支持同时连接个数
50 const int config_le_sm_support_enable = 0; //是否支持加密配对
51 const int config_le_gatt_server_num = 1; //支持server角色个数
52 const int config_le_gatt_client_num = 1; //支持client角色个数
53
54 #else
55
56 //le 配置,可以优化代码和RAM
57 const int config_le_hci_connection_num = 1;//支持同时连接个数
58 const int config_le_sm_support_enable = 0; //是否支持加密配对
59 const int config_le_gatt_server_num = 1; //支持server角色个数
60 const int config_le_gatt_client_num = 1; //支持client角色个数
61 #endif
```

183.695V310 以及旧版本 SDK 当歌曲内容恰好存放于设备地址的底部,播放歌曲会异常退出设备 20220614 WGH

修改以下文件: apps\common\usb\host\usb_storage.c

```
/* 大于0的表示读取的字节数(Byte) */
static int _usb_stor_read(struct device *device, void *pBuf, u32 num_lba, u32 lba)
{
    struct usb_host_device *host_dev = device_to_usbdev(device);
    struct mass_storage *disk = host_device2disk(host_dev);
    const usb_dev usb_id = host_device2id(host_dev);

    const u32 txmaxp = usb_stor_txmaxp(disk);
    const u32 rxmaxp = usb_stor_rxmaxp(disk);
    const u32 curlun = usb_stor_get_curlun(disk);

    u8 read_retry = 0;
    int ret;
    u32 rx_cnt;
    /* u32 rx_len = num_lba * disk->capacity[curlun].block_size; */
    u32 rx_len = num_lba * 512; //filesystem uses the fixed BLOCK_SIZE

    if (disk->capacity[curlun].block_size > 512) {
        // 扇区大于512Byte
        log_error("%s:%d\n", __func__, __LINE__);
        return _usb_stor_read_big_block(device, disk->capacity[curlun].block_size, pBuf, num_lba, lba)
    }

    if (lba + num_lba - 1 >= disk->capacity[curlun].block_num) {
        log_error("%s:%d\n", __func__, __LINE__);
        return -DEV_ERR_OVER_CAPACITY;
    }

    #if (UDISK_READ_BIGBLOCK_ASYNC_ENABLE || UDISK_READ_512_ASYNC_ENABLE)
    if (disk->async_en) {
        return _usb_stor_read_async(device, pBuf, num_lba, lba);
    }
    #endif

    ret = _usb_stor_read_cbw_request(device, num_lba, lba);
    if (ret < DEV_ERR_NONE) {
        log_error("%s:%d\n", __func__, __LINE__);
        goto __exit;
    }
}
```

```
static int _usb_stor_write(struct device *device, void *pBuf, u32 num_lba, u32 lba)
{
    struct usb_host_device *host_dev = device_to_usbdev(device);
    struct mass_storage *disk = host_device2disk(host_dev);
    const usb_dev usb_id = host_device2id(host_dev);
    u32 ret;

    const u32 txmaxp = usb_stor_txmaxp(disk);
    const u32 rxmaxp = usb_stor_rxmaxp(disk);
    const u32 curlun = usb_stor_get_curlun(disk);

    if (lba + num_lba - 1 >= disk->capacity[curlun].block_num) {
        log_error("%s %d", __func__, __LINE__);
        return -DEV_ERR_OVER_CAPACITY;
    }

    if (disk->capacity[curlun].block_size > 512) {
        log_error("%s disk block size %d not support write", __func__, disk->capacity[curlun].block_size);
        return -DEV_ERR_NO_WRITE;
    }

    if (disk->read_only) {
        log_error("%s %d", __func__, __LINE__);
        return -DEV_ERR_NO_WRITE;
    }

    /* u32 tx_len = num_lba * disk->capacity[curlun].block_size; */
    u32 tx_len = num_lba * 512; //filesystem uses the fixed BLOCK_SIZE
}
```

184.AC696 1.3.SDK 打开人声消除的宏定义，声音失真

声音输入声音叠加后导致失真

The image shows a code editor on the left and an oscilloscope on the right. The code defines various audio output modes, with `DAC_OUTPUT_MONO_L` highlighted. The oscilloscope displays a distorted waveform with a red arrow pointing to it, labeled "DAC-L的波形". A table of statistics is visible on the oscilloscope screen:

频率 (C1-34)	频率 (C2-0)	峰值值 (C1-778)
当前 1,000kHz	当前 OFF	当前 2.780V
平均 1,000kHz	平均 ---	平均 1.681V
最小 990.00Hz	最小 ---	最小 120.00mV
最大 1,001kHz	最大 ---	最大 2.840V
标准 563.28mHz	标准 ---	标准 1.283V

Red annotations in Chinese are present: "配成左声道输出失真" (Distorted when configured as left channel output) and "改为DAC_OUTPUT_LR 波形不会失真" (Change to DAC_OUTPUT_LR, waveform will not be distorted).

修改方法:

替换 media.a 库

【金山文档】 [双变单转换默认除 2_ac696n_soundbox_sdk_v1.3.3_media](#)

185. ac696n_sdk_release_v0.3.0 SDK VIVO 手机连接打开 SIRI 蓝牙断连问题

需更新 media.a 库，百度网盘连接如下：

链接：https://pan.baidu.com/s/1gf_XuJrK_O9URjMbeuDIwA

提取码：id39

Version: ac696n_earphone_v0.3.0

CommitID: 18c54c2

CommitDate: 2022-01-13

Date: 2022-01-13

Author: GZR

Log: -

- 1、支持立体声通话时候的回音消除
- 2、修复vivo S10 Pro 手机（工程机）SIRI断连问题
- 3、算法ram不复用蓝牙buf，避免申请不到的情况

说明：该补丁是在“G板补丁”基础上打的

186. AC696X AUX 输入信号有效值没有超过 600MV，但从 DAC 输出出现失真问题处理 -----20220705 WTS

- 1、如果 AUX 走的模拟通道，把 AUX 模拟增益调到 0:

```
linein_api.c - SDK - Visual Studio Code [管理员]
... C board_config.h C board_ac695x_demo_cfg.h C linein_api.c X C audio_linein.h C setup.c C aud
apps > soundbox > task_manager > linein > C linein_api.c > _linein_way_analog_start()
93  */
94  /*-----*/
95  static inline void __linein_way_analog_start()
96  {
97      app_audio_state_switch(__this->audio_state, get_max_sys_vol());//
98      app_audio_set_volume(__this->audio_state, __this->volume, 1);
99      if (!app_audio_get_volume(__this->audio_state)) {
100         audio_linein_mute(1); //模拟输出时候, dac为0也有数据
101     }
102
103     if (TCFG_LINEIN_LR_CH & (BIT(0) | BIT(1))) {
104         audio_linein0_open(TCFG_LINEIN_LR_CH, 1);
105     } else if (TCFG_LINEIN_LR_CH & (BIT(2) | BIT(3))) {
106         audio_linein1_open(TCFG_LINEIN_LR_CH, 1);
107     } else if (TCFG_LINEIN_LR_CH & (BIT(4) | BIT(5))) {
108         audio_linein2_open(TCFG_LINEIN_LR_CH, 1);
109     }
110
111     if (TCFG_LINEIN_LR_CH != AUDIO_LIN0_LR && TCFG_LINEIN_LR_CH != AUDIO_LIN1_LR && TCFG_LINEI
112         audio_linein_ch_combine(1, 1);
113     }
114
115     audio_linein_gain(0); // high gain
116     if (app_audio_get_volume(__this->audio_state)) {
117         audio_linein_mute(0);
118         app_audio_set_volume(__this->audio_state, app_audio_get_volume(__this->audio_state), 1
119     }
120     //模拟输出时候, dac为0也有数据
121 }
122
123 /*-----*/
124 /**@brief   linein 使用dac IO输入方式
125  @param   无
```

2、如果 AUX 走的是 AD 采样通道，请将采样初始化时的 AUX 增益调到 0:

```
audio_digital_vol.c x bt.c x audio_enc_recoder.c x adkey.c x pwm_led.c x adc_api.c x board_ac6082_demo_cfg.h x linein_dev.c x
261     return NULL;
262 }
263
264     memset(linein, 0x0, sizeof(struct linein_sample_hdl));
265
266     linein->store_pcm_buf = malloc(ADC_LINEIN_STORE_PCM_SIZE);
267     if (!linein->store_pcm_buf) {
268         return NULL;
269     }
270
271     if (source != 0xff) {
272         audio_adc_linein_open(&linein->linein_ch, (source << 2), &adc_hdl);
273     }
274
275     cbuf_init(&linein->cbuf, linein->store_pcm_buf, ADC_LINEIN_STORE_PCM_SIZE);
276     linein->sample_rate = sample_rate;
277     linein->audio_track = audio_local_sample_track_open(linein->channel_num, sample_rate, 1000);
278     os_sem_create(&linein->sem, 0);
279     if (source == 0xff) {
280         return linein;
281     }
282     linein->channel_num = adc_hdl.channel;
283     audio_adc_linein_set_sample_rate(&linein->linein_ch, sample_rate);
284     audio_adc_linein_set_gain(&linein->linein_ch, 0);
285     audio_adc_set_buffs(&linein->linein_ch, linein->adc_buf, ADC_CH_NUM * ADC_LINEIN_IRQ_POINTS * 2, ADC_BUF_NUM);
286
287     linein->sample_output.handler = linein_sample_output_handler;
288     linein->sample_output.priv = linein;
289
290     audio_adc_add_output_handler(&adc_hdl, &linein->sample_output);
291
292     audio_adc_linein_start(&linein->linein_ch);
293
294     return linein;
295 }
296
297 void linein_sample_close(void *hdl)
```

改为0

187. AC696X_170SDK 使用默认 DEMO 板级调式或烧录出的样机蓝牙 MAC 地址变成固定（没有随机生成）问题-----20220709 LZK

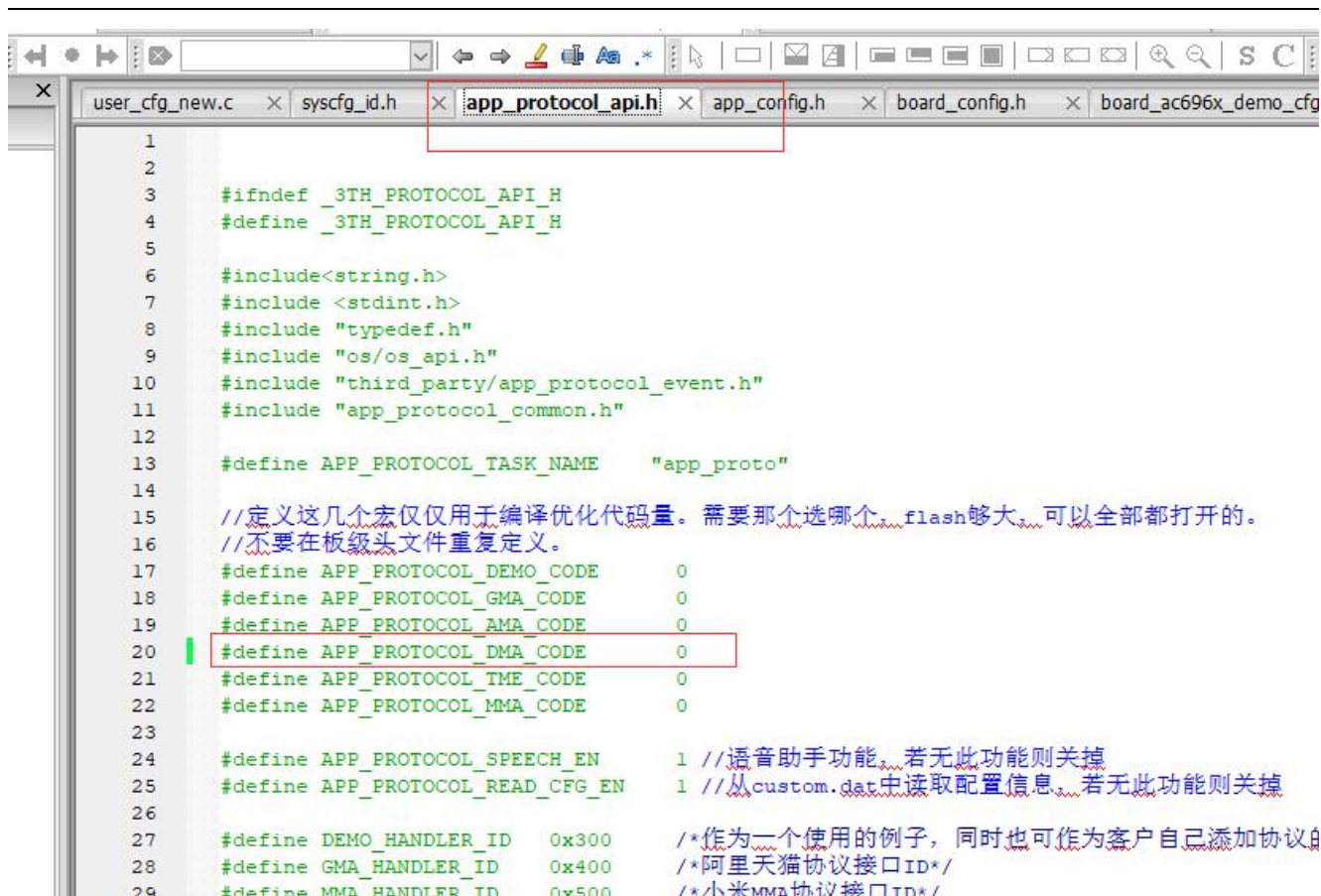
处理方法：

- 1、预定义 CONFIG_TWS_COMMON_ADDR_SELECT 宏：

```
user_cfg_new.c × app_config.h × board_config.h × board_ac696x_demo_cfg.h × board_ac6366c_chargebox_inside_cfg.h
145 #define TCFG_APP_PC_EN 0
146
147 #undef TCFG_PC_ENABLE
148 #define TCFG_PC_ENABLE 0
149
150 #undef TCFG_UDISK_ENABLE
151 #define TCFG_UDISK_ENABLE 0
152
153 /* #undef TCFG_UART0_ENABLE
154 #define TCFG_UART0_ENABLE DISABLE_THIS_MOUDLE */
155
156 #endif //(CONFIG_BT_MODE != BT_NORMAL)
157
158 //确保上面 undef 后在include usb
159 #include "usb_common_def.h"
160
161
162 #define CONFIG_TWS_COMMON_ADDR_SELECT 0xFF
163
164
165
166 #if TCFG_USER_TWS_ENABLE
167
168 #define CONFIG_TWS_COMMON_ADDR_AUTO 0 /* 自动生成TWS配对后的MAC地址 */
169 #define CONFIG_TWS_COMMON_ADDR_USED_LEFT 1 /* 使用左耳的MAC地址作为TWS配对后的地址
170 可配合烧写器MAC地址自增功能一起使用
171 多台交叉配对会出现MAC地址相同情况 */
172
173 #ifdef CONFIG_APP_BT_ENABLE
174 #define CONFIG_TWS_USE_COMMON_ADDR 0 /* TWS 使用公共地址 */
175 #define CONFIG_TWS_COMMON_ADDR_SELECT CONFIG_TWS_COMMON_ADDR_USED_LEFT
176 #else
177 #define CONFIG_TWS_USE_COMMON_ADDR 1 /* TWS 使用公共地址 */
178 #define CONFIG_TWS_COMMON_ADDR_SELECT CONFIG_TWS_COMMON_ADDR_AUTO
179 #endif
180 #endif
```

增加

2、APP_PROTOCOL_DMA_CODE 这个宏不带 APP 时，把它设置为 0



```
1
2
3 #ifndef _3TH_PROTOCOL_API_H
4 #define _3TH_PROTOCOL_API_H
5
6 #include<string.h>
7 #include <stdint.h>
8 #include "typedef.h"
9 #include "os/os_api.h"
10 #include "third_party/app_protocol_event.h"
11 #include "app_protocol_common.h"
12
13 #define APP_PROTOCOL_TASK_NAME    "app_proto"
14
15 //定义这几个宏仅仅用于编译优化代码量。需要那个选哪个。flash够大。可以全部都打开的。
16 //不要在板级头文件重复定义。
17 #define APP_PROTOCOL_DEMO_CODE    0
18 #define APP_PROTOCOL_GMA_CODE    0
19 #define APP_PROTOCOL_AMA_CODE    0
20 #define APP_PROTOCOL_DMA_CODE    0
21 #define APP_PROTOCOL_TME_CODE    0
22 #define APP_PROTOCOL_MMA_CODE    0
23
24 #define APP_PROTOCOL_SPEECH_EN    1 //语音助手功能。若无此功能则关掉
25 #define APP_PROTOCOL_READ_CFG_EN  1 //从custom.dat中读取配置信息。若无此功能则关掉
26
27 #define DEMO_HANDLER_ID    0x300 /*作为一个使用的例子，同时也可作为客户自己添加协议的
28 #define GMA_HANDLER_ID    0x400 /*阿里天猫协议接口ID*/
29 #define MMA_HANDLER_ID    0x500 /*小米MMA协议接口ID*/
```

188. AC696X_170SDK 或者打漏电补丁后，某些 IO 无法输出打印信息（DP,DM,PC4 等） WGH

```
static u8 debug_io_dir = 0;
static u8 debug_io_die = 0;
static u8 debug_io_dieh = 0;
static u8 debug_io_pu = 0;
static u8 debug_io_pd = 0;
static const u32 gpio_regs[] = {
    (u32) JL_PORTA,
    (u32) JL_PORTB,
    (u32) JL_PORTC,
    (u32) JL_PORTD,
};
int debug_io_status_save(u32 gpio)
{
    if (gpio > IO_PORT_MAX) {
        return -EINVAL;
    }
}
```

```
struct gpio_reg *g;
u32 mask;
if (gpio >= IO_MAX_NUM) {
    gpio = gpio - USB_IO_OFFSET - IO_MAX_NUM;
    debug_io_dir = (JL_USB_IO->CON0 & BIT(gpio + 2));
    debug_io_pd = (JL_USB_IO->CON0 & BIT(gpio + 4));
    debug_io_pu = (JL_USB_IO->CON0 & BIT(gpio + 6));
    debug_io_die = (JL_USB_IO->CON0 & BIT(gpio + 9));
    debug_io_dieh = (JL_USB_IO->CON0 & BIT(gpio + 13));
    return 0;
}
g = (struct gpio_reg *)gpio_regs[gpio / IO_GROUP_NUM];
if (!g) {
    return -EINVAL;
}
mask = BIT((gpio) % IO_GROUP_NUM);
debug_io_dir = (g->dir & mask);
debug_io_die = (g->die & mask);
debug_io_dieh = (g->dieh & mask);
debug_io_pu = (g->pu & mask);
debug_io_pd = (g->pd & mask);

return 0;
}
void debug_io_status_post(u32 gpio)
{
    gpio_set_direction(gpio, debug_io_dir);
    gpio_set_die(gpio, debug_io_die);
    gpio_set_dieh(gpio, debug_io_dieh);
    gpio_set_pull_up(gpio, debug_io_pu);
    gpio_set_pull_down(gpio, debug_io_pd);
}
```

将代码放在下面修改:

```
1124 static void board_power_wakeup_init(void)
1125 {
1126     debug_io_status_save(TCFG_UART0_TX_PORT);
1127     power_wakeup_init(&wk_param);
1128     debug_io_status_post(TCFG_UART0_TX_PORT);
1129     key_wakeup_disable();
```

189. AC696n_earphone_v0.2.0 设置立体声通话有整句回音处理方法 20220714---- WTS

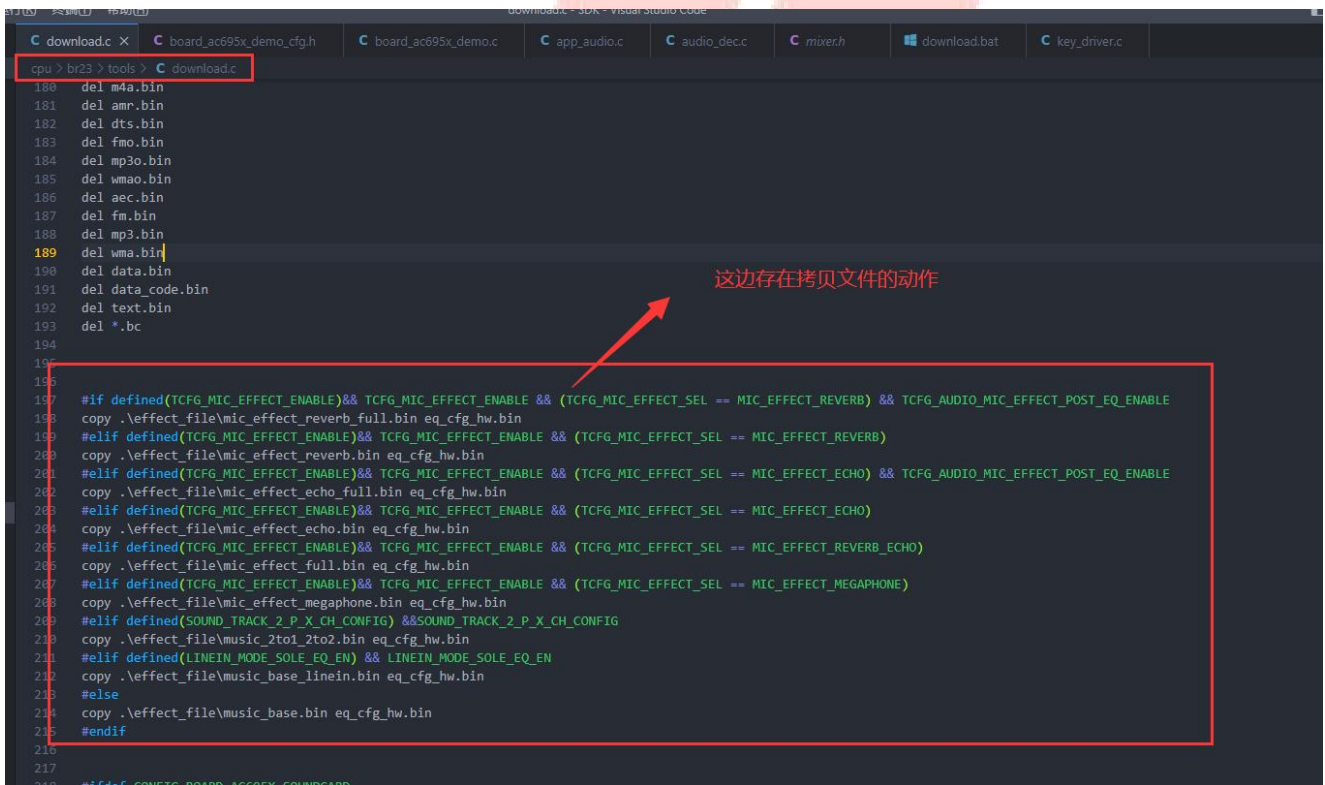
打以下补丁：

【金山文档】 ac696n_earphone_v0.2.0_立体声通话回音消除加 G 板补丁
<https://kdocs.cn/l/ce3hCT9oD2Jd>

190. AC696n_earphone_v1.7.0 替换 eq_cfg_hw.bin 的正确位置 20220815---- WGH

鉴于许多客户反馈 695V310 和 696V170 版本说替换 eq_cfg_hw.bin 无效问题，在此说明的原因，和解决方法。
原因：

695V310 和 696V170 添加了新音效，tool 文件夹目录下的 download.c 文件做了更改，会对 tools\effect_file 文件夹下 .bin 文件做文件拷贝的动作。



```
cpu > br23 > tools > C download.c
180 del m4a.bin
181 del amr.bin
182 del dts.bin
183 del fmo.bin
184 del mp3o.bin
185 del wmao.bin
186 del aec.bin
187 del fm.bin
188 del mp3.bin
189 del wma.bin
190 del data.bin
191 del data_code.bin
192 del text.bin
193 del *.bc
194
195
196
197 #if defined(TCFG_MIC_EFFECT_ENABLE)&& TCFG_MIC_EFFECT_ENABLE && (TCFG_MIC_EFFECT_SEL == MIC_EFFECT_REVERB) && TCFG_AUDIO_MIC_EFFECT_POST_EQ_ENABLE
198 copy .\effect_file\mic_effect_reverb_full.bin eq_cfg_hw.bin
199 #elif defined(TCFG_MIC_EFFECT_ENABLE)&& TCFG_MIC_EFFECT_ENABLE && (TCFG_MIC_EFFECT_SEL == MIC_EFFECT_REVERB)
200 copy .\effect_file\mic_effect_reverb.bin eq_cfg_hw.bin
201 #elif defined(TCFG_MIC_EFFECT_ENABLE)&& TCFG_MIC_EFFECT_ENABLE && (TCFG_MIC_EFFECT_SEL == MIC_EFFECT_ECHO) && TCFG_AUDIO_MIC_EFFECT_POST_EQ_ENABLE
202 copy .\effect_file\mic_effect_echo_full.bin eq_cfg_hw.bin
203 #elif defined(TCFG_MIC_EFFECT_ENABLE)&& TCFG_MIC_EFFECT_ENABLE && (TCFG_MIC_EFFECT_SEL == MIC_EFFECT_ECHO)
204 copy .\effect_file\mic_effect_echo.bin eq_cfg_hw.bin
205 #elif defined(TCFG_MIC_EFFECT_ENABLE)&& TCFG_MIC_EFFECT_ENABLE && (TCFG_MIC_EFFECT_SEL == MIC_EFFECT_REVERB_ECHO)
206 copy .\effect_file\mic_effect_full.bin eq_cfg_hw.bin
207 #elif defined(TCFG_MIC_EFFECT_ENABLE)&& TCFG_MIC_EFFECT_ENABLE && (TCFG_MIC_EFFECT_SEL == MIC_EFFECT_MEGAPHONE)
208 copy .\effect_file\mic_effect_megaphone.bin eq_cfg_hw.bin
209 #elif defined(SOUND_TRACK_2_P_X_CH_CONFIG) && SOUND_TRACK_2_P_X_CH_CONFIG
210 copy .\effect_file\music_2to1_2to2.bin eq_cfg_hw.bin
211 #elif defined(LINEIN_MODE_SOLE_EQ_EN) && LINEIN_MODE_SOLE_EQ_EN
212 copy .\effect_file\music_base_linein.bin eq_cfg_hw.bin
213 #else
214 copy .\effect_file\music_base.bin eq_cfg_hw.bin
215 #endif
216
217
218 #endif CONFIG_BOARD_AC695X_SOUNDICARD
```

最后执行到每个.bat 批处理文件后，会对 tool 文件夹下 eq_cfg_hw.bin 进行拷贝，作为下载进 flash 的资源文件，所以会出现替换 tool 文件夹下的 eq_cfg_hw.bin 无用的现象等。

```

C download.c 9+ > download.bat X C board_ac695x_demo_cfg.h C board_ac695x_demo.c C app_audio.c C audio_decc C mixerh C key_driver.c
cpu > br23 > tools > download > standard > download.bat
1 |echo off
2
3 |cd %~dp0
4
5 |copy ..\..\script.ver .
6 |copy ..\..\uboot.boot .
7 |copy ..\..\tone.cfg .
8 |copy ..\..\cfg_tool.bin .
9 |copy ..\..\app.bin .
10 |copy ..\..\br23loader.bin .
11 |copy ..\..\eq_cfg_hw.bin .
12 |copy ..\..\ota_all.bin .
13 |copy ..\..\ota_nor.bin .
14
15
16 |..\..\isd_download.exe -tonorflash -dev br23 -boot 0x12000 -div8 -wait 300 -uboot uboot.boot -app app.bin -res tone.cfg cfg_tool.bin eq_cfg_hw.bin -format all %1
17 |:-format all
18 |:-reboot 2500
19

```

解决方法:

根据不同的宏定义, 替换相应的 bin 文件, 比如默认的需要修改 music_base.bin

<< SDK > cpu > br23 > tools > effect_file

在 effect_file 中搜索

名称	修改日期	类型	大小
mic_effect_echo.bin	2022/2/28 16:45	FTE Binary Expo...	5 KB
mic_effect_echo_full.bin	2022/2/28 16:45	FTE Binary Expo...	6 KB
mic_effect_echo_voicechanger.bin	2022/2/28 16:45	FTE Binary Expo...	6 KB
mic_effect_full.bin	2022/2/26 15:48	FTE Binary Expo...	6 KB
mic_effect_megaphone.bin	2022/2/26 15:48	FTE Binary Expo...	3 KB
mic_effect_reverb.bin	2022/2/26 15:48	FTE Binary Expo...	5 KB
mic_effect_reverb_full.bin	2022/2/26 15:48	FTE Binary Expo...	6 KB
music_2to1_2to2.bin	2022/2/26 15:48	FTE Binary Expo...	2 KB
music_advance.bin	2022/2/26 15:48	FTE Binary Expo...	2 KB
music_base.bin	2022/2/26 15:48	FTE Binary Expo...	2 KB
music_base_linein.bin	2022/2/26 15:48	FTE Binary Expo...	2 KB

191.AC696n_soundbox_sdk_v1.2.2 做 TF 和 U 盘共用, 播放 TF 的提示音时插入 U 盘有概率不能识别 U 盘 20220907---- FS

解决方案:

dev_multiplex_api.c 文件中, mult_usb_mount_before(usb_dev usb_id)函数, 删除 if(mult_flag.active == SD_ACTIVE)判断, 如图:

```
1 dev_multiplex_api.c
316 int mult_usb_mount_before(usb_dev usb_id)
317 {
318     u32 reset_delay = 0;;
319 #if TCFG_USB_DM_MULTIPLEX_WITH_SD_DAT0
320     music_player_stop(1); //停止解码
321     /* if (mult_flag.active == SD_ACTIVE) { */
322     |     mult_sdio_suspend();
323     |     reset_delay = USB_RESET_DELAY;
324     /* } */
325     mult_usb_resume(usb_id, 0, reset_delay);
326     mult_flag.sd_status = MULT_IO_SUSPEND;
327     mult_flag.usb_status = MULT_IO_ACTIVE;
328     mult_flag.active = USB_ACTIVE;
329 #endif
330     return 0;
331 }
332
```