Partybox Audio 开发指南

文件标识：RK-KF-YF-575

发布版本：V1.0.0

日期：2024-07-24

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

本文档主要描述 Partybox Audio 相关流程，以及调试方法。

产品版本

芯片名称	内核版本
RK3308B/RK3308H	5.10

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

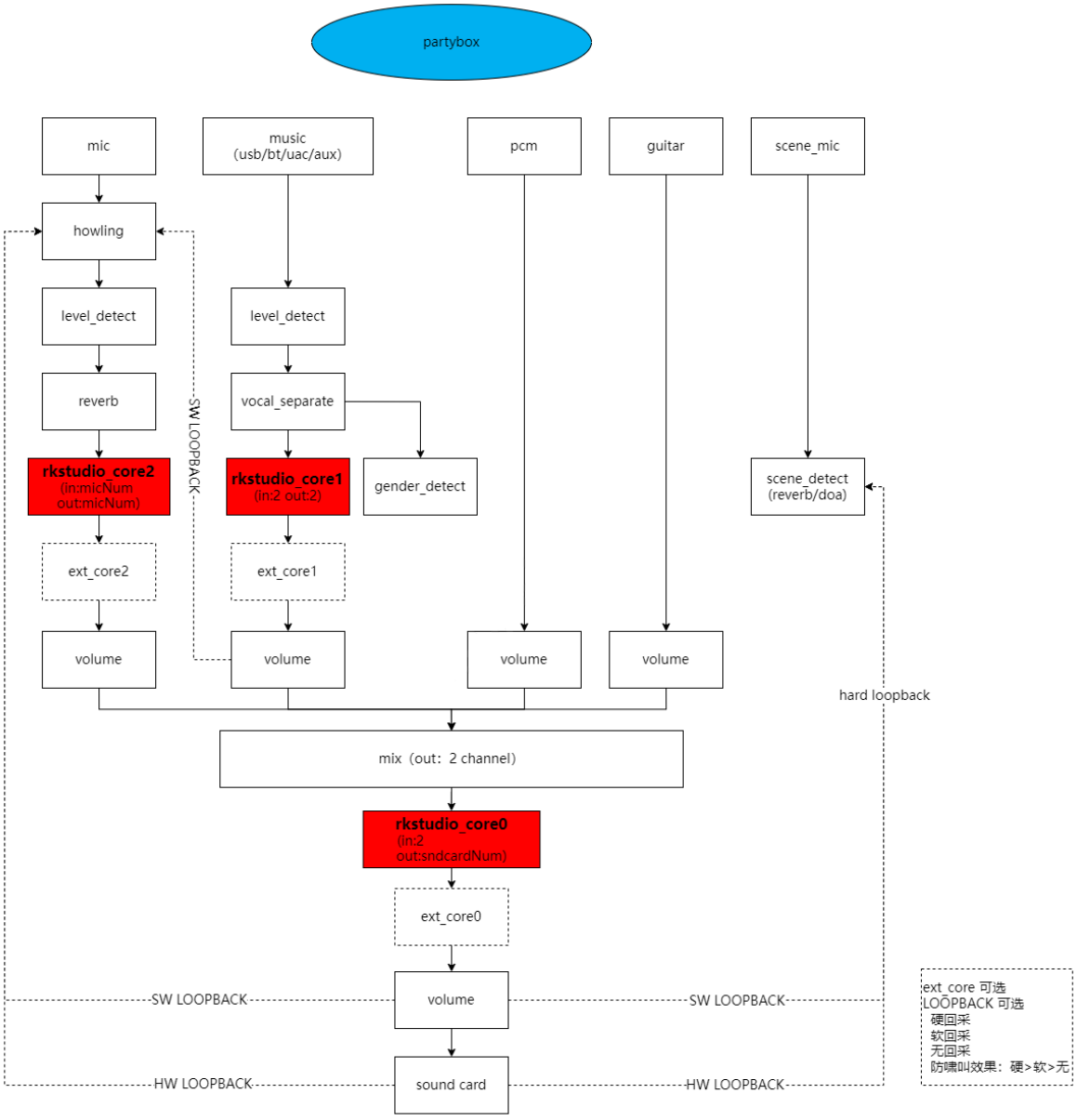
版本号	作者	修改日期	修改说明
V1.0.0	LYH	2024-07-24	初始版本

目录

1. 功能描述

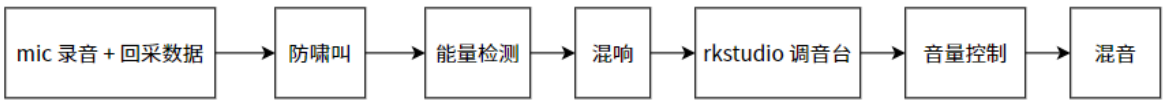
1. 支持 mic 录音。
2. 支持 mic 相关算法：包含去啸叫(howling)，加混响(reverb)， 能量检测(level detect)，音效等。
3. 支持音乐播放：U盘播放(解码)，蓝牙/UAC/AUX (PCM数据)。
4. 支持音乐相关算法：去人声，去吉他，男女声识别，能量检测等。
5. 支持 guitar 录音。
6. 支持混音输出，以及混音后音效调整。
7. 支持场景识别：室内外，左右声道等。
8. 支持外部注册音频处理算法。

2. 系统框图



2.1 系统框图说明

1. mic 通路。



a) 其中回采方式可选：

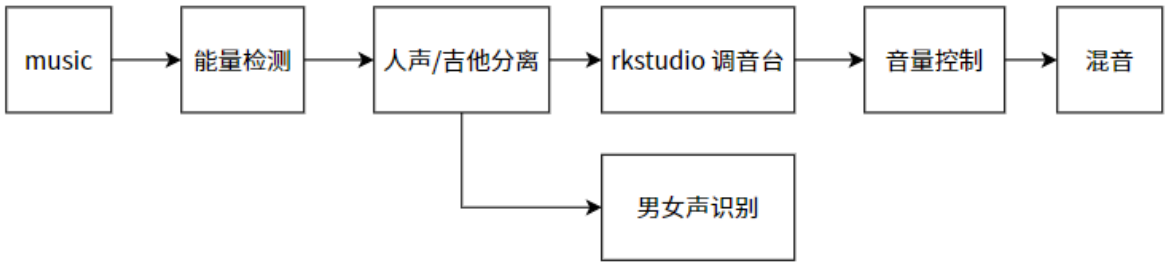
- 硬回采：功放后模拟信号回采送算法。
- 软回采1：送声卡前数字信号回采送算法。
- 软回采2：音乐混音前数字信号回采送算法。

- 无回采。

以上回采方式任选 1 种即可，回采效果：硬回采 > 软回采 > 无回采。

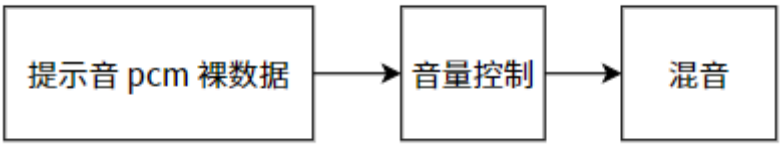
b) mic 音效在 rkstudio core2 中调整，输入输出通道输由 mic 数量决定，如： 1 个mic， rkstudio 为 1 进 1 出；2 个 mic， rkstudio 为 2 进 2 出。

2. music 通路。

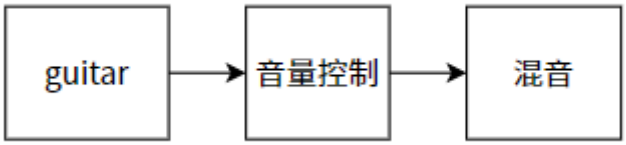


音乐音效在 rkstudio core1 中调，输入输出通道为 2 进 2 出。

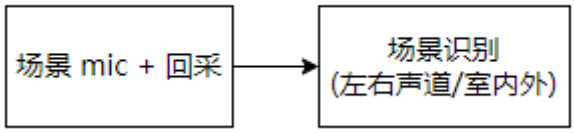
3. 提示音通路。



4. guitar 通路。



5. 场景识别通路。

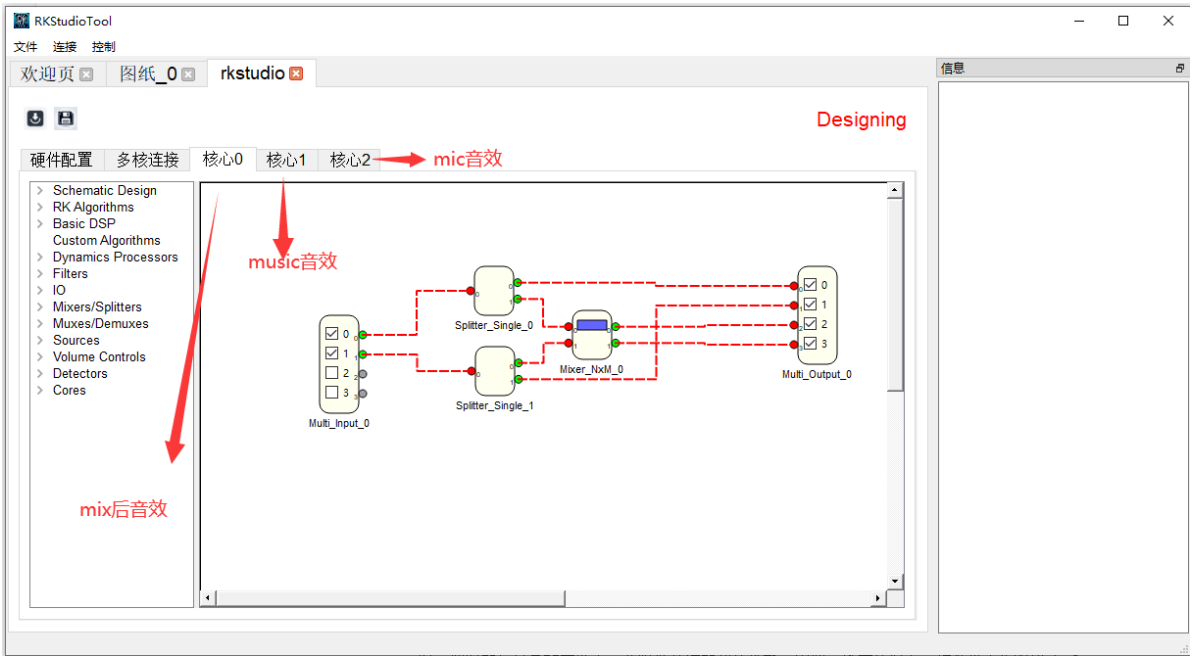


6. 混音播放通路。

混音输出的音效在 rkstudio core0 中调，输入为 2 声道，输出看实际声卡为几声道打开。

7. 外部音频处理算法可选择是否注册，目前支持在三个位置注册算法。

2.2 rkstudio调音台界面


RKStudioTool
文件 连接 控制
欢迎页 图纸_0 rkstudio
信息
Designing
硬件配置 多核连接 核心0 核心1 核心2
mic音效
Schematic Design
RK Algorithms
Basic DSP
Custom Algorithms
Dynamics Processors
Filters
IO
Mixers/Splitters
Muxes/Demuxes
Sources
Volume Controls
Detectors
Cores
music音效
mix后音效
Multi_Input_0
Splitter_Single_0
Splitter_Single_1
Mixer_NxM_0
Multi_Output_0

3. Audio解码支持格式列表

音频格式	支持情况
PCM	√
WAV	√
FLAC	√
MP3	√
WMA	√
APE	√
OPUS	√
VORBIS	√

4. API参考

4.1 rc_pb_create

【描述】
创建 partybox 实例。

【语法】
rc_s32 rc_pb_create(rc_pb_ctx *ctx, struct rc_pb_attr *attr);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输出
attr	partybox 属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

4.2 rc_pb_destroy

【描述】
销毁 partybox 实例。

【语法】
rc_s32 rc_pb_destroy(rc_pb_ctx ctx);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

4.3 rc_pb_set_volume

【描述】
设置主音量(对应混音后的音量)。

【语法】
rc_s32 rc_pb_set_volume(rc_pb_ctx ctx, rc_float volume_db);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
volume_db	音量(单位 db)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

4.4 rc_pb_get_volume

【描述】
获取主音量(对应混音后的音量)。

【语法】
rc_s32 rc_pb_get_volume(rc_pb_ctx ctx, rc_float *volume_db);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
volume_db	音量(单位 db)。	输出

【返回值】

返回值	描述
0	成功。
非0	失败。

4.5 rc_pb_set_param

【描述】
设置主参数：包含 rkstudio_core0 参数，amix 参数等。

【语法】
rc_s32 rc_pb_set_param(rc_pb_ctx ctx, struct rc_pb_param *param);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
param	主参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

4.6 rc_pb_get_param

【描述】
获取主参数。

【语法】
rc_s32 rc_pb_get_param(rc_pb_ctx ctx, struct rc_pb_param *param);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
param	主参数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败。

4.7 rc_pb_player_start

【描述】
启动播放器。

【语法】
rc_s32 rc_pb_player_start(rc_pb_ctx ctx, enum rc_pb_play_src src, struct rc_pb_player_attr *attr);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	播放器类型。	输入
attr	播放器属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

4.8 rc_pb_player_stop

【描述】
停止播放。

【语法】
rc_s32 rc_pb_player_stop(rc_pb_ctx ctx, enum rc_pb_play_src src);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	播放器类型。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

4.9 rc_pb_player_pause

【描述】
暂停播放。

【语法】
rc_s32 rc_pb_player_pause(rc_pb_ctx ctx, enum rc_pb_play_src src);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	播放器类型。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

4.10 rc_pb_player_resume

【描述】
恢复播放。

【语法】
rc_s32 rc_pb_player_resume(rc_pb_ctx ctx, enum rc_pb_play_src src);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	播放器类型。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

4.11 rc_pb_player_dequeue_frame

【描述】
获取一个可用的帧，用于 PCM 播放器播放提示音。拿到可用帧后填充 PCM 数据，之后 queue 归还，播放器内部会将该数据参与混音输出。

【语法】
rc_s32 rc_pb_player_dequeue_frame(rc_pb_ctx ctx, enum rc_pb_play_src src, struct rc_pb_frame_info *frame_info, rc_s32 ms);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	播放器类型。	输入
frame_info	音频帧信息。	输出
ms	超时时间，-1为阻塞直到获取成功。单位 ms。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

【注意】

- 仅 PCM 播放器支持(RC_PB_PLAY_SRC_PCM)。

4.12 rc_pb_player_queue_frame

【描述】
归还音频帧，播放器内部会将该数据参与混音输出。

【语法】
rc_s32 rc_pb_player_queue_frame(rc_pb_ctx ctx, enum rc_pb_play_src src, struct rc_pb_frame_info *frame_info, rc_s32 ms);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	播放器类型。	输入
frame_info	音频帧信息。	输出
ms	超时时间，-1为阻塞直到获取成功。单位 ms。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

【注意】

- 仅 PCM 播放器支持(RC_PB_PLAY_SRC_PCM)。

4.13 rc_pb_player_get_position

【描述】
获取当前播放位置。

【语法】
rc_s32 rc_pb_player_get_position(rc_pb_ctx ctx, enum rc_pb_play_src src, rc_s64 *usec);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	播放器类型。	输入
usec	当前位置。单位 us。	输出

【返回值】

返回值	描述
0	成功。
非0	失败。

【注意】

- 仅本地播放器支持(RC_PB_PLAY_SRC_LOCAL)。

4.14 rc_pb_player_get_duration

【描述】
获取当前音频总时长。

【语法】
rc_s32 rc_pb_player_get_duration(rc_pb_ctx ctx, enum rc_pb_play_src src, rc_s64 *usec);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	播放器类型。	输入
usec	总时长。单位 us。	输出

【返回值】

返回值	描述
0	成功。
非0	失败。

【注意】

- 仅本地播放器支持(RC_PB_PLAY_SRC_LOCAL)。

4.15 rc_pb_player_set_loop

【描述】
设置循环播放。

【语法】
rc_s32 rc_pb_player_set_loop(rc_pb_ctx ctx, enum rc_pb_play_src src, rc_bool loop);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	播放器类型。	输入
loop	0: 播一次。 1: 循环播放。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

【注意】

- 仅本地播放器支持(RC_PB_PLAY_SRC_LOCAL)。

4.16 rc_pb_player_seek

【描述】
跳转到指定位置播放。

【语法】
rc_s32 rc_pb_player_seek(rc_pb_ctx ctx, enum rc_pb_play_src src, rc_s64 usec);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	播放器类型。	输入
usec	跳转位置。单位 us。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

【注意】

- 仅本地播放器支持(RC_PB_PLAY_SRC_LOCAL)。

4.17 rc_pb_player_set_volume

【描述】
设置音乐音量。

【语法】
rc_s32 rc_pb_player_set_volume(rc_pb_ctx ctx, enum rc_pb_play_src src, rc_float volume_db);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	播放器类型。	输入
volume_db	音量(单位 db)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

4.18 rc_pb_player_get_volume

【描述】
获取音乐音量。

【语法】
rc_s32 rc_pb_player_get_volume(rc_pb_ctx ctx, enum rc_pb_play_src src, rc_float *volume_db);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	播放器类型。	输入
volume_db	音量(单位 db)。	输出

【返回值】

返回值	描述
0	成功。
非0	失败。

4.19 rc_pb_player_set_param

【描述】
设置播放器参数：包含 rkstudio_core1 参数，人生分离参数等。

【语法】
rc_s32 rc_pb_player_set_param(rc_pb_ctx ctx, enum rc_pb_play_src src, struct rc_pb_param *param);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	播放器类型。	输入
param	播放器参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

4.20 rc_pb_player_get_param

【描述】
获取播放器参数。

【语法】
rc_s32 rc_pb_player_get_param(rc_pb_ctx ctx, enum rc_pb_play_src src, struct rc_pb_param *param);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	播放器类型。	输入
param	播放器参数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败。

4.21 rc_pb_player_get_energy

【描述】
获取音乐能量。

【语法】
rc_s32 rc_pb_player_get_energy(rc_pb_ctx ctx, enum rc_pb_play_src src, struct rc_pb_energy *energy);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	播放器类型。	输入
energy	能量。	输出

【返回值】

返回值	描述
0	成功。
非0	失败。

【注意】

- PCM 播放器不支持 (RC_PB_PLAY_SRC_PCM)。

4.22 rc_pb_player_release_energy

【描述】
释放音乐能量。

【语法】
rc_s32 rc_pb_player_release_energy(rc_pb_ctx ctx, enum rc_pb_play_src src, struct rc_pb_energy *energy);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	播放器类型。	输入
energy	能量。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

【注意】

- PCM 播放器不支持 (RC_PB_PLAY_SRC_PCM)。

4.23 rc_pb_recorder_start

【描述】
启动录音。

【语法】
rc_s32 rc_pb_recorder_start(rc_pb_ctx ctx);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

4.24 rc_pb_recorder_stop

【描述】
停止录音。

【语法】
rc_s32 rc_pb_recorder_stop(rc_pb_ctx ctx);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

4.25 rc_pb_recorder_mute

【描述】
控制录音是否静音 (默认不静音)。

【语法】
rc_s32 rc_pb_recorder_mute(rc_pb_ctx ctx, enum rc_pb_rec_src src, rc_s32 idx, rc_bool mute);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	录音源类型。	输入
idx	通道号。如两个 mic，支持分别控制 mute。	输入
mute	0: 关闭静音。 1: 开启静音。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

4.26 rc_pb_recorder_set_volume

【描述】
设置录音音量。

【语法】
rc_s32 rc_pb_recorder_set_volume(rc_pb_ctx ctx, enum rc_pb_rec_src src, rc_s32 idx, rc_float volume_db);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	录音源类型。	输入
idx	通道号。如两个 mic，支持分别控制音量。	输入
volume_db	音量(单位 db)。	输出

【返回值】

返回值	描述
0	成功。
非0	失败。

4.27 rc_pb_recorder_get_volume

【描述】
获取录音音量。

【语法】
rc_s32 rc_pb_recorder_get_volume(rc_pb_ctx ctx, enum rc_pb_rec_src src, rc_s32 idx, rc_float *volume_db);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	录音源类型。	输入
idx	通道号。如两个 mic，支持分别控制音量。	输入
volume_db	音量(单位 db)。	输出

【返回值】

返回值	描述
0	成功。
非0	失败。

4.28 rc_pb_recorder_set_param

【描述】
设置录音参数：包含 rkstudio_core2 参数，防啸叫参数，混响参数等。

【语法】
rc_s32 rc_pb_recorder_set_param(rc_pb_ctx ctx, enum rc_pb_rec_src src, rc_s32 idx, struct rc_pb_param *param);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	录音源类型。	输入
idx	通道号。	输入
param	录音参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

4.29 rc_pb_recorder_get_param

【描述】
获取录音参数。

【语法】
rc_s32 rc_pb_recorder_get_param(rc_pb_ctx ctx, enum rc_pb_rec_src src, rc_s32 idx, struct rc_pb_param *param);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	录音源类型。	输入
idx	通道号。	输入
param	录音参数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败。

4.30 rc_pb_recorder_get_energy

【描述】
获取录音能量。

【语法】
rc_s32 rc_pb_recorder_get_energy(rc_pb_ctx ctx, enum rc_pb_rec_src src, rc_s32 idx, struct rc_pb_energy *energy);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	录音源类型。	输入
idx	通道号。	输入
energy	能量。	输出

【返回值】

返回值	描述
0	成功。
非0	失败。

4.31 rc_pb_recorder_release_energy

【描述】
释放录音能量。

【语法】
rc_s32 rc_pb_recorder_release_energy(rc_pb_ctx ctx, enum rc_pb_rec_src src, rc_s32 idx, struct rc_pb_energy *energy);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	录音源类型。	输入
idx	通道号。	输入
energy	能量。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

4.32 rc_pb_recorder_dequeue_frame

【描述】
获取录音帧，获取的是混音前的数据，经过了 howling，reverb， rkstudio 等算法。

【语法】
rc_s32 rc_pb_recorder_dequeue_frame(rc_pb_ctx ctx, enum rc_pb_rec_src src, struct rc_pb_frame_info *frame_info, rc_s32 ms);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	录音源类型。	输入
frame_info	音频帧信息。	输出
ms	超时时间，-1为阻塞直到获取成功。单位 ms。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

【注意】

- 仅支持 mic 录音源 (RC_PB_REC_SRC_MIC)。

4.33 rc_pb_recorder_queue_frame

【描述】
归还录音帧，使新的录音帧数据可以填充到这里。

【语法】
rc_s32 rc_pb_recorder_queue_frame(rc_pb_ctx ctx, enum rc_pb_rec_src src, struct rc_pb_frame_info *frame_info, rc_s32 ms);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
src	录音源类型。	输入
frame_info	音频帧信息。	输入
ms	超时时间，-1为阻塞直到获取成功。单位 ms。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

【注意】

- 如果应用归还不及时，将会导致无可用 buf 去保存新的录音数据，导致断音。
- 录音可用帧个数配置由启动参数 struct rc_pb_recorder_attr 中 pool_cnt 指定。
- 获取录音帧内部会开启一个线程处理。如果没有这种需求，将 pool_cnt 配置为 0， 减少CPU消耗。

4.34 rc_pb_scene_detect_start

【描述】
启动场景识别，包含左右声道识别，室内外识别等，具体算法可由参数配置。

【语法】
rc_s32 rc_pb_scene_detect_start(rc_pb_ctx ctx, struct rc_pb_scene_detect_attr *attr);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
attr	场景识别属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

4.35 rc_pb_scene_detect_stop

【描述】
停止场景识别。

【语法】
rc_s32 rc_pb_scene_detect_stop(rc_pb_ctx ctx);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

4.36 rc_pb_scene_get_result

【描述】
获取场景识别结果。

【语法】
rc_s32 rc_pb_scene_get_result(rc_pb_ctx ctx, enum rc_pb_scene_detect_mode mode, rc_float *result);

【参数】

参数名	描述	输入/输出
ctx	partybox 上下文。	输入
mode	场景识别模式。	输入
result	场景识别结果。	输出

【返回值】

返回值	描述
0	成功。
非0	失败。

4.37 rc_pb_register_filter

【描述】
注册第三方音频处理插件。

【语法】
rc_s32 rc_pb_register_filter(enum rc_pb_filter_pos pos, struct rc_pb_filter *filter, void *arg);

【参数】

参数名	描述	输入/输出
pos	插件位置。	输入
filter	插件回调。	输入
arg	回调参数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

【注意】

- 用于客户不使用 rkstudio 调音或想加入一些第三方算法场景。
- 需要在 rc_pb_create 前注册才能生效。

4.38 rc_pb_unregister_filter

【描述】
注销音频处理插件。

【语法】
rc_s32 rc_pb_unregister_filter(enum rc_pb_filter_pos pos);

【参数】

参数名	描述	输入/输出
pos	插件位置。	输入

【返回值】

返回值	描述
0	成功。
非0	失败。

【注意】

- 需要在 rc_pb_destroy 后销毁。

5. 数据类型

5.1 rc_pb_attr

【说明】
partybox 属性。

【定义】

```
struct rc_pb_attr {
    char                        *card_name;
    rc_u32                       sample_rate;
    rc_u32                       channels;
    rc_u32                       bit_width;
    notifyfun_t                  notify;
    void                        *opaque;
    rc_float                     volume_db;
    struct rc_pb_recorder_attr *record_attr;
};
```

【成员】

成员名称	描述
card_name	播放声卡名。
sample_rate	播放采样率。
channels	播放声道。
bit_width	播放数据位宽。
notify	通知回调函数指针。
opaque	传入回调的指针。
volume_db	主音量(单位 db)。
record_attr	录音属性。

【注意】

- record_attr 设置 NULL， 不开启录音，只有播放功能。
- 回调参数说明

```
typedef void (*notifyfun_t)(enum rc_pb_event event, rc_s32 cmd, void
*opaque);
event：回调事件 （播放事件，语音事件等）。
cmd：命令，暂为语音命令。
opaque：传入回调的指针。
```

5.2 rc_pb_player_attr

【说明】
播放器属性。

【定义】

```
struct rc_pb_player_attr {
    const char *url;
    const char *headers;
    char       *card_name;
    rc_u32      sample_rate;
    rc_u32      channels;
    rc_u32      bit_width;
    rc_u32      valid_bit_width;
    rc_u32      valid_start_bit;
    rc_u32      pool_size;
    rc_u32      pool_cnt;
    rc_u32      mute_ms;
    rc_bool     basic;
    struct rc_pb_param_level_detect detect;
};
```

【成员】

成员名称	描述
url	片源地址(本地播放使用)。
headers	地址头信息(本地播放使用)，暂为启用，全配 NULL。
card_name	声卡名 (声卡播放使用)。
sample_rate	采样率 PCM 提示音 (声卡播放使用)。
channels	声道数 (声卡播放使用)。
bit_width	位宽 (声卡播放使用)。
valid_bit_width	有效数据位宽 (声卡播放使用)。
valid_start_bit	有效数据起始位 (声卡播放使用)。
pool_size	内存池单块内存大小 (PCM 播放使用)。
pool_cnt	内存池内存个数 (PCM 播放使用)。
mute_ms	mute 声卡启动后前几 ms 的数据。
basic	基础播放。
detect	能量检测属性。

【注意】

- 用不到的属性配置 NULL/0。如本地播放时，声卡信息配 NULL/0。

- basic 为 rc_true 时代表只要基础播放，无高级算法(如人声分离算法)，适用于 TWS/BIS 等多台机器同步的场景。

5.3 rc_pb_play_src

【说明】
播放器播放源。

【定义】

```
enum rc_pb_play_src {
    RC_PB_PLAY_SRC_LOCAL = 0,
    RC_PB_PLAY_SRC_BT,
    RC_PB_PLAY_SRC_UAC,
    RC_PB_PLAY_SRC_PCM,
    RC_PB_PLAY_SRC_BUTT
};
```

【成员】

成员名称	描述
RC_PB_PLAY_SRC_LOCAL	本地播放 (需解码)。
RC_PB_PLAY_SRC_BT	蓝牙 (蓝牙声卡)。
RC_PB_PLAY_SRC_UAC	UAC (UAC声卡)。
RC_PB_PLAY_SRC_PCM	PCM提示音 (PCM裸数据)。

5.4 rc_pb_param

【说明】
partybox 动态更新参数。

【定义】

```
struct rc_pb_param {
    enum rc_pb_param_type type;
    union {
        struct rc_pb_param_howling        howling;
        struct rc_pb_param_reverb         reverb;
        struct rc_pb_param_vocal_separate vocal;
        struct rc_pb_param_amix           amix;
        struct rc_pb_param_rkstudio       rkstudio;
        struct rc_pb_param_scene_detect   scene;
    };
};
```

【成员】

成员名称	描述
type	参数类型。
howling	防啸叫参数。
reverb	混响参数。
vocal	分离参数。
amix	amix控制参数。
rkstudio	rkstudio参数。
scene	场景识别参数。

【注意】

- 参数为联合体，每次只能更新一种参数，参数类型由 type 指定。

5.5 rc_pb_param_type

【说明】
partybox 参数类型。

【定义】

```
enum rc_pb_param_type {
    RC_PB_PARAM_TYPE_3A = 0,
    RC_PB_PARAM_TYPE_REVERB,
    RC_PB_PARAM_TYPE_VOLCAL_SEPARATE,
    RC_PB_PARAM_TYPE_AMIX,
    RC_PB_PARAM_TYPE_RKSTUDIO,
    RC_PB_PARAM_TYPE_SCENE,
    RC_PB_PARAM_TYPE_BUTT
};
```

【成员】

成员名称	描述
RC_PB_PARAM_TYPE_3A	防啸叫。
RC_PB_PARAM_TYPE_REVERB	混响。
RC_PB_PARAM_TYPE_VOLCAL_SEPARATE	分离。
RC_PB_PARAM_TYPE_AMIX	amix控制。
RC_PB_PARAM_TYPE_RKSTUDIO	rkstudio。
RC_PB_PARAM_TYPE_SCENE	场景识别。

5.6 rc_pb_param_howling

【说明】
防啸叫参数。

【定义】

```
struct rc_pb_param_howling {
    rc_bool bypass;
};
```

【成员】

成员名称	描述
bypass	0: 开启。 1: 关闭。

5.7 rc_pb_param_reverb

【说明】
混响参数。

【定义】

```
struct rc_pb_param_reverb {
    rc_bool bypass;
    enum rc_pb_reverb_mode mode;
    rc_s32 dry_level;
    rc_s32 wet_level;
};
```

【成员】

成员名称	描述
bypass	0: 开启。 1: 关闭。
mode	混响模式。
dry_level	干信号等级。 取值范围：[0, 100]。
wet_level	湿信号等级。 取值范围：[0, 100]。

【注意】

- mode 为 RC_PB_REVERB_MODE_USER 时， dry / wet 才有用，可用户自定义。

5.8 rc_pb_reverb_mode

【说明】
混响模式。

【定义】

```
enum rc_pb_reverb_mode {
    RC_PB_REVERB_MODE_USER = 0,
    RC_PB_REVERB_MODE_STUDIO,
    RC_PB_REVERB_MODE_KTV,
    RC_PB_REVERB_MODE_CONCERT,
    RC_PB_REVERB_MODE_ECHO,
    RC_PB_REVERB_MODE_BUTT
};
```

【成员】

成员名称	描述
RC_PB_REVERB_MODE_USER	自定义。
RC_PB_REVERB_MODE_STUDIO	录音室。
RC_PB_REVERB_MODE_KTV	KTV 模式。
RC_PB_REVERB_MODE_CONCERT	音乐会模式。
RC_PB_REVERB_MODE_ECHO	回声模式。

5.9 rc_pb_param_vocal_separate

【说明】
分离参数。

【定义】

```
struct rc_pb_param_vocal_separate {
    rc_bool bypass;
    rc_u32  human_level;         /* RW; Range: [0, 100]; */
    rc_u32  other_level;         /* RW; Range: [0, 100]; */
    rc_u32  reserve_level[32];   /* RW; Range: [0, 100]; */
    const char *lib_name;
};
```

【成员】

成员名称	描述
bypass	0: 开启。 1: 关闭。
human_level	目标保留比例。 取值范围：[0, 100]。
other_level	其他保留比例。 取值范围：[0, 100]。
reserve_level	预留参数。
lib_name	库名。

【成员】

- RK3308 吉他分离和人声分离是分别的库，不支持同时消除，需动态切换。切换配置库名:

```
吉他分离
param.vocal.lib_name = "librkaudio_effect_guitar.so";
人声分离
param.vocal.lib_name = "librkaudio_effect_vocal.so";
```

- 只调整比例时，lib_name 设置 NULL，此时操作的库是最后一次配置的库名。
- 人声分离时，human_level 为保留人声比例，other_level 为除去人声的其他声音比例。
- 吉他分离时，human_level 为保留吉他比例，other_level 为除去吉他的其他声音比例。

5.10 rc_pb_param_amix

【说明】
amix控制参数。

【定义】

```
struct rc_pb_param_amix {
    rc_u32      card;
    const char *control;
    const char *values;
};
```

【成员】

成员名称	描述
card	操作的声卡序号。
control	控制节点。
values	控制值。

举例：设置PPM

```
param.amix.card = 0;
param.amix.control = "PCM Clk Compensation In PPM";
param.amix.values = 1000;
```

5.11 rc_pb_param_rkstudio

【说明】
rkstudio 参数。

【定义】

```
struct rc_pb_param_rkstudio {
    rc_bool bypass;
    const char *uri;
    enum rc_pb_rkstudio_cmd cmd;
    rc_u32    id;
    rc_u32    addr;
    rc_float *data;
    rc_u32    cnt;
};
```

【成员】

成员名称	描述
bypass	0: 开启。 1: 关闭。
uri	配置 bin 文件路径。
cmd	命令。
id	core id。 对应混音后core 0, 音乐 core 1, 麦克风 core2。
addr	空间参数地址，rkstudio保存时，会同步生成一份记录参数地址的头文件。
data	参数数组地址，一共 cnt 个参数。
cnt	参数个数。

【注意】

- 如果要整张图全部更新，需要配置 uri，同时 data 置 NULL。
- 不能随意 bypass ，如 mix 后 rkstudio_core0 中实现了 2 进 4 出功能，送声卡为 4 声道数据，如果 bypass，数据转换没人做，送给声卡的数据会异常。

5.12 rc_pb_rkstudio_cmd

【说明】
rkstudio 命令。

【定义】

```
enum rc_pb_rkstudio_cmd {
    RC_PB_RKSTUDIO_CMD_DOWNLOAD_GRAPH = 0,
    RC_PB_RKSTUDIO_CMD_SET_PARAM,
    RC_PB_RKSTUDIO_CMD_GET_PARAM,
    RC_PB_RKSTUDIO_CMD_BUTT
};
```

【成员】

成员名称	描述
RC_PB_RKSTUDIO_CMD_DOWNLOAD_GRAPH	下载图。
RC_PB_RKSTUDIO_CMD_SET_PARAM	设置参数。
RC_PB_RKSTUDIO_CMD_GET_PARAM	读取参数。

5.13 rc_pb_param_scene_detect

【说明】
场景识别参数。

【定义】

```
struct rc_pb_param_scene_detect {
    rc_bool bypass;
    enum rc_pb_scene_detect_mode scene_mode;
    rc_float result;
};
```

【成员】

成员名称	描述
bypass	0: 开启。 1: 关闭。
scene_mode	模式。
result	结果。

【注意】

- DOA : result 为角度，0 - 180。
- REVERB: result 0 室内，2 室外。
- GENDER: result 0 无法识别，1 男生，2 女生。

5.14 rc_pb_event

【说明】
partybox 事件，回调中会指明事件类型。

【定义】

```
enum rc_pb_event {
    RC_PB_EVENT_PLAYBACK_ERROR = 0,
    RC_PB_EVENT_PLAYBACK_COMPLETE,
    RC_PB_EVENT_AWAKEN,
    RC_PB_EVENT_BUTT
};
```

【成员】

成员名称	描述
RC_PB_EVENT_PLAYBACK_ERROR	音频播放错误。
RC_PB_EVENT_PLAYBACK_COMPLETE	音频播放完成。
RC_PB_EVENT_AWAKEN	语音唤醒。

【注意】

- 播放事件只有在 RC_PB_PLAY_SRC_LOCAL 模式会上报。
- 语音唤醒暂不支持。

5.15 rc_pb_wake_up_cmd

【说明】
partybox 语音唤醒命令。

【定义】

```
enum rc_pb_wake_up_cmd {
    RC_PB_WAKE_UP_CMD_START_PLAYER = 1,
    RC_PB_WAKE_UP_CMD_PAUSE_PLARER,
    RC_PB_WAKE_UP_CMD_STOP_PLARER,
    RC_PB_WAKE_UP_CMD_PREV,
    RC_PB_WAKE_UP_CMD_NEXT,
    RC_PB_WAKE_UP_CMD_VOLUME_UP,
    RC_PB_WAKE_UP_CMD_VOLUME_DOWN,
    RC_PB_WAKE_UP_CMD_ORIGINAL_SINGER_OPEN,
    RC_PB_WAKE_UP_CMD_ORIGINAL_SINGER_CLOSE,

    RC_PB_WAKE_UP_CMD_RECIEVE = 100,
    RC_PB_WAKE_UP_CMD_RECIEVE_BUT_NO_TASK,

    RC_PB_WAKE_UP_CMD_BUTT
};
```

【成员】

成员名称	描述
RC_PB_WAKE_UP_CMD_START_PLAYER	唤醒词：播放歌曲。
RC_PB_WAKE_UP_CMD_PAUSE_PLARER	唤醒词：暂停播放。
RC_PB_WAKE_UP_CMD_STOP_PLARER	唤醒词：停止原唱。
RC_PB_WAKE_UP_CMD_PREV	唤醒词：上一首。
RC_PB_WAKE_UP_CMD_NEXT	唤醒词：下一首。
RC_PB_WAKE_UP_CMD_VOLUME_UP	唤醒词：声音大点。
RC_PB_WAKE_UP_CMD_VOLUME_DOWN	唤醒词：声音小点。
RC_PB_WAKE_UP_CMD_ORIGINAL_SINGER_OPEN	唤醒词：打开原唱。
RC_PB_WAKE_UP_CMD_ORIGINAL_SINGER_CLOSE	唤醒词：关闭原唱。
RC_PB_WAKE_UP_CMD_RECIEVE	接收到唤醒词。 唤醒词：小瑞小瑞。
RC_PB_WAKE_UP_CMD_RECIEVE_BUT_NO_TASK	接收到唤醒词，但没有下一条指令。

【注意】

- 语音唤醒暂不支持。

5.16 rc_pb_param_level_detect

【说明】
能量检测属性。

【定义】

```
struct rc_pb_param_level_detect {
    rc_u32 rms_tc;
    rc_u32 hold_time;
    rc_u32 decay_time;
    rc_u32 detect_per_frm;  /* RW; default 10 */
    rc_u32 band_cnt;
};
```

【成员】

成员名称	描述
rms_tc	信号变化的响应速度。 取值范围：[0, 1000]。 默认值 200，越大变化越慢。
hold_time	包络下降前保持当前峰值的持续时间。 取值范围：[0, 1000]。 默认值 0，越大变化越慢。
decay_time	包络下降速度。 取值范围：[0, 1000]。 默认值 200，越大变化越慢。
detect_per_frm	每几帧检测一次。
band_cnt	频带个数。

【注意】

- detect_per_frm，对于 mic 来说一帧 2.66ms， 对于 music，一帧 16ms。

5.17 rc_pb_energy

【说明】
能量值。

【定义】

```
struct rc_pb_energy {
    rc_float    *energy_vec;
    rc_pb_frame  frame;
};
```

【成员】

成员名称	描述
energy_vec	能量，包含频带以及对应的能量。
frame	音频帧。

【注意】

- energy_vec 一维数组，两两一组，n 存储频带，n+1 存储能量。举例：

```
for (rc_s32 i = 0; i < detect.band_cnt; i++) {
    RK_LOGD("freq[%5.0f]HZ energy[%5.0f]DB",
            energy.energy_vec[i],
            energy.energy_vec[detect.band_cnt + i]);
}
```

5.18 rc_pb_frame_info

【说明】
音频帧信息。

【定义】

```
struct rc_pb_frame_info {
    rc_pb_frame frame;
    void       *data;
    rc_u32      size;
    rc_u32      sample_rate;
    rc_u32      channels;
    rc_u32      bit_width;
    rc_u32      seq;
    rc_u64      time_stamp;
};
```

【成员】

成员名称	描述
frame	音频帧。
data	音频数据虚拟起始地址。
size	音频大小(单位byte)。
sample_rate	采样率。
channels	通道数。
bit_width	位宽。
seq	序号。
time_stamp	时间戳。

5.19 rc_pb_recorder_attr

【说明】
partybox 录音属性。

【定义】

```
struct rc_pb_recorder_attr {
    char   *card_name;
    rc_u32  sample_rate;
    rc_u32  channels;
    rc_u32  bit_width;
    rc_u32  chn_layout;
    rc_u32  ref_layout;
    rc_u32  rec_layout;
```

```
    rc_u32  pool_cnt;  /* RW; 0 means no need get frame, can reduce cpu
usage */
    rc_u32  mute_ms;
    struct rc_pb_param_level_detect detect;
    enum rc_pb_ref_mode ref_mode;
    struct rc_pb_recorder_gt_attr *guitar;
    struct rc_pb_recorder_ref_ind_attr *ref;
};
```

【成员】

成员名称	描述
card_name	录音声卡名。
sample_rate	录音采样率。
channels	录音声道。
bit_width	录音数据位宽。
chn_layout	有效录音通道布局。
ref_layout	回采通道布局。
rec_layout	麦克通道布局。
pool_cnt	内存池 buf 总个数。
mute_ms	mute 声卡启动后前几 ms 的数据。
detect	能量检测。
ref_mode	回采模式。
guitar	吉他属性。
ref	回采属性。

【注意】

- layout 声道布局，每一个 bit 位表示一个声道。如 8 声道 (0 - 7)，其中 0 2 3 4 5 6 7 有效，1 无效，二进制表示 0b'11111101， 也就是 0xfd。
- pool_cnt 指定录音后数据经过算法处理后，保存在内存池中的 buf 个数，如 UAC 录音功能使用，此功能内部会开启一个线程处理。如果没有这种需求，pool_cnt 配置为 0， 减少 CPU 消耗。
- ref_mode 如果是 RC_PB_REF_MODE_HARD_IND，表示回采是单独的声卡，需要配置 ref 属性，其余情况 ref 设置 NULL。
- guitar，如果没有吉他，设置 NULL。

5.20 rc_pb_rec_src

【说明】
录音源。

【定义】

```
enum rc_pb_rec_src {
    RC_PB_REC_SRC_MIC = 0,
    RC_PB_REC_SRC_GUITAR,
    RC_PB_REC_SRC_BUTT
};
```

【成员】

成员名称	描述
RC_PB_REC_SRC_MIC	麦克风。
RC_PB_REC_SRC_GUITAR	吉他。

5.21 rc_pb_ref_mode

【说明】
回采模式。

【定义】

```
enum rc_pb_ref_mode {
    RC_PB_REF_MODE_NONE = 0,
    RC_PB_REF_MODE_SOFT,
    RC_PB_REF_MODE_HARD_COMBO,
    RC_PB_REF_MODE_HARD_IND
};
```

【成员】

成员名称	描述
RC_PB_REF_MODE_NONE	无回采。
RC_PB_REF_MODE_SOFT	软回采。
RC_PB_REF_MODE_HARD_COMBO	硬回采，回采 + mic 共用一个声卡。
RC_PB_REF_MODE_HARD_IND	硬回采，回采独立声卡。

5.22 rc_pb_recorder_ref_ind_attr

【说明】
回采独立声卡属性。

【定义】

```
struct rc_pb_recorder_ref_ind_attr {
    char   *card_name;
    rc_u32  sample_rate;
    rc_u32  channels;
    rc_u32  bit_width;
    rc_u32  mute_ms;
};
```

【成员】

成员名称	描述
card_name	回采声卡名。
sample_rate	回采采样率。
channels	回采声道。
bit_width	回采数据位宽。
mute_ms	mute 声卡启动后前几 ms 的数据。

5.23 rc_pb_recorder_gt_attr

【说明】
吉他声卡属性。

【定义】

```
struct rc_pb_recorder_gt_attr {
    enum rc_pb_gt_card_type type;
    union {
        struct rc_pb_gt_attr_ind    independent;
        struct rc_pb_gt_attr_combo  combo;
    };
};
```

【成员】

成员名称	描述
type	声卡类型。
independent	独立声卡属性。
combo	共用声卡属性。

【注意】

- 参数为联合体，参数类型由 type 指定。

5.24 rc_pb_gt_card_type

【说明】
吉他类型。

【定义】

```
enum rc_pb_gt_card_type {
    RC_PB_GUITAR_CARD_TYPE_IND = 0,
    RC_PB_GUITAR_CARD_TYPE_COMBO,
    RC_PB_GUITAR_CARD_TYPE_BUTT
};
```

【成员】

成员名称	描述
RC_PB_GUITAR_CARD_TYPE_IND	guitar 独立声卡读取。
RC_PB_GUITAR_CARD_TYPE_COMBO	guitar 和 mic 共用一个声卡。

5.25 rc_pb_gt_attr_ind

【说明】
吉他独立声卡属性。

【定义】

```
struct rc_pb_gt_attr_ind {
    char   *card_name;
    rc_u32  sample_rate;
    rc_u32  channels;
    rc_u32  bit_width;
    rc_u32  mute_ms;
};
```

【成员】

成员名称	描述
card_name	吉他声卡名。
sample_rate	吉他采样率。
channels	吉他声道。
bit_width	吉他数据位宽。
mute_ms	mute 声卡启动后前几 ms 的数据。

5.26 rc_pb_gt_attr_combo

【说明】
吉他共用声卡属性。

【定义】

```
struct rc_pb_gt_attr_combo {
    rc_u32  channel_status[PB_MAX_AUDIO_CHN_NUM];  /* RW; 0:not guitar 1:guitar */
};
```

【成员】

成员名称	描述
channel_status	通道状态，1 代表是guitar。

【注意】

- 如 8 声道数据(0 - 7)，其中 0 是吉他，需配置：

```
channel_status[PB_MAX_AUDIO_CHN_NUM] = {1, 0, 0, 0, 0, 0, 0, 0};
```

- partybox 内部会先分离 guitar 、 mic、ref，之后将 mic，ref 送入防啸叫算法，所以配置 mic，ref 的 layout 参数时，当做 guitar 不存在。
- 举例 guitar，mic 是同一个声卡，0 是 guitar，12 是 mic。ref 是独立声卡， 4 声道。该如何配？
    1. 由于会先分离走 guitar，所以 mic 和 ref 认为是单独的，这样 guitar分离走后， 剩余数据中 01 变成了 mic。
    2. mic 声卡要多开 4 声道，预留给回采，回采数据将会从另一个声卡搬运到预留的通道处，此例子中我们预留 2345 给回采。
    3. 实际用到了 1 个 guitar，2 个 mic，4 个回采，一共 7 个声道，但是**驱动不支持奇数声道**打开声卡，所以按 8 声道打开。

```
    static struct rc_pb_recorder_gt_attr guitar_attr;
    rc_u32 channel_status[PB_MAX_AUDIO_CHN_NUM] = {1, 0, 0, 0, 0, 0, 0, 0};
    guitar_attr.type = RC_PB_GUITAR_CARD_TYPE_COMBO;
    memcpy(guitar_attr.combo.channel_status, channel_status, sizeof(channel_status));

    static struct rc_pb_recorder_ref_ind_attr ref_ind_attr;
    ref_ind_attr.card_name = "hw:2,0";
    ref_ind_attr.sample_rate = 48000;
    ref_ind_attr.channels    = 4;
    ref_ind_attr.bit_width   = 16;

    static struct rc_pb_recorder_attr recorder_attr;
    recorder_attr.card_name = "hw:0,0";
    recorder_attr.sample_rate = 44100;
    recorder_attr.channels    = 8;
    recorder_attr.bit_width   = 16;
    recorder_attr.chn_layout  = 0x3f;
    recorder_attr.ref_layout  = 0x3c;
```

```
    recorder_attr.rec_layout  = 0x03;
    recorder_attr.pool_cnt    = 0;
    recorder_attr.ref_mode    = RC_PB_REF_MODE_HARD_IND;
    recorder_attr.detect      = detect;
    recorder_attr.guitar      = &guitar_attr;
    recorder_attr.ref         = &ref_ind_attr;
```

5.27 rc_pb_scene_detect_attr

【说明】
场景识别属性。

【定义】

```
struct rc_pb_scene_detect_attr {
    char   *card_name;
    rc_u32  sample_rate;
    rc_u32  channels;
    rc_u32  bit_width;
    rc_u32  ref_layout;
    rc_u32  rec_layout;
    rc_u32  mute_ms;
    enum rc_pb_ref_mode ref_mode;
    enum rc_pb_scene_detect_mode scene_mode;
};
```

【成员】

成员名称	描述
card_name	场景识别声卡名。
sample_rate	场景识别采样率。
channels	场景识别声道。
bit_width	场景识别数据位宽。
ref_layout	回采通道布局。
rec_layout	麦克通道布局。
mute_ms	mute 声卡启动后前几 ms 的数据。
ref_mode	回采模式。
scene_mode	场景识别模式。

5.28 rc_pb_scene_detect_mode

【说明】
场景识别模式。

【定义】

```
enum rc_pb_scene_detect_mode {
    RC_PB_SCENE_MODE_DOA    = 1 << 0,
    RC_PB_SCENE_MODE_REVERB = 1 << 1,
    RC_PB_SCENE_MODE_GENDER = 1 << 2
};
```

【成员】

成员名称	描述
RC_PB_SCENE_MODE_DOA	到达角度，用于左右声道识别。
RC_PB_SCENE_MODE_REVERB	混响，用于室内外识别。
RC_PB_SCENE_MODE_GENDER	男女声识别。

5.29 rc_pb_filter_pos

【说明】
插件位置。

【定义】

```
enum rc_pb_filter_pos {
    RC_PB_FILTER_POS_AFTER_CORE0 = 0,
    RC_PB_FILTER_POS_AFTER_CORE1,
    RC_PB_FILTER_POS_AFTER_CORE2,
    RC_PB_FILTER_POS_BUTT
};
```

【成员】

成员名称	描述
RC_PB_FILTER_POS_AFTER_CORE0	rkstudio_core0之后。
RC_PB_FILTER_POS_AFTER_CORE1	rkstudio_core1之后。
RC_PB_FILTER_POS_AFTER_CORE2	rkstudio_core0之后。

5.30 rc_pb_filter

【说明】
插件回调。

【定义】

```
struct rc_pb_filter {
    rc_s32 (*open)(struct rc_pb_filter_attr *attr, void *filter);
    rc_s32 (*process)(void *filter, struct rc_pb_filter_param *param);
    rc_s32 (*flush)(void *filter);
    rc_s32 (*close)(void *filter);
};
```

【成员】

成员名称	描述
open	打开插件回调。
process	处理回调，每次处理 frame_cnt 帧数据。
flush	清楚缓存回调，close前调用。
close	关闭插件回调。

【注意】

- flush 可以不注册，其他回调需注册 。
- void * filter, filter 就是注册插件时的 arg 参数，回调时将 arg 传入。

5.31 rc_pb_filter_attr

【说明】
插件属性。

【定义】

```
struct rc_pb_filter_attr {
    rc_u32 in_bit_width;
    rc_u32 in_sample_rate;
    rc_u32 in_channels;
    rc_u32 out_bit_width;
    rc_u32 out_sample_rate;
    rc_u32 out_channels;
    rc_u32 frame_cnt;
};
```

【成员】

成员名称	描述
in_bit_width	输入数据位宽。
in_sample_rate	输入数据采样率。
in_channels	输入数据声道。
out_bit_width	输出数据位宽。
out_sample_rate	输出数据采样率。
out_channels	输出数据声道。
frame_cnt	每次回调处理的帧数。

【注意】

- 输入参数由系统决定，输出参数默认和输入一致。
- frame_cnt 建议按默认，不去调整，调整后内部需要拼帧/拆帧。

5.32 rc_pb_filter_param

【说明】
插件参数。

【定义】

```
struct rc_pb_filter_param {
    void   *in_data;
    rc_u32  in_len;

    void   *out_data;
    rc_u32  out_len;
};
```

【成员】

成员名称	描述
in_data	输入数据内存地址。
in_len	输入数据长度。
out_data	输出数据内存地址。
out_len	输出数据长度。

【注意】

- 长度由注册时 rc_pb_filter_attr 决定 。
- 访问时不要越界。
- 音频数据是交织排布的。

6. DUMP

6.1 模块说明

6.1.1 AO

【调试信息】

```
root@rk3308b-buildroot:/# dumpsys ao
------------------------------------------------------------------------------
DUMP OF SERVICE ao:
---------------------- ao dev attr -----------------------------
ao_dev    snd_rate  snd_channel    snd_bit_Width  data_rate data_channel
data_bit_width chn_cnt   expand_flag    frm_num        frm_size
0         48000     2              16bit          48000     stereo
16bit          0         0              1              0
---------------------- ao dev extend status ----------------------------
ao_dev         track_mode     mute           volume
0              0              N              0.00
---------------------- ao dev advanced parameter ----------------------------
-
ao_dev tst_mode avail_min prd_cnt prd_size prd_step sil_size sil_thr
start_delay stop_delay bound
0      enable   0         2       128      0        0        0       0
    0          0
---------------------- ao chn attr -----------------------------
ao_dev    ao_chn    card_name        snd_open       state
resample_open  in_rate   in_ch     out_rate  out_ch
0         0         hw:0,0           open           start          N
     0         0         48000     2
---------------------- ao chn status -----------------------------
ao_dev              ao_chn              frm_len             frm_total_cnt
    frm_total_len       underrun_cnt  send_rate
0                   0                   0                   0
    0                   2             0.0000
---------------------- ao node status -----------------------------
ao_dev              ao_chn              node                in_size
    in_cnt              proc_cnt            write_size
0                   0                   0                   512
    1                   1385                709120

------------------------------------------------------------------------------
END DUMP OF SERVICE ao:
```

【调试信息分析】

记录当前音频输出属性配置以及状态信息。

【参数说明】

参数模块	参数名	描述
音频输出通道属性	ao_dev	设备号
	snd_rate	打开声卡的采样率
	snd_channel	打开声卡的声道数
	snd_bit_width	打开声卡的位宽
	data_rate	发送数据的采样率 范围：[8k,96k]
	data_channel	发送数据的声道数 mono：单声道 stereo：双声道
	data_bit_width	发送数据的采样精度 范围：[8bit,16bit]
	chn_cnt	暂未使用
	expand_flag	暂未使用
	frm_num	设置送帧最大缓冲buffer数
	frm_size	设置送帧buffer大小，暂未使用
音频输出设备扩展信息	ao_dev	设备号
	track_mode	声道模式 0：normal 1：both_left 2：both_right 3：exchange 4：mix 5：left_mute 6：right_mute 5：both_mute
	mute	静音功能是否开启 Y：开启 N：关闭
	volume	音量值
音频输入高级参数信息	ao_dev	设备号
	tst_mode	使能时间戳模式： enable：使能 disable：不使能
	avail_min	最小可用帧
	prd_cnt	音频缓冲个数
	prd_size	每次传输的数据长度，单位：音频帧
	prd_step	每次传输的数据步进，单位：音频帧

参数模块	参数名	描述
	sil_size	静音帧大小，单位：音频帧
	sil_thr	静音帧阈值，单位：音频帧
	start_delay	启动传输延时，单位：us
	stop_delay	停止传输延时，单位：us
	bound	缓存区边界，单位：音频帧
音频输出通道属性	ao_dev	设备号
	ao_chn	通道号
	card_name	声卡名
	snd_open	声卡是否打开: close:关闭 open:打开
	state	通道状态 idle：闲置状态 pause：暂停状态 start：工作状态
	resample_open	重采样是否开启 Y：开启 N：关闭
	in_rate	重采样的源采样率
	in_ch	重采样的源声道数
	out_rate	重采样的目标采样率
	out_ch	重采样的目标声道数
音频输出通道信息	ao_dev	设备号
	ao_chn	通道号
	frm_len	ao送帧当前帧的长度
	frm_total_cnt	ao送帧累计帧数
	frm_total_len	ao送帧累计长度
	underrun_cnt	underrun次数
	send_rate	ao送帧采样率
音频输出通道节点信息	ao_dev	设备号
	ao_chn	通道号
	node	节点号
	in_size	当前节点inputBuffer缓冲长度

参数模块	参数名	描述
	in_cnt	当前节点inputBuffer缓冲个数
	proc_cnt	process函数线程累计计数（只统计plackback节点）
	write_size	写入声卡的buffer长度（只统计plackback节点）

6.1.2 AI

【调试信息】

```
root@rk3308b-buildroot:/# dumpsys ai
--------------------------------------------------------------------------------
DUMP OF SERVICE ai:
---------------------- ai dev attr -----------------------------
ai_dev    snd_rate  snd_channel    snd_bit_Width  data_rate data_channel
data_bit_width chn_cnt   expand_flag    frm_num        frm_size
0         48000     4              16bit          48000     NULL
16bit          0         0              0              0
---------------------- ai dev extend status ----------------------------
ai_dev    track_mode
0         0
---------------------- ai dev advanced parameter ----------------------------
-
ai_dev tst_mode avail_min prd_cnt prd_size prd_step sil_size sil_thr
start_delay stop_delay bound
0      enable   0         2       128      0        0        0       0
    0          0
---------------------- ai chn attr -----------------------------
ai_dev    ai_chn    card_name        snd_open       state
resample_open  in_rate   in_ch     out_rate  out_ch    valid_start_bit
valid_bitwidth
0         0         mic              open           start          N
     48000     4         48000     0         0               16bit

---------------------- ai chn status -----------------------------
ai_dev              ai_chn              frm_len             frm_total_cnt
    frm_total_len       overrun_cnt
0                   0                   0                   0
    0                   0
---------------------- ai node status -----------------------------
ai_dev              ai_chn              node                in_size
    in_cnt
0                   0                   0                   0
    0
0                   0                   1                   0
    0
0                   0                   2                   0
    0
0                   0                   3                   0
    0

```

```
27 ------------------------------------------------------------------------------
   ---
28 END DUMP OF SERVICE ai:
```

【调试信息分析】

记录当前音频输入属性配置以及状态信息。

【参数说明】

参数模块	参数名	描述
音频输入设备属性	ai_dev	设备号
	snd_rate	打开声卡的采样率
	snd_channel	打开声卡的声道数
	snd_bit_width	打开声卡的位宽
	data_rate	获取数据的采样率
	data_channel	获取数据的声道数 mono：单声道 stereo：双声道
	data_bit_width	获取数据的采样精度 范围：[8bit,16bit]
	chn_cnt	暂未使用
	expand_flag	暂未使用
	frm_num	ai暂未使用
	frm_size	应用设置取帧的长度（byte）。
音频输入设备扩展信息	ai_dev	设备号
	track_mode	声道模式 0：normal 1：both_left 2：both_right 3：exchange 4：mix 5：left_mute 6：right_mute 5：both_mute
音频输入高级参数信息	ai_dev	设备号
	tst_mode	使能时间戳模式： enable：使能 disable：不使能
	avail_min	最小可用帧
	prd_cnt	音频缓冲个数
	prd_size	每次传输的数据长度，单位：音频帧
	prd_step	每次传输的数据步进，单位：音频帧
	sil_size	静音帧大小，单位：音频帧
	sil_thr	静音帧阈值，单位：音频帧
	start_delay	启动传输延时，单位：us

参数模块	参数名	描述
	stop_delay	停止传输延时，单位：us
	boundary	缓存区边界，单位：音频帧
音频输入通道属性	ai_dev	设备号
	ai_chn	通道号
	card_name	声卡名
	snd_open	声卡是否打开: close:关闭 open:打开
	state	通道状态 idle：闲置状态 pause：暂停状态 start：工作状态
	resample_open	重采样是否开启 Y：开启 N：关闭
	in_rate	重采样的源采样率
	in_ch	重采样的源声道数
	out_rate	重采样的目标采样率
	out_ch	重采样的目标声道数
音频输入通道信息	ai_dev	设备号
	ai_chn	通道号
	frm_len	ai取帧当前帧的长度
	frm_total_cnt	ai取帧累计帧数
	overrun_cnt	overrun次数
音频输入通道节点信息	ai_dev	设备号
	ai_chn	通道号
	node	节点号
	in_size	当前节点inputBuffer缓冲长度
	in_cnt	当前节点inputBuffer缓冲个数

6.1.3 AF

【调试信息】

```
root@rk3308b-buildroot:/# dumpsys af
------------------------------------------------------------------------------
DUMP OF SERVICE af:
---------------------- af chn attr -----------------------------
chn_id  type        in_buf_cnt  out_buf_cnt  rate  chn  bitwidth  depth
1       algo_combo  1           1            48000 1    16bit     0
---------------------- af chn attr -----------------------------
chn_id  type    in_buf_cnt     rate    chn   bitwidth  chn_layout
ref_layout  mic_layout  ref_type  bypass
1       skv     0              48000   4     16bit     0xf         0x3
  0x4         2         0
---------------------- af chn attr -----------------------------
chn_id  type    in_buf_cnt  out_buf_cnt  rate  chn  bitwidth  depth
frame_cnt det_per_frm band_cntdet_chn  rms  hold  decay  bypass
1       lv_det  0           0            48000 1    16bit     1      128
  2           10      1        200  0     200    1
---------------------- af chn attr -----------------------------
chn_id  type    in_buf_cnt  out_buf_cnt  rate  chn  bitwidth  depth
frame_cnt  bypass  mode
1       reverb  0           0            48000 1    16bit     0      128
   0       2
---------------------- af chn attr -----------------------------
chn_id  type        in_buf_cnt  out_buf_cnt  in_chn  out_chn  rate  bitwidth
depth  frame_cnt  bypass
1       rkstudio    0           0            1       1        48000 16bit
  0      128        1
---------------------- af chn attr -----------------------------
chn_id  type        in_buf_cnt  out_buf_cnt  rate  chn  bitwidth  volume
1       volume_ind  0           0            48000 1    16bit     0.00

---------------------- af chn attr -----------------------------
chn_id  type        in_buf_cnt  out_buf_cnt  rate  chn  bitwidth  depth
2       algo_combo  1           1            48000 2    16bit     0
---------------------- af chn attr -----------------------------
chn_id  type    in_buf_cnt  out_buf_cnt  mixer_input_num  rate  out_chn
bitwidth                in_chn                              status
                      gain
2       mixer   0           0            5                48000 2
16bit     0:1 1:2 2:2 3:2 4:2 5:0 6:0 7:0   0:1 1:1 2:0 3:0 4:0 5:0 6:0 7:0
   0:0.0 1:0.0 2:0.0 3:-80.0 4:-80.0 5:0.0 6:0.0 7:-80.0 8:-80.0 9:0.0
10:0.0 11:-80.0 12:-80.0 13:0.0
---------------------- af chn attr -----------------------------
chn_id  type        in_buf_cnt  out_buf_cnt  in_chn  out_chn  rate  bitwidth
depth  frame_cnt  bypass
2       rkstudio    0           0            2       2        48000 16bit
  0      128        0
---------------------- af chn attr -----------------------------
chn_id  type        in_buf_cnt  out_buf_cnt  rate  chn  bitwidth  volume
2       volume      0           0            48000 2    16bit     0.00

---------------------- af chn attr -----------------------------
```

```
chn_id  type        in_buf_cnt  out_buf_cnt  rate  chn  bitwidth  volume
8       volume      1           1            48000 2    16bit     0.00

---------------------- af chn attr -----------------------------
chn_id  type      in_buf_cnt     in_rate   in_ch in_bitwidth     out_rate
out_ch    out_bitwidth
10      resample  4              44100     2     16bit           48000     2
        16bit

---------------------- af chn attr -----------------------------
chn_id  type    in_buf_cnt  out_buf_cnt  rate  chn  bitwidth  depth
frame_cnt det_per_frm band_cntdet_chn  rms  hold  decay  bypass
11      lv_det  4           4            48000 2    16bit     1      768
  2           10      0        200  0     200    0

---------------------- af chn attr -----------------------------
chn_id  type    in_buf_cnt  out_buf_cnt  rate  chn  bitwidth  bypass
human_level  guitar_level  other_level
12      vocal   4           4            48000 2    16bit     1       15
      100           100

---------------------- af chn attr -----------------------------
chn_id  type        in_buf_cnt  out_buf_cnt  rate  chn  bitwidth  depth
13      algo_combo  4           4            48000 2    16bit     0
---------------------- af chn attr -----------------------------
chn_id  type        in_buf_cnt  out_buf_cnt  in_chn  out_chn  rate  bitwidth
depth  frame_cnt  bypass
13      rkstudio    0           0            2       2        48000 16bit
  0      768        1
---------------------- af chn attr -----------------------------
chn_id  type        in_buf_cnt  out_buf_cnt  rate  chn  bitwidth  volume
13      volume      0           0            48000 2    16bit     0.00

---------------------- af chn attr -----------------------------
chn_id  type    in_buf_cnt     rate    chn   bitwidth  ref_layout
mic_layout  ref_type  scene_mode  bypass
15      scene   4              48000   2     16bit     0x0         0x3
  0         0x4         0


------------------------------------------------------------------------
---
END DUMP OF SERVICE af:
```

【调试信息分析】

记录当前音频处理属性配置以及状态信息。

【参数说明】

参数模块	参数名	描述
音频处理设备属性	chn_id	通道号
	type	音频处理算法

6.1.4 SYS

【调试信息】

```
root@rk3308b-buildroot:/# dumpsys sys
--------------------------------------------------------------------------------
DUMP OF SERVICE sys:
---------------------- bind relation table -----------------------------
src_mod   src_dev   src_chn   dst_mod   dst_dev   dst_chn   src_recv_cnt   dst_recv_cnt   src_recv_rate  dst_recv_rate
af        0         2         ao        0         0         0              4047           0.00           375.00
af        0         1         af        2         2         4047           0              374.99         0.00
ai        0         0         af        0         1         -1             4047           -1.00          374.99
af        0         8         af        5         2         555            0              63.09          0.00
af        0         11        af        0         12        329            536            36.98          60.24
af        0         12        af        0         13        536            532            60.24          62.52
af        0         13        af        3         2         532            0              62.52          0.00
af        0         10        af        0         11        334            329            40.38          36.98


--------------------------------------------------------------------------------
END DUMP OF SERVICE sys:
```

【调试信息分析】

记录当前 SYS 模块的使用情况。

【参数说明】

参数模块	参数名	描述
bind relation table 模块通道间的绑定关系	src_mod	绑定关系中第一级的模块名，数据由第一级发送给第二级。
	src_dev	绑定关系中第一级的设备号，数据由第一级发送给第二级。
	src_chn	绑定关系中第一级的通道号，数据由第一级发送给第二级。
	dst_mod	绑定关系中第二级的模块名，数据由第一级发送给第二级。
	dst_dev	绑定关系中第二级的设备号，数据由第一级发送给第二级
	dst_chn	绑定关系中第二级的通道号，数据由第一级发送给第二级。
	src_recv_cnt	第一级接收到的数据计数（一般以帧为单位）。
	dst_recv_cnt	第一级向第二级的发送数据计数（一般以帧为单位）。
	src_recv_rate	第一级接收到的数据帧率。
	dst_recv_rate	第一级向第二级的发送数据帧率。

【举例说明】
上述 dumpsys sys 中 af 10 下级绑定 af 11, af 11 下级绑定 af 12，12 绑定 13，af 13 绑定 af 通道 2，设备号 3。dumpsys af 查看 af 可知，af 10 resample，11 lv_det，12 vocal，13 algo_combo(rkstudio + volume)，af 2 mixer，pipeline 为 resample -> level_detect -> vocal_separate -> rkstudio -> volume -> mix。

6.2 dump数据

```
#mic通路
#mic原始数据
dumpsys record -m ai -d 0 -c 0 -o /tmp/ai_out_48k_4ch.pcm  -i 0 -n 50000;
#howling输入输出
echo "path:/tmp/howling_in_48k_3ch.pcm /tmp/howling_out_48k_1ch.pcm cnt:10000" > /tmp/howling_debug;
#reverb输入输出
echo "path:/tmp/reverb_in_48k_1ch.pcm /tmp/reverb_out_48k_1ch.pcm cnt:10000" > /tmp/reverb_debug;
#rkstuido_core2+volume输出
dumpsys record -m af -d 0 -c 1 -o /tmp/mic_algo_out_48k_1ch.pcm  -i 0 -n 50000;

#独立回采输入
dumpsys record -m ai -d 4 -c 0 -o /tmp/ai_ref_out_48k.pcm  -i 0 -n 50000;

#guitar
dumpsys record -m ai -d 2 -c 0 -o /tmp/ai_guitar_out_48k.pcm  -i 0 -n 50000;
#guitar volume输入
```

```
dumpsys record -m af -d 0 -c 31 -o /tmp/guitar_vol_in_48k.pcm  -i 0 -n 50000;

#usb通路
#resample输入
dumpsys record -m af -d 0 -c 10 -o /tmp/usb_resample_in.pcm  -i 1 -n 50000;
#resample输出
dumpsys record -m af -d 0 -c 10 -o /tmp/usb_resample_out_48k_2ch.pcm  -i 0 -n 50000;
#level输出
dumpsys record -m af -d 0 -c 11 -o /tmp/usb_level_out_48k_2ch.pcm  -i 0 -n 50000;
#vocal输出
dumpsys record -m af -d 0 -c 12 -o /tmp/usb_vocal_out_48k_2ch.pcm  -i 0 -n 50000;
#rkstuido_core1+volume输出
dumpsys record -m af -d 0 -c 13 -o /tmp/usb_rkstudio_vol_out_48k_2ch.pcm  -i 0 -n 50000;

#bt/uac通路
#bt 原始数据
dumpsys record -m ai -d 1 -c 0 -o /tmp/bt_ai_out_48k_2ch.pcm  -i 0 -n 50000;
#level输出
dumpsys record -m af -d 0 -c 21 -o /tmp/bt_level_out_48k_2ch.pcm  -i 0 -n 5000;
#vocal输出
dumpsys record -m af -d 0 -c 22 -o /tmp/bt_vocal_out_48k_2ch.pcm  -i 0 -n 50000;
#rkstuido_core1+volume输出
dumpsys record -m af -d 0 -c 23 -o /tmp/bt_rkstudio_vol_out_48k_2ch.pcm  -i 0 -n 50000;

#rkstudio输入输出
echo "path:/tmp/rkstudio_in_48k.pcm /tmp/rkstudio_out_48k.pcm cnt:10000" > /tmp/rkstudio_debug;
送声卡volume输出
dumpsys record -m ao -d 0 -c 0 -o /tmp/main_volume_out_48k_2ch.pcm  -i 1 -n 25000;

#场景mic数据
dumpsys record -m ai -d 3 -c 0 -o /tmp/scene_ai_out_48k_4ch.pcm  -i 0 -n 50000;
#场景识别，完整数据
export rt_debug_scene_path=/tmp/
#场景识别，动态抓取数据
echo "path:/tmp/scene_in_48k.pcm /tmp/scene_out_48k.pcm cnt:10000" > /tmp/scene_debug;

#混音数据
export rt_debug_mixer_path=/tmp/

#rkstudio_core0输入输出
echo "path:/tmp/rkstudio_core0_in.pcm /tmp/rkstudio_core0_out.pcm cnt:10000" > /tmp/rkstudio_core0_debug;
#rkstudio_core1输入输出
echo "path:/tmp/rkstudio_core1_in.pcm /tmp/rkstudio_core1_out.pcm cnt:10000" > /tmp/rkstudio_core1_debug;
#rkstudio_core2输入输出
```

```
echo "path:/tmp/rkstudio_core2_in.pcm /tmp/rkstudio_core2_out.pcm cnt:10000" > /tmp/rkstudio_core2_debug;

#外部注册core0算法数据
echo "path:/tmp/ext_core0_in.pcm /tmp/ext_core0_out.pcm cnt:10000" > /tmp/ext_core0_debug;
#外部注册core1算法数据
echo "path:/tmp/ext_core1_in.pcm /tmp/ext_core1_out.pcm cnt:10000" > /tmp/ext_core1_debug
#外部注册core2算法数据
echo "path:/tmp/ext_core2_in.pcm /tmp/ext_core2_out.pcm cnt:10000" > /tmp/ext_core2_debug
```

【注意】

- dumpsys record命令：
  -m：模块
  -d：设备号
  -c：通道号
  -o：保存文件名
  -i： 0 表示输出，1 表示出入
  -n：帧数
- echo命令：
  path： 第一个为输入文件名，第二个输出文件名
  cnt：帧数
- export方式抓取数据（混音数据、场景识别），export 要在应用启动前执行，且同一个终端窗口中执行，确保环境变量生效。

6.3 环境变量

一些参数由环境变量配置，如果没指定，走默认参数。

```
防啸叫算法参数
export rt_cfg_path_3a=/oem/config_howling.json
室内外，左右声道算法参数
export rt_cfg_path_reverb_doa_detect=/oem/config_reverb_doa_detect.json
男女声算法参数
export rt_cfg_path_gender_detect=/oem/config_gender_detect.json
rkstudio_core0参数
export rt_cfg_path_rkstudio=/oem/rkstudio.bin
rkstudio_core1参数
export rt_cfg_path_rkstudio_player=/oem/eq_drc_player.bin
rkstudio_core2参数
export rt_cfg_path_rkstudio_recorder=/oem/eq_drc_recorder.bin
rkstudio_core1是否bypass
export player_rkstudio_bypass=0
rkstudio_core2是否bypass
export recorder_rkstudio_bypass=1
mic能量检测否bypass
export recorder_level_bypass=1
男女声是否bypass
export player_gender_bypass=1
```

【注意】

- rkstudio 3个 core 如果使用同一个 json，该 json 中需要有 3 个 core 的参数 (即工具中画了 3 个 core 的图)，这样每个 core 才能加载正确自己对应的图。
- rkstudio 3个 core 如果各自用一个 json，该 json 中可以只有一张图的参数(即工具中只画了 core0 的图)，这张图给这个 core 专用。

7. 常见问题

7.1 dumpsys工具无法使用

dumpsys 命令输出如下错误。

```
connect failed, reason: Connection refused
sendDump: send_message failed
Confirm MPI system has been initialized and running?
--------------------------------------------------------------------------------
DUMP OF SERVICE all:
--------------------------------------------------------------------------------
END DUMP OF SERVICE all:
```

- 确认应用进程已调用 rc_pb_create 函数调用成功，在系统初始化时，会创建 Dumpsys Server 来接收命令消息，可通过 netstat -nlt 来确认是否已初始化成功，确认端口 3893 连接是否存在，输出如下：

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:3893          0.0.0.0:*               LISTEN
```

- 确认应用进程未调用 rc_pb_destroy，若已调用，将退出 Dumpsys Server，无法处理 dumpsys 命令消息。
- 确认应用进程处于运行状态，可通过ps命令来确认当前应用是否已退出。
- 确认当前设备是否有 127.0.0.1 回环地址，可通过 ifconfig 来确认，输出如下：

```
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:756 errors:0 dropped:0 overruns:0 frame:0
          TX packets:756 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:55888 (54.5 KiB)  TX bytes:55888 (54.5 KiB)
```

如果不存在这个配置， 可通过如下命令来手动配置

```
 ifconfig lo 127.0.0.1
```

7.2 声卡对接调试

在对接 partybox 之前，先确定 arecored/aplay 是否可以正常录音/放音，可以后再由 partybox 去打开声卡。

7.2.1 查看音频驱动注册的声卡

```
查看注册声卡命令
root@rk3308b-buildroot:/# cat /proc/asound/cards
 0 [rockchiprk3308v]: rockchip_rk3308 - rockchip,rk3308-vad
                      rockchip,rk3308-vad
 1 [rockchiprk3308p]: rockchip_rk3308 - rockchip,rk3308-pcm
                      rockchip,rk3308-pcm
 2 [UAC1Gadget     ]: UAC1_Gadget - UAC1_Gadget
                      UAC1_Gadget 0
 7 [Loopback       ]: Loopback - Loopback
                      Loopback 1
```

如果声卡初始化成功，那么通过如上命令，可以看到注册成功后的声卡信息。反之，如果通过如上命令看不到声卡，则说明声卡驱动注册失败，需要检查声卡驱动。

7.2.2 用命令行测试声卡播放和录音功能

- 播放测试

使用 aplay 命令播放PCM文件，测试下声卡的播放功能。

```
killall rkpartybox # 退出应用，避免占用声卡
aplay -Dhw:0,0 /userdata/t48.pcm -c 2 -r 48000 -f S16_LE --period-size 128 --buffer-size 256
参数说明：
-Dhw:x,0 : x代表的声卡序号
-c 声道数
-r 采样率
-f 格式
```

- 录音测试

录音：使用 arecord 命令录制 PCM 文件，测试声卡的录音功能。

```
killall rkpartybox # 退出应用，避免占用声卡
arecord -Dhw:0,0 -c 2 -r 48000 -f S16_LE -t raw /tmp/rec.pcm --period-size 128 --buffer-size 256
参数说明：
-Dhw:x,0 : x代表的声卡序号
-c 声道数
-r 采样率
-f 格式
-t 输出文件格式
```

- 查看声卡硬件参数

  注意要声卡打开后看，声卡号要匹配。录音为 pcm0c，放音 pcm0p。

```
录音
cat /proc/asound/card0/pcm0c/sub0/hw_params
access: RW_INTERLEAVED
format: S16_LE
subformat: STD
```

```
channels: 2
rate: 48000 (48000/1)
period_size: 128
buffer_size: 256

放音
root@rk3308b-buildroot:/# cat /proc/asound/card0/pcm0p/sub0/hw_params
access: RW_INTERLEAVED
format: S16_LE
subformat: STD
channels: 2
rate: 48000 (48000/1)
period_size: 128
buffer_size: 256
```

7.3 声音异常

需要抓音频数据，对比输入输出，数据如果有问题，需逐级定位到是 pipeline 中那个模块引入的。抓数据命令见dump数据

7.4 断音卡顿

7.4.1 确认mclk时钟是否正常

启动声卡后打印 clk_summary

```
cat /sys/kernel/debug/clk/clk_summary
```

根据dts配置和硬件图找到对应clk值，查看是否正常。

比如某个声卡(rockchip,spdif-tx1)播放卡顿。确认spdif-tx1的mlck值。如果是mclk_spdif1的频率是12000000是不对的。

```
#cat /sys/kernel/debug/clk/clk_summary
.....
clk_spdif1                1        1        0    12000000          0     0
 50000
  mclk_spdif1             1        2        0    12000000          0     0
 50000
.....
```

需要修改对应声卡的dts配置，添加分频：

```
simple-audio-card,mclk-fs = <128>;
```

计算mclk公式: mclk = sample_rate * mclk_fs。
比如打开声卡采样率44100Hz，mclk = 44100 * 128 = 5644800Hz。
修改后mclk打印如下：

```
#cat /sys/kernel/debug/clk/clk_summary
.....
clk_spdif1                1        1        0    5644800          0     0
 50000
  mclk_spdif1             1        2        0    5644800          0     0
 50000
.....
```

7.4.2 xrun 断音问题

该问题为调度不及时，没及时从声卡取走数据(录音overrun)/没及时送给声卡数据(放音underrun)。需要trace 查看。
开trace需要内核配合开启一些配置

7.4.2.1 xrun ftrace

- trace 数据说明

Trace Event	Description
snd_pcm:applptr	应用指针更新
snd_pcm:hw_ptr_error	dma 指针出错
snd_pcm:xrun	xrun
snd_pcm:hwptr	dma 指针更新

- 内核配置

```
CONFIG_SND_DEBUG=y
CONFIG_SND_DEBUG_VERBOSE=y
CONFIG_SND_PCM_XRUN_DEBUG=y
CONFIG_FUNCTION_TRACER=y
CONFIG_FUNCTION_GRAPH_TRACER=y
CONFIG_STACK_TRACER=y
CONFIG_DYNAMIC_FTRACE=y
```

- 抓trace命令

```
echo performance > /sys/devices/system/cpu/cpufreq/policy0/scaling_governor

rm /tmp/ftrace_alsa.log
echo 1 > /sys/kernel/debug/tracing/events/snd_pcm/enable
cat /sys/kernel/debug/tracing/set_event
echo 0 > /sys/kernel/debug/tracing/tracing_on
echo 0 > /sys/kernel/debug/tracing/trace
echo 1 > /sys/kernel/debug/tracing/tracing_on

cat /sys/kernel/debug/tracing/trace_pipe > /tmp/ftrace_alsa.log
```

- 操作方法

打开应用程序。
执行抓 trace 脚本， 复现后将 /tmp/ftrace_alsa.log 提供给技术支持人员。

7.4.2.2 调度 ftrace

- 内核配置

```
CONFIG_FUNCTION_TRACER=y
CONFIG_FUNCTION_GRAPH_TRACER=y
CONFIG_CONTEXT_SWITCH_TRACER=y
CONFIG_NOP_TRACER=y
# CONFIG_SCHED_TRACER is not set
CONFIG_EVENT_TRACING=y
CONFIG_FTRACE=y
CONFIG_BLK_DEV_IO_TRACE=y
```

- 抓trace命令

```
echo performance > /sys/devices/system/cpu/cpufreq/policy0/scaling_governor
rm /tmp/ftrace.log
#echo NO_TTWU_QUEUE > /sys/kernel/debug/sched_features
sleep 1s
cd /sys/kernel/debug/tracing/events
echo 0 > sched/enable
echo 1 > sched/sched_switch/enable
echo 1 > sched/sched_wakeup/enable
echo 1 > block/enable
echo 1 > scsi/enable
echo 1 > irq/enable
echo 1 > workqueue/enable
cd ..
echo 96000 > buffer_size_kb
echo 1 > options/record-tgid
echo global > trace_clock
echo 1 > tracing_on
sleep 1s
echo 0 > tracing_on
cat trace > /tmp/ftrace.log
```

- 操作方法

打开应用程序。
执行抓 trace 脚本， 复现后将 /tmp/ftrace.log 提供给技术支持人员。

7.5 防啸叫调试

- 数据抓取 howling输入输出
- 数据确认

  确认每个通道是否有数据（ mic + ref ），无数据请检查硬件、驱动是否正常， 可使用 aplay / arecord 确认 。
  确认数据是否截幅，若截幅需调整增益，确保送给 howling 数据无截幅。

- 参数文档

  参数配置文件：`/oem/config_howling.json`
  参数文档：
  `app/rkpartybox/doc/Rockchip_Developer_Guide_Howling_Suppresion_Tuning.pdf`

7.6 rkstudio 调音台

工具和相应文档放在 ftp 服务器，如下：

```
ftp://www.rockchip.com.cn
用户名：rkwifi
密码：Cng9280H8t
目录：/15-RK3308/RkStudioTool
```