# AUDIO 接口设计说明书
# V1.0

**珠海市杰理科技股份有限公司**

**Zhuhai Jieli Technologyco.,LTD**





地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

# 修改记录

| 版本 | 更新日期 | 描述 |
| --- | --- | --- |
| V1.0 | 2020-09-17 | |
| V1.1 | | |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

# 目录






地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

# 1. 文档介绍

## 1.1. 文档目的

## 1.2. 参考文献

[1].

## 1.3. 术语与缩写词

| 缩写、术语 | 解 释 |
|---|---|
| AP | Application，应用程序 |
| | |
| | |
| | |
| | |



地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

# 2. 功能概述

# 3. 其他系统/模块的调用关系

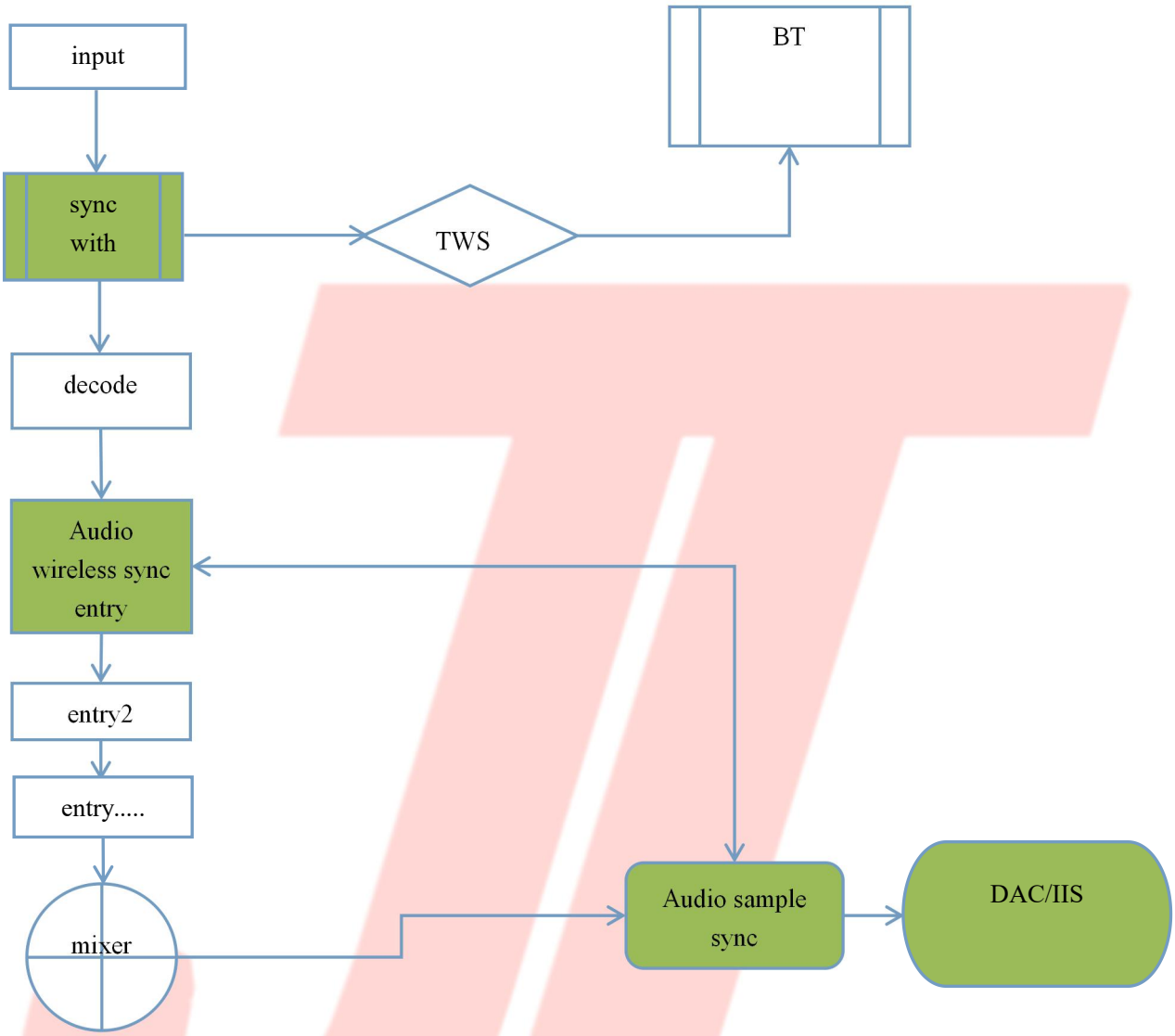
# 4. 性能要求

# 5. 音频同步接口模块功能介绍

## 5.1. 流程




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com




## 5.2. AUDIO WIRELESS SYNC

### 5.2.1. 功能介绍

Audio wireless sync 属于蓝牙无线音频同步功能的子模块，负责将本地播放输出与蓝牙系统的关联起来同步。Audio wireless sync 模块负责对接输入，读取输出模块的信息，计算误差及修正的结果值，将最终结果值配置到 Audio sample sync（即同步的 SRC 模块），改变实际播放速率达到控制输入与输出同步的目的。

无线音频同步功能支持 TWS 音频同步，需蓝牙配置 CONFIG_BTCTLER_TWS_ENABLE 才有效。




地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

### 5.2.2. 接口介绍

#### 5.2.2.1. 打开

| | |
|---|---|
| 函数原型 | void *audio_wireless_sync_open(struct audio_wireless_sync_info *info, struct audio_stream_entry **entry); |
| 功能描述 | 打开同步模块，配置同步所需要的信息参数，获取需要接入数据流的 entry 节点 |
| 参数说明 | 1、info：蓝牙无线音频同步的配置信息参数<br>struct audio_wireless_sync_info {<br>u8 target; //本地目标设备<br>u8 protocol; //音频同步协议类型<br>u8 reset_enable; //内部检测异常后返回复位信息的使能<br>u16 sample_rate; //解码采样率<br>u16 output_rate; //输出采样率<br>void *sample_ch; //同步采样通道<br>const struct audio_tws_conn_ops *tws_ops; //TWS 相关的调用接口<br>};<br>2、entry : audio 数据流节点，参数为二级指针，打开同步后将获取同步的数据流节点 entry 实体的指针供外部对接使用 |
| 输出 | 输出无类型指针，指针为内部申请结构 context；<br>NULL 则属于模块打开异常 |
| 例子 | a2dp_sync->context = audio_wireless_sync_open(&info, &a2dp_sync->entry); |
| 关联模块 | audio_sample_sync，蓝牙收发 API(TWS) |
| 补充说明 | TWS 同步使能需配置 CONFIG_BTCTLER_TWS_ENABLE=1，并配置 tws_ops 参数 |

#### 5.2.2.2. 增加需要同步的输出级

| | |
|---|---|
| 函数原型 | int audio_wireless_sync_add_output(void *c, struct audio_wireless_sync_info *info); |
| 功能描述 | 添加需要同步的输出到同步模块 |
| 参数说明 | 1、info：蓝牙无线音频同步的配置信息参数，和打开时需要配置的一个输出参数一致 |
| 输出 | 0：成功<br>非 0：添加输出同步失败 |
| 例子 | err = audio_wireless_sync_add_output(a2dp_sync->context, &info); |
| 关联模块 | audio_sample_sync，蓝牙收发 API(TWS) |
| 补充说明 | |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

#### 5.2.2.3. 与接收数据流延时同步

| | |
|---|---|
| 函数原型 | int audio_wireless_sync_with_stream(void *c, struct remote_stream_info *info); |
| 功能描述 | 和蓝牙接收音频的数据做同步，作用于输出来维持蓝牙收数的延时水平 |
| 参数说明 | 1、c：音频同步模块 context 指针<br>2、info：蓝牙音频接收端数据波动及音频包信息<br>struct remote_stream_info {<br>u8 rx_delay; //音频接收数据流延时波动<br>u16 seqn; //音频数据包号信息<br>u32 remain_len; //蓝牙接收 buffer 剩余空间<br>};<br>#define RX_DELAY_NULL 0 //延时未抖动<br>#define RX_DELAY_UP 1 //延时上浮<br>#define RX_DELAY_DOWN 2 //延时下浮 |
| 输出 | 0：同步成功<br>非 0：同步失败，需复位数据流（设置 reset_enable 才有失败返回） |
| 例子 | struct remote_stream_info info = {0};<br>info.rx_delay = RX_DELAY_UP;<br>info.seqn = dec->seqn;<br>info.remain_len = a2dp_media_get_remain_buffer_size();<br>err = audio_wireless_sync_with_stream(dec->sync, &info); |
| 关联模块 | audio_sample_sync，蓝牙收发 API(TWS)、蓝牙解码 |
| 补充说明 | |

#### 5.2.2.4. 设置丢样点

| | |
|---|---|
| 函数原型 | int audio_wireless_sync_drop_samples(void *c, int samples); |
| 功能描述 | 设置经过同步模块需要丢的音频样点数量 |
| 参数说明 | 1、c: 音频同步模块 context 指针<br>2、samples：样点个数 |
| 输出 | 0：正常返回<br>非 0：异常返回（大部分因为传入空指针） |
| 例子 | audio_wireless_sync_drop_samples(dec->sync, samples); |
| 关联模块 | audio_sample_sync |
| 补充说明 | |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

#### 5.2.2.5. 复位经过同步的声音

| | |
|---|---|
| 函数原型 | int audio_wireless_sync_sound_reset(void *c, int time); |
| 功能描述 | TWS 需要重新对齐使用的接口，用于将声音静音一段时间，两边对齐后再恢复正常音频数据 |
| 参数说明 | 1、c: 音频同步模块 context 指针<br>2、time : 需要复位的时间（ms 单位） |
| 输出 | 0：正常返回<br>非 0：异常返回（大部分因为传入空指针） |
| 例子 | audio_wireless_sync_sound_reset(dec->sync, 500); |
| 关联模块 | |
| 补充说明 | |

#### 5.2.2.6. 查询同步是否工作

| | |
|---|---|
| 函数原型 | int audio_wireless_sync_is_working(void *c); |
| 功能描述 | 查询同步模块是否处于工作状态 |
| 参数说明 | 1、c: 音频同步模块 context 指针 |
| 输出 | 0：未开始工作<br>1：正在工作 |
| 例子 | if (audio_wireless_sync_is_working(a2dp_dec->sync)) {...} |
| 关联模块 | audio_sample_sync |
| 补充说明 | |

#### 5.2.2.7. 设置 TWS 对齐时间

| | |
|---|---|
| 函数原型 | int audio_wireless_sync_set_tws_time(void *c, int time) |
| 功能描述 | 在未开始解码前，将 TWS 约定的统一时间配置到同步模块实现初步对齐 |
| 参数说明 | 1、c: 音频同步模块 context 指针<br>2、time : TWS 约定的时间，一定是 TWS 蓝牙的主机基准时间 |
| 输出 | 0：成功<br>非 0：异常返回（大部分因为传入空指针 |
| 例子 | int time = bt_tws_future_slot_time(0) + msecs_to_bt_time(200); |




地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | audio_wireless_sync_set_tws_time(dec->sync, time); |
| 关联模块 | audio_sample_sync，蓝牙收发 API(TWS) |
| 补充说明 | TWS 同步使能需配置 CONFIG_BTCTLER_TWS_ENABLE=1，并配置 tws_ops 参数 |

#### 5.2.2.8. 关闭

| | |
|---|---|
| 函数原型 | void audio_wireless_sync_close(void *c); |
| 功能描述 | 关闭模块，释放资源 |
| 参数说明 | 1、c: 音频同步模块 context 指针 |
| 输出 | 无 |
| 例子 | audio_wireless_sync_close(a2dp_sync->context);<br>a2dp_sync->context = NULL; |
| 关联模块 | 蓝牙收发 API(TWS) |
| 补充说明 | |

## 5.3. AUDIO SAMPLE SYNC

### 5.3.1. 功能介绍

Audio sample sync 模块属于同步流程中的子模块，负责音频数据输出到 DAC/IIS/...等输出级的变采样以及输出统计功能。DAC/IIS 等输出模块将消耗信息更新到 audio sample sync，Audio wireless sync 通过 audio sample sync 读取到音频在 DAC/IIS 等模块的输入输出位置等关键信息用作计算，再将结果反馈到 audio sample sync 模块实现控制 DAC/IIS 等模块播放的目的。

### 5.3.2. 接口介绍

#### 5.3.2.1. 模块打开

| | |
|---|---|
| 函数原型 | struct audio_sample_sync *audio_sample_sync_open(u8 stream_mode); |
| 功能描述 | Audio sample sync 模块打开 |
| 参数说明 | stream_mode : 接入数据流的方式 |
| 输出 | 返回 struct audio_sample_sync 结构指针<br>NULL 为打开失败 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 例子 | struct audio sample_sync *sample_sync = audio_sample_sync_open(0); |
| 关联模块 | |
| 补充说明 | |

#### 5.3.2.2. 输出设备使用模块接口

| | |
|---|---|
| 函数原型 | int audio_sample_sync_init(struct audio_sample_sync *s,<br>int sample_rate,<br>u8 data_channels,<br>u8 position_num); |
| 功能描述 | 输出模块 DAC/IIS 在被配置后的初始化动作 |
| 参数说明 | 1、s: audio_sample_sync 结构指针<br>2、sample_rate : DAC/IIS/...配置的输出采样率<br>3、data_channels : DAC/IIS/...输出 channel 个数<br>4、position_num : 记录位置信息个数，硬件为 cfifo 则无需配置，默认为 0；硬件为 pingpong buffer 则需要配置个数 2~3 个即可。 |
| 输出 | 0：初始化成功<br>非 0：初始化失败 |
| 例子 | audio_sample_sync_init(dac->sample_sync, dac->sample_rate, dac->channel, 0); |
| 关联模块 | DAC/IIS/..., SRC_BASE |
| 补充说明 | 仅属于接入到输出设备端的配置，外部无需配置 |

| | |
|---|---|
| 函数原型 | int audio_sample_sync_set_fifo_handler(struct audio_sample_sync *s,<br>void *buffer,<br>int (*data_len)(void *priv),<br>s16 * (*write_alloc)(void *priv, int *len),<br>int (*write_update)(void *priv, int len)); |
| 功能描述 | CFIFO 设备类型的输出绑定 audio_sample_sync 配置的模块输出和 fifo 查询函数接口 |
| 参数说明 | 1、s: audio_sample_sync 结构指针<br>2、buffer : 绑定 audio sample sync 的 buffer 私有指针<br>3、data_len : 查询设备的样点缓冲函数指针<br>4、write_alloc : 变采样申请可写入空间的函数指针<br>5、write_update : 变采样完成后更新写入数据的函数指针。 |
| 输出 | 0：成功 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | 非 0：配置失败 |
| 例子 | audio_sample_sync_set_fifo_handler(dac->sample_sync,<br>(void *)dac,<br>audio_dac_buf_samples,<br>(s16 * (*)(void *priv, int *len))audio_dac_get_write_ptr,<br>(int (*)(void *priv, int len))audio_dac_update_write_ptr); |
| 关联模块 | DAC/IIS/..., SRC_BASE |
| 补充说明 | 仅属于接入到输出设备端的配置，外部无需配置，设计上暂时无非 cbuf 存储类型的。 |

| | |
|---|---|
| 函数原型 | int audio_sample_sync_set_stream_handler(struct audio_sample_sync *s,<br>void *buffer,<br>int (*data_len)(void *priv)); |
| 功能描述 | Sample sync 未接入到输出设备端的数据接口函数设置（这时 audio sample sync 作为 audio stream entry 的节点） |
| 参数说明 | 1、s: audio_sample_sync 结构指针<br>2、buffer：绑定 audio sample sync 的 buffer 私有指针<br>3、data_len：查询设备的样点缓冲函数指针 |
| 输出 | 0：成功<br>非 0：配置失败 |
| 例子 | audio_sample_sync_set_stream_handler(dac->sample_sync,<br>dac,<br>audio_dac_buf_samples); |
| 关联模块 | DAC/IIS/..., SRC_BASE，audio_stream |
| 补充说明 | 仅属于接入到输出设备端的配置，外部无需配置 |

| | |
|---|---|
| 函数原型 | int audio_sample_sync_write(struct audio_sample_sync *s, void *buf, int len); |
| 功能描述 | Sample sync 接入到输出设备数据流写入后再输出到输出设备 DAC/IIS/... |
| 参数说明 | 1、s: audio_sample_sync 结构指针<br>2、buf: 音频数据地址<br>3、len：写入数据长度（byte） |
| 输出 | 返回实际写入到模块的长度 |
| 例子 | if (dac->sample_sync) {<br>return audio_sample_sync_write(dac->sample_sync, data, data_len);<br>} |
| 关联模块 | DAC/IIS/..., SRC_BASE |




地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int audio_sample_start_by_sync_time(struct audio_sample_sync *s, void *priv, void (*callback)(void *)); |
| 功能描述 | 输出启动根据同步设置的时间启动 |
| 参数说明 | 1、s: audio_sample_sync 结构指针<br>2、priv: 设置的私有指针<br>3、callback : 达到启动条件的 callback 函数指针 |
| 输出 | 0：到达启动时间或触发启动条件<br>非 0：未到达启动时间 |
| 例子 | if (dac->sample_sync) {<br>int err = audio_sample_start_by_sync_time(dac->sample_sync, (void *)dac, audio_dac_release_fifo_data);<br>} |
| 关联模块 | DAC/IIS/..., SRC_BASE |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int audio_sample_sync_output_begin(struct audio_sample_sync *s, int samples); |
| 功能描述 | 设置 audio sample sync 起始 |
| 参数说明 | 1、s: audio_sample_sync 结构指针<br>2、samples: 起始样点个数 |
| 输出 | 默认正常返回 |
| 例子 | if (dac->sample_sync) {<br>audio_sample_sync_output_begin(dac->sample_sync, samples);<br>} |
| 关联模块 | DAC/IIS/..., SRC_BASE |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int audio_sample_sync_output_miss_data(struct audio_sample_sync *s); |
| 功能描述 | 输出级丢失采样点后通知给 audio_sample_sync |
| 参数说明 | 1、s: audio_sample_sync 结构指针 |
| 输出 | 默认正常返回 |
| 例子 | if (dac->sample_sync) { |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | audio_sample_sync_output_miss_data(dac->sample_sync);<br>} |
| 关联模块 | DAC/IIS/..., SRC_BASE |
| 补充说明 | DAC/IIS/...输出丢失样点通知到 audio sample sync 给上层处理，丢失采样点一般为数据流中有跑不过来的问题 |

| | |
|---|---|
| 函数原型 | int audio_sample_sync_update_count(struct audio_sample_sync *s, int samples); |
| 功能描述 | 输出级更新 audio sample sync 输出计数 |
| 参数说明 | 1、s: audio_sample_sync 结构指针<br>2、samples: 样点个数 |
| 输出 | 默认正常返回 |
| 例子 | if (dac->sample_sync) {<br>audio_sample_sync_update_count(dac->sample_sync, samples);<br>} |
| 关联模块 | DAC/IIS/..., SRC_BASE |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int audio_irq_update_sample_sync_position(struct audio_sample_sync *s, int irq_sample_num); |
| 功能描述 | Audio 输出中断更新数据计数给 audio sample sync |
| 参数说明 | 1、s: audio_sample_sync 结构指针<br>2、samples: 当次中断消耗样点 |
| 输出 | 默认正常返回 |
| 例子 | if (iis->sample_sync) {<br>audio_irq_update_sample_sync_position(iis->sample_sync, samples);<br>} |
| 关联模块 | DAC/IIS/..., SRC_BASE |
| 补充说明 | 该接口主要用作中断类型输出的 audio 模块，常见为 pingpong buffer 的 IIS 等 |

#### 5.3.2.3. AUDIO WIRELESS SYNC 使用模块接口

| | |
|---|---|
| 函数原型 | void audio_sample_sync_set_event_handler(struct audio_sample_sync *s, |



地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | void *priv,<br>int (*handler)(void *priv, void *ch, u8 event)); |
| 功能描述 | 设置 audio sample sync 输出端事件回调函数 |
| 参数说明 | 1、s: audio_sample_sync 结构指针<br>2、priv : 回调私有指针<br>3、handler: 回调 handle 函数<br>priv : 设置的私有指针<br>ch : 当前调用事件函数的 audio_sample_sync 指针<br>event : 事件值<br>常用事件值：<br>#define AUDIO_SYNC_OUTPUT_IDLE 0<br>#define AUDIO_SYNC_OUTPUT_PROBE 1 //输出前处理<br>#define AUDIO_SYNC_OUTPUT_START 2 //输出开始<br>#define AUDIO_SYNC_OUTPUT_MISS_DATA 3 //输出采样丢失样点<br>#define AUDIO_SYNC_OUTPUT_ALIGN_COMPLETE 4 //输出对齐完成 |
| 输出 | 无返回值 |
| 例子 | static int audio_sample_sync_ch_event_handler(void *priv, void *ch, u8 event)<br>{<br>switch (event) {<br>case AUDIO_SYNC_OUTPUT_START:<br><br>break;<br>...<br>default:<br>break;<br>}<br>return 0;<br>}<br><br>audio_sample_sync_set_event_handler(ch->sample_ch, ctx,<br>audio_sample_ch_sync_event_handler); |
| 关联模块 | DAC/IIS/... |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int audio_sample_sync_get_in_position(struct audio_sample_sync *s, struct audio_input_position *pos); |
| 功能描述 | 查询 audio sample sync 模块当前输入的位置信息 |
| 参数说明 | 1、s: audio_sample_sync 结构指针<br>2、pos : 输入位置结构指针<br>struct audio_input_position { |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | u32 buf_num; //输出 buf 当前缓冲样点个数<br>u32 bt_time; //蓝牙时间<br>u32 clkoffset; //与主机的偏差<br>int bitoff; //与主机偏差值的相位<br>u16 bt_time_phase; //蓝牙时间的相位<br>u32 pcm_position; //样点输入位置<br>}; |
| 输出 | 0:正常<br>非 0：异常（暂时无非 0 返回） |
| 例子 | {<br>struct audio_input_position pos;<br>audio_sample_sync_get_in_position(ch->sample_ch, &pos);<br>} |
| 关联模块 | DAC/IIS/... |
| 补充说明 | 该功能仅在 TWS 位置对齐时会使用 |

| | |
|---|---|
| 函数原型 | int audio_sample_sync_get_out_position(struct audio_sample_sync *s, struct audio_output_position *pos); |
| 功能描述 | 查询 audio sample sync 模块当前输出的位置信息 |
| 参数说明 | 3、s: audio_sample_sync 结构指针<br>4、pos：输出位置结构指针，位置为整样点对应的蓝牙时间<br>struct audio_output_position {<br>u32 out_samples; //输出样点个数<br>u32 bt_time; //蓝牙时间<br>u16 bt_time_phase; //蓝牙时间相位<br>}; |
| 输出 | 0:正常<br>非 0：异常（暂时无非 0 返回） |
| 例子 | {<br>struct audio_output_position pos;<br>audio_sample_sync_get_out_position(ch->sample_ch, &pos);<br>} |
| 关联模块 | DAC/IIS/... |
| 补充说明 | 不支持任意蓝牙时间对应带小数位样点位置 |

| | |
|---|---|
| 函数原型 | int audio_sample_sync_flush_data(struct audio_sample_sync *s); |
| 功能描述 | 强制刷新当前 sample sync 处于变采样的数据 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 参数说明 | 1、s: audio_sample_sync 结构指针 |
| 输出 | 0:正常<br>非 0：异常（暂时无非 0 返回） |
| 例子 | {<br>audio_sample_sync_flush_data(ch->sample_ch);<br>...//计算位置信息<br>} |
| 关联模块 | DAC/IIS/...，SRC_BASE |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | void audio_sample_sync_position_correct(struct audio_sample_sync *s, int num); |
| 功能描述 | audio sample sync 位置纠正 |
| 参数说明 | 1、s: audio_sample_sync 结构指针<br>2、num : 位置纠正的个数 |
| 输出 | 无返回值 |
| 例子 | {<br>...//统计 TWS 之间输入位置偏差<br>audio_sample_sync_position_correct(ch->sample_ch, tws_diff_num);<br>} |
| 关联模块 | DAC/IIS/...，SRC_BASE |
| 补充说明 | 该功能仅在 TWS 位置对齐时会使用 |

| | |
|---|---|
| 函数原型 | int audio_sample_sync_align_control(struct audio_sample_sync *s, int in_rate, int out_rate, int points, s16 phase_diff); |
| 功能描述 | audio sample sync 控制快速对齐功能 |
| 参数说明 | 1、s: audio_sample_sync 结构指针<br>2、in_rate : 输入速率<br>3、out_rate : 输出速率<br>4、ponits : 快速对齐输出的样点<br>5、phase_diff:相位差（目前不再使用） |
| 输出 | 目前只有 0 返回值 |
| 例子 | {<br>...//计算 TWS 之间位置差值及需要调整的速度<br>audio_sample_sync_align_control(ch->sample_ch, src_in_rate, src_out_rate, fast_align_num, 0);<br>} |




地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 关联模块 | DAC/IIS/...，SRC_BASE |
| 补充说明 | 该功能仅在 TWS 位置对齐时会使用 |

| | |
|---|---|
| 函数原型 | int audio_sample_sync_rate_control(struct audio_sample_sync *s, int in_rate, int out_rate); |
| 功能描述 | audio sample sync 本地输出速率控制 |
| 参数说明 | 1、s: audio_sample_sync 结构指针<br>2、in_rate : 输入速率<br>3、out_rate : 输出速率 |
| 输出 | 目前只有 0 返回值 |
| 例子 | {<br>.....//计算本地跟随蓝牙的速度结果<br>audio_sample_sync_rate_control(ch->sample_ch, ctx->base_in_rate, sample_speed);<br>} |
| 关联模块 | DAC/IIS/...，SRC_BASE |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int audio_sample_sync_set_bt_time(struct audio_sample_sync *s, u32 bt_clkn, int phase); |
| 功能描述 | audio sample sync 按蓝牙时间开启输出的配置 |
| 参数说明 | 1、s: audio_sample_sync 结构指针<br>2、bt_clkn : 蓝牙时钟 clkn<br>3、phase : 蓝牙时钟相位 |
| 输出 | 目前只有 0 返回值 |
| 例子 | {<br>...<br>u32 bt_clkn;<br>u16 phase;<br>bt_clkn = tws_bt_time_to_local_time(time, &phase);<br><br>audio_sample_sync_set_bt_time(ch->sample_ch, bt_clkn, phase);<br>} |
| 关联模块 | DAC/IIS/...，SRC_BASE |
| 补充说明 | 该功能仅在 TWS 起始对齐时会使用 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

## 5.4. DAC/IIS 配置同步接口

### 5.4.1. 功能介绍

各个硬件输出模块添加 audio sample sync 进行同步，配置 audio sample sync 作为同步 SRC 和同步信息的处理媒介。

### 5.4.2. DAC 同步接口

| | |
|---|---|
| 函数原型 | int audio_dac_add_sample_sync(struct audio_dac_hdl *dac, void *sample_sync, u8 in_dac); |
| 功能描述 | DAC 输出加入音频同步 |
| 参数说明 | 1、dac : dac 结构 handle 指针<br>2、sample_sync : 打开的 audio sample sync 指针<br>3、in_dac : 选择 audio sample sync 是否内置到 DAC |
| 输出 | 0 : 成功<br>非 0 : audio sample sync 加入到 DAC 失败返回 |
| 例子 | {<br>void *sample_sync = audio_sample_sync_open(0);<br>/*选择 SYNC 模块接入到 DAC 内部*/<br>audio_dac_add_sample_sync(&dac_hdl, sample_sync, 1);<br>} |
| 关联模块 | DAC, audio sample sync |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int audio_dac_remove_sample_sync(struct audio_dac_hdl *dac, void *sample_sync); |
| 功能描述 | DAC 删除同步变采样模块 |
| 参数说明 | 1、dac : dac 结构 handle 指针<br>2、sample_sync : 配置的 audio sample sync 指针 |
| 输出 | 0 : 成功<br>非 0 : 删除 audio sample sync 失败 |
| 例子 | {<br>audio_dac_remove_sample_sync(&dac_hdl, a2dp_sync->sample_sync);<br>} |
| 关联模块 | DAC, audio sample sync |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| 补充说明 | |
|---|---|

### 5.4.3. IIS 同步接口

| 函数原型 | int audio_iis_add_sample_sync(struct audio_iis_hdl *iis, void *sample_sync, u8 in_iis); |
|---|---|
| 功能描述 | DAC 输出加入音频同步 |
| 参数说明 | 1、iis : iis 结构 handle 指针<br>2、sample_sync : 打开的 audio sample sync 指针<br>3、in_iis : 选择 audio sample sync 是否内置到 IIS |
| 输出 | 0 : 成功<br>非 0 : audio sample sync 加入到 IIS 失败返回 |
| 例子 | {<br>void *sample_sync = audio_sample_sync_open(0);<br>/*选择 SYNC 模块接入到 IIS 内部*/<br>audio_iis_add_sample_sync(&iis_hdl, sample_sync, 1);<br>} |
| 关联模块 | IIS, audio sample sync |
| 补充说明 | |

| 函数原型 | int audio_iis_remove_sample_sync(struct audio_iis_hdl *iis, void *sample_sync); |
|---|---|
| 功能描述 | IIS 删除同步变采样模块 |
| 参数说明 | 1、iis : iis 结构 handle 指针<br>2、sample_sync : 配置的 audio sample sync 指针 |
| 输出 | 0 : 成功<br>非 0 : 删除 audio sample sync 失败 |
| 例子 | {<br>audio_iis_remove_sample_sync(&iis_hdl, dec->sample_sync);<br>} |
| 关联模块 | IIS, audio sample sync |
| 补充说明 | |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

# 6. RESAMPLE 模块功能介绍

## 6.1. RESAMPLE 通用接口

### 6.1.1. 功能介绍

Resample 模块为音频变采样模块，包含接入数据流的变采样以及直接使用的单次变采样功能。

### 6.1.2. 接口介绍

#### 6.1.2.1. Resample 打开

| | |
|---|---|
| 函数原型 | struct audio_resample_context *audio_resample_open(u8 ch_num,<br>u8 hardware_first,<br>void *output_priv,<br>int (*output_handler)(void *priv, void *data, int len)); |
| 功能描述 | resample 模块打开和初始化 |
| 参数说明 | 1、ch_num : 音频数据通道个数，一接入数据流的需要配置<br>2、hardware_first : 优先使用硬件变采样模块<br>3、output_priv : 配置输出私有数据指针<br>4、output_handler : 配置输出回调 handle 的函数指针 |
| 输出 | 返回 struct audio_resample_contex 结构指针<br>NULL 为打开失败 |
| 例子 | static int resample_output_data_handler(void *priv, void *data, int len)<br>{<br><br>}<br><br>struct audio_resample_context *ctx = audio_resample_open(2, 1, priv,<br>resample_output_data_handler); |
| 关联模块 | SRC |
| 补充说明 | |

#### 6.1.2.2. 设置数据流采样率

| | |
|---|---|
| 函数原型 | int audio_resample_set_sample_rate(struct audio_resample_context *ctx, int |




地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | in_sample_rate, int out_sample_rate); |
| 功能描述 | resample 模块数据流配置输入输出采样率 |
| 参数说明 | 1、ctx : resample context 指针<br>2、in_sample_rate: 输入采样率<br>3、out_sample_rate : 输出采样率 |
| 输出 | 0: 配置成功<br>非 0: 配置失败 |
| 例子 | {<br>struct audio_resample_context *ctx = audio_resample_open(2, 1, priv, resample_output_data_handler);<br>audio_resample_set_sample_rate(ctx, 16000, 44100);<br>} |
| 关联模块 | SRC |
| 补充说明 | 数据流类型的音频变采样必须设置采样率 |

#### 6.1.2.3. 变采样写入

| | |
|---|---|
| 函数原型 | int audio_resample_stream_write(struct audio_resample_context *ctx, void *data, int len); |
| 功能描述 | resample 模块数据流写入 |
| 参数说明 | 1、ctx : resample context 指针<br>2、data: 写入数据地址<br>3、len : 写入数据长度 |
| 输出 | 返回写入的长度 |
| 例子 | {<br>int len = 0;<br>len = audio_resample_stream_write(ctx, data, len);<br>} |
| 关联模块 | SRC |
| 补充说明 | |

#### 6.1.2.4. 单次变采样

| | |
|---|---|
| 函数原型 | int audio_resample(struct audio_resample_context *ctx,<br>struct audio_resample_data *in,<br>struct audio_resample_data *out); |



地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 功能描述 | resample 模块数据流写入 |
| 参数说明 | 1、ctx : resample context 指针<br>2、in: 输入的变采样数据<br>3、out: 输出的变采样数据<br>struct audio_resample_data {<br>void *buffer; //变采样数据缓冲地址<br>int sample_rate; //采样率<br>int sample_num; //缓冲内样点个数<br>int offset; //已处理的样点偏移<br>u8 ch_num; //样点的通道个数<br>}; |
| 输出 | 0:变采样成功<br>非 0:变采样中途失败 |
| 例子 | {<br>struct audio_resample_data in;<br>struct audio_resample_data out;<br><br>in.buffer = input_data;<br>in.count = (input_len >> 1) / hdl->channel;<br>in.ch_num = hdl->channel;<br>in.sample_rate = in_sr;<br>in.offset = 0;<br><br>out.buffer = output_buffer;<br>out.count = output_buf_len / 2 / hdl->channel;<br>out.ch_num = hdl->channel;<br>out.sample_rate = out_sr;<br>out.offset = 0;<br>audio_resample(hdl->resample_ctx, &in, &out);<br>/*in.offset 为输入消耗到的样点，out.offset 为输出到的样点*/<br>} |
| 关联模块 | SRC |
| 补充说明 | |

### 6.1.2.5. 微调同步

| | |
|---|---|
| 函数原型 | int audio_resample_set_sample_sync(struct audio_resample_context *ctx, int in_correct_rate, int out_correct_rate) ; |
| 功能描述 | resample 模块数据流微调采样率同步 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 参数说明 | 1、ctx : resample context 指针<br>2、in_correct_rate: 微调输入采样率<br>3、out_correct_rate : 微调输出采样率 |
| 输出 | 0: 配置成功<br>非 0: 配置失败 |
| 例子 | {<br>struct audio_resample_context *ctx = audio_resample_open(2, 1, priv, resample_output_data_handler);<br>audio_resample_set_sample_rate(ctx, 16000, 44100);<br>audio_resample_set_sample_sync(ctx, 1, 18);<br>} |
| 关联模块 | SRC |
| 补充说明 | 必须先设置输入输出采样率后才可以同步，否则设置无效 |

#### 6.1.2.6. 模块关闭

| | |
|---|---|
| 函数原型 | void audio_resample_close(struct audio_resample_context *ctx); |
| 功能描述 | resample 模块关闭 |
| 参数说明 | ctx : resample context 指针 |
| 输出 | 无返回值 |
| 例子 | audio_resample_close(hdl->resample_ctx); |
| 关联模块 | SRC |
| 补充说明 | |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

# 7. DAC 模块功能介绍

## 7.1. DAC 通用接口

### 7.1.1. 功能介绍

dac 模块主要负责把不同采样率的音频数字信号转换为模拟信号输出，同时支持调节模块的数字增益和模拟增益，都可以达到调节音量的功能。 695x 的 dac 模块使用一个环形 buffer 与 cpu 进行音频数据交互，环形 buffer 由 cpu 分配配置到 dac 寄存器中，dac 数字模块启动后就会通过 dma 方式从环形 buffer 读出音频数据然后转换为模拟信号输出。

数据流部分作为输出模块接入 audio stream 使用，增益调节部分由音量控制部分或应用层单独调用。

### 7.1.2. 接口介绍

#### 7.1.2.1. 初始化

| | |
|---|---|
| 函数原型 | int audio_dac_init(struct audio_dac_hdl *dac, const struct dac_platform_data *pd) |
| 功能描述 | 初始化 dac 的配置参数和相关基础配置 |
| 参数说明 | * \param[in] dac // audio dac 句柄，应用层申请传入<br>* \param[in] pd // audio dac 参数配置数组，在板级文件中配置 |
| 输出 | * \return 返回操作结果<br>* \retval 0 初始化完成<br>* \retval -1 初始化出错 |
| 例子 | audio_dac_init(&dac_hdl, &dac_data); |
| 关联模块 | app_audio.c |
| 补充说明 | |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

#### 7.1.2.2. 设置 buffer

| | |
|---|---|
| 函数原型 | int audio_dac_set_buff(struct audio_dac_hdl *dac, s16 *buf, int len) |
| 功能描述 | 配置 dac 模块使用的 buffer |
| 参数说明 | * \param[in] dac // audio dac 句柄，应用层申请传入<br>* \param[in] buf // buffer 地址<br>* \param[in] len // buffer 长度字节数 |
| 输出 | * \return 返回操作结果<br>* \retval 0 初始化完成<br>* \retval -1 初始化出错 |
| 例子 | audio_dac_set_buff(&dac_hdl, dac_buff, sizeof(dac_buff)); |
| 关联模块 | app_audio.c |
| 补充说明 | 696x dac 模块内部会通过一个环形结构 buffer 与 cpu 进行音频数据交互，此函数是设置这个 buffer 的地址和长度供 dac 模块使用。 |

#### 7.1.2.3. dac trim 校准

| | |
|---|---|
| 函数原型 | int audio_dac_do_trim(struct audio_dac_hdl *dac, struct audio_dac_trim *dac_trim, u8 fast_trim) |
| 功能描述 | dac 模块 trim 校准 |
| 参数说明 | * \param[in] dac // audio dac 句柄，应用层申请传入<br>* \param[out] dac_trim // 保存 trim 值的结构体，调用时传入地址<br>* \param[in] fast_trim// 是否启用快速校准 |
| 输出 | * \return 返回操作结果<br>* \retval 0 初始化完成<br>* \retval -1 初始化出错 |
| 例子 | audio_dac_do_trim(&dac_hdl, &dac_trim, 0); |
| 关联模块 | app_audio.c, vm |
| 补充说明 | 由于每个芯片的个体差异，dac 模块需要通过校准消除这个偏差，可以在第一次开机校准一次，后面可以直接从 flash 读出来写入 trim 寄存器 |

| | |
|---|---|
| 函数原型 | int audio_dac_set_trim_value(struct audio_dac_hdl *dac, struct audio_dac_trim *dac_trim) |
| 功能描述 | 将 trim 值写入 dac 寄存器 |
| 参数说明 | * \param[in] dac // audio dac 句柄，应用层申请传入<br>* \param[in] dac_trim // 保存 trim 值的结构体，调用时传入地址 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 输出 | * \return 返回操作结果<br>* \retval 0 初始化完成<br>* \retval -1 初始化出错 |
| 例子 | audio_dac_set_trim_value(&dac_hdl, &dac_trim); |
| 关联模块 | app_audio.c, vm |
| 补充说明 | |

#### 7.1.2.4. 设置启动延时和最大延时

| | |
|---|---|
| 函数原型 | int audio_dac_set_delay_time(struct audio_dac_hdl *dac, int start_ms, int max_ms) |
| 功能描述 | 设置 dac 模块启动时间和最大延时 |
| 参数说明 | * \param[in] dac // audio dac 句柄，应用层申请传入<br>* \param[in] start_ms // 启动时间，dac 环形 buffer 长度达到这个时间才会启动 dac<br>* \param[in] max_ms // 最大延时，dac 环形 buffer 中限制容纳的最大的数据长度 |
| 输出 | * \return 返回操作结果<br>* \retval 0 初始化完成<br>* \retval -1 初始化出错 |
| 例子 | audio_dac_set_delay_time(&dac_hdl, 30, 50); |
| 关联模块 | app_audio.c |
| 补充说明 | |

#### 7.1.2.5. 设置与获取 dac 采样率

| | |
|---|---|
| 函数原型 | int audio_dac_get_sample_rate(struct audio_dac_hdl *dac) |
| 功能描述 | 获取 dac 采样率 |
| 参数说明 | * \param[in] dac // audio dac 句柄，应用层申请传入<br>* \param[in] sample_rate // 设置 dac 采样率 |
| 输出 | * \return 返回操作结果<br>* \retval 当前 dac 的采样率 |
| 例子 | sample_rate = audio_dac_get_sample_rate(&dac_hdl); |
| 关联模块 | app_audio.c,audio_stream |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int audio_dac_set_sample_rate(struct audio_dac_hdl *dac, int sample_rate) |
| 功能描述 | 设置 dac 采样率 |
| 参数说明 | * \param[in] dac // audio dac 句柄，应用层申请传入 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | * \param[in] sample_rate // 设置 dac 采样率 |
| 输出 | * \return 返回操作结果<br>* \retval 0 初始化完成<br>* \retval -1 初始化出错 |
| 例子 | audio_dac_set_sample_rate(&dac_hdl, 44100); |
| 关联模块 | app_audio.c,audio_stream |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int audio_dac_sample_rate_select(struct audio_dac_hdl *dac, u32 sample_rate, u8 high) |
| 功能描述 | 匹配 dac 支持的采样率 |
| 参数说明 | * \param[in] dac // audio dac 句柄，应用层申请传入<br>* \param[in] sample_rate // 设置 dac 采样率<br>* \param[in] high // 0 - 低一级采样率，1 - 高一级采样率 |
| 输出 | * \return 返回操作结果<br>* \retval 匹配后的 dac 支持的采样率 |
| 例子 | Sample_rate = audio_dac_sample_rate_select(&dac_hdl,44100, 1); |
| 关联模块 | app_audio.c,audio_stream |
| 补充说明 | |

### 7.1.2.6. 关闭 dac 模块

| | |
|---|---|
| 函数原型 | int audio_dac_close(struct audio_dac_hdl *dac) |
| 功能描述 | 关闭 dac 模块 |
| 参数说明 | * \param[in] dac // audio dac 句柄，应用层申请传入 |
| 输出 | * \return 返回操作结果<br>* \retval 0 初始化完成<br>* \retval -1 初始化出错 |
| 例子 | audio_dac_close(&dac_hdl); |
| 关联模块 | app_audio.c |
| 补充说明 | |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

### 7.1.3. 输出和启停

#### 7.1.3.1. dac 数字输出打开

| 函数原型 | int audio_dac_start(struct audio_dac_hdl *dac) |
|---|---|
| 功能描述 | dac 数字输出打开 |
| 参数说明 | * \param[in] dac // audio dac 句柄，应用层申请传入 |
| 输出 | * \return 返回操作结果<br>* \retval 0 初始化完成<br>* \retval -1 初始化出错 |
| 例子 | audio_dac_start(&dac_hdl); |
| 关联模块 | app_audio.c |
| 补充说明 | |

#### 7.1.3.2. dac 数字输出关闭

| 函数原型 | int audio_dac_stop(struct audio_dac_hdl *dac) |
|---|---|
| 功能描述 | dac 数字输出关闭 |
| 参数说明 | * \param[in] dac // audio dac 句柄，应用层申请传入 |
| 输出 | * \return 返回操作结果<br>* \retval 0 初始化完成<br>* \retval -1 初始化出错 |
| 例子 | audio_dac_stop(&dac_hdl); |
| 关联模块 | app_audio.c |
| 补充说明 | |

#### 7.1.3.3. dac 写入音频数据

| 函数原型 | int audio_dac_write(struct audio_dac_hdl *dac, void *buf, int len) |
|---|---|
| 功能描述 | dac 数字输出关闭 |
| 参数说明 | * \param[in] dac // audio dac 句柄，应用层申请传入<br>* \param[in] buf // 音频数据的地址<br>* \param[in] len // 音频数据字节长度 |
| 输出 | * \return 返回操作结果<br>* \retval 成功写入的长度 |



地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 例子 | rlen = audio_dac_write(&dac_hdl, buf, len); |
| 关联模块 | app_audio.c，audio_stream |
| 补充说明 | |

### 7.1.4. 增益调节

#### 7.1.4.1. dac 模拟增益设置

| | |
|---|---|
| 函数原型 | int audio_dac_ch_analog_gain_set(struct audio_dac_hdl *dac, u8 ch, u8 again) |
| 功能描述 | dac 模拟增益设置 |
| 参数说明 | * \param[in] dac // audio dac 句柄，应用层申请传入<br>* \param[in] ch // 通道，需要设置的通道就把对应 BIT 置一，例如 0x01 说明这次只设置 通道 0 (dac out L)<br>* \param[in] again // 设置的模拟增益 (范围：0-30) |
| 输出 | * \return 返回操作结果<br>* \retval 0 初始化完成<br>* \retval -1 初始化出错 |
| 例子 | audio_dac_ch_analog_gain_set(&dac_hdl, BIT(0)\|BIT(1), 15); |
| 关联模块 | app_audio.c, audio_vol |
| 补充说明 | |

#### 7.1.4.2. dac 数字增益设置

| | |
|---|---|
| 函数原型 | int audio_dac_ch_digital_gain_set(struct audio_dac_hdl *dac, u8 ch, u32 dgain) |
| 功能描述 | dac 模拟增益设置 |
| 参数说明 | * \param[in] dac // audio dac 句柄，应用层申请传入<br>* \param[in] ch // 通道，需要设置的通道就把对应 bit 置一，例如 0x01 说明这次只设置 通道 0 (dac out L)<br>* \param[in] dgain // 设置的数字增益 (范围：0-16384) |
| 输出 | * \return 返回操作结果<br>* \retval 0 初始化完成<br>* \retval -1 初始化出错 |
| 例子 | audio_dac_ch_digital_gain_set(&dac_hdl, BIT(0)\|BIT(1), 16384); |
| 关联模块 | app_audio.c, audio_vol |
| 补充说明 | |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

### 7.1.5. audio stream 节点接入

在 dac 初始化(audio_dac_init)后，可以使用 dac 句柄中的 entry 成员接入 audio stream。
使用例子：

```
entries[entry_cnt++] = &dac_hdl.entry;
```

### 7.1.6. 其他接口

#### 7.1.6.1. dac 输出引脚设置高阻

| | |
|---|---|
| 函数原型 | void audio_dac_ch_mute(struct audio_dac_hdl *dac, u8 ch, u8 mute) |
| 功能描述 | dac 输出引脚设成高阻 |
| 参数说明 | * \param[in] dac // audio dac 句柄，应用层申请传入<br>* \param[in] ch // 通道，需要设置的通道就把对应 bit 置一，例如 0x01 说明这次只设置 通道 0 (dac out L)<br>* \param[in] mute // 1：高阻 0：正常 |
| 输出 | |
| 例子 | audio_dac_ch_mute(&dac_hdl, BIT(0)\|BIT(1), 1); |
| 关联模块 | app_audio.c, automute |
| 补充说明 | |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

# 8. MIXER 模块功能介绍

## 8.1. MIXER 接口

### 8.1.1. 功能介绍

该模块用于多个通道叠加成一个通道。

所有通道都填满时才往 mixer 下一级输出，但可以设置超时忽略该通道。

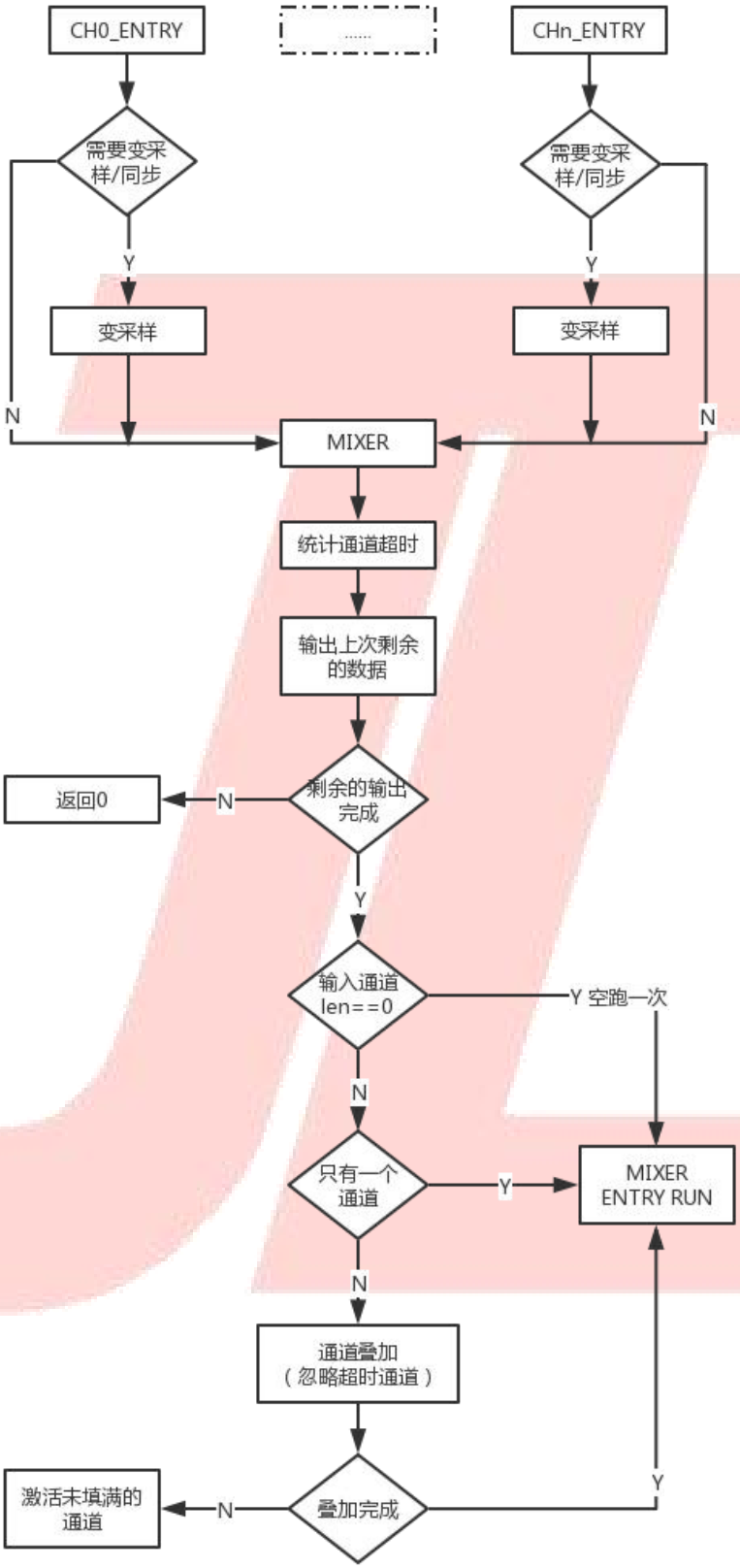
Mixer 模块可以根据通道采样率和 mixer 模块采样率进行自动变采样；可以根据设置的通道超时，叠加时自动忽略已经超时的通道。

Mixer 模块关联的其他模块有：SRC，audio_stream。

Mixer 模块框架如下图所示：



地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com







地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

### 8.1.2. 数据结构介绍

```
 // mixer Buf 同步设置参数
struct audio_mixer_ch_sync_info {
    int begin_per;      // 起始百分比
    int top_per;        // 最大百分比
    int bottom_per;         // 最小百分比
    u8 inc_step;        // 每次调整增加步伐
    u8 dec_step;        // 每次调整减少步伐
    u8 max_step;        // 最大调整步伐
    void *priv;             // get_total,get_size 私有句柄
    int (*get_total)(void *priv);   // 获取 buf 总长
    int (*get_size)(void *priv);    // 获取 buf 数据长度
};

// mixer
struct audio_mixer {
    struct list_head head;      // 链表头
    s16 *output;        // 输出 buf
    u16 points;             // 输出 buf 总点数
    u16 remain_points; // 输出剩余点数
    u32 process_len;    // 输出了多少
    u8  channel_num; // 声道数
    u8  sample_sync; // 空中包数据同步标记
    u16 sample_rate;    // 当前 mixer 的采样率
    MIXER_SR_TYPE sr_type;  // 采样率设置类型
    void (*evt_handler)(struct audio_mixer *, int);     // 事件返回接口
    u16(*check_sr)(struct audio_mixer *, u16 sr);       // 检查采样率
    volatile u8 active;   // 活动标记。1-正在运行
    struct audio_stream *stream;        // 音频流
    struct audio_stream_entry entry;    // 音频流入口，后级接 dac 等
    struct audio_stream_group group; // 用于链接前级
};

// mixer 通道
struct audio_mixer_ch {
    u32 start : 1;          // 启动标记。1-已经启动。输出时标记一次
    u32 open : 1;           // 打开标记。1-已经打开。输出标记为 1，reset 标记为 0
    u32 pause : 1;          // 暂停标记。1-暂停
    u32 no_wait : 1;        // 不等待有数
```



地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

```
    u32 lose : 1;          // 丢数标记
    u32 src_en : 1;        // 变采样使能
    u32 src_always : 1; // 不管采样率是否相同都做变采样
    u32 sample_sync : 1;// 空中包数据同步标记
    u32 sync_en : 1;       // 同步使能
    u32 sync_always : 1;// 不管采样率是否相同都做同步
    u16 offset;                 // 当前通道在输出 buf 中的偏移位置
    u16 sample_rate;    // 当前通道采样率
    u16 lose_time;      // 超过该时间还没有数据，则以为可以丢数。no_wait 置 1 有效
    unsigned long lose_limit_time;     // 丢数超时中间运算变量
    struct list_head list_entry;       // 链表
    struct audio_mixer *mixer;     // mixer 句柄
    struct audio_src_handle *src; // 变采样
    struct audio_buf_sync_hdl sync;    // 同步
    struct audio_mixer_ch_sync_info sync_info;// 同步参数
    void *priv;            // 事件回调私有句柄
    void (*event_handler)(void *priv, int event, int param);      // 事件回调接口
    struct audio_stream_entry entry;   // 音频流入口，通道后面不应该再接其他的音频流，最后由 mixer 合并后输出
};
```

### 8.1.3. 接口介绍

| 函数原型 | int audio_mixer_open(struct audio_mixer *mixer) |
|---|---|
| 功能描述 | 打开一个 mixer 模块 |
| 参数说明 | * \param[in][out]　　*mixer　// mixer 模块句柄 |
| 输出 | * \return　　返回相应的操作消息处理值<br>* \retval　　非 0. 打开失败<br>* \retval　　0 打开成功 |
| 例子 | Ret = audio_mixer_open(&test_mixer); |
| 关联模块 | |
| 补充说明 | 和 audio_mixer_close()函数配套使用 |

| 函数原型 | void audio_mixer_close(struct audio_mixer *mixer) |
|---|---|
| 功能描述 | 关闭一个 mixer 模块 |
| 参数说明 | * \param[in][out]　　*mixer　// mixer 模块句柄 |
| 输出 | * \return　　无 |
| 例子 | audio_mixer_close(&test_mixer); |



地址：珠海市吉大石花西路 107 号 9 栋综合楼　邮编：519015
电话：0756-6313088　传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 关联模块 | |
| 补充说明 | 和 audio_mixer_open()函数配套使用 |

| | |
|---|---|
| 函数原型 | void audio_mixer_set_event_handler(struct audio_mixer *mixer,<br>void (*handler)(struct audio_mixer *, int)) |
| 功能描述 | 设置 Mixer 模块事件回调 |
| 参数说明 | * \param[in][out] *mixer // mixer 模块句柄<br>* \param[in] *handler // mixer 事件回调<br>enum {<br>MIXER_EVENT_OPEN, // 打开一个通道<br>MIXER_EVENT_CLOSE, // 关闭一个通道<br>MIXER_EVENT_SR_CHANGE, // mixer 采样率变化<br>}; |
| 输出 | * \return 无 |
| 例子 | static void test_mixer_event_handler(struct audio_mixer *mixer, int event)<br>{<br>switch (event) {<br>case MIXER_EVENT_OPEN:<br>break;<br>case MIXER_EVENT_CLOSE:<br>break;<br>.....<br>}<br>}<br>audio_mixer_set_event_handler(&test_mixer, test_mixer_event_handler); |
| 关联模块 | |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | void audio_mixer_set_check_sr_handler(struct audio_mixer *mixer,<br>u16(*check_sr)(struct audio_mixer *, u16)) |
| 功能描述 | 设置采样率检测回调 |
| 参数说明 | * \param[in][out] *mixer // mixer 模块句柄<br>* \param[in] *check_sr // mixer 采样率检测回调 |
| 输出 | * \return 无 |
| 例子 | Static u16 test_mixer_check_sr(struct audio_mixer *mixer, u16 sr)<br>{<br>If (sr <= 44100) {<br>Return 44100;<br>}<br>Return sr; |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | }<br>audio_mixer_set_check_sr_handler(&test_mixer, test_mixer_check_sr); |
| 关联模块 | |
| 补充说明 | 检测 mixer 模块采样率是否可用，并返回可以使用的采样率 |

| | |
|---|---|
| 函数原型 | void audio_mixer_set_output_buf(struct audio_mixer *mixer, s16 *buf, u16 len) |
| 功能描述 | 设置输出 buf |
| 参数说明 | * \param[in][out] *mixer // mixer 模块句柄<br>* \param[in] *buf // buf 地址<br>* \param[in] len // buf 长度 |
| 输出 | * \return 无 |
| 例子 | S16 test_mix_buf[512];<br>audio_mixer_set_output_buf(&test_mixer, test_mix_buf, sizeof(test_mix_buf)); |
| 关联模块 | |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | void audio_mixer_set_channel_num(struct audio_mixer *mixer, u8 channel_num) |
| 功能描述 | 设置 mixer 模块声道数 |
| 参数说明 | * \param[in][out] *mixer // mixer 模块句柄<br>* \param[in] channel_num // mixer 声道数 |
| 输出 | * \return 无 |
| 例子 | U8 ch_num = audio_output_channel_num();<br>audio_mixer_set_channel_num(&test_mixer, ch_num); |
| 关联模块 | |
| 补充说明 | 通道声道数需要和 mixer 声道数相同 |

| | |
|---|---|
| 函数原型 | void audio_mixer_set_sample_rate(struct audio_mixer *mixer, MIXER_SR_TYPE type, u16 sample_rate) |
| 功能描述 | 设置 mixer 采样率类型 |
| 参数说明 | * \param[in][out] *mixer // mixer 模块句柄<br>* \param[in] type // 采样率设置类型<br>typedef enum {<br>MIXER_SR_MAX = 0, // 最大采样率<br>MIXER_SR_MIN, // 最小采样率<br>MIXER_SR_FIRST, // 第一个通道的采样率<br>MIXER_SR_LAST, // 最后一个通道的采样率<br>MIXER_SR_SPEC, // set 指定采样率；get 当前 mixer 采样率<br>} MIXER_SR_TYPE;<br>* \param[in] sample_rate // mixer 采样率（MIXER_SR_SPEC 有效） |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 输出 | * \return 无 |
| 例子 | audio_mixer_set_sample_rate(&test_mixer, MIXER_SR_SPEC, 44100); |
| 关联模块 | |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int audio_mixer_get_sample_rate_by_type(struct audio_mixer *mixer, MIXER_SR_TYPE type) |
| 功能描述 | 按条件获取 mixer 采样率 |
| 参数说明 | * \param[in][out] *mixer // mixer 模块句柄<br>* \param[in] type // 采样率获取类型<br>typedef enum {<br>MIXER_SR_MAX = 0, // 最大采样率<br>MIXER_SR_MIN, // 最小采样率<br>MIXER_SR_FIRST, // 第一个通道的采样率<br>MIXER_SR_LAST, // 最后一个通道的采样率<br>MIXER_SR_SPEC, // set 指定采样率；get 当前 mixer 采样率<br>} MIXER_SR_TYPE; |
| 输出 | * \return 采样率 |
| 例子 | U16 sr = audio_mixer_get_sample_rate_by_type(&test_mixer,MIXER_SR_MAX); |
| 关联模块 | |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int audio_mixer_get_sample_rate(struct audio_mixer *mixer) |
| 功能描述 | 获取采样率（按 mixer->sr_type 获取） |
| 参数说明 | * \param[in][out] *mixer // mixer 模块句柄 |
| 输出 | * \return 采样率 |
| 例子 | U16 sr = audio_mixer_get_sample_rate(&test_mixer); |
| 关联模块 | |
| 补充说明 | mixer->sr_type 是 audio_mixer_set_sample_rate()函数设置的类型，默认为 0 |

| | |
|---|---|
| 函数原型 | int audio_mixer_get_ch_num(struct audio_mixer *mixer) |
| 功能描述 | 获取通道总数 |
| 参数说明 | * \param[in][out] *mixer // mixer 模块句柄 |
| 输出 | * \return 通道数 |
| 例子 | int mix_num = audio_mixer_get_ch_num(&test_mixer); |
| 关联模块 | |
| 补充说明 | |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 函数原型 | int audio_mixer_get_active_ch_num(struct audio_mixer *mixer) |
| 功能描述 | 获取非暂停通道总数 |
| 参数说明 | * \param[in][out] *mixer // mixer 模块句柄 |
| 输出 | * \return 通道数 |
| 例子 | int mix_num = audio_mixer_get_active_ch_num(&test_mixer); |
| 关联模块 | |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int audio_mixer_data_len(struct audio_mixer *mixer) |
| 功能描述 | 获取 mixer 剩余长度 |
| 参数说明 | * \param[in][out] *mixer // mixer 模块句柄 |
| 输出 | * \return 数据长度 |
| 例子 | int mix_data = audio_mixer_data_len(&test_mixer); |
| 关联模块 | |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int audio_mixer_ch_open(struct audio_mixer_ch *ch, struct audio_mixer *mixer); |
| 功能描述 | 打开一个 mixer 通道（放在链表结尾处） |
| 参数说明 | * \param[in][out] *ch // mixer 通道句柄<br>* \param[in][out] *mixer // mixer 模块句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 打开失败<br>* \retval 0 打开成功 |
| 例子 | Ret = audio_mixer_ch_open(&test_ch0, &test_mixer); |
| 关联模块 | |
| 补充说明 | 和 audio_mixer_ch_close()函数配套使用 |

| | |
|---|---|
| 函数原型 | int audio_mixer_ch_open_head(struct audio_mixer_ch *ch, struct audio_mixer *mixer); |
| 功能描述 | 打开一个 mixer 通道（放在链表起始处） |
| 参数说明 | * \param[in][out] *ch // mixer 通道句柄<br>* \param[in][out] *mixer // mixer 模块句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 打开失败<br>* \retval 0 打开成功 |
| 例子 | Ret = audio_mixer_ch_open_head(&test_ch0, &test_mixer); |
| 关联模块 | |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| 补充说明 | 和 audio_mixer_ch_close()函数配套使用 |
|---|---|

| 函数原型 | void audio_mixer_ch_close(struct audio_mixer_ch *ch) |
|---|---|
| 功能描述 | 关闭通道 |
| 参数说明 | * \param[in][out] *ch // mixer 通道句柄 |
| 输出 | * \return 无 |
| 例子 | audio_mixer_ch_close(&test_ch0); |
| 关联模块 | |
| 补充说明 | 和 audio_mixer_ch_open()或者 audio_mixer_ch_open_head()函数配套使用 |

| 函数原型 | void audio_mixer_ch_set_event_handler(struct audio_mixer_ch *ch, void *priv, void (*handler)(void *, int, int)); |
|---|---|
| 功能描述 | 设置通道事件回调 |
| 参数说明 | * \param[in][out] *ch // mixer 通道句柄<br>* \param[in] *priv // 通道事件回调私有参数<br>* \param[in] *handler // 通道事件回调<br>enum {// mixer 通道事件回调类型<br>MIXER_EVENT_CH_OPEN, // 通道打开<br>MIXER_EVENT_CH_CLOSE, // 通道关闭<br>MIXER_EVENT_CH_RESET, // 通道重新开始<br>MIXER_EVENT_CH_LOST, // 通道丢数（超时）<br>MIXER_EVENT_CH_OUT_SR_CHANGE, // mixer 采样率变化<br>}; |
| 输出 | * \return 无 |
| 例子 | Static void test_ch0_event_handler(void *priv, int event, int param)<br>{<br>Switch (event) {<br>Case MIXER_EVENT_CH_OPEN:<br>Break;<br>Case MIXER_EVENT_CH_LOST:<br>// printf(“lost len:%d \n”, param);<br>Break;<br>.....<br>}<br>audio_mixer_ch_set_event_handler(&test_ch0, NULL, test_ch0_event_handler); |
| 关联模块 | |
| 补充说明 | |

| 函数原型 | void audio_mixer_ch_set_sample_rate(struct audio_mixer_ch *ch, u16 sample_rate) |
|---|---|




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 功能描述 | 设置通道采样率 |
| 参数说明 | * \param[in][out] *ch // mixer 通道句柄<br>* \param[in] sample_rate // 采样率 |
| 输出 | * \return 无 |
| 例子 | audio_mixer_ch_set_sample_rate(&test_ch0, 16000); |
| 关联模块 | |
| 补充说明 | 可以不用设置，在数据流输出那里会根据采样率自动变化 |

| | |
|---|---|
| 函数原型 | void audio_mixer_ch_set_src(struct audio_mixer_ch *ch, u8 src_en, u8 always) |
| 功能描述 | 设置通道变采样 |
| 参数说明 | * \param[in][out] *ch // mixer 通道句柄<br>* \param[in] src_en // 变采样使能<br>* \param[in] always // 不管采样率是否相同，都要变采样 |
| 输出 | * \return 无 |
| 例子 | audio_mixer_ch_set_src(&test_ch0, 1, 0); |
| 关联模块 | |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | void audio_mixer_ch_set_sync(struct audio_mixer_ch *ch, struct audio_mixer_ch_sync_info *info, u8 sync_en, u8 always) |
| 功能描述 | 设置通道同步 |
| 参数说明 | * \param[in][out] *ch // mixer 通道句柄<br>* \param[in] *info // 同步参数<br>* \param[in] sync_en // 同步使能<br>* \param[in] always // 不管采样率是否相同，都要变采样 |
| 输出 | * \return 无 |
| 例子 | Static int test_get_total(void *priv)<br>{<br>Return total_len;<br>}<br>Static int test_get_cur(void *priv)<br>{<br>Return cur_len;<br>}<br>Struct audio_mixer_ch_sync_info info = {0};<br>Info.priv = NULL;<br>Info.get_total = test_get_total;<br>Info.get_size = test_get_cur;<br>audio_mixer_ch_set_sync(&test_ch0, &info, 1, 1); |
| 关联模块 | |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 补充说明 | 该同步是根据输入 buf 变化幅度动态更改变采样的采样率值 |

| | |
|---|---|
| 函数原型 | void audio_mixer_ch_set_no_wait(struct audio_mixer_ch *ch, u8 no_wait, u16 time_ms) |
| 功能描述 | 设置通道没数据时不等待 |
| 参数说明 | * \param[in][out] *ch // mixer 通道句柄<br>* \param[in] no_wait // 超时不等待使能<br>* \param[in] time_ms // 超时时长 |
| 输出 | * \return 无 |
| 例子 | audio_mixer_ch_set_no_wait(&test_ch0, 1, 30); |
| 关联模块 | |
| 补充说明 | 超时直接丢数，叠加不等待该通道填满 |

| | |
|---|---|
| 函数原型 | void audio_mixer_ch_pause(struct audio_mixer_ch *ch, u8 pause) |
| 功能描述 | 通道暂停 |
| 参数说明 | * \param[in][out] *ch // mixer 通道句柄<br>* \param[in] pause // 暂停使能 |
| 输出 | * \return 无 |
| 例子 | audio_mixer_ch_pause(&test_ch0, 1); |
| 关联模块 | |
| 补充说明 | 暂停后叠加不等待该通道填满 |

| | |
|---|---|
| 函数原型 | int audio_mixer_ch_reset(struct audio_mixer_ch *ch) |
| 功能描述 | 通道重启 |
| 参数说明 | * \param[in][out] *ch // mixer 通道句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | audio_mixer_ch_reset(&test_ch0); |
| 关联模块 | |
| 补充说明 | 该函数会清除已经写入 mixer 模块的数据 |

| | |
|---|---|
| 函数原型 | int audio_mixer_ch_data_len(struct audio_mixer_ch *ch) |
| 功能描述 | 获取通道剩余长度 |
| 参数说明 | * \param[in][out] *ch // mixer 通道句柄 |
| 输出 | * \return 通道数据长度 |
| 例子 | Len = audio_mixer_ch_data_len(&test_ch0); |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 关联模块 | |
| 补充说明 | 如果已经填到 mixer，则返回 mixer 剩余长度 |

| | |
|---|---|
| 函数原型 | int audio_mixer_ch_write(struct audio_mixer_ch *ch, s16 *data, int len) |
| 功能描述 | 通道数据输出 |
| 参数说明 | * \param[in][out] *ch // mixer 通道句柄<br>* \param[in] *data // 数据地址<br>* \param[in] len // 数据长度 |
| 输出 | * \return 实际输出长度 |
| 例子 | Len = audio_mixer_ch_write(&test_ch0, buf, sizeof(buf)); |
| 关联模块 | |
| 补充说明 | 在 audio_stream 数据流中会自动调用该函数 |

| | |
|---|---|
| 函数原型 | void audio_mixer_ch_sample_sync_enable(struct audio_mixer_ch *ch, u8 enable) |
| 功能描述 | 通道空中包数据同步使能 |
| 参数说明 | * \param[in][out] *ch // mixer 通道句柄<br>* \param[in] enable // 使能 |
| 输出 | * \return 无 |
| 例子 | audio_mixer_ch_sample_sync_enable(&test_ch0, 1); |
| 关联模块 | |
| 补充说明 | 用于在数据流中标记空中包（A2DP/ESCO 等）同步功能 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

# 9. DECODER 模块功能介绍

## 9.1. DECODER 接口

### 9.1.1. 功能介绍

该模块用于解码处理，可以轮询处理多个解码（链表结构）。

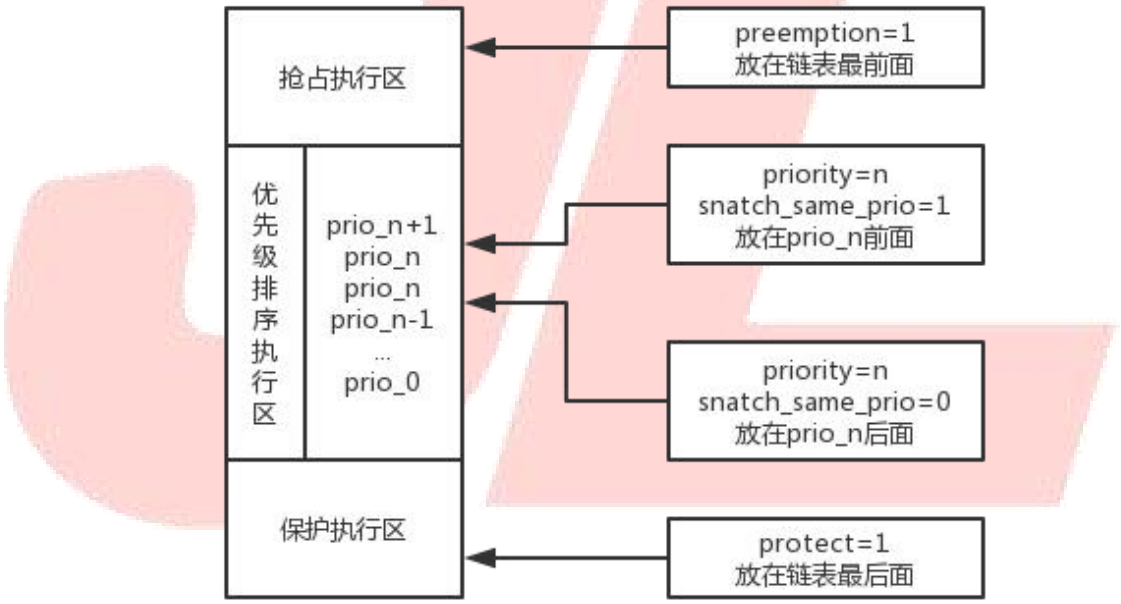
Decoder 支持文件解码（MP3/WMA 等）和 frame 解码（SBC/AAC 等），解码处理本身不关心是什么解码器，由上层应用指定。

Decoder 的解码格式可以由上层直接指定，也可以按条件自动查找解码格式（需解码格式有格式检查才行，如 WMA 有格式检查，可以自动查找；SBC 没有格式检查，不能自动查找）。

Decoder 后级输出不能完全输出时，会挂起该解码，需要激活才能继续解码。

Decoder 模块关联的其他模块有：audio_stream。

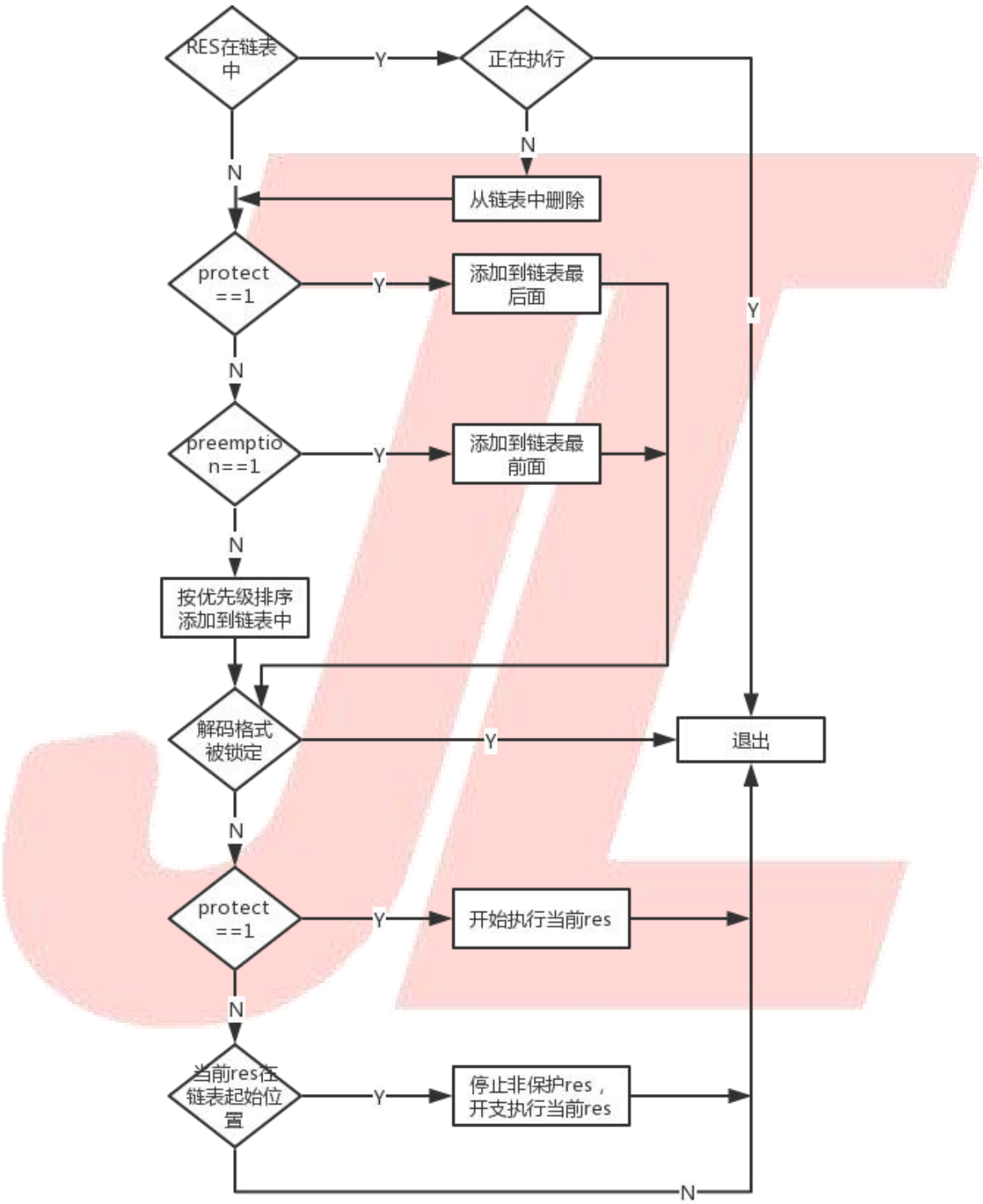
Decoder 链表（task->wait）结构如下图所示：






地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

Decoder 链表（task->wait）添加处理流程如下图所示：

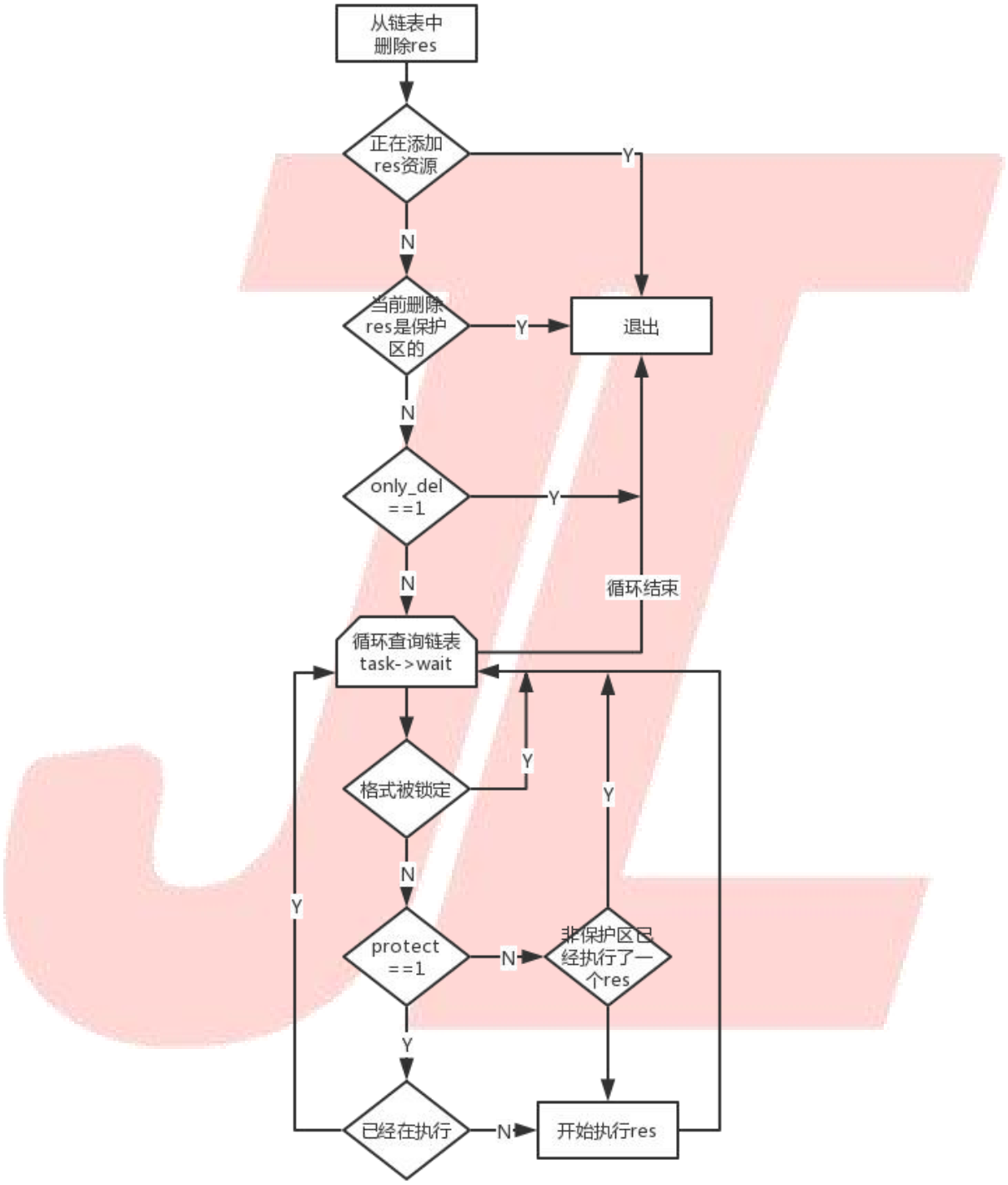






地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

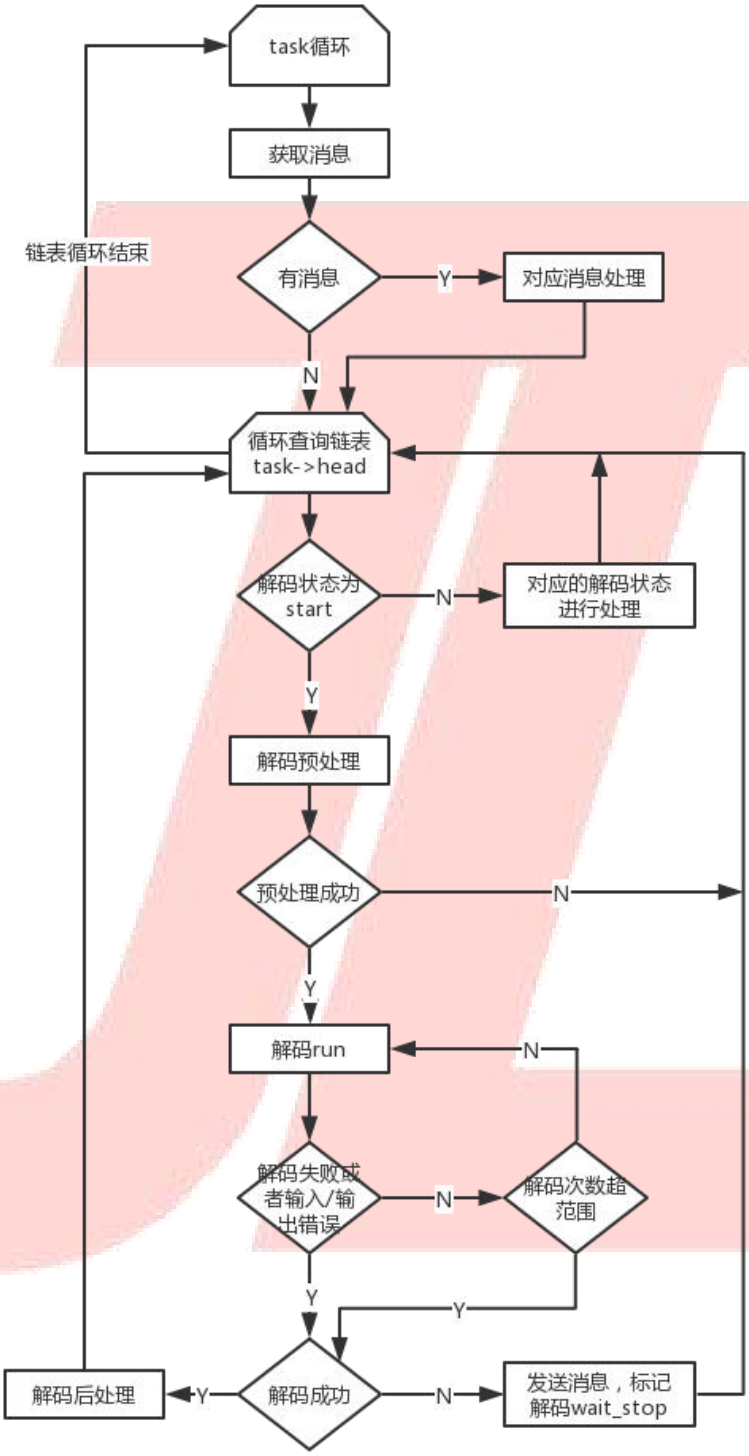
Decoder 链表（task->wait）删除处理流程如下图所示：






地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

Decoder 任务轮询链表（task->head）处理流程如下图所示：






地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

### 9.1.2. 数据结构介绍

```
// decoder 处理
struct audio_decoder {
    struct list_head list_entry;            // 链表。用于解码任务中轮询处理
    struct audio_decoder_task *task;        // 解码任务
    struct audio_fmt fmt;                   // 解码格式
    const char *evt_owner;                  // 用于接受消息的任务名称
    const struct audio_dec_input *input;                // 解码输入接口
    const struct audio_decoder_ops *dec_ops;            // 解码器接口
    const struct audio_dec_handler *dec_handler;        // 解码处理回调
    void (*evt_handler)(struct audio_decoder *dec, int, int *);  // 事件回调接口
    void *dec_priv;             // 解码器对应句柄
    void *bp;                   // 断点
    //u16 id;                   // ID 号
    u16 pick : 1;               // 本地拆包解码标记
    u16 tws : 1;                // 本地 tws 解码标记
    u16 resume_flag : 1;        // 解码激活标记
    u16 output_err : 1;         // 解码输出错误
    u16 read_err : 1;           // 解码读取错误
    u16 reserved : 11;          // 保留
    u8 run_max;                 // 正常解码最大次数
    //u8 output_channel;            // 解码输出通道
    u8 state;                   // 解码状态
    u8 err;                     // 解码结束错误类型标记
    u8 remain;                  // 解码输出完成标记
    u32 magic;                  // 事件回调的私有标记
    u32 process_len;            // 数据流处理长度
    struct audio_stream_entry entry;    // 音频流入口
};

// 解码任务
struct audio_decoder_task {
    struct list_head head;          // 用于解码任务中的轮询处理
    struct list_head wait;          // 用于多个解码抢占/排序等
    struct prevent_task_fill *prevent_fill;   // 防止解码任务一直占满 cpu
    const char *name;           // 任务名称
    int wakeup_timer;           // 定时唤醒
    int fmt_lock;           // 格式锁
```




地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

```
    int is_add_wait;     // 正在添加 res 资源
    OS_SEM sem;          // 任务执行一轮 post 一次信号量
};

// 解码处理回调
struct audio_dec_handler {
    int (*dec_probe)(struct audio_decoder *);    // 预处理
    // 解码输出。当前架构中解码会自动输出到数据流中，不必要上层实现该函数
    int (*dec_output)(struct audio_decoder *, s16 *data, int len, void *priv);
    int (*dec_post)(struct audio_decoder *);     // 后处理
    int (*dec_stop)(struct audio_decoder *);     // 解码结束
};

// 解码输入接口
struct audio_dec_input {
    u32 coding_type;   // 解码器类型
    // 定义在 p_more_coding_type 数组中的解码器会依照顺序依次检测
    // 先检测 coding_type 再检测 p_more_coding_type
    u32 *p_more_coding_type;
    u32 data_type : 8;  // 数据类型。AUDIO_INPUT_FRAME / AUDIO_INPUT_FILE
    union {
        struct { //数据类型为 AUDIO_INPUT_FILE
            int (*fread)(struct audio_decoder *, void *buf, u32 len);
            int (*fseek)(struct audio_decoder *, u32 offset, int seek_mode);
            int (*ftell)(struct audio_decoder *);
            int (*flen)(struct audio_decoder *);
        } file;
        struct { //数据类型为 AUDIO_INPUT_FRAME
            int (*fget)(struct audio_decoder *, u8 **frame);
            void (*fput)(struct audio_decoder *, u8 *frame);
            int (*ffetch)(struct audio_decoder *, u8 **frame);
        } frame;
    } ops;
};

// decoder 链表
struct audio_res_wait {
    struct list_head list_entry;    // 链表。用于多个解码抢占/排序等
    u8 priority;            // 优先级
    u8 preemption : 1;  // 抢断
    u8 protect : 1;         // 保护（叠加）
    u8 only_del : 1;        // 仅删除
    u8 snatch_same_prio : 1; // 放在同优先级的前面
```




地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

```
    u8 is_work : 1;      // 已经运行
    u32 format;          // 格式锁
    int (*handler)(struct audio_res_wait *, int event); // 解码处理  AUDIO_RES_GET / AUDIO_RES_PUT
};
```

### 9.1.3. 接口介绍

| | |
|---|---|
| 函数原型 | int audio_decoder_task_create(struct audio_decoder_task *task, const char *name) |
| 功能描述 | 创建一个 decoder 任务 |
| 参数说明 | * \param[in][out]  *task  // 解码任务句柄<br>* \param[in]  *name  // 解码任务名称 |
| 输出 | * \return  返回相应的操作消息处理值<br>* \retval  非 0. 失败<br>* \retval  0 成功 |
| 例子 | struct audio_decoder_task test_decoder = {0};<br>Ret = audio_decoder_task_create(&test_decoder, “test_dec”); |
| 关联模块 | |
| 补充说明 | 解码任务需要在 task_info_table 数组中注册。任务名称不超过 12Byte |

| | |
|---|---|
| 函数原型 | int audio_decoder_task_add_wait(struct audio_decoder_task *task, struct audio_res_wait *wait) |
| 功能描述 | 添加一个资源到 decoder 中 |
| 参数说明 | * \param[in][out]  *task  // 解码任务句柄<br>* \param[in][out]  *wait  // 解码资源 |
| 输出 | * \return  返回相应的操作消息处理值<br>* \retval  非 0. 失败<br>* \retval  0 成功 |
| 例子 | Static int test_dec_wait_res_handler(struct audio_res_wait *wait, int event)<br>{<br>  If (event == AUDIO_RES_GET) {<br>  Test_dec_start();<br>  } else if (event == AUDIO_RES_PUT) {<br>  Test_dec_stop();<br>  }<br>}<br>struct audio_res_wait test_wait = {0};<br>If (叠加方式) {<br>  Test_wait.protect = 1;<br>} else if (抢占方式) { |




地址：珠海市吉大石花西路 107 号 9 栋综合楼  邮编：519015
电话：0756-6313088  传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | Test_wait.preemption = 1;<br>} else { // 按优先级顺序播放<br>Test_wait.priority = n;<br>If (放在同优先级的前面) {<br>Test_wait.snatch_same_prio = 1;<br>}<br>}<br>Test_wait.handler = test_dec_wait_res_handler;<br>Ret = audio_decoder_task_add_wait(&test_decoder, &test_wait); |
| 关联模块 | |
| 补充说明 | 和 audio_decoder_task_del_wait()配套使用 |

| | |
|---|---|
| 函数原型 | void audio_decoder_task_del_wait(struct audio_decoder_task *task, struct audio_res_wait *wait) |
| 功能描述 | 从 decoder 中删除一个资源 |
| 参数说明 | * \param[in][out] *task // 解码任务句柄<br>* \param[in][out] *wait // 解码资源 |
| 输出 | * \return 无 |
| 例子 | audio_decoder_task_del_wait(&test_decoder, &test_wait); |
| 关联模块 | |
| 补充说明 | 和 audio_decoder_task_add_wait()配套使用 |

| | |
|---|---|
| 函数原型 | int audio_decoder_task_wait_state(struct audio_decoder_task *task) |
| 功能描述 | 查询 decoder 资源链表 task->wait 的总数 |
| 参数说明 | * \param[in] *task // 解码任务句柄 |
| 输出 | * \return 资源总数 |
| 例子 | Int res_num = audio_decoder_task_wait_state(&test_decoder); |
| 关联模块 | |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int audio_decoder_resume_all(struct audio_decoder_task *task) |
| 功能描述 | 激活所有 task->head 中的解码 |
| 参数说明 | * \param[in][out] *task // 解码任务句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 消息发送失败（但解码状态会改变）<br>* \retval 0 成功 |
| 例子 | Ret = audio_decoder_resume_all(&test_decoder); |
| 关联模块 | |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 补充说明 | 仅激活状态为 DEC_STA_WAIT_SUSPEND 类型的解码。仅改变状态并且发送消息。 |

| | |
|---|---|
| 函数原型 | int audio_decoder_resume_all_by_sem(struct audio_decoder_task *task, int time_out) |
| 功能描述 | 激活所有 task->head 中的解码，并且超时等待解码任务执行一轮 |
| 参数说明 | * \param[in][out] *task // 解码任务句柄<br>* \param[in] time_out // 超时等待时长 time_out*10 ms |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | Ret = audio_decoder_resume_all_by_sem(&test_decoder, 1); |
| 关联模块 | |
| 补充说明 | 仅激活状态为 DEC_STA_WAIT_SUSPEND 类型的解码 |

| | |
|---|---|
| 函数原型 | int audio_decoder_fmt_lock(struct audio_decoder_task *task, int fmt) |
| 功能描述 | 解码任务添加格式锁 |
| 参数说明 | * \param[in][out] *task // 解码任务句柄<br>* \param[in] fmt // 锁定的格式 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval -EFAULT. 该格式已经被锁定<br>* \retval 0 成功 |
| 例子 | Ret = audio_decoder_fmt_lock(&test_decoder, AUDIO_CODING_AAC); |
| 关联模块 | |
| 补充说明 | 被锁定的格式其他解码不能再使用。和 audio_decoder_fmt_unlock()配套使用 |

| | |
|---|---|
| 函数原型 | int audio_decoder_fmt_unlock(struct audio_decoder_task *task, int fmt) |
| 功能描述 | 解码任务删除格式锁 |
| 参数说明 | * \param[in][out] *task // 解码任务句柄<br>* \param[in] fmt // 删除锁定的格式 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | Ret = audio_decoder_fmt_unlock(&test_decoder, AUDIO_CODING_AAC); |
| 关联模块 | |
| 补充说明 | 被锁定的格式其他解码不能再使用。和 audio_decoder_fmt_lock()配套使用 |

| | |
|---|---|
| 函数原型 | int audio_decoder_open(struct audio_decoder *dec, const struct audio_dec_input |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | *input,struct audio_decoder_task *task) |
| 功能描述 | 打开一个解码 |
| 参数说明 | * \param[in][out] *dec // 解码句柄<br>* \param[in] *input // 解码输入接口<br>* \param[in] *task // 解码任务句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | Static const struct audio_dec_input test_audio_dec_file_input = {<br>.coding_type = AUDIO_CODING_G729,<br>.data_type = AUDIO_INPUT_FILE,<br>.ops = {<br>.file = {<br>.fread = test_audio_dec_file_fread,<br>.fseek = test_audio_dec_file_fseek,<br>.flen = test_audio_dec_file_flen,<br>}<br>}<br>};<br>Static const struct audio_dec_input test_audio_dec_frame_input = {<br>.coding_type = AUDIO_CODING_SBC,<br>.data_type = AUDIO_INPUT_FRAME,<br>.ops = {<br>.frame = {<br>.fget = test_audio_dec_frame_get_data,<br>.fput = test_audio_dec_frame_put_data,<br>.ffetch = test_audio_dec_frame_fetch_data,<br>}<br>}<br>};<br>struct audio_decoder test_dec0 = {0};<br>If (file 数据类型的解码) {<br>Ret = audio_decoder_open(&test_dec0, &test_audio_dec_file_input, &test_decoder);<br>} else { // frame 数据类型的解码<br>Ret = audio_decoder_open(&test_dec0, &test_audio_dec_frame_input, &test_decoder);<br>} |
| 关联模块 | |
| 补充说明 | 和 audio_decoder_close()配套使用 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 函数原型 | int audio_decoder_data_type(void *_dec) |
| 功能描述 | 获取解码数据格式 |
| 参数说明 | * \param[in][out] *dec // 解码句柄 |
| 输出 | * \return 数据格式 |
| 例子 | Int data_type = audio_decoder_data_type(&test_dec0); |
| 关联模块 | |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int audio_decoder_get_fmt(struct audio_decoder *dec, struct audio_fmt **fmt) |
| 功能描述 | 获取解码格式信息 |
| 参数说明 | * \param[in][out] *dec // 解码句柄<br>* \param[in][out] **fmt // 解码格式 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | struct audio_fmt *pfmt = NULL;<br>Ret = audio_decoder_get_fmt(&test_dec0, &pfmt); |
| 关联模块 | |
| 补充说明 | 当还没有获取到解码器接口时，会尝试自动获取解码器，并获取格式信息 |

| | |
|---|---|
| 函数原型 | int audio_decoder_set_fmt(struct audio_decoder *dec, struct audio_fmt *fmt) |
| 功能描述 | 以指定解码类型查找解码器 |
| 参数说明 | * \param[in][out] *dec // 解码句柄<br>* \param[in] *fmt // 解码格式 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | struct audio_fmt fmt = {0};<br>If (指定解码类型并且指定采样率和通道) {<br>Fmt.coding_type = AUDIO_CODING_SBC;<br>Fmt.sample_rate = 16000;<br>Fmt.channel = 2;<br>} else { // 仅指定解码类型<br>Fmt.coding_type = AUDIO_CODING_MP3;<br>}<br>Ret = audio_decoder_set_fmt(&test_dec0, pfmt); |
| 关联模块 | |
| 补充说明 | 当解码格式中没有指定采样率和通道，会尝试从解码器中获取格式信息 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 函数原型 | void audio_decoder_set_handler(struct audio_decoder *dec, const struct audio_dec_handler *handler) |
| 功能描述 | 设置解码处理回调 |
| 参数说明 | * \param[in][out] *dec // 解码句柄<br>* \param[in] *handler // 解码处理回调 |
| 输出 | * \return 无 |
| 例子 | Static int test_audio_dec_probe_handler(struct audio_decoder *decoder)<br>{<br>If (该解码不运行并且解码任务执行完一轮后不等待消息池 pend) {<br>// 一般情况下需要调用 audio_decoder_suspend()函数挂起该解码<br>Return 非 0 并且非-EINVAL 值;<br>} else { // 该解码不运行，解码任务等待 pend<br>Return -EINVAL;<br>}<br>// 正常解码<br>Return 0;<br>}<br>Static int test_audio_dec_post_handler(struct audio_decoder *decoder)<br>{<br>Return 0;<br>}<br>Static int test_audio_dec_stop_handler(struct audio_decoder *decoder)<br>{<br>Return 0;<br>}<br>Static const struct audio_dec_handler test_audio_dec_handler = {<br>.dec_probe = test_audio_dec_probe_handler,<br>.dec_probe = test_audio_dec_post_handler,<br>.dec_probe = test_audio_dec_stop_handler,<br>};<br>audio_decoder_set_handler(&test_dec0, &test_audio_dec_handler); |
| 关联模块 | |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int audio_decoder_set_output_channel(struct audio_decoder *dec, enum audio_channel ch_type) |
| 功能描述 | 设置解码声道类型 |
| 参数说明 | * \param[in][out] *dec // 解码句柄<br>* \param[in] ch_type // 声道类型<br>enum audio_channel {<br>AUDIO_CH_LR = 0, //立体声 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | AUDIO_CH_L, //左声道（单声道）<br>AUDIO_CH_R, //右声道（单声道）<br>AUDIO_CH_DIFF, //差分（单声道）<br>AUDIO_CH_DUAL_L, //双声道都为左<br>AUDIO_CH_DUAL_R, //双声道都为右<br>AUDIO_CH_DUAL_LR, //双声道为左右混合<br>AUDIO_CH_QUAD, //四声道（LRLR）<br>AUDIO_CH_MAX = 0xff,<br>}; |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | Ret = audio_decoder_set_output_channel(&test_dec0, AUDIO_CH_LR); |
| 关联模块 | |
| 补充说明 | 该函数直接调用解码器中的函数，需要保证解码器还没有关闭 |

| | |
|---|---|
| 函数原型 | int audio_decoder_start(struct audio_decoder *dec) |
| 功能描述 | 解码器开始运行，并且解码任务开始运行该解码 |
| 参数说明 | * \param[in][out] *dec // 解码句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | Ret = audio_decoder_start(&test_dec0); |
| 关联模块 | |
| 补充说明 | 会先调用解码器中的 start 函数，成功后才让解码任务开始解码 |

| | |
|---|---|
| 函数原型 | int audio_decoder_stop(struct audio_decoder *dec) |
| 功能描述 | 解码任务停止该解码 |
| 参数说明 | * \param[in][out] *dec // 解码句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | Ret = audio_decoder_stop(&test_dec0); |
| 关联模块 | |
| 补充说明 | 仅让解码任务停止该解码。如果该解码运行的时候有错误，会返回事件 |

| | |
|---|---|
| 函数原型 | int audio_decoder_pause(struct audio_decoder *dec) |
| 功能描述 | 解码任务暂停该解码 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 参数说明 | * \param[in][out] *dec // 解码句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | Ret = audio_decoder_pause(&test_dec0); |
| 关联模块 | |
| 补充说明 | 仅让解码任务暂停该解码 |

| | |
|---|---|
| 函数原型 | int audio_decoder_suspend(struct audio_decoder *dec) |
| 功能描述 | 解码任务挂起该解码 |
| 参数说明 | * \param[in][out] *dec // 解码句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | Ret = audio_decoder_suspend(&test_dec0); |
| 关联模块 | |
| 补充说明 | 解码处于 DEC_STA_WAIT_STOP 获取 DEC_STA_WAIT_PAUSE 状态时无法被挂起 |

| | |
|---|---|
| 函数原型 | int audio_decoder_resume(struct audio_decoder *dec) |
| 功能描述 | 解码任务激活该解码 |
| 参数说明 | * \param[in][out] *dec // 解码句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | Ret = audio_decoder_resume(&test_dec0); |
| 关联模块 | |
| 补充说明 | 解码为 DEC_STA_WAIT_SUSPEND 状态时才能被激活 |

| | |
|---|---|
| 函数原型 | int audio_decoder_close(struct audio_decoder *dec) |
| 功能描述 | 关闭解码 |
| 参数说明 | * \param[in][out] *dec // 解码句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | Ret = audio_decoder_stop(&test_dec0); |
| 关联模块 | |
| 补充说明 | 发信息给解码任务，等待解码任务关闭该解码完成才退出 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 函数原型 | int audio_decoder_reset(struct audio_decoder *dec) |
| 功能描述 | 解码重新开始 |
| 参数说明 | * \param[in][out] *dec // 解码句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | Ret = audio_decoder_reset(&test_dec0); |
| 关联模块 | |
| 补充说明 | 该函数直接调用解码器中的函数，需要保证解码器还没有关闭。一般只有 frame 数据类型的解码器才有该功能，如 SBC 解码等 |

| | |
|---|---|
| 函数原型 | void audio_decoder_set_breakpoint(struct audio_decoder *dec, struct audio_dec_breakpoint *bp) |
| 功能描述 | 设置断点句柄 |
| 参数说明 | * \param[in][out] *dec // 解码句柄<br>* \param[in] *bp // 断点句柄 |
| 输出 | * \return 无 |
| 例子 | struct audio_dec_breakpoint *test_bp = NULL;<br>Test_bp = zalloc(sizeof(struct audio_dec_breakpoint) + BP_DATA_LEN);<br>Test_bp->data_len = BP_DATA_LEN;<br>Vm_read(BP_ID, Test_bp, sizeof(struct audio_dec_breakpoint) + BP_DATA_LEN);<br>Ret = audio_decoder_set_breakpoint(&test_dec0, test_bp);<br>Ret = audio_decoder_get_fmt(&test_dec0, &pfmt); |
| 关联模块 | |
| 补充说明 | 该函数需要在 get_fmt 或者 set_fmt 之前调用 |

| | |
|---|---|
| 函数原型 | int audio_decoder_get_breakpoint(struct audio_decoder *dec, struct audio_dec_breakpoint *bp) |
| 功能描述 | 获取断点信息 |
| 参数说明 | * \param[in][out] *dec // 解码句柄<br>* \param[in] *bp // 断点句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | struct audio_dec_breakpoint *test_bp = NULL;<br>Test_bp = zalloc(sizeof(struct audio_dec_breakpoint) + BP_DATA_LEN);<br>Test_bp->data_len = BP_DATA_LEN;<br>Ret = audio_decoder_get_breakpoint(&test_dec0, test_bp); |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 关联模块 | |
| 补充说明 | 发信息给解码任务，等待解码任务获取该解码断点完成才退出 |

| | |
|---|---|
| 函数原型 | int audio_decoder_forward(struct audio_decoder *dec, int step_s) |
| 功能描述 | 解码快进 |
| 参数说明 | * \param[in][out] *dec // 解码句柄<br>* \param[in] step_s // 快进步伐，单位：秒 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | Ret = audio_decoder_forward(&test_dec0); |
| 关联模块 | |
| 补充说明 | 该函数直接调用解码器中的函数，需要保证解码器还没有关闭。 |

| | |
|---|---|
| 函数原型 | int audio_decoder_rewind(struct audio_decoder *dec, int step_s) |
| 功能描述 | 解码快退 |
| 参数说明 | * \param[in][out] *dec // 解码句柄<br>* \param[in] step_s // 快退步伐，单位：秒 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | Ret = audio_decoder_rewind(&test_dec0); |
| 关联模块 | |
| 补充说明 | 该函数直接调用解码器中的函数，需要保证解码器还没有关闭。 |

| | |
|---|---|
| 函数原型 | int audio_decoder_get_total_time(struct audio_decoder *dec) |
| 功能描述 | 获取解码总时间 |
| 参数说明 | * \param[in][out] *dec // 解码句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | Ret = audio_decoder_get_fmt(&test_dec0, &pfmt);<br>Int total_time = audio_decoder_get_total_time(&test_dec0); |
| 关联模块 | |
| 补充说明 | 在获取解码格式信息的时候已经保存了解码总时间 |

| | |
|---|---|
| 函数原型 | int audio_decoder_get_play_time(struct audio_decoder *dec) |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 功能描述 | 获取解码当前时间 |
| 参数说明 | * \param[in][out] *dec // 解码句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | Int cur_time = audio_decoder_get_play_time(&test_dec0); |
| 关联模块 | |
| 补充说明 | 该函数直接调用解码器中的函数，需要保证解码器还没有关闭。 |

| | |
|---|---|
| 函数原型 | void audio_decoder_set_pick_stu(struct audio_decoder *dec, u8 pick) |
| 功能描述 | 使能解码拆包处理 |
| 参数说明 | * \param[in][out] *dec // 解码句柄<br>* \param[in] pick // 拆包使能 |
| 输出 | * \return 无 |
| 例子 | audio_decoder_set_pick_stu(&test_dec0, 1);<br>Ret = audio_decoder_get_fmt(&test_dec0, &pfmt); |
| 关联模块 | |
| 补充说明 | 本地解码，目前仅 mp3（mp3_decstream_lib.a）和 wma（wma_decstream_lib.a）有效。该函数需要在 get_fmt 或者 set_fmt 之前调用 |

| | |
|---|---|
| 函数原型 | int audio_decoder_get_pick_stu(struct audio_decoder *dec) |
| 功能描述 | 获取解码拆包处理使能状态 |
| 参数说明 | * \param[in][out] *dec // 解码句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | Ret = audio_decoder_get_pick_stu(&test_dec0); |
| 关联模块 | |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | void audio_decoder_set_tws_stu(struct audio_decoder *dec, u8 tws) |
| 功能描述 | 使能解码器按数据流方式处理 |
| 参数说明 | * \param[in][out] *dec // 解码句柄<br>* \param[in] tws // 数据流方式处理使能 |
| 输出 | * \return 无 |
| 例子 | audio_decoder_set_tws_stu(&test_dec0, 1);<br>Ret = audio_decoder_get_fmt(&test_dec0, &pfmt); |
| 关联模块 | |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 补充说明 | 数据流方式代表文件读取返回的长度可以不是需要的长度，可以挂起等待下一次执行（文件方式读取返回长度不同会认为已经解码结束），其他的不变。该函数需要在 get_fmt 或者 set_fmt 之前调用 |

| | |
|---|---|
| 函数原型 | int audio_decoder_get_tws_stu(struct audio_decoder *dec) |
| 功能描述 | 获取按数据流方式处理状态 |
| 参数说明 | * \param[in][out] *dec // 解码句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | Ret = audio_decoder_get_tws_stu(&test_dec0); |
| 关联模块 | |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | void audio_decoder_set_run_max(struct audio_decoder *dec, u8 run_max) |
| 功能描述 | 设置解码最大正常运行次数 |
| 参数说明 | * \param[in][out] *dec // 解码句柄<br>* \param[in] run_max // 最大正常运行次数 |
| 输出 | * \return 无 |
| 例子 | audio_decoder_set_run_max(&test_dec0, 10) |
| 关联模块 | |
| 补充说明 | 一般用在叠加的情况下。如 wav 可能解码一次输出 2k 数据，g729 解码一次输出 200 个数据，可以让 g729 正常多运行几次。正常运行是指解码输入输出都是完整的，且没有解码结束 |

| | |
|---|---|
| 函数原型 | int audio_decoder_running_number(struct audio_decoder_task *task) |
| 功能描述 | 处于运行中的解码个数 |
| 参数说明 | * \param[in][out] *task // 解码任务句柄 |
| 输出 | * \return 解码个数 |
| 例子 | Int num = audio_decoder_running_number(&test_decoder); |
| 关联模块 | |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int audio_decoder_ioctrl(struct audio_decoder *dec, u32 cmd, void *parm) |
| 功能描述 | 用于传递给解码器的一些命令控制 |
| 参数说明 | * \param[in][out] *dec // 解码句柄<br>* \param[in][out] *cmd // 命令 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | * \param[in][out] *parm // 命令参数 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | // 设置循环播放<br>struct audio_repeat_mode_param rep = {0};<br>rep.flag = 1; //使能<br>rep.headcut_frame = 2; //依据需求砍掉前面几帧，仅 mp3 格式有效<br>rep.tailcut_frame = 2; //依据需求砍掉后面几帧，仅 mp3 格式有效<br>rep.repeat_callback = file_dec_repeat_cb;<br>rep.callback_priv = dec;<br>rep.repair_buf = &dec->repair_buf;<br>audio_decoder_ioctrl(&test_dec0,AUDIO_IOCTRL_CMD_REPEAT_PLAY, &rep); |
| 关联模块 | |
| 补充说明 | 发信息给解码任务，等待解码任务处理完成才退出 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

# 10. 音效处理功能介绍

## 10.1. EQ/DRC

### 10.1.1. 功能介绍

EQ（Equalizer），即均衡器，通过调整各个频段的信号增益值，对特定的频段进行放大或者衰减，实现对原始声音信号的增强和补偿，提升歌曲的整体听感。常见包括正常、摇滚、流行、舞曲、古典、柔和、爵士和自定义等。

DRC（Dynamic Range Control），动态范围控制，通过限制音频信号的动态范围，防止音频信号溢出和负载出现过载的情况出现。

本章节 EQ/DRC 模块主要提供包含模块打开，模块关闭，模块参数控制，模块系数更新，等接口给应用调用。

### 10.1.2. 接口介绍

#### 10.1.2.1. eq、drc 打开

| 函数原型 | struct audio_eq_drc *audio_eq_drc_open(struct audio_eq_drc_parm *parm) |
|---|---|
| 功能描述 | 打开 eq drc |
| 参数说明 | struct audio_eq_drc_parm *parm<br>struct audio_eq_drc_parm {<br>u8 eq_en: 1; //eq 是否使能 1:使能 0:关闭<br>u8 drc_en: 1; //drc 是否使能 1:使能 0:关闭<br>u8 high_bass: 1; //高低音是否使能， 1：使能 0：关闭<br>u8 async_en: 1; //是否使能异步 eq 1:使能 0：关闭<br>u8 out_32bit: 1; //eq 后是否输出 32bit 1:使能： 0 关闭<br>u8 mode_en: 1; //没离线文件时，是否支持使用默认系数表做 eq<br>u8 online_en: 1; //是否支持在线调试 1：支持 0：不支持<br>u8 ch_num: 3; //输入通道数<br>/*四声道部分未支持*/<br>u8 divide_en: 1; //各个声道 eq drc 效果是否独立，0：使用同个效果<br>u8 four_ch: 1; //四声道 eq drc 是否使能， 1：使能 0：关闭<br>u8 input_four_ch_dat: 1; //输入数据是否是 4 声道 1：是 0：不是<br>u8 eq_name_four; //四通道时，RL RR 通道的 eq_name |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | u8 eq_name; //FL FR 通道的 eq_name 普通音乐 eq 使用 song_eq_mode,通话下行 eq 使用 call_eq_mode<br>u16 sr; //采样率<br>audio_eq_filter_cb eq_cb; //获取 eq 系数的回调函数，high_bass 使能后，该回调无用<br>audio_drc_filter_cb drc_cb;//获取 drc 系数的回调<br>}; |
| 输出 | * \return 返回相应 eq、drc 操作句柄 |
| 例子 | 普通音乐 eq: 双声道、44100 采样率的 eq、drc 打开 例子<br>struct audio_eq_drc *eq_drc = NULL;<br>u8 ch_num = 2;<br>u16 sample_rate = 44100;<br>struct audio_eq_drc_parm effect_parm = {0};<br>effect_parm.eq_en = 1;<br>effect_parm.drc_en = 1;<br>effect_parm.drc_cb = drc_get_filter_info;<br>if (effect_parm.eq_en) {<br>effect_parm.async_en = 1;<br>effect_parm.out_32bit = 1;<br>effect_parm.online_en = 1;<br>effect_parm.mode_en = 1;<br>}<br>effect_parm.eq_name = song_eq_mode;<br>effect_parm.ch_num = ch_num;<br>effect_parm.sr = sample_rate;<br>effect_parm.eq_cb = eq_get_filter_info;<br>eq_drc = audio_eq_drc_open(&effect_parm);<br><br>高低音：双声道、44100 采样率 高低音打开例子<br>struct audio_eq_drc *eq_drc = NULL;<br>u8 ch_num = 2;<br>u16 sample_rate = 44100;<br>struct audio_eq_drc_parm effect_parm = {0};<br>effect_parm.high_bass = 1;<br>effect_parm.eq_en = 1;<br>effect_parm.drc_en = 1;//根据需要是否打开限幅器，此处打开限幅器<br>effect_parm.drc_cb = high_bass_drc_get_filter_info;//自定义的限幅器系数回调<br>if (effect_parm.eq_en) {<br>effect_parm.async_en = 1;<br>effect_parm.out_32bit = 1; |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | effect_parm.online_en = 0;<br>effect_parm.mode_en = 0;<br>}<br>effect_parm.eq_name = song_eq_mode;<br><br>effect_parm.ch_num = ch_num;<br>effect_parm.sr = sample_rate;<br>eq_drc = audio_eq_drc_open(&effect_parm);<br><br>通话下行 eq:单声道、16k 采样率 eq 打开<br>struct audio_eq_drc *eq_drc = NULL;<br>u8 ch_num = 1;<br>u16 sample_rate = 16000;<br>struct audio_eq_drc_parm effect_parm = {0};<br>effect_parm.eq_en = 1;<br>if (effect_parm.eq_en) {<br>effect_parm.async_en = 1;<br>effect_parm.online_en = 1;<br>effect_parm.mode_en = 0;<br>}<br>effect_parm.eq_name = call_eq_mode;<br>effect_parm.ch_num = ch_num;<br>effect_parm.sr = sample_rate;<br>effect_parm.eq_cb = eq_phone_get_filter_info;<br>eq_drc = audio_eq_drc_open(&effect_parm); |
| 关联模块 | |
| 补充说明 | |

### 10.1.2.2. eq、drc 关闭

根据 eq、drc 打开的句柄，关闭 eq、drc

| | |
|---|---|
| 函数原型 | void audio_eq_drc_close(struct audio_eq_drc *hdl); |
| 功能描述 | 根据 eq、drc 打开的句柄，关闭 eq、drc |
| 参数说明 | * \param[in] hdl 是 audio_eq_drc_open()的返回值 |
| 输出 | * \return 无 |
| 例子 | audio_eq_drc_close(eq_drc) |
| 关联模块 | |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| 补充说明 | |
|---|---|

### 10.1.2.3. eq、drc 功能控制

根据传递的 cmd 值，设置需要更新的参数。

| 函数原型 | int audio_eq_drc_parm_update(struct audio_eq_drc *hdl, u32 cmd, void *parm) |
|---|---|
| 功能描述 | 根据传递的 cmd 值，设置需要更新的参数 |
| 参数说明 | * \param[in] hdl 操作句柄<br>* \param[in] cmd: AUDIO_EQ_SET_CH<br>AUDIO_EQ_CLR_DAT<br>AUDIO_EQ_HIGH<br>AUDIO_EQ_BASS<br>AUDIO_EQ_GET_DATA_LEN<br>AUDIO_EQ_HIGH_BASS_DIS<br>* \param[in] parm:根据 cmd 传递对应参数 |
| 输出 | * \return AUDIO_EQ_GET_DATA_LEN 时，返回 len,其余返回 0 |
| 关联模块 | |
| 例子 | 通道设置例子<br>u8 ch = 2;<br>audio_eq_drc_parm_update(eq_drc,AUDIO_EQ_SET_CH, (void*)ch );<br><br>异步 eq 数据清空例子<br>audio_eq_drc_parm_update(eq_drc,AUDIO_EQ_CLR_DAT, NULL);<br><br>高音调节例子<br>struct high_bass hb = {0};<br>hb.freq = 12000;//高音中心截止频率 Hz, 0：使用内部默认中心截止频率<br>hb.gain = 0;//高音增益 范围 -12~12<br>audio_eq_drc_parm_update(eq_drc, AUDIO_EQ_HIGH, (void *)&hb);<br><br>低音调节例子<br>struct high_bass hb = {0};<br>hb.freq = 200;//低音音中心截止频率 Hz， 0 使用内部默认中心截止频率<br>hb.gain = 0;//低音增益 范围 -12~12<br>audio_eq_drc_parm_update(eq_drc, AUDIO_EQ_BASS, (void *)&hb);<br><br>高低音打开后，进入某模式不做高低音例子<br>u8 dis = 1;<br>audio_eq_drc_parm_update(eq_drc, AUDIO_EQ_HIGH_BASS_DIS, (void |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | *)dis );<br>高低音打开后，退出某模式还原高低音处理例子<br>u8 dis = 0;<br>audio_eq_drc_parm_update(eq_drc, AUDIO_EQ_HIGH_BASS_DIS, (void *)dis ); |
| 补充说明 | |

#### 10.1.2.4. 普通音乐 eq 系数回调接口

| | |
|---|---|
| 函数原型 | int eq_get_filter_info(struct audio_eq *eq, int sr, struct audio_eq_filter_info *info) |
| 功能描述 | 普通音乐 Eq 系数回调，离线文件系数或者默认系数表可使用该回调 |
| 参数说明 | * \param[in] 参数内部调用时传递 |
| 输出 | * \return 0:更新系数成功。 -1：系数更新失败 |
| 关联模块 | |
| 例子 | ...<br>effect_parm.eq_cb = eq_get_filter_info;<br>eq_drc = audio_eq_drc_open(&effect_parm); |
| 补充说明 | 使用者亦可自定义该回调 |

#### 10.1.2.5. 普通音乐 drc 系数回调接口

| | |
|---|---|
| 函数原型 | int drc_get_filter_info(struct audio_drc *drc, struct audio_drc_filter_info *info) |
| 功能描述 | 普通音乐 drc 系数回调，离线文件系数或者默认系数表可使用该回调 |
| 参数说明 | * \param[in] 参数内部调用时传递 |
| 输出 | * \return 0:更新系数成功。 -1：系数更新失败 |
| 关联模块 | |
| 例子 | ...<br>effect_parm.drc_cb = drc_get_filter_info;<br>eq_drc = audio_eq_drc_open(&effect_parm); |
| 补充说明 | 使用者亦可自定义该回调 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

#### 10.1.2.6. 通话上行 eq 系数回调接口

| | |
|---|---|
| 函数原型 | int aec_ul_eq_filter(struct audio_eq *eq, int sr, struct audio_eq_filter_info *info) |
| 功能描述 | aec 通话上行 eq 系数回调 |
| 参数说明 | * \param[in] 参数内部调用时传递 |
| 输出 | * \return 0:更新系数成功。 -1：系数更新失败 |
| 关联模块 | |
| 例子 | |
| 补充说明 | |

#### 10.1.2.7. 通话下行 eq 系数回调接口

| | |
|---|---|
| 函数原型 | int eq_phone_get_filter_info(struct audio_eq *eq, int sr, struct audio_eq_filter_info *info) |
| 功能描述 | 通话下行 eq 系数回调 |
| 参数说明 | * \param[in] 参数内部调用时传递 |
| 输出 | * \return 0:更新系数成功。 -1：系数更新失败 |
| 关联模块 | |
| 例子 | |
| 补充说明 | |

#### 10.1.2.8. 无离线效果文件时，音乐 eq 效果模式设置接口

| | |
|---|---|
| 函数原型 | int eq_mode_set(u8 mode) |
| 功能描述 | 无离线文件时，用默认系数表的 eq 效果模式设置 |
| 参数说明 | * \param[in] mode:<br>EQ_MODE_NORMAL = 0,<br>EQ_MODE_ROCK,<br>EQ_MODE_POP,<br>EQ_MODE_CLASSIC,<br>EQ_MODE_JAZZ,<br>EQ_MODE_COUNTRY,<br>EQ_MODE_CUSTOM,//自定义<br>EQ_MODE_MAX, |
| 输出 | * \return 0 |
| 关联模块 | |
| 例子 | eq_mode_set(EQ_MODE_JAZZ) |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| 补充说明 | |
|---|---|

#### 10.1.2.9. 无离线效果文件时，音乐 eq 效果模式切换接口

| 函数原型 | int eq_mode_sw(void) |
|---|---|
| 功能描述 | 无离线文件时，用默认系数表的 eq 效果模式切换 |
| 参数说明 | * \param[in]　无 |
| 输出 | * \return　0 |
| 关联模块 | |
| 例子 | eq_mode_sw() |
| 补充说明 | |

#### 10.1.2.10. 无离线效果文件时，获取音乐 eq 当前效果模式接口

| 函数原型 | int eq_mode_get_cur(void) |
|---|---|
| 功能描述 | 无离线文件时，获取当前 eq 效果模式 |
| 参数说明 | * \param[in]　无 |
| 输出 | * \return<br>EQ_MODE_NORMAL = 0,<br>EQ_MODE_ROCK,<br>EQ_MODE_POP,<br>EQ_MODE_CLASSIC,<br>EQ_MODE_JAZZ,<br>EQ_MODE_COUNTRY,<br>EQ_MODE_CUSTOM,//自定义 |
| 关联模块 | |
| 例子 | int mode = eq_mode_get_cur() |
| 补充说明 | |

#### 10.1.2.11. 无离线效果文件时，设置音乐自定义模式 eq 段增益接口

| 函数原型 | int eq_mode_set_custom_param(u16 index, int gain) |
|---|---|
| 功能描述 | 无离线文件时，设置用户自定义 eq 效果模式时的增益 |
| 参数说明 | * \param[in]　index:第几段<br>* \param[in]　gain:增益（-12~12） |




地址：珠海市吉大石花西路 107 号 9 栋综合楼　邮编：519015
电话：0756-6313088　传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 输出 | * \return 0 |
| 关联模块 | |
| 例子 | 设置自定义模式第 0 段 eq 的，增益为 -2db<br>eq_mode_set_custom_param(0, -2) |
| 补充说明 | |

### 10.1.2.12. 无离线效果文件时，设置音乐自定义模式 eq 段中心截止频率、增益接口

| | |
|---|---|
| 函数原型 | int eq_mode_set_custom_info(u16 index, int freq, int gain) |
| 功能描述 | 无离线文件时，设置用户自定义 eq 效果模式时的中心截止频率和增益 |
| 参数说明 | * \param[in] index:第几段<br>* \param[in] freq:中心截止频率（22Hz~22k Hz）<br>* \param[in] gain:增益（-12~12） |
| 输出 | * \return 0 |
| 关联模块 | |
| 例子 | 设置自定义模式第 0 段 eq 的，中心截止频率为 400Hz，增益为 -2db<br>eq_mode_set_custom_param(0, 400, -2) |
| 补充说明 | |

### 10.1.2.13. 无离线效果文件时，获取相应模式下 eq 段增益接口

| | |
|---|---|
| 函数原型 | s8 eq_mode_get_gain(u8 mode, u16 index) |
| 功能描述 | 无离线文件时，获取某 eq 效果模式,某段 eq 的增益 |
| 参数说明 | * \param[in] index:第几段<br>* \param[in] mode:<br>EQ_MODE_NORMAL = 0,<br>EQ_MODE_ROCK,<br>EQ_MODE_POP,<br>EQ_MODE_CLASSIC,<br>EQ_MODE_JAZZ,<br>EQ_MODE_COUNTRY,<br>EQ_MODE_CUSTOM,//自定义 |
| 输出 | * \return val:增益值（-12~12） |
| 关联模块 | |
| 例子 | 获取 EQ_MODE_COUNTRY，第 2 段 eq 的增益<br>S8 val = eq_mode_get_gain(EQ_MODE_COUNTRY, 2); |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| 补充说明 | |
|---|---|

#### 10.1.2.14. 无离线效果文件时，获取相应模式下 eq 段中心截止频率接口

| 函数原型 | int eq_mode_get_freq(u8 mode, u16 index) |
|---|---|
| 功能描述 | 无离线文件时，获取某 eq 效果模式,某段 eq 的中心截止频率 |
| 参数说明 | * \param[in] index:第几段<br>* \param[in] mode:<br>EQ_MODE_NORMAL = 0,<br>EQ_MODE_ROCK,<br>EQ_MODE_POP,<br>EQ_MODE_CLASSIC,<br>EQ_MODE_JAZZ,<br>EQ_MODE_COUNTRY,<br>EQ_MODE_CUSTOM,//自定义 |
| 输出 | * \return val:增益值（22Hz~22kHz） |
| 关联模块 | |
| 例子 | 获取 EQ_MODE_COUNTRY，第 2 段 eq 的中心截止频率<br>Int freq = eq_mode_get_freq(EQ_MODE_COUNTRY, 2); |
| 补充说明 | |

## 10.2. 等响度、虚拟低音、环绕音效

### 10.2.1. 功能介绍

等响度、虚拟低音、环绕音效 模块主要提供包含模块打开，模块关闭，模块参数控制，等接口给应用调用

### 10.2.2. 接口介绍

#### 10.2.2.1. 等响度打开

| 函数原型 | equal_loudness_hdl *audio_equal_loudness_open(equalloudness_open_parm *_parm) |
|---|---|
| 功能描述 | 打开等响度 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 参数说明 | typedef struct _equalloudness_open_parm {<br>u16 sr; //采样率<br>u8 channel; //通道数<br>u8 threadhold_vol; //触发等响度软件数字音量阈值<br><br>int (*alpha_cb)(float *alpha, u8 *volume, u8 threadhold_vol);//该函数根据，系统软件的数字音量,参数返回 alpha 值<br>} equalloudness_open_parm; |
| 输出 | * \return 返回相应等响度操作句柄 |
| 例子 | 等响度 打开例子<br>equal_loudness_hdl *loudness = NULL;<br>equalloudness_open_parm parm = {0};<br>parm.threadhold_vol = 17;<br>parm.sr = sample_rate;<br>parm.channel = ch_num;<br>parm.alpha_cb = get_alpha;<br>loudness = audio_equal_loudness_open(&parm); |
| 关联模块 | |
| 补充说明 | 开了等响度后，eq 将无效，所以使用等响度时，需将 eq 模块关掉 |

#### 10.2.2.2. 等响度关闭

根据等响度打开的句柄，关闭关闭等响度

| | |
|---|---|
| 函数原型 | int audio_equal_loudness_close(equal_loudness_hdl *_hdl); |
| 功能描述 | 根据打开的句柄，关闭等响度 |
| 参数说明 | * \param[in] hdl 是 audio_equal_loudness_open()的返回值 |
| 输出 | * \return 0：成功 -1：未打开等想响度 |
| 例子 | audio_equal_loudness_close(loudness) |
| 关联模块 | |
| 补充说明 | |

#### 10.2.2.3. 等响度功能控制

设置启动更新等响度的 启动阈值。

| | |
|---|---|
| 函数原型 | int audio_equal_loudness_parm_update(equal_loudness_hdl *_hdl, u32 cmd, equalloudness_update_parm *_parm) |



地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 功能描述 | 设置启动更新等响度的 启动阈值 |
| 参数说明 | * \param[in] hdl 操作句柄<br>* \param[in] cmd: 无<br>* \param[in] _parm:阈值设置结构体 |
| 输出 | * \return 0：成功， -1：失败 |
| 关联模块 | |
| 例子 | equalloudness_update_parm par = {0};<br>Par.threadhold_vol = 18;//启动等响度的数字音量阈值<br>audio_equal_loudness_parm_update(loudness, 0, &par); |
| 补充说明 | |

#### 10.2.2.4. 环绕音效打开

根据参数，打开环绕音效

| | |
|---|---|
| 函数原型 | surround_hdl *audio_surround_open(surround_open_parm *_parm) |
| 功能描述 | 环绕音效打开 |
| 参数说明 | typedef struct _surround_open_parm {<br>u8 channel; //通道数<br>} surround_open_parm; |
| 输出 | * \return 环绕音效句柄 |
| 关联模块 | |
| 例子 | surround_hdl *surround = NULL;<br>surround_open_parm parm = {0};<br>parm.channel = ch_num;<br>surround = audio_surround_open(&parm); |
| 补充说明 | |

#### 10.2.2.5. 环绕音效关闭

根据打开的句柄，关闭环绕音效

| | |
|---|---|
| 函数原型 | int audio_surround_close(surround_hdl *_hdl) |
| 功能描述 | 关闭环绕音效 |
| 参数说明 | * \param[in] 打开的句柄 |
| 输出 | * \return 0：成功 -1：未打开 |
| 关联模块 | |
| 例子 | audio_surround_close(surround); |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| 补充说明 | |
|---|---|

### 10.2.2.6. 环绕音效切换

根据 cmd 值，切换到相应环绕音效

| 函数原型 | int audio_surround_parm_update(surround_hdl *_hdl, u32 cmd, surround_update_parm *_parm) |
|---|---|
| 功能描述 | 根据 cmd 值，切换相应默认音效 |
| 参数说明 | * \param[in] hdl 操作句柄<br>* \param[in] cmd: //使用以下宏定义<br>EFFECT_3D_PANORAMA = 0, //3d 全景<br>EFFECT_3D_ROTATES, //3d 环绕<br>EFFECT_FLOATING_VOICE, //流动人声<br>EFFECT_GLORY_OF_KINGS, //王者荣耀<br>EFFECT_FOUR_SENSION_BATTLEFIELD, //四季战场<br><br>* \param[in] _parm:给 NULL 时，使用内置默认效果，也可以根据需要调整结构体参数，调整效果<br><br>typedef struct _surround_update_parm {<br>int surround_type;//音效类型<br>int rotatestep;//旋转速度<br>int damping;//高频衰减速度<br>int feedback;//整体衰减速度<br>int roomsize;//空间大小<br>} surround_update_parm; |
| 输出 | * \return 0：成功， -1：失败 |
| 关联模块 | |
| 例子 | 设置 3d 全景 音效<br>audio_surround_parm_update(surround, EFFECT_3D_PANORAMA, NULL); |
| 补充说明 | |

### 10.2.2.7. 虚拟低音打开

根据采样率与通道数，打开虚拟低音

| 函数原型 | vbass_hdl *audio_vbass_open(vbass_open_parm *_parm) |
|---|---|




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 功能描述 | 根据采样率与通道数打开虚拟低音 |
| 参数说明 | typedef struct _vbass_open_parm {<br>u16 sr; //输入音频采样率<br>u8 channel;//输入音频声道数<br>} vbass_open_parm; |
| 输出 | * \return 句柄 |
| 关联模块 | |
| 例子 | vbass_hdl *vbass = NULL;<br>vbass_open_parm parm = {0};<br>parm.sr = sample_rate;<br>parm.channel = ch_num;<br>vbass = audio_vbass_open(&parm); |
| 补充说明 | |

#### 10.2.2.8. 虚拟低音关闭

| | |
|---|---|
| 函数原型 | int audio_vbass_close(vbass_hdl *_hdl) |
| 功能描述 | 关闭虚拟低音 |
| 参数说明 | * \param[in] 打开的句柄 |
| 输出 | * \return 0：成功 -1：未打开 |
| 关联模块 | |
| 例子 | audio_vbass_close(vbass); |
| 补充说明 | |

#### 10.2.2.9. 虚拟低音参数更新

| | |
|---|---|
| 函数原型 | int audio_vbass_parm_update(vbass_hdl *_hdl, u32 cmd, vbass_update_parm *_parm) |
| 功能描述 | 设置低音直接频率与强度 |
| 参数说明 | * \param[in] hdl 操作句柄<br>* \param[in] cmd: 0<br><br>* \param[in] _parm:<br>typedef struct _vbass_update_parm {<br>int bass_f;//外放的低音截止频率 Hz<br>int level;//增强强度(4096 等于 1db， 建议范围：4096 到 16384) |



地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | } vbass_update_parm; |
| 输出 | * \return 0：成功， -1：失败 |
| 关联模块 | |
| 例子 | vbass_update_parm def_parm = {0};<br>def_parm.bass_f = 300;//低音截止频率 Hz<br>def_parm.level = 8192;//增强强度<br>audio_vbass_parm_update(vbass, 0, &def_parm); |
| 补充说明 | |

## 10.3. 变声

### 10.3.1. 功能介绍

。变声模块主要实现变声功能：如娃娃音、怪兽音、男变女、女变男。模块主要提供包含模块打开、模块关闭、模块数据处理、模块参数切换等接口给 AP 应用调用。

变声模块只涉及 AP 层，与其他没有任何关系。

变声模块在 16K 采样率情况下占用空间 7K 空间，约占用时钟 3M。

### 10.3.2. 接口介绍

#### 10.3.2.1. Pitch 模块打开

按照输入参数打开 pitc 变声模块，申请所需的运算空间获取后续运行、关闭的句柄。

| | |
|---|---|
| 函数原型 | s_pitch_hdl *open_pitch(PITCH_SHIFT_PARM *param) |
| 功能描述 | 打开 pitch 变声模块函数 |
| 参数说明 | * \param[in] param //变声参数<br>typedef struct PITCH_SHIFT_PARM_ {<br>u32 sr; //input audio samplerate<br>u32 shiftv; //pitch rate: <8192(pitch up), >8192(pitch down)<br>u32 effect_v;<br>s32 formant_shift;<br>} PITCH_SHIFT_PARM;<br>enum {<br>EFFECT_PITCH_SHIFT = 0x00,<br>EFFECT_VOICECHANGE_KIN0,<br>EFFECT_VOICECHANGE_KIN1, |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | EFFECT_ROBORT,<br>EFFECT_AUTOTUNE<br>}; |
| 输出 | * \return 返回 s_pitch_hdl 结构指针<br>* \retval >1. pitch 变声模块运算及设置的句柄<br>* \retval NULL 打开 pitch 模块失败 |
| 例子 | effect->p_pitch_hdl = open_pitch(&effect_pitch_parm_default); |
| 关联模块 | |
| 补充说明 | 输入参数为 NULL 时 使用内部默认参数 |

#### 10.3.2.2. Pitch 变声模块关闭

该功能主要是关闭 pitch 变声模块，释放模块申请的空间。

| | |
|---|---|
| 函数原型 | void close_pitch(s_pitch_hdl *picth_hdl); |
| 功能描述 | 关闭 pitch 变声函数 |
| 参数说明 | * \param[in] picth_hdl //pitch 变声句柄 |
| 输出 | * \return 无 |
| 关联模块 | |
| 例子 | close_pitch(effect->p_pitch_hdl); |
| 补充说明 | |

#### 10.3.2.3. Pitch 变声参数更新

主要是 pitch 变声参数设置。

| | |
|---|---|
| 函数原型 | void update_pict_parm(s_pitch_hdl *picth_hdl); |
| 功能描述 | pitch 变声参数设置 |
| 参数说明 | * \param[in] picth_hdl //pitch 变声句柄 |
| 输出 | * \retrun 无 |
| 关联模块 | |
| 例子 | |
| 补充说明 | |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

#### 10.3.2.4. Pitc 变声参数获取

主要是获取 pitch 变声参数变。

| | |
|---|---|
| 函数原型 | PITCH_SHIFT_PARM *get_pitch_parm(void); |
| 功能描述 | 取 pitch 变声参数变 |
| 参数说明 | * 无 |
| 输出 | * \return 变声参数结构指针 PITCH_SHIFT_PARM* |
| 关联模块 | |
| 例子 | |
| 补充说明 | 配合变声参数更新使用，先获取参数，修改后调用更新接口更新 |

#### 10.3.2.5. Pitch 变声数据处理

主要是对输入音频数据进行变声处理

| | |
|---|---|
| 函数原型 | void pitch_run(s_pitch_hdl *picth_hdl, s16 *indata, s16 *outdata, int len, u8 ch_num); |
| 功能描述 | 对输入音频数据进行 echo 混响处理 |
| 参数说明 | * \param[in] picth_hdl //pitch 变声句柄.<br>* \param[in] indata //待处理的音频数据指针地址.<br>* \param[out] outdata //变声处理后的数据输出指针地址.<br>* \param[in] len //待处理的音频数据长度 BYTE.<br>* \param[in] ch_num//待处理的音频数据通道数. |
| 输出 | * \return 无 |
| 关联模块 | |
| 例子 | |
| 补充说明 | 若模块串入 stream 流，无需显式调用处理接口 |

#### 10.3.2.6. 变声处理暂停

暂停变声处理，暂停后调用 pitch_run 接口处理的数据将会不带处理效果。

| | |
|---|---|
| 函数原型 | void pause_pitch(s_pitch_hdl *pitch_hdl, u8 run_mark); |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 功能描述 | 暂停变声处理 |
| 参数说明 | * \param[in] pitch_hdl//变声句柄.<br>* \param[in] run_mark//暂停标记：<br>1： 启动暂停<br>0： 正常运行 |
| 输出 | * \return 无 |
| 关联模块 | |
| 例子 | |
| 补充说明 | |

## 10.4. 混响

### 10.4.1. 功能介绍

混响 模块包含 2 种模式：echo 和 reverb。echo 用于 K 歌 mic；reverb 用于声卡。模块主要提供包含模块打开、模块关闭、模块数据处理、模块参数切换等接口给 AP 应用调用。

混响模块只涉及 AP 层，与其他没有任何关系。

Echo 模式在 16K 采样率情况下占用空间 12K，占用时钟 8M;

Reverb 模式在 44.1K 采样率情况下占用空间 46K，（自带单声道变双声道功能）

### 10.4.2. 接口介绍

#### 10.4.2.1. Echo 混响打开

按照输入参数打开 echo 混响模块，申请所需的运算空间获取后续运行、关闭的句柄。

| | |
|---|---|
| 函数原型 | ECHO_API_STRUCT *open_echo(ECHO_PARM_SET *echo_seting, u16 sample_rate) |
| 功能描述 | 打开 echo 混响模块函数 |
| 参数说明 | * \param[in] echo_seting //混响参数<br>typedef struct _EF_ECHO__PARM_ {<br>unsigned int delay; //回声的延时时间 0-200ms<br>unsigned int decayval; // 0-70%<br>unsigned int direct_sound_enable; //直达声使能 0/1<br>unsigned int filt_enable; //发散滤波器使能<br>} ECHO_PARM_SET;<br>* \param[in ] sample_rate //待处理数据的采样率； |
| 输出 | * \return 返回 ECHO_API_STRUCT 结构指针<br>* \retval >1. echo 混响模块运算及设置的句柄 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | * \retval NULL 打开 echo 混响模块失败 |
| 例子 | effect->p_echo_hdl=open_echo(&effect_echo_parm_default,effect->parm.sample_rate); |
| 关联模块 | |
| 补充说明 | 输入参数为 NULL 时 使用内部默认参数 |

#### 10.4.2.2. Reverb 混响打开

按照输入参数打开 reverb 混响模块，申请所需的运算空间获取后续运行、关闭的句柄。

| | |
|---|---|
| 函数原型 | REVERBN_API_STRUCT*open_reverb(REVERBN_PARM_SET*reverb_seting, u16 sample_rate) |
| 功能描述 | 打开 reverb 混响模块函数 |
| 参数说明 | * \param[in] reverb_seti //混响参数<br>typedef struct REVERBN_PARM_SET {<br>int dry; // 0-200%<br>int wet; //0-300%<br>int delay; //0-100ms<br>int rot60; //100-15000ms<br>int Erwet; // 5%- 250%<br>int Erfactor; // 50%-250%<br>int Ewidth; // -100% - 100%<br>int Ertolate; // 0- 100%<br>int predelay; // 0- 20ms<br>int width; //0% - 100%<br>int diffusion; //0% - 100%<br>int dampinglpf; //0-18k<br>int basslpf; //0-1.1k<br>int bassB; // 0-80%<br>int inputlpf; //0-18k<br>int outputlpf; // 0-18k<br>} REVERBN_PARM_SET;<br>* \param[in ] sample_rate //待处理数据的采样率； |
| 输出 | * \return 返回 REVERBN_API_STRUCT 结构指针<br>* \retval >1. echo 混响模块运算及设置的句柄<br>* \retval NULL 打开 echo 混响模块失败 |
| 例子 | effect->p_reverb_hdl=open_reverb(&effect_reverb_parm_default,effect->parm.sample_rate); |
| 关联模块 | |
| 补充说明 | 输入参数为 NULL 时 使用内部默认参数 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

#### 10.4.2.3. Echo 混响关闭

该功能主要是关闭 echo 混响模块，释放模块申请的空间。

| | |
|---|---|
| 函数原型 | void close_echo(ECHO_API_STRUCT *echo_api_obj) |
| 功能描述 | 关闭 echo 混响函数 |
| 参数说明 | * \param[in] echo_api_ob //echo 混响句柄 |
| 输出 | * \return 无 |
| 关联模块 | |
| 例子 | close_echo(effect->p_echo_hdl); |
| 补充说明 | |

#### 10.4.2.4. Reverb 混响关闭

该功能主要是关闭 echo 混响模块，释放模块申请的空间。

| | |
|---|---|
| 函数原型 | void close_reverb(REVERBN_API_STRUCT *reverb_api_obj) |
| 功能描述 | 关闭 reverb 混响函数 |
| 参数说明 | * \param[in] reverb_api_obj //reverb 混响句柄 |
| 输出 | * \return 无 |
| 关联模块 | |
| 例子 | close_reverb(effect->p_reverb_hdl); |
| 补充说明 | |

#### 10.4.2.5. echo 混响参数更新

主要是 echo 混响参数设置。

| | |
|---|---|
| 函数原型 | void update_echo_parm(ECHO_API_STRUCT *echo_api_obj, ECHO_PARM_SET *echo_seting) |
| 功能描述 | 打开资源文件 |
| 参数说明 | * \param[in] echo_api_obj //echo 混响句柄<br>* \param[in] echo_setingj //新的 echo 混响设置参数 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | |
| 输出 | * \retrun 无 |
| 关联模块 | |
| 例子 | |
| 补充说明 | |

#### 10.4.2.6. reverb 混响参数更新

主要是 reverb 混响参数变更设置。

| | |
|---|---|
| 函数原型 | void update_reverb_parm(REVERBN_API_STRUCT *reverb_api_obj, REVERBN_PARM_SET *reverb_seting) |
| 功能描述 | reverb 混响参数变更 |
| 参数说明 | * \param[in] reverb_api_ob//reverb 混响句柄.<br>* \param[in] reverb_seting //新的 reverb 混响设置参数 |
| 输出 | * \return 无 |
| 关联模块 | |
| 例子 | |
| 补充说明 | |

#### 10.4.2.7. echo 混响数据处理

主要是对输入音频数据进行 echo 混响处理

| | |
|---|---|
| 函数原型 | void run_echo(ECHO_API_STRUCT *p_echo_obj, short *in, short *out, int len) |
| 功能描述 | 对输入音频数据进行 echo 混响处理 |
| 参数说明 | * \param[in] p_echo_obj //echo 混响句柄.<br>* \param[in] in //待处理的音频数据指针地址.<br>* \param[out] out //echo 混响处理后的数据输出指针地址.<br>* \param[in] len //待处理的音频数据长度 BYTE. |
| 输出 | * \return 无 |
| 关联模块 | |
| 例子 | |
| 补充说明 | 若模块串入 stream 流，无需显式调用处理接口 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

#### 10.4.2.8. Reverb 混响处理数据

主要是对输入音频数据进行 reverb 混响处理，数据要求：单声道入双声道出

| | |
|---|---|
| 函数原型 | void run_reverb(REVERBN_API_STRUCT *reverb_api_obj, short *in, short *out, int len) |
| 功能描述 | 对输入音频数据进行 echo 混响处理 |
| 参数说明 | * \param[in] scrollstr //reverb 混响句柄.<br>* \param[in] in //待处理的音频数据指针地址.<br>* \param[out] out///echo 混响处理后的数据输出指针地址<br>* \param[in] len//待处理的音频数据长度 BYTE |
| 输出 | * \return 无 |
| 关联模块 | |
| 例子 | |
| 补充说明 | 若模块串入 stream 流，无需显式调用处理接口 |
| | |

#### 10.4.2.9. Pause_echo

| | |
|---|---|
| 函数原型 | void pause_echo(ECHO_API_STRUCT *echo_api_obj, u8 run_mark) |
| 功能描述 | 暂停 echo 混响处理 |
| 参数说明 | * \param[in] echo_api_obj//echo 混响句柄.<br>* \param[in] run_mark//暂停标记：<br>1： 启动暂停<br>0： 正常运行 |
| 输出 | * \return 无 |
| 关联模块 | |
| 例子 | |
| 补充说明 | |

#### 10.4.2.10. pause_reverb

| | |
|---|---|
| 函数原型 | void pause_reverb(REVERBN_API_STRUCT *reverb_api_obj, u8 run_mark) |
| 功能描述 | 暂停 reverb 混响处理 |



地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 参数说明 | * \param[in] reverb_api_obj//混响句柄<br>* \param[in] run_mark//暂停标记：<br>1： 启动暂停<br>0： 正常运行 |
| 输出 | * \return<br>* \retval TRUE 成功<br>* \retval FALSE 失败 |
| 关联模块 | |
| 例子 | |
| 补充说明 | |

## 10.5. 啸叫抑制

### 10.5.1. 功能介绍

啸叫抑制模块主要是防止啸叫。模块主要提供包含模块打开、模块关闭、模块数据处理、模块参数切换等接口给 AP 应用调用。

啸叫抑制模块只涉及 AP 层，与其他没有任何关系。

啸叫抑制（移频）模块在 16K 采样率情况下占用约空间 2K 空间，约占用时钟 10M

### 10.5.2. 接口介绍

#### 10.5.2.1. Howling 模块打开

按照输入参数打开啸叫抑制模块，申请所需的运算空间获取后续运行、关闭的句柄。

| | |
|---|---|
| 函数原型 | HOWLING_API_STRUCT *open_howling(void *howl_para, u16 sample_rate, u8 channel, u8 mode); |
| 功能描述 | 打开啸叫抑制模块函数 |
| 参数说明 | * \param[in] howl_para //私有参数 预留使用<br>* \param[in] sample_rate //待处理数据的采样率<br>* \param[in] channel //待处理数据的通道数<br>* \param[in] mode //选择啸叫抑制模式 预留 固定传 1 |
| 输出 | * \return 返回 HOWLING_API_STRUCT 结构指针<br>* \retval >1. 啸叫抑制模块运算及设置的句柄 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | * \retval NULL 打开啸叫抑制模块失败 |
| 例子 | effect->p_howling_hdl = open_howling(NULL, effect->parm.sample_rate, 0, 1); |
| 关联模块 | |
| 补充说明 | 输入参数为 NULL 时 使用内部默认参数 |

#### 10.5.2.2. Howling 模块关闭

该功能主要是关闭啸叫抑制模块，释放模块申请的空间。

| | |
|---|---|
| 函数原型 | void close_howling(HOWLING_API_STRUCT *holing_hdl); |
| 功能描述 | 关闭啸叫抑制函数 |
| 参数说明 | * \param[in] holing_hdl //啸叫抑制句柄 |
| 输出 | * \return 无 |
| 关联模块 | |
| 例子 | close_howling(effect->p_howling_hdl); |
| 补充说明 | |

#### 10.5.2.3. Howling 模块数据处理

主要是对输入音频数据进行啸叫抑制处理

| | |
|---|---|
| 函数原型 | void run_howling(HOWLING_API_STRUCT *howling_hdl, short *in, short *out, int len); |
| 功能描述 | 对输入音频数据进行 echo 混响处理 |
| 参数说明 | * \param[in] howling_hdl /啸叫抑制句柄.<br>* \param[in] in //待处理的音频数据指针地址.<br>* \param[out] out //变声处理后的数据输出指针地址.<br>* \param[in] len //待处理的音频数据长度 BYTE. |
| 输出 | * \return 无 |
| 关联模块 | |
| 例子 | |
| 补充说明 | 若模块串入 stream 流，无需显式调用处理接口 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

# 11. 应用模块功能介绍

## 11.1. DEC_APP

### 11.1.1. 功能介绍

本模块用于封装 decoder 的通用流程，上层简单调用就可以实现解码应用。
本模块关联的其他模块有：DECODER。

### 11.1.2. 数据结构介绍

```
// dec app
struct audio_dec_app_hdl {
    u32 mask;       // 固定为 AUDIO_DEC_APP_MASK
    u32 id;         // 唯一标识符，随机值
    struct list_head list_entry;            // 链表
    struct audio_stream *stream;            // 音频流
    struct audio_decoder decoder;           // 解码器
    struct audio_res_wait wait;             // res 资源（解码顺序、抢占等）
    struct audio_mixer_ch mix_ch;           // 叠加通道
    enum audio_channel ch_type;             // 声道类型
    u32 status : 3;             // 状态
    u32 ch_num : 4;             // channel 通道数
    u32 out_ch_num : 4;         // 输出声道数
    u32 tmp_pause : 1;          // 临时暂停（被其他解码打断）
    u32 dec_mix : 1;            // 1:叠加模式
    u32 remain : 1;             // 输出不完成标记
    u32 frame_type : 1;         // 1:frame 格式；0:file 格式
    u32 close_by_res_put : 1;     //被打断就自动 close
    u16 frame_pkt_len;          // frame 格式帧长
    u16 frame_data_len;         // frame 格式当前数据长度
    u8 *frame_buf;              // frame 格式帧 buf
    u32 dec_type;               // 指定解码格式，start 后为实际解码格式
    u16 sample_rate;            // 指定采样率，start 后为实际解码采样率
    u16 resume_tmr_id;          // time 激活 ID
    struct audio_decoder_task *p_decode_task;  // 解码任务
```



地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

```
    struct audio_mixer *p_mixer;        // 叠加器
    int (*evt_cb)(void *, int event, int *param); // 事件回调
    void *evt_priv;             // 事件回调句柄
    struct audio_dec_input dec_input;       // 解码输入接口
    struct audio_dec_input *input;          // 指定使用 input 接口
    struct audio_dec_handler *handler;      // 指定使用 handler 接口
    void *file_hdl;                         // 文件句柄
    struct audio_dec_app_file_hdl *file;    // 文件操作接口
    void *app_hdl;      // 指向上一级句柄
};

// 普通文件解码
struct audio_dec_file_app_hdl {
    struct audio_dec_app_hdl *dec;          // decapp 句柄
    struct audio_dec_format_hdl  *format; // 后缀转换成解码类型的结构体数组
    u32     flag;           // 标签，可用于设置 audio 通道等
    void *file_hdl;     // 文件句柄
    void *priv;         // 私有参数
};

// 正弦波解码
struct audio_dec_sine_app_hdl {
    struct audio_dec_app_hdl *dec;    // decapp 句柄
    void *sin_maker;          // 正弦波句柄
    struct audio_sin_param sin_parm[AUDIO_DEC_SINE_APP_NUM_MAX];    // 正弦波数组
    struct audio_sin_param *sin_src;        // 上层传入的正弦波数组
    u8      sin_num;          // 正弦波数组大小
    u8      sin_repeat;       // 循环次数
    u16     sin_default_sr;   // 采样率
    u32     sin_volume;       // 音量
    u32     flag;             // 标签，可用于设置 audio 通道等
    void *file_hdl;           // 正弦波文件句柄
    void *priv;               // 私有参数
};
```

### 11.1.3. 通用接口

| 函数原型 | struct audio_dec_app_hdl *audio_dec_app_create(void *priv, int (*evt_cb)(void *, int event, int *param), u8 mix) |
|---|---|
| 功能描述 | 创建一个 dec_app 模块 |
| 参数说明 | * \param[in]  *priv  // 事件回调私有句柄 |



地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | * \param[in] *evt_cb // 事件回调接口<br>* \param[in] *mix // 1-叠加播放，0-抢占播放 |
| 输出 | * \return dec_app 句柄 |
| 例子 | |
| 关联模块 | |
| 补充说明 | 该函数会统一调用 audio_dec_app_create_parram_init()弱函数。由上层重新实现该弱函数，设置通用的 p_decode_task、p_mixer 和 out_ch_num 等值。也可以在该函数之后，audio_dec_app_open()之前重新设置想要的参数 |

| | |
|---|---|
| 函数原型 | int audio_dec_app_open(struct audio_dec_app_hdl *dec) |
| 功能描述 | 打开 dec_app 解码 |
| 参数说明 | * \param[in][out] *dec // dec_app 句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval false. 失败<br>* \retval true 成功 |
| 例子 | |
| 关联模块 | |
| 补充说明 | 和 audio_dec_app_close()配套使用 |

| | |
|---|---|
| 函数原型 | void audio_dec_app_close(struct audio_dec_app_hdl *dec) |
| 功能描述 | 关闭 dec_app 解码 |
| 参数说明 | * \param[in][out] *dec // dec_app 句柄 |
| 输出 | * \return 无 |
| 例子 | |
| 关联模块 | |
| 补充说明 | 和 audio_dec_app_open()配套使用 |

| | |
|---|---|
| 函数原型 | void audio_dec_app_set_file_info(struct audio_dec_app_hdl *dec, void *file_hdl) |
| 功能描述 | 设置文件句柄 |
| 参数说明 | * \param[in][out] *dec // dec_app 句柄<br>* \param[in] *file_hdl // 文件句柄 |
| 输出 | * \return 无 |
| 例子 | |
| 关联模块 | |
| 补充说明 | 在 audio_dec_app_open()前调用 |

| | |
|---|---|
| 函数原型 | void audio_dec_app_set_frame_info(struct audio_dec_app_hdl *dec, u16 pkt_len, |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | u32 coding_type) |
| 功能描述 | 设置解码为 frame 类型 |
| 参数说明 | * \param[in][out] *dec // dec_app 句柄<br>* \param[in] pkt_len // frame 帧长<br>* \param[in] coding_type // frame 解码类型 |
| 输出 | * \return 无 |
| 例子 | |
| 关联模块 | |
| 补充说明 | 在 audio_dec_app_open()前调用。Frame 类型的解码会根据帧长申请 frame 所需的空间 |

| | |
|---|---|
| 函数原型 | int audio_dec_app_pp(struct audio_dec_app_hdl *dec) |
| 功能描述 | Dec_app 解码暂停/播放控制 |
| 参数说明 | * \param[in][out] *dec // dec_app 句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval false. 失败<br>* \retval true 成功 |
| 例子 | |
| 关联模块 | |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int audio_dec_app_get_status(struct audio_dec_app_hdl *dec) |
| 功能描述 | 获取 dec_app 状态 |
| 参数说明 | * \param[in][out] *dec // dec_app 句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 负数 句柄错误<br>enum {<br>AUDIO_DEC_APP_STATUS_STOP = 0,<br>AUDIO_DEC_APP_STATUS_PLAY,<br>AUDIO_DEC_APP_STATUS_PAUSE,<br>}; |
| 例子 | |
| 关联模块 | |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int audio_dec_app_check_hdl(struct audio_dec_app_hdl *dec) |
| 功能描述 | 检查 dec_app 句柄是否正常 |
| 参数说明 | * \param[in][out] *dec // dec_app 句柄 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval false. 失败<br>* \retval true 正常 |
| 例子 | |
| 关联模块 | |
| 补充说明 | 由于解码和控制是在不同的 task 中，句柄可能已经被释放 |

### 11.1.4. 普通文件解码接口

| | |
|---|---|
| 函数原型 | struct audio_dec_file_app_hdl *audio_dec_file_app_create(char *name, u8 mix) |
| 功能描述 | 根据名字创建一个 dec_app 解码 |
| 参数说明 | * \param[in] *name // 文件名<br>* \param[in] mix // 1-叠加播放，0-抢占播放 |
| 输出 | * \return file_dec 句柄 |
| 例子 | |
| 关联模块 | |
| 补充说明 | 该函数会打开文件，并且根据文件名找到对应的解码类型（也可以不需要，由 decoder 自动匹配）。当然也可以在 open 之前重新手动更改。该函数默认使能解码被抢断就自动结束功能（dec->close_by_res_put） |

| | |
|---|---|
| 函数原型 | int audio_dec_file_app_open(struct audio_dec_file_app_hdl *file_dec) |
| 功能描述 | 打开 file_dec 解码 |
| 参数说明 | * \param[in][out] *file_dec // 解码句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval false. 失败<br>* \retval true 成功 |
| 例子 | |
| 关联模块 | |
| 补充说明 | 和 audio_dec_file_app_close()配套使用 |

| | |
|---|---|
| 函数原型 | void audio_dec_file_app_close(struct audio_dec_file_app_hdl *file_dec) |
| 功能描述 | 关闭 file_dec 解码 |
| 参数说明 | * \param[in][out] *file_dec // 解码句柄 |
| 输出 | * \return 无 |
| 例子 | |
| 关联模块 | |
| 补充说明 | 和 audio_dec_file_app_open()配套使用 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

### 11.1.5. 正弦波解码接口

| | |
|---|---|
| 函数原型 | struct audio_dec_sine_app_hdl *audio_dec_sine_app_create(char *name, u8 mix) |
| 功能描述 | 根据名字创建一个正弦波解码 |
| 参数说明 | * \param[in] *name // 文件名<br>* \param[in] mix // 1-叠加播放，0-抢占播放 |
| 输出 | * \return 正弦波解码句柄 |
| 例子 | |
| 关联模块 | |
| 补充说明 | 该函数会打开文件，设置一些正弦波默认参数。默认使能解码被抢断就自动结束功能（dec->close_by_res_put） |

| | |
|---|---|
| 函数原型 | struct audio_dec_sine_app_hdl *audio_dec_sine_app_create_by_parm(struct audio_sin_param *sin, u8 sin_num, u8 mix) |
| 功能描述 | 根据正弦波参数数组创建一个正弦波解码 |
| 参数说明 | * \param[in] *sin // 正弦波参数数组<br>* \param[in] sin_num // 数组长度<br>* \param[in] mix // 1-叠加播放，0-抢占播放 |
| 输出 | * \return 正弦波解码句柄 |
| 例子 | |
| 关联模块 | |
| 补充说明 | 默认使能解码被抢断就自动结束功能（dec->close_by_res_put） |

| | |
|---|---|
| 函数原型 | int audio_dec_sine_app_open(struct audio_dec_sine_app_hdl *sine_dec) |
| 功能描述 | 打开 sin_dec 解码 |
| 参数说明 | * \param[in][out] *sine_dec // 解码句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval false. 失败<br>* \retval true 成功 |
| 例子 | |
| 关联模块 | |
| 补充说明 | 和 audio_dec_sine_app_close()配套使用 |

| | |
|---|---|
| 函数原型 | void audio_dec_sine_app_close(struct audio_dec_sine_app_hdl *sine_dec) |
| 功能描述 | 关闭 sin_dec 解码 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| 参数说明 | * \param[in][out]　　*sine_dec // 解码句柄 |
|---|---|
| 输出 | * \return　　无 |
| 例子 | |
| 关联模块 | |
| 补充说明 | 和 audio_dec_sine_app_open()配套使用 |

| 函数原型 | void audio_dec_sine_app_probe(struct audio_dec_sine_app_hdl *sine_dec) |
|---|---|
| 功能描述 | 正弦波解码预处理 |
| 参数说明 | * \param[in][out]　　*sine_dec // 解码句柄 |
| 输出 | * \return　　无 |
| 例子 | |
| 关联模块 | |
| 补充说明 | 从正弦波数组或者正弦波文件中转换一些解码信息 |

### 11.1.6. 示例

#### 11.1.6.1. 简单的普通文件解码

```
struct audio_dec_file_app_hdl *hdl;
hdl = audio_dec_file_app_create(TONE_POWER_ON, 1);
if (hdl) {
    audio_dec_file_app_open(hdl);
}
// 可以不用释放，解码结束后会自动释放，也可以调用 audio_dec_file_app_close()手动关闭
```

#### 11.1.6.2. 简单的正弦波文件解码

```
struct audio_dec_sine_app_hdl *hdl;
hdl = audio_dec_sine_app_create(SDFILE_RES_ROOT_PATH"tone/vol_max.sin", 1);
if (hdl) {
    audio_dec_sine_app_open(hdl);
}
// 可以不用释放，解码结束后会自动释放，也可以调用 audio_dec_sine_app_close()手动关闭
```




地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

#### 11.1.6.3. 简单的正弦波数组解码

```
static const struct audio_sin_param sine_test[] = {
    {200 << 9, 4000, 0, 100},
};
struct audio_dec_sine_app_hdl *hdl;
hdl = audio_dec_sine_app_create_by_parm(sine_test, ARRAY_SIZE(sine_test), 1);
if (hdl) {
    audio_dec_sine_app_open(hdl);
}
// 可以不用释放，解码结束后会自动释放，也可以调用 audio_dec_sine_app_close()手动关闭
```

#### 11.1.6.4. 更改解码参数

Dec_app 模块 create 与 open 解码分开，create 的时候会设置一些默认参数让解码器能跑起来，也可以在 open 之前重新设置一些参数用以达到自己想要的效果。

比如以 frame 方式启动一个 msbc 文件解码：

```
hdl = audio_dec_file_app_create(SDFILE_RES_ROOT_PATH"tone/test.msbc", 1);
if (hdl) {
    audio_dec_app_set_frame_info(hdl->dec, 128, AUDIO_CODING_MSBC);// 指定为 msbc 解码
    audio_dec_file_app_open(hdl);
}
```

一些通用的参数设置示例：

```
hdl = audio_dec_file_app_create(TONE_POWER_ON, 1);
if (hdl) {
    If (不用抢占或者叠加的方式播放，按优先级排序播放) {
        Hdl->dec.dec_mix = 0;
        Hdl->dec.wait.protect = 0;
        Hdl->dec.wait.preemption = 0;
        Hdl->dec.wait.priority = n;
        Hdl->dec.wait.snatch_same_prio = 1; // 1-放在同优先级的前面，0-放同优先级后面
    }
    If (被其他解码打断后不结束，而是暂停解码，等打断释放后重新解码) {
        Hdl->dec->close_by_res_put = 0;
    }
    If (使用指定的 input 接口。比如文件是加密的，在 input 中解码等) {
        Hdl->dec->input = 指定 input;
    }
    If (使用指定的解码 handler。一般不用改过，都可以在事件回调中处理) {
        Hdl->dec->handler = 指定 handler;
```




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

```
    }
    If (指定查找解码类型) {
        Hdl->dec->dec_type = AUDIO_CODING_MP3 | AUDIO_CODING_WMA...;
    }
    If (指定声道类型) {
        // 默认为 AUDIO_CH_MAX，会根据 dec->out_ch_num 值自动设定
        Hdl->dec->ch_type = AUDIO_CH_LR;
    }
    audio_dec_file_app_open(hdl);
}
```

Dec_app 可以设置回调接口。在回调中可以重定义数据流、设置时钟等。

```
Static int test_dec_evt_cb(void priv, int event, int *param)
{
    struct audio_dec_file_app_hdl *file_dec;
    Switch (event) {
    // 下面的项是在解码任务里调用的
    Case AUDIO_DEC_APP_EVENT_DEC_PROBE:
        // 解码 run 预处理，返回值见 decoder 模块说明
        // 如果是正弦波，解码预处理的时候一定要添加以下处理
        // if (sine_dec->sin_maker) {
        //         break;
        // }
        // audio_dec_sine_app_probe(sine_dec);
        // if (!sine_dec->sin_maker) {
        //         return -ENOENT;
        // }
        Break;
    Case AUDIO_DEC_APP_EVENT_DEC_OUTPUT:// param[0]:data, param[1]:len
        // 解码 run 数据输出，不判断返回值
        Break;
    Case AUDIO_DEC_APP_EVENT_DEC_POST:
        // 解码 run 后处理，返回值见 decoder 模块说明
        Break;
    Case AUDIO_DEC_APP_EVENT_DEC_STOP:
        // 解码结束后的处理
        Break;

    // 下面的项是在控制任务里调用的
    Case AUDIO_DEC_APP_EVENT_START_INIT_OK:
        // 该项在 audio_decoder_start()之前，可以在这里重新定义数据流，设置时钟等
        If { // 更改数据流，删除原有的数据流，需要在回调中重新设置
            if (file_dec->dec->stream) {
```



地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

```
                audio_stream_del_entry(&file_dec->dec->mix_ch.entry);
                audio_stream_del_entry(&file_dec->dec->decoder.entry);
                audio_stream_close(file_dec->dec->stream);
                file_dec->dec->stream = NULL;
            }
            struct audio_stream_entry *entries[8] = {NULL};
            u8 entry_cnt = 0;
            entries[entry_cnt++] = &file_dec->dec->decoder.entry;
            {
                // 添加一个 eq/drc
                test_tone_eq_drc = file_eq_drc_open(file_dec->dec->sample_rate, file_dec->dec->ch_num);
                entries[entry_cnt++] = &test_tone_eq_drc->entry;
            }
            entries[entry_cnt++] = &file_dec->dec->mix_ch.entry;
            file_dec->dec->stream = audio_stream_open(file_dec->dec, tone_test_stream_resume);
            audio_stream_add_list(file_dec->dec->stream, entries, entry_cnt);
        }
        // 设置时钟
        clock_add(DEC_TONE_CLK);
        clock_set_cur();
        // 设置 audio state 等
        audio_dec_file_app_init_ok(file_dec);
        Break;
    Case AUDIO_DEC_APP_EVENT_START_OK:
        // 该项在 audio_decoder_start()之后，启动成功
        Break;
    Case AUDIO_DEC_APP_EVENT_START_ERR:
        // 启动失败
        Break;
    Case AUDIO_DEC_APP_EVENT_DEC_CLOSE:
        // 解码关闭，该项在 decoder 关闭、mixer 关闭之后，数据流删除之前，因此需要在此处删除自定义的数据流
        {
            // 如果有添加自己的数据流等，需要在这里删除
            if (test_tone_eq_drc) {
                file_eq_drc_close(test_tone_eq_drc);
                test_tone_eq_drc = NULL;
            }
        }
        Break;
    Case AUDIO_DEC_APP_EVENT_PLAY_END:
        // 解码结束发消息给上层，上层 task 收到结束消息时的处理
        audio_dec_file_app_play_end(file_dec);
```




地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

```
            Break;
        }
        Return 0;
}
hdl = audio_dec_file_app_create(TONE_POWER_ON, 1);
if (hdl) {
        Hdl->dec->evt_cb = test_dec_evt_cb;
        Hdl->dec->priv = hdl;
        audio_dec_file_app_open(hdl);
}
```




地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

## 11.2. TONE 通用接口

### 11.2.1. 功能介绍

该模块主要用于提示音播放。

该模块基于 DEC_APP 开发，可重入，因此可以创建多个解码，可以用于普通的叠加播放。

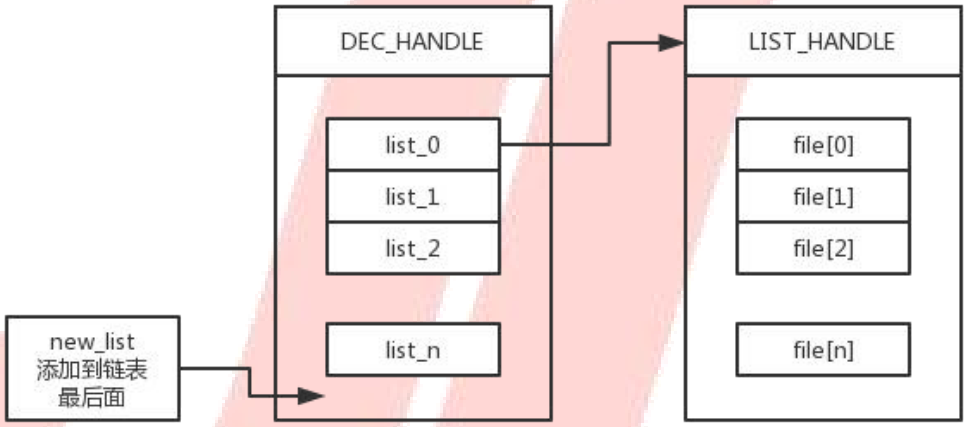
该模块为链表结构，可以串联多个 LIST 依次播放，list 中使用 file 数组，可以实现 file 循环播放等。

TONE 模块可以支持正弦波序号（需要添加正弦波数组转换获取回调函数）播放、正弦波文件播放、普通文件播放。

SDK 中在 tone_player.c 中实现了 tone 应用，共用*tone_dec 句柄，不可重入，但调用简单。

TONE 模块关联的其他模块有：DECODER，MIXER。

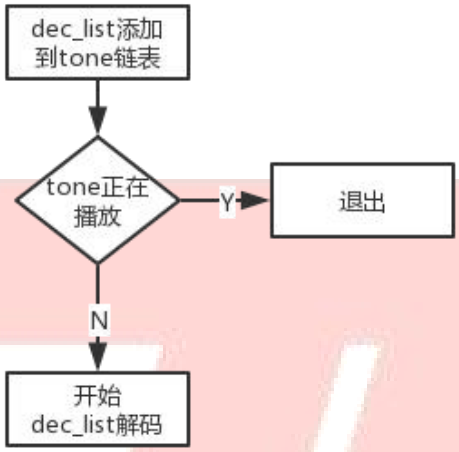
TONE 模块链表框架如下图所示：






地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

Tone 添加一个 dec_list 解码流程图（具体的 list 解码见 dec_list 解码流程图）如下图所示：

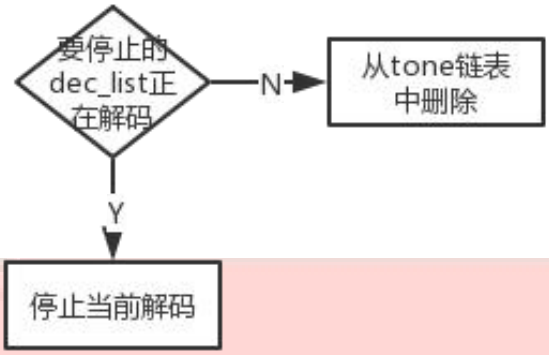



Tone 停止所有解码流程图如下图所示：

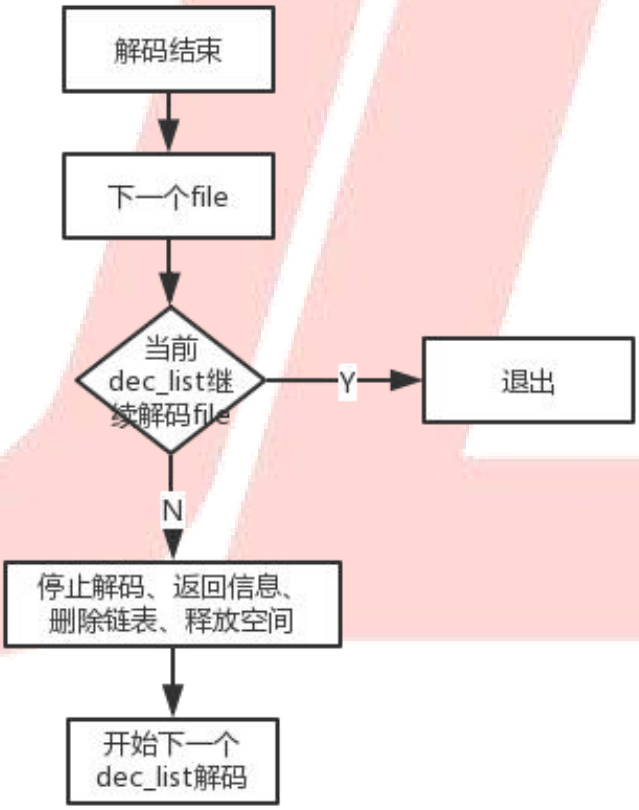

Tone 停止指定解码，如果当前要停止的没有正在播放，仅从链表中删除，不停止当前解码。流程图如下图所示：



地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com



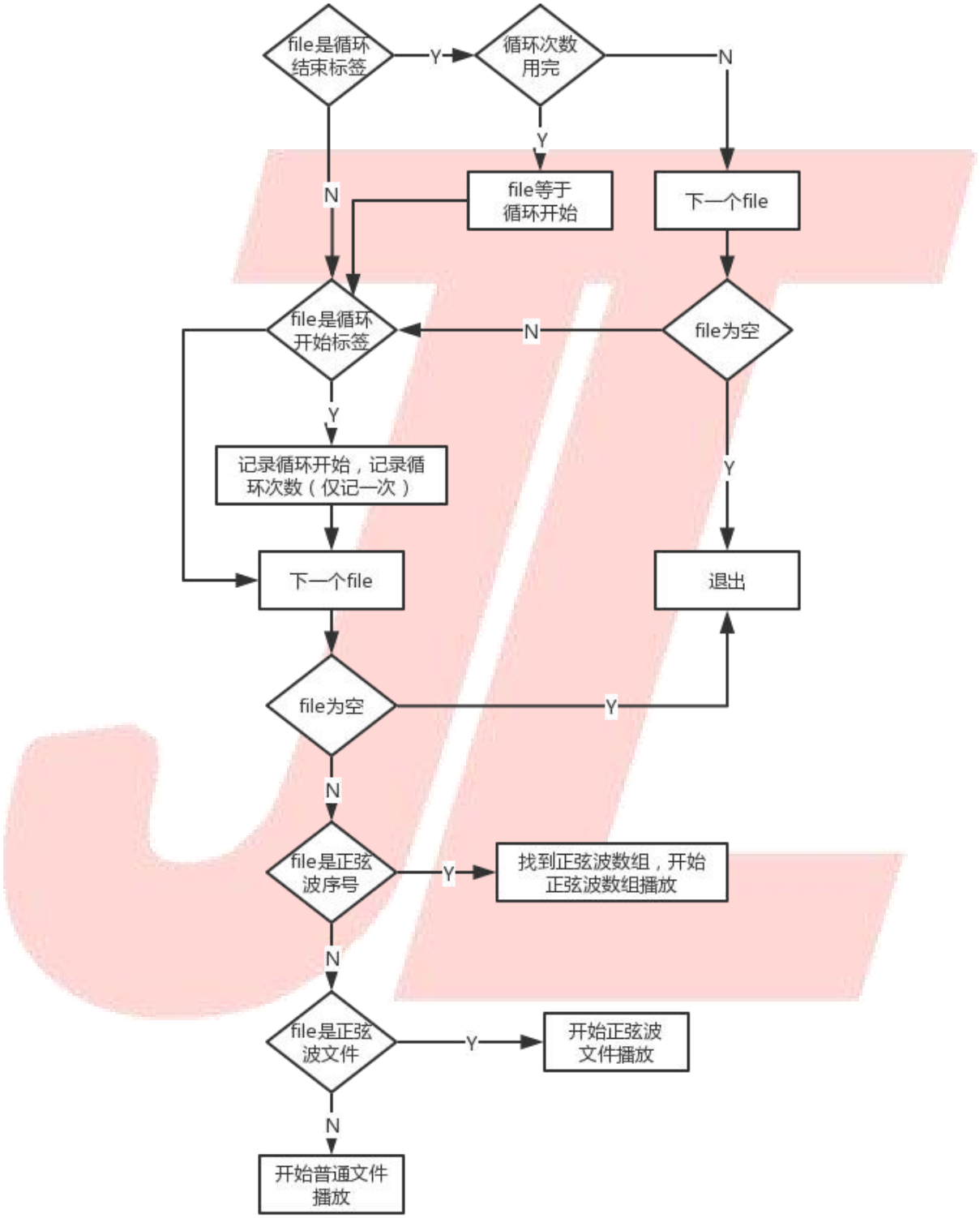

Tone 当前解码结束发送信息到任务中，任务中获取消息后自动开始下一个解码，流程图如下图所示：






地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

Tone 模块 dec_list 播放流程图如下图所示：







地址：珠海市吉大石花西路 107 号 9 栋综合楼　　邮编：519015
电话：0756-6313088　　传真：0756-6313081
网站：www.zh-jieli.com

### 11.2.2. 数据结构介绍

```
// TONE DECODER
struct tone_dec_handle {
    struct list_head head;      // 链表头
    struct audio_dec_sine_app_hdl *dec_sin;     // 文件播放句柄
    struct audio_dec_file_app_hdl *dec_file;    // sine 播放句柄
    struct tone_dec_list_handle *cur_list;      // 当前播放 list
    struct sin_param *(*get_sine)(u8 id, u8 *num);   // 按序列号获取 sine 数组
    OS_MUTEX mutex;             // 互斥
};

// TONE LIST DECODER
struct tone_dec_list_handle {
    struct list_head list_entry;    // 链表
    u8 preemption : 1;          // 打断
    u8 idx;                     // 循环播放序号
    u8 repeat_begin;            // 循环播放起始序号
    u16 loop;                   // 循环播放次数
    char **file_list;           // 文件名
    const char *evt_owner;                          // 事件接受任务
    void (*evt_handler)(void *priv, int flag);      // 事件回调
    void *evt_priv;                                 // 事件回调私有句柄
    void (*stream_handler)(void *priv, int event, struct audio_dec_app_hdl *);// 数据流设置回调
    void *stream_priv;                              // 数据流设置回调私有句柄
};
```

### 11.2.3. 接口介绍

| | |
|---|---|
| 函数原型 | struct tone_dec_handle *tone_dec_create(void) |
| 功能描述 | 创建一个 tone 句柄 |
| 参数说明 | * \param 无 |
| 输出 | * \return tone 句柄 |
| 例子 | struct tone_dec_handle *test_tone = tone_dec_create(); |
| 关联模块 | |
| 补充说明 | 和 tone_dec_stop()函数配套使用 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 函数原型 | void tone_dec_set_sin_get_hdl(struct tone_dec_handle *dec, struct sin_param *(*get_sine)(u8 id, u8 *num)) |
| 功能描述 | 设置 sine 数组获取回调 |
| 参数说明 | * \param[in][out] *dec // tone 句柄<br>* \param[in] *get_sine // 正弦波序号转数组函数 |
| 输出 | * \return 无 |
| 例子 | Static const struct sin_param test_sine_tab[] = {<br>{200<<9, 4000, 0, 100},<br>};<br>Static const struct sin_param *test_get_sine_tab(u8 id, u8 *num)<br>{<br>If (id == n) {<br>*num = ARRAY_SIZE(test_sine_tab);<br>Return test_sine_tab;<br>}<br>Return NULL;<br>}<br>tone_dec_set_sin_get_hdl(test_tone, test_get_sine_tab); |
| 关联模块 | |
| 补充说明 | 如果有正弦波序号播放，需要实现该接口 |

| | |
|---|---|
| 函数原型 | struct tone_dec_list_handle *tone_dec_list_create(struct tone_dec_handle *dec,<br>const char **file_list,<br>u8 preemption,<br>void (*evt_handler)(void *priv, int flag),<br>void *evt_priv,<br>void (*stream_handler)(void *priv, int event, struct audio_dec_app_hdl *app_dec),<br>void *stream_priv); |
| 功能描述 | 创建提示音播放 list 句柄 |
| 参数说明 | * \param[in][out] *dec // tone 句柄<br>* \param[in] **file_list // 文件列表<br>* \param[in] preemption // 1-抢占方式播放；0-叠加方式播放<br>* \param[in] *evt_handler // 播放结束事件回调，flag0 正常结束，1 被打断<br>* \param[in] *evt_priv // 播放结束事件回调私有句柄<br>* \param[in] *stream_handler // 数据流开始/结束设置回调<br>* \param[in] *stream_priv // 数据流开始/结束设置回调私有参数 |
| 输出 | * \return dec_list 句柄 |
| 例子 | static void tone_test_stream_resume(void *p) |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

```
{ // 激活解码
    struct audio_dec_app_hdl *app_dec = p;
    audio_decoder_resume(&app_dec->decoder);
}
static void tone_test_stream_handler(void *priv, int event, struct audio_dec_app_hdl
*app_dec)
{
    switch (event) {
    case AUDIO_DEC_APP_EVENT_START_INIT_OK:
        y_printf("AUDIO_DEC_APP_EVENT_START_INIT_OK \n");
        // 数据流开始设置
        struct audio_stream_entry *entries[8] = {NULL};
        u8 entry_cnt = 0;
        // 第一个数据流是解码输出
        entries[entry_cnt++] = &app_dec->decoder.entry;
        { // 私有数据流在这里添加
            // 打开并添加一个 eq 音效数据流
            test_tone_eq_drc = file_eq_drc_open(app_dec->sample_rate,
app_dec->ch_num);
            entries[entry_cnt++] = &test_tone_eq_drc->entry;
        }
        // 最后一个数据流是 MIXER
        entries[entry_cnt++] = &app_dec->mix_ch.entry;
        app_dec->stream = audio_stream_open(app_dec,
tone_test_stream_resume);
        audio_stream_add_list(app_dec->stream, entries, entry_cnt);
        break;
    case AUDIO_DEC_APP_EVENT_DEC_CLOSE:
        y_printf("AUDIO_DEC_APP_EVENT_DEC_CLOSE \n");
        // 数据流结束设置
        {
            // 删除私有的数据流
            if (test_tone_eq_drc) {
                file_eq_drc_close(test_tone_eq_drc);
                test_tone_eq_drc = NULL;
            }
        }
        // tone 中的 decoder 和 mixer 数据流会在内部删除，上层不必要处理
        break;
    }
}
```




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | test_tone_dec = tone_dec_create();<br>static char *single_file[2] = {NULL};<br>single_file[0] = (char *)TONE_POWER_OFF;<br>single_file[1] = NULL;<br>struct tone_dec_list_handle *dec_list = tone_dec_list_create(test_tone_dec, single_file, 1, NULL, NULL, tone_test_stream_handler, NULL);<br>tone_dec_list_add_play(test_tone_dec, dec_list); |
| 关联模块 | |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int tone_dec_list_add_play(struct tone_dec_handle *dec, struct tone_dec_list_handle *dec_list) |
| 功能描述 | 添加 dec_list 播放 |
| 参数说明 | * \param[in][out] *dec // tone 句柄<br>* \param[in] *dec_list // dec_list 句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval false. 失败<br>* \retval true 成功 |
| 例子 | |
| 关联模块 | |
| 补充说明 | tone_dec_list_create()之后调用 |

| | |
|---|---|
| 函数原型 | void tone_dec_stop(struct tone_dec_handle **ppdec,<br>u8 push_event,<br>u8 end_flag) |
| 功能描述 | 停止所有 tone 解码 |
| 参数说明 | * \param[in][out] **ppdec // tone 句柄<br>* \param[in] push_event // 解码是否返回事件<br>* \param[in] end_flag // 返回事件中带有的标记 |
| 输出 | * \return 无 |
| 例子 | tone_dec_stop(&test_tone, 1, TONE_DEC_STOP_BY_OTHER_PLAY); |
| 关联模块 | |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | void tone_dec_stop_spec_file(struct tone_dec_handle **ppdec,<br>char *file_name,<br>u8 push_event,<br>u8 end_flag); |
| 功能描述 | 停止指定文件解码 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 参数说明 | * \param[in][out] **ppdec // tone 句柄<br>* \param[in] *file_name // 文件名<br>* \param[in] push_event // 解码是否返回事件<br>* \param[in] end_flag // 返回事件中带有的标记 |
| 输出 | * \return 无 |
| 例子 | tone_dec_stop(&test_tone, (char *)TONE_POWER_OFF, 0, 0); |
| 关联模块 | |
| 补充说明 | 仅匹配 dec_list 中的第一个 file |

| | |
|---|---|
| 函数原型 | int tone_play_with_callback_by_name(char *name,<br>u8 preemption,<br>void (*evt_handler)(void *priv, int flag),<br>void *evt_priv); |
| 功能描述 | 按名字播放提示音 |
| 参数说明 | * \param[in] *name // 带有路径的文件名<br>* \param[in] preemption // 1-抢占方式播放；0-叠加方式播放<br>* \param[in] *evt_handler // 播放结束事件回调，flag0 正常结束，1 被打断<br>* \param[in] *evt_priv // 播放结束事件回调私有句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | #define MY_TONE_FLAG 0x12120000<br>U32 my_tone_cur_flag = MY_TONE_FLAG;<br>Static void my_tone_evt_handler(void *priv, int flag)<br>{<br>If (flag == 0) {// 正常结束<br>If ((u32)priv == my_tone_cur_flag) { // 播放结束<br>}<br>} else { // 被打断<br>}<br>}<br>my_tone_cur_flag = MY_TONE_FLAG + rand32() %0xffff;<br>tone_play_with_callback_by_name(TONE_POWER_OFF, 1, my_tone_evt_handler, (void*)my_tone_cur_flag); |
| 关联模块 | |
| 补充说明 | 该函数在 tone_player.c 中，共用*tone_dec 句柄，调用时会关闭当前正在播放的 |

| | |
|---|---|
| 函数原型 | int tone_play_with_callback_by_list(const char **list,<br>u8 preemption, |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| | void (*evt_handler)(void *priv, int flag),<br>void *evt_priv); |
| 功能描述 | 按名字列表播放提示音 |
| 参数说明 | * \param[in] *name // 带有路径的文件名列表<br>* \param[in] preemption // 1-抢占方式播放；0-叠加方式播放<br>* \param[in] *evt_handler // 播放结束事件回调，flag0 正常结束，1 被打断<br>* \param[in] *evt_priv // 播放结束事件回调私有句柄 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | Static char file_list[n] = {NULL};<br>U8 cnt = 0;<br>file_list[cnt++] = FILE_NAME0;<br>file_list[cnt++] = FILE_NAME1;<br>If (循环播放) {<br>U8 loop_num = 2; // 循环次数<br>file_list[cnt++] = (U32)TONE_REPEAT_BEGIN(loop_num);<br>file_list[cnt++] = LOOP_FILE_NAME0;<br>file_list[cnt++] = LOOP_FILE_NAME1;<br>file_list[cnt++] = (U32)TONE_REPEAT_END();<br>}<br>file_list[cnt++] = NULL;<br>tone_play_with_callback_by_name(TONE_POWER_OFF, 1, NULL, NULL) |
| 关联模块 | |
| 补充说明 | 该函数在 tone_player.c 中，共用*tone_dec 句柄，调用时会关闭当前正在播放的 |

| | |
|---|---|
| 函数原型 | int tone_get_status() |
| 功能描述 | 获取提示音播放状态 |
| 参数说明 | * \param[in][out] 无 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval TONE_START 正在播放（可能已经解码结束了的）<br>* \retval TONE_STOP 播放结束 |
| 例子 | |
| 关联模块 | |
| 补充说明 | 该函数在 tone_player.c 中，共用*tone_dec 句柄。可能已经播放完了，但状态还未改变（decoder 返回消息给上层任务，可能上层任务还未处理） |

| | |
|---|---|
| 函数原型 | int tone_get_dec_status() |
| 功能描述 | 获取提示音播放解码状态 |
| 参数说明 | * \param[in][out] 无 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval TONE_START 正在解码<br>* \retval TONE_STOP 解码结束 |
| 例子 | |
| 关联模块 | |
| 补充说明 | 该函数在 tone_player.c 中，共用*tone_dec 句柄。 |

| | |
|---|---|
| 函数原型 | int tone_play_stop(void) |
| 功能描述 | 停止播放 |
| 参数说明 | * \param[in][out] 无 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | tone_play_stop(); |
| 关联模块 | |
| 补充说明 | 该函数在 tone_player.c 中，共用*tone_dec 句柄 |

| | |
|---|---|
| 函数原型 | int tone_play_stop_by_path(char *path) |
| 功能描述 | 停止指定文件播放 |
| 参数说明 | * \param[in] *path // 带有路径的文件名 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval 非 0. 失败<br>* \retval 0 成功 |
| 例子 | tone_play_stop_by_path(TONE_POWER_OFF); |
| 关联模块 | |
| 补充说明 | 该函数在 tone_player.c 中，共用*tone_dec 句柄 |

| | |
|---|---|
| 函数原型 | int tone_dec_wait_stop(u32 timeout_ms) |
| 功能描述 | 超时等待解码结束 |
| 参数说明 | * \param[in] timeout_ms // 超时时长 |
| 输出 | * \return 返回相应的操作消息处理值<br>* \retval TONE_START 正在解码<br>* \retval TONE_STOP 解码结束 |
| 例子 | |
| 关联模块 | |
| 补充说明 | 该函数在 tone_player.c 中，共用*tone_dec 句柄 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

## 11.3. 音量管理接口

### 11.3.1. 功能介绍

**模式一 提示音跟随当前音量(提示音大小会跟随系统音量改变)**

在此模式下，系统音量主要调节 dac 的数字增益和模拟增益。提示音播放有两种情况，一种是提示音播放时已经有解码声音输出，这时候播放提示音不再重新设置音量，而是以当前解码音量为准；一种是提示播放时没有其他解码声音输出，这时候播放提示音会设置为默认提示音音量。优点是系统信噪比会比较好，缺点是提示音音量做不到固定效果。

**模式二 提示音固定音量(提示音大小保持一直不变)**

在此模式下，系统音量调节为固定 dac 的数字增益和模拟增益，调节每路解码的数据音量。提示音播放的时候，可以选择自动将除提示音外其他解码通道声音减小或者静音。优点是音量效果比较好，可以固定提示音效果，缺点是模拟音量需要固定，在播放音量低时候信噪比会比较差。

### 11.3.2. 接口模块

#### 11.3.2.1. 初始化与关闭

| | |
|---|---|
| 函数原型 | int app_audio_volume_init(void *param) |
| 功能描述 | 初始化音量管理的配置参数 |
| 参数说明 | * \param[in] param // 配置参数句柄 |
| 输出 | * \return 返回操作结果<br>* \retval 0 操作完成<br>* \retval -1 操作出错 |
| 例子 | app_audio_volume_init(vol_param); |
| 关联模块 | app_audio.c |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int app_audio_volume_uninit(void) |
| 功能描述 | 释放音量管理的资源 |
| 参数说明 | |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 输出 | * \return 返回操作结果<br>* \retval 0 操作完成<br>* \retval -1 操作出错 |
| 例子 | app_audio_volume_uninit(); |
| 关联模块 | app_audio.c |
| 补充说明 | |

#### 11.3.2.2. 模式切换

系统音量一共分 3 个模式的音量，分别是音乐音量、通话音量和提示音音量。都可以独立设置，在切换到不同模式时会自动切换到对应的音量等级。

| | |
|---|---|
| 函数原型 | int app_audio_volume_state_switch(u8 state, u16 max_volume) |
| 功能描述 | 切换当前模式 |
| 参数说明 | * \param[in] state // 需要切换到的模式<br>* \param[in] max_volume // 模式限制最大音量 |
| 输出 | * \return 返回操作结果<br>* \retval 0 操作完成<br>* \retval -1 操作出错 |
| 例子 | app_audio_volume_state_switch(APP_AUDIO_STATE_MUSIC, 30); |
| 关联模块 | app_audio.c |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int app_audio_volume_state_exit(u8 state) |
| 功能描述 | 退出当前模式 |
| 参数说明 | * \param[in] state // 当前模式 |
| 输出 | * \return 返回操作结果<br>* \retval 0 操作完成<br>* \retval -1 操作出错 |
| 例子 | app_audio_volume_state_exit(APP_AUDIO_STATE_MUSIC); |
| 关联模块 | app_audio.c |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int app_audio_volume_state_get(void) |
| 功能描述 | 获取当前模式 |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| 参数说明 | |
|---|---|
| 输出 | * \return 返回当前模式<br>* \retval 当前模式<br>* \retval -1 操作出错 |
| 例子 | u8 cur_state = app_audio_volume_state_get(void); |
| 关联模块 | app_audio.c |
| 补充说明 | |

#### 11.3.2.3. 音量调节

| 函数原型 | int app_audio_volume_set(u8 state, u16 volume, u8 fade) |
|---|---|
| 功能描述 | 调节某个模式的音量，如果调节的是当前模式则立即生效，如果是其他模式则切换到相应模式才生效 |
| 参数说明 | * \param[in] state // 模式<br>* \param[in] volume // 音量等级<br>* \param[in] fade // 是否淡入淡出 |
| 输出 | * \return 返回操作结果<br>* \retval 0 操作完成<br>* \retval -1 操作出错 |
| 例子 | app_audio_volume_set(APP_AUDIO_STATE_MUSIC, 30, 1); |
| 关联模块 | app_audio.c |
| 补充说明 | |

| 函数原型 | int app_audio_volume_get(u8 state) |
|---|---|
| 功能描述 | 获取某个模式的音量等级 |
| 参数说明 | * \param[in] state // 模式 |
| 输出 | * \return 返回音量等级<br>* \retval 音量等级 |
| 例子 | int music_vol = app_audio_volume_get(APP_AUDIO_STATE_MUSIC); |
| 关联模块 | app_audio.c |
| 补充说明 | |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com

| | |
|---|---|
| 函数原型 | int app_audio_volume_mute(u8 mute) |
| 功能描述 | 音量静音或者恢复静音前状态 |
| 参数说明 | * \param[in] mute// 1:静音 0:恢复 |
| 输出 | * \return 返回操作结果<br>* \retval 0 操作完成<br>* \retval -1 操作出错 |
| 例子 | app_audio_volume_mute(1); |
| 关联模块 | app_audio.c |
| 补充说明 | |

| | |
|---|---|
| 函数原型 | int app_audio_volume_updown(u8 state, int updown) |
| 功能描述 | 调节某个模式的音量加一定等级，如果 updown 为正数则为增加，负数则减少 |
| 参数说明 | * \param[in] state // 模式<br>* \param[in] updown// 增加音量等级 |
| 输出 | * \return 返回操作结果<br>* \retval 0 操作完成<br>* \retval -1 操作出错 |
| 例子 | app_audio_volume_updown(APP_AUDIO_STATE_MUSIC, -4); |
| 关联模块 | app_audio.c |
| 补充说明 | |




地址：珠海市吉大石花西路 107 号 9 栋综合楼 邮编：519015
电话：0756-6313088 传真：0756-6313081
网站：www.zh-jieli.com