



# AX AUDIO API 文档

文档版本: V1.3

发布日期: 2022/03/11

AXERA CONFIDENTIAL FOR Sipeed

前 言 .....	6
<b>目 录</b>	
修订历史 .....	7
1 概述 .....	8
1.1 概述 .....	9
1.2 功能描述 .....	9
1.2.1 音频输入和音频输出 .....	9
1.2.2 声音质量增强 .....	9
1.2.3 音频编码和解码 .....	11
1.2.4 设备节点的对应关系 .....	12
1.2.5 软件流程 .....	12
2 API 参考 .....	19
2.1 Tinyalsa 音频输入输出 .....	20
pcm_params_get .....	22
pcm_params_free .....	24
pcm_params_get_mask .....	25
pcm_params_get_max .....	27
pcm_open .....	29
pcm_open_by_name .....	31
pcm_close .....	33
pcm_is_ready .....	34
pcm_get_channels .....	35
pcm_get_config .....	36
pcm_get_rate .....	37

pcm_get_format .....	38
pcm_get_file_descriptor .....	39
pcm_get_error .....	40
pcm_set_config .....	41
pcm_format_to_bits .....	42
pcm_get_buffer_size .....	43
pcm_frames_to_bytes .....	44
pcm_bytes_to_frames .....	45
pcm_writel .....	46
pcm_readl .....	48
2.2 声音质量增强 .....	50
AX_AUDIO_PROCESS_Init .....	50
AX_AUDIO_PROCESS_DelInit .....	51
AX_AUDIO_PROCESS_Proc .....	52
AX_AUDIO_InterleavedToNoninterleaved16 .....	53
AX_AUDIO_MonoToStereo16 .....	54
2.3 音频编码 .....	55
AX_AENC_Init .....	56
AX_AENC_DelInit .....	57
AX_AENC_CreateChn .....	58
AX_AENC_DestroyChn .....	60
AX_AENC_SendFrame .....	61
AX_AENC_GetStream .....	62
AX_AENC_ReleaseStream .....	64
AX_AENC_RegisterEncoder .....	66
AX_AENC_UnRegisterEncoder .....	68

2.4 音频解码.....	69
AX_ADEC_Init .....	70
AX_ADEC_DelInit.....	71
AX_ADEC_CreateChn .....	72
AX_ADEC_DestroyChn.....	74
AX_ADEC_SendStream .....	75
AX_ADEC_ClearChnBuf .....	77
AX_ADEC_GetFrame .....	78
AX_ADEC_ReleaseFrame.....	80
AX_ADEC_SendEndOfStream.....	82
AX_ADEC_RegisterDecoder .....	83
AX_ADEC_UnRegisterDecoder.....	85
<b>3 数据结构 .....</b>	<b>86</b>
3.1 Tinyalsa 音频输入输出 .....	87
pcm_format .....	87
pcm_mask .....	90
pcm_config .....	91
pcm_param.....	93
3.2 声音质量增强.....	96
AEC_MODE_E.....	96
SUPPRESSION_LEVEL_E .....	96
AEC_FLOAT_CONFIG_S.....	97
ROUTING_MODE_E .....	98
AEC_FIXED_CONFIG_S.....	99
AEC_CONFIG_S.....	100
AGGRESSIVENESS_LEVEL_E .....	101

NS_CONFIG_S .....	101
AGC_MODE_E .....	102
AGC_CONFIG_S .....	103
AUDIO_PROCESS_ATTR_S .....	104
3.3 音频编码 .....	106
AX_AENC_CHN_ATTR_S .....	106
AX_AUDIO_BIT_WIDTH_E .....	108
AX_AUDIO_SOUND_MODE_E .....	109
AX_AUDIO_FRAME_S .....	110
AX_AENC_ENCODER_S .....	112
AX_AUDIO_STREAM_S .....	114
3.4 音频解码 .....	116
AX_ADEC_MODE_E .....	116
AX_ADEC_CHN_ATTR_S .....	118
AX_ADEC_DECODER_S .....	119
<b>4 错误码 .....</b>	<b>121</b>
4.1 音频编码错误码 .....	122
4.2 音频解码错误码 .....	122
<b>5 调试信息 .....</b>	<b>124</b>

## 权利声明

爱芯元智半导体(上海)有限公司或其许可人保留一切权利。

非经权利人书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

## 注意

您购买的产品、服务或特性等应受商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非商业合同另有约定，本公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

AXERA CONFIDENTIAL FOR SPEED

## 适用产品



AX620A

## 前言

## 适读人群

- 软件开发工程师
- 技术支持工程师

## 符号与格式定义

符号/格式	说明
<code>xxx</code>	表示您可以执行的命令行。
<i>斜体</i>	表示变量。如，“安装目录/AX620A_SDK_Vx.x.x/build 目录”中的“安装目录”是一个变量，由您的实际环境决定。
 说明/备注：	表示您在使用产品的过程中，我们向您说明的事项。
 注意：	表示您在使用产品的过程中，需要您特别注意的事项。

文档版本	发布时间	修订说明
V1.0	2021/08/25	文档初版
V1.1	2021/10/15	更新错误码
V1.2	2022/03/04	增加初始化、反初始化 API
V1.3	2022/03/11	增加声音质量增强（AEC、NS、AGC）

### 修订历史

AXERA CONFIDENTIAL FOR Speed



---

本章节包含：

[1.1 概述](#)

[1.2 功能描述](#)

## 1 概述

AXERA CONFIDENTIAL FOR Sipeed

## 1.1 概述

音频（AUDIO）模块，包括音频输入、音频输出、声音质量增强、音频编码和音频解码五个子模块。音频输入和输出模块通过第三方开源软件库 Tinyalsa 对 AX620A 芯片音频接口的控制，从而实现音频输入和输出功能，音频编码和解码模块提供对 G711、AAC 格式的音频编解码功能。

## 1.2 功能描述

### 1.2.1 音频输入和音频输出

#### ➤ 音频输入输出接口

音频输入输出接口使用第三方开源软件库 Tinyalsa 和 Audio Codec 进行对接，完成声音的录制和播放。

对每个输入输出接口，软件根据该接口支持的功能，分为音频输入接口和音频输出接口。例如：pcmC0D0c 只支持音频输入，pcmC0D0p 只支持音频输出。

### 1.2.2 声音质量增强

#### ➤ AEC

AEC 为回声消除（Acoustic Echo Cancellation）模块，主要工作在需要进行去除回声的场景下：如 IPC 对讲，远端语音在设备上播放，此时在本地通过 MIC 采集语音数据，它支持消除录制的语音数据中的设备播放的声音（回声）。

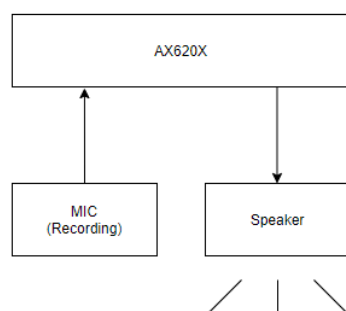


图1-1 回声消除示意图

与其他功能模块只需要 Sin 数据不同，AEC 模块需要 Sin（Signal In）和 Rin（Reference In）两路数据进行算法处理，最终得到处理后的 sou（Signal Out）数据。其中，Sin 为加入了回声的近端输入，Rin 为参考帧（回声）数据。成功启用回声消除需要具备一定的条件：单声道模式，工作采样率 8kHz、16kHz，且 MIC 采集语音数据的帧长和远程语音播放的帧长必须相同。

#### ➤ NS

NS 为噪声抑制（Noise Suppression）模块，主要工作在需要去除外接噪声，保留语音输入的场景下。

NS 会去除一些环境声音，主要保留语音数据，并会带来一定的细节丢失。所以 NS 算法更适用于 NVR 和 IPC 场景。在这两个场景下，我们更希望能够着重保留人声，滤除其他噪声。

工作条件：

- 单声道模式
- 工作采样率 8kHz（帧长 80）或者工作采样率 16kHz（帧长 160）

#### ➤ AGC

AGC 为自动增益控制（Auto Gain Control）模块，主要负责增益控制输出电平，在声音输入音量有大小变化时，能将输出音量控制在比较一致的范围内，主要工作在需要保证声音不至于过大或过小的场景下。

AGC 更多起到的作用是放大输入源的声音，以保证音源过小时，经过算法处理后的声音依然很大。

工作条件：

- 单声道模式
- 工作采样率 8kHz（帧长 80）或者工作采样率 16kHz（帧长 160）

1.2.3 音频编码和解码

➤ 音频编解码流程

AX620A SDK 音频的编解码类型 G711、AAC 使用 CPU 软件编解码，核心编解码器工作在用户态。

➤ 音频编解码协议

AX620A 支持的音频编解码协议如下表所示：

表1-1 音频编码协议

协议	采样率	帧长（采样点）	码率 （kbps）	压缩率	CPU 消耗	描述
G711	8kHz	80/160/240/320/480	64	2	1MHz	优点：语音质量最好；CPU 消耗小；支持广泛，协议免费 缺点：压缩效率低
AAC	48kHz	1024	128	12	2MHz	优点：语音质量好；压缩效率高；支持广泛 缺点：CPU 消耗

						较高
--	--	--	--	--	--	----

1.2.4 设备节点的对应关系

➤ AX620A ALSA 设备节点的对应关系如下图所示：

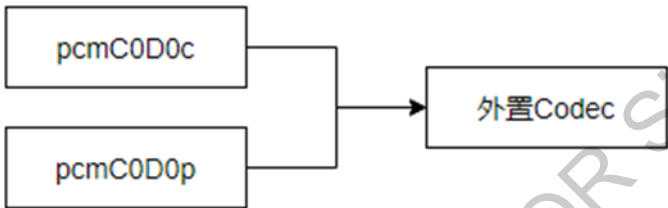


图1-2 ALSA 设备节点的对应关系

1.2.5 软件流程

➤ 录音流程包括：配置声卡参数，打开 PCM 设备，读入音频数据，关闭 PCM 设备。如下图所示：

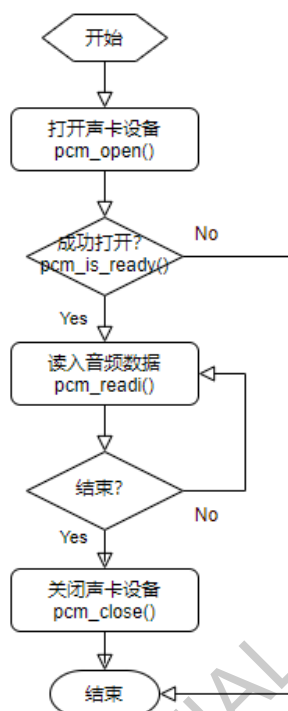


图1-3 录音流程图

- 播放流程包括：配置声卡参数，打开 PCM 设备，写入音频数据，关闭 PCM 设备。如下图所示：

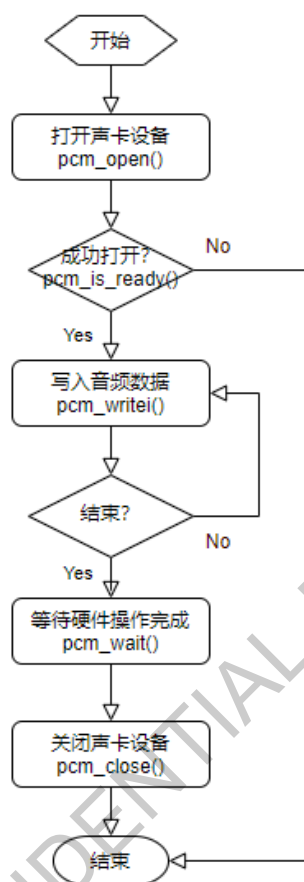


图1-4 播放流程图

- 编码流程包括：配置声卡参数，打开 PCM 设备，初始化编码器，创建音频编码通道，读入音频数据，发送音频编码帧，获取编码后码流，释放编码后码流，销毁音频编码通道，反初始化编码器，关闭 PCM 设备。如下图所示：

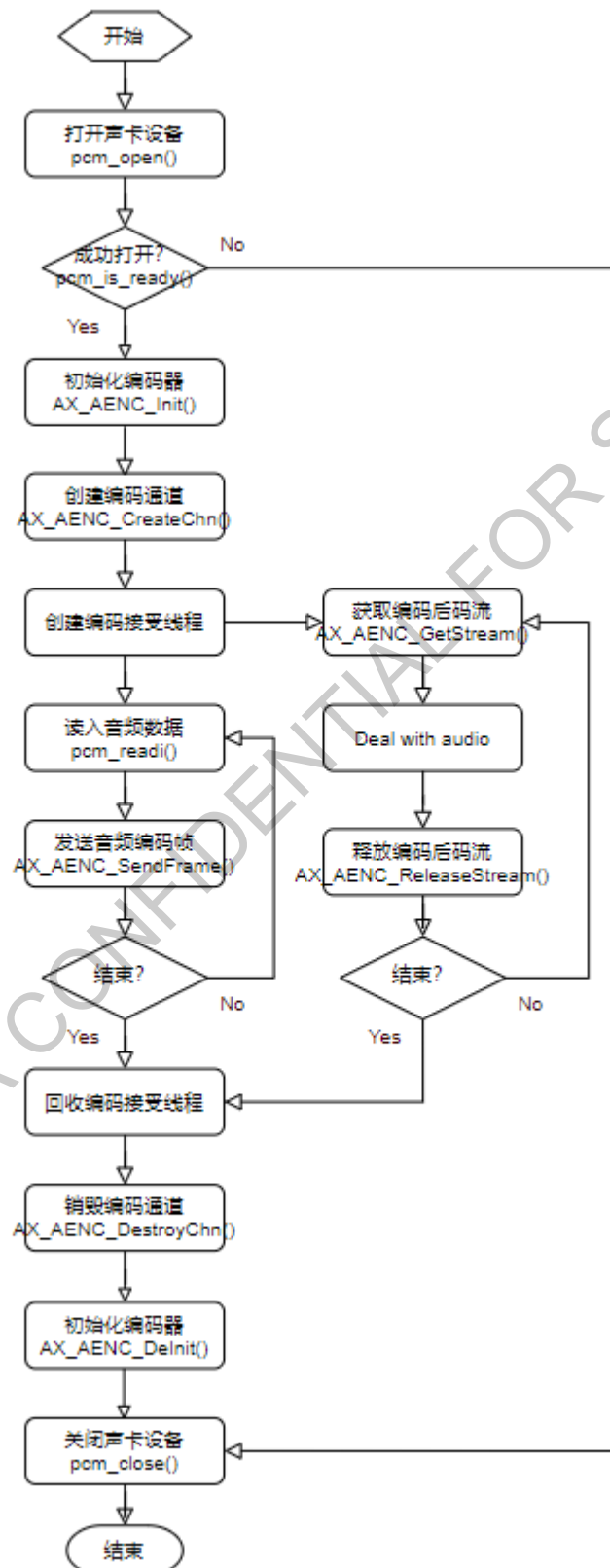




图1-5 编码流程图

- 解码流程包括：配置声卡参数，打开 PCM 设备，初始化解码器，创建音频解码通道，读入音频码流，向解码通道发送码流，获取音频解码帧数据，释放音频解码帧数据，发送码流结束标识符，销毁音频解码通道，反初始化解码器，关闭 PCM 设备。如下图所示：

AXERA CONFIDENTIAL FOR Sipeed

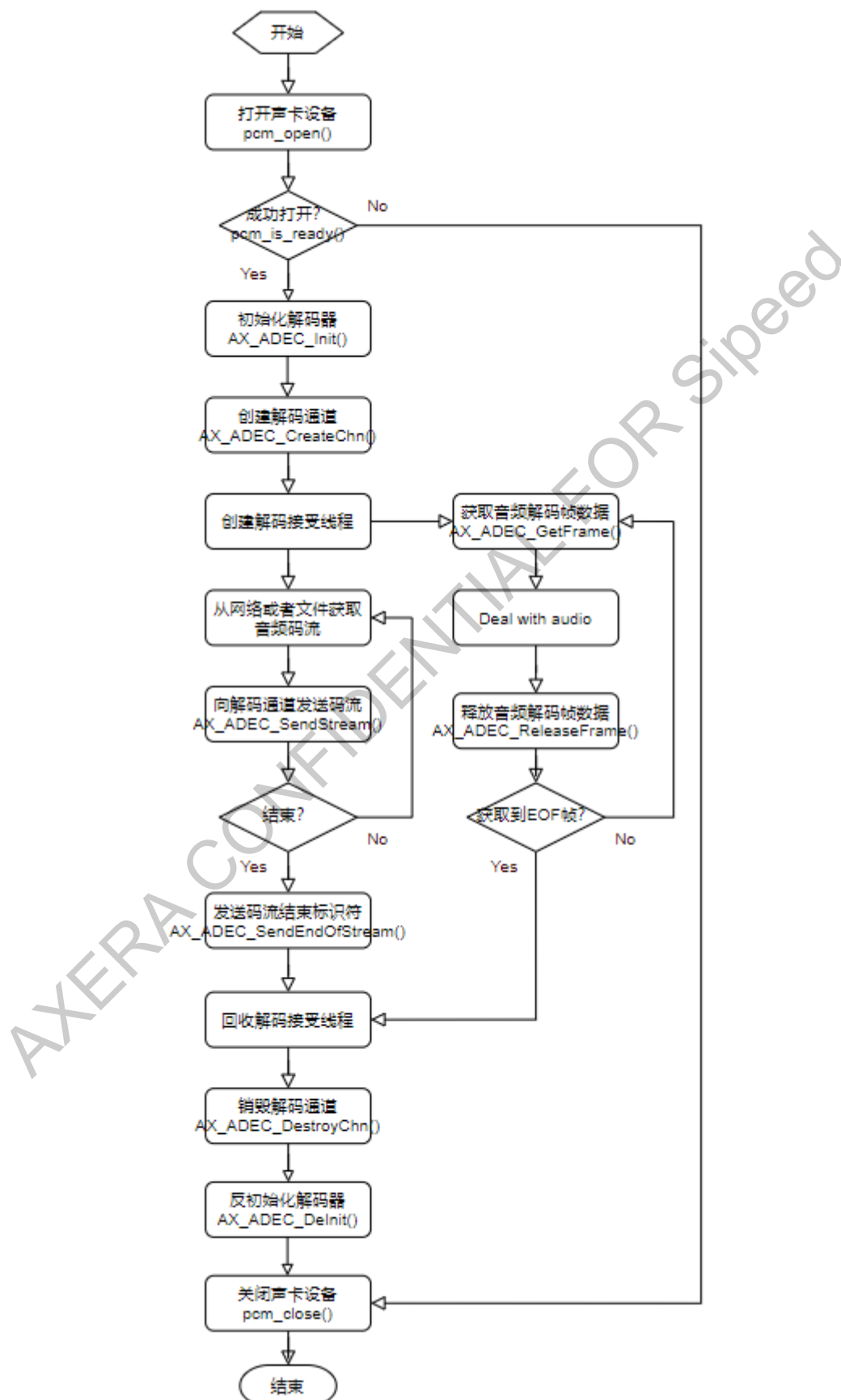


图1-6 解码流程图

AXERA CONFIDENTIAL FOR Sipeed

本章节包含：

## 2 API 参考

### 2.1 Tinyalsa 音频输入输出

### 2.2 声音质量增强

- AX\_AUDIO\_PROCESS\_Init：初始化声音增强模块。
- AX\_AUDIO\_PROCESS\_DeInit：反初始化声音增强模块。
- AX\_AUDIO\_PROCESS\_Proc：声音增强音频处理。
- AX\_AUDIO\_InterleavedToNoninterleaved16：交错转非交错（16bit）。
- AX\_AUDIO\_MonoToStereo16：单声道转立体声（16bit）。

## AX\_AUDIO\_PROCESS\_Init

### 【描述】

初始化声音增强模块。

### 【语法】

```
AX_S32 AX_AUDIO_PROCESS_Init(const AUDIO_PROCESS_ATTR_S *pstAudioProcessAttr)
```

### 【参数】

参数名称	描述	输入/输出
pstAudioProcessAttr	声音增强属性指针	输入

### 【返回值】

返回值	描述
0	成功

非 0	失败
-----	----

**【需求】**

- 头文件：ax\_audio\_process.h
- 库文件：libax\_audio\_3a.so

**【注意】**

无

**【举例】**

无

**【相关主题】**

无

## AX\_AUDIO\_PROCESS\_DeInit

**【描述】**

反初始化声音增强模块。

**【语法】**

```
AX_S32 AX_AUDIO_PROCESS_DeInit()
```

**【参数】**

无

**【返回值】**

返回值	描述
0	成功
非 0	失败

**【需求】**

➤ 头文件: ax\_audio\_process.h

➤ 库文件: libax\_audio\_3a.so

#### 【注意】

无

#### 【举例】

无

#### 【相关主题】

无

## AX\_AUDIO\_PROCESS\_Proc

#### 【描述】

声音增强音频处理。

#### 【语法】

```
AX_S32 AX_AUDIO_PROCESS_Proc(AX_VOID *in_data, AX_VOID *ref_data, AX_VOID *out_data)
```

#### 【参数】

参数名称	描述	输入/输出
in_data	输入音频	输入
ref_data	参考音频	输入
out_data	输出音频	输出

#### 【返回值】

返回值	描述
0	成功
非 0	失败

**【需求】**

- 头文件：ax\_audio\_process.h
- 库文件：libax\_audio\_3a.so

**【注意】**

无

**【举例】**

无

**【相关主题】**

无

## AX\_AUDIO\_InterleavedToNoninterleaved16

**【描述】**

交错转非交错（16bit）。

**【语法】**

```
AX_VOID AX_AUDIO_InterleavedToNoninterleaved16(AX_S16 *interleaved, AX_U32  
frameCount, AX_S16 *left, AX_S16 *right)
```

**【参数】**

参数名称	描述	输入/输出
interleaved	交错格式音频	输入
frameCount	帧数	输入
left	非交错格式左声道	输出
right	非交错格式右声道	输出

**【返回值】**

无

**【需求】**

- 头文件：ax\_audio\_process.h
- 库文件：libax\_audio\_3a.so

**【注意】**

无

**【举例】**

无

**【相关主题】**

无

## AX\_AUDIO\_MonoToStereo16

**【描述】**

单声道转立体声（16bit）。

**【语法】**

```
AX_VOID AX_AUDIO_MonoToStereo16(AX_S16 *mono, AX_U32 frameCount, AX_S16
*stereo)
```

**【参数】**

参数名称	描述	输入/输出
mono	单声道音频	输入
frameCount	帧数	输入
stereo	立体声音频	输出

**【返回值】**

无

**【需求】**



➤ 头文件: ax\_audio\_process.h

➤ 库文件: libax\_audio\_3a.so

**【注意】**

无

**【举例】**

无

**【相关主题】**

无

音频编码

[2.4 音频解码](#)

## 2.1 Tinyalsa 音频输入输出

- `pcm_params_get`: 获取 PCM 设备硬件参数。
- `pcm_params_free`: 释放 PCM 设备硬件参数结构体。
- `pcm_params_get_mask`: 获取 PCM 设备硬件参数掩码。
- `pcm_params_get_min`: 获取 PCM 参数最小值。
- `pcm_params_get_max`: 获取 PCM 参数最大值。
- `pcm_open`: 打开 PCM 设备。
- `pcm_open_by_name`: 打开特定 PCM 设备。
- `pcm_close`: 关闭 PCM 设备。
- `pcm_is_ready`: 检查 PCM 设备是否成功打开。
- `pcm_get_channels`: 获取通道数量。
- `pcm_get_config`: 获取 PCM 设备配置。
- `pcm_get_rate`: 获取 PCM 设备采样率。
- `pcm_get_format`: 获取 PCM 设备音频格式。
- `pcm_get_file_descriptor`: 获取 PCM 设备文件描述符。
- `pcm_get_error`: 返回上一次出错信息。
- `pcm_set_config`: 设置 PCM 设备配置。
- `pcm_format_to_bits`: 返回音频格式包含比特数。
- `pcm_get_buffer_size`: 获取 PCM 设备缓存大小。
- `pcm_frames_to_bytes`: 返回音频帧包含比特数。
- `pcm_bytes_to_frames`: 返回比特数包含音频帧数。
- `pcm_writew`: 向 PCM 设备写入音频数据。

- `pcm_readi`: 从 PCM 设备读取音频数据。

AXERA CONFIDENTIAL FOR Sipeed

## pcm\_params\_get

### 【描述】

获取 PCM 设备硬件参数。

### 【语法】

```
struct pcm_params *pcm_params_get(unsigned int card, unsigned int device,  
                                   unsigned int flags)
```

### 【参数】

参数名称	描述	输入/输出
card	声卡号	输入
device	声卡设备号	输入
flags	PCM 设备标志	输入

### 【返回值】

返回值	描述
0	失败
非 0	成功

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

AXERA CONFIDENTIAL FOR Sipeed

## pcm\_params\_free

### 【描述】

释放 PCM 设备硬件参数结构体。

### 【语法】

```
void pcm_params_free(struct pcm_params *pcm_params)
```

### 【参数】

参数名称	描述	输入/输出
pcm_params	PCM 设备硬件参数结构体指针	输入

### 【返回值】

无

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## pcm\_params\_get\_mask

### 【描述】

获取 PCM 设备硬件参数掩码。

### 【语法】

```
const struct pcm_mask *pcm_params_get_mask(const struct pcm_params
*pcm_params, enum pcm_param param)
```

### 【参数】

参数名称	描述	输入/输出
pcm_params	PCM 设备硬件参数结构体指针	输入
param	参数枚举	输入

### 【返回值】

返回值	描述
0	失败
非 0	成功

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

AXERA CONFIDENTIAL FOR Sipeed



## pcm\_params\_get\_max

### 【描述】

获取 PCM 参数最大值。

### 【语法】

```
unsigned int pcm_params_get_max(const struct pcm_params *pcm_params, enum  
pcm_param param)
```

### 【参数】

参数名称	描述	输入/输出
pcm_params	PCM 设备硬件参数结构体指针	输入
param	参数枚举	输入

### 【返回值】

返回值	描述
0	失败
非 0	成功，其值为最大值

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

AXERA CONFIDENTIAL FOR Sipeed

## pcm\_open

### 【描述】

打开 PCM 设备。

### 【语法】

```
struct pcm *pcm_open(unsigned int card, unsigned int device, unsigned int flags, const struct pcm_config *config)
```

### 【参数】

参数名称	描述	输入/输出
card	声卡号	输入
device	逻辑设备号	输入
flags	PCM 设备标志	输入
config	参数结构体	输入

### 【返回值】

返回值	描述
0	失败
非 0	成功

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

**【相关主题】**

无

AXERA CONFIDENTIAL FOR Sipeed

## pcm\_open\_by\_name

### 【描述】

打开特定 PCM 设备。

### 【语法】

```
struct pcm *pcm_open_by_name(const char *name, unsigned int flags, const
struct pcm_config *config)
```

### 【参数】

参数名称	描述	输入/输出
name	设备名	输入
flags	设备标志	输入
config	参数结构体	输入

### 【返回值】

返回值	描述
0	失败
非 0	成功，其值为最小值

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

AXERA CONFIDENTIAL FOR Sipeed

## pcm\_close

### 【描述】

关闭 PCM 设备。

### 【语法】

```
int pcm_close(struct pcm *pcm)
```

### 【参数】

参数名称	描述	输入/输出
pcm	pcm 句柄	输入

### 【返回值】

返回值	描述
0	成功

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## pcm\_is\_ready

### 【描述】

检查 PCM 设备是否成功打开。

### 【语法】

```
int pcm_is_ready(const struct pcm *pcm)
```

### 【参数】

参数名称	描述	输入/输出
pcm	pcm 句柄	输入

### 【返回值】

返回值	描述
0	失败
非 0	成功

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无



## pcm\_get\_channels

### 【描述】

获取通道数量。

### 【语法】

```
unsigned int pcm_get_channels(const struct pcm *pcm)
```

### 【参数】

参数名称	描述	输入/输出
pcm	pcm 句柄	输入

### 【返回值】

返回值	描述
非 0	通道数

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## pcm\_get\_config

### 【描述】

获取 PCM 设备配置。

### 【语法】

```
const struct pcm_config * pcm_get_config(const struct pcm *pcm)
```

### 【参数】

参数名称	描述	输入/输出
pcm	pcm 句柄	输入

### 【返回值】

返回值	描述
0	失败
非 0	成功

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## pcm\_get\_rate

### 【描述】

获取 PCM 设备采样率。

### 【语法】

```
unsigned int pcm_get_rate(const struct pcm *pcm)
```

### 【参数】

参数名称	描述	输入/输出
pcm	pcm 句柄	输入

### 【返回值】

返回值	描述
非 0	采样率

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## pcm\_get\_format

### 【描述】

获取 PCM 设备音频格式。

### 【语法】

```
enum pcm_format pcm_get_format(const struct pcm *pcm)
```

### 【参数】

参数名称	描述	输入/输出
pcm	pcm 句柄	输入

### 【返回值】

返回值	描述
enum pcm_format	音频格式

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## pcm\_get\_file\_descriptor

### 【描述】

获取 PCM 设备文件描述符。

### 【语法】

```
int pcm_get_file_descriptor(const struct pcm *pcm)
```

### 【参数】

参数名称	描述	输入/输出
pcm	pcm 句柄	输入

### 【返回值】

返回值	描述
int	文件描述符

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## pcm\_get\_error

### 【描述】

返回上一次出错信息。

### 【语法】

```
const char *pcm_get_error(const struct pcm *pcm)
```

### 【参数】

参数名称	描述	输入/输出
pcm	pcm 句柄	输出

### 【返回值】

返回值	描述
const char *	错误描述

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## pcm\_set\_config

### 【描述】

设置 PCM 设备配置。

### 【语法】

```
int pcm_set_config(struct pcm *pcm, const struct pcm_config *config)
```

### 【参数】

参数名称	描述	输入/输出
pcm	pcm 句柄	输入
config	配置结构体	输入

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## pcm\_format\_to\_bits

### 【描述】

返回音频格式占用比特数。

### 【语法】

```
unsigned int pcm_format_to_bits(enum pcm_format format)
```

### 【参数】

参数名称	描述	输入/输出
format	音频格式	输入

### 【返回值】

返回值	描述
unsigned int	比特数

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无



## pcm\_get\_buffer\_size

### 【描述】

获取 PCM 设备缓存大小。

### 【语法】

```
unsigned int pcm_get_buffer_size(const struct pcm *pcm)
```

### 【参数】

参数名称	描述	输入/输出
pcm	pcm 句柄	输入

### 【返回值】

返回值	描述
unsigned int	缓存

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## pcm\_frames\_to\_bytes

### 【描述】

返回音频帧包含比特数。

### 【语法】

```
unsigned int pcm_frames_to_bytes(const struct pcm *pcm, unsigned int frames)
```

### 【参数】

参数名称	描述	输入/输出
pcm	pcm 句柄	输入
frames	音频帧数	输入

### 【返回值】

返回值	描述
unsigned int	比特数

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## pcm\_bytes\_to\_frames

### 【描述】

返回比特数包含音频帧数。

### 【语法】

```
unsigned int pcm_bytes_to_frames(const struct pcm *pcm, unsigned int bytes)
```

### 【参数】

参数名称	描述	输入/输出
pcm	pcm 句柄	输入
bytes	字节数	输入

### 【返回值】

返回值	描述
unsigned int	音频帧数

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## pcm\_writei

### 【描述】

向 PCM 设备写入音频数据。

### 【语法】

```
int pcm_writei(struct pcm *pcm, const void *data, unsigned int frame_count)
```

### 【参数】

参数名称	描述	输入/输出
pcm	pcm 句柄	输入
data	音频数据指针	输入
frame_count	帧数	输入

### 【返回值】

返回值	描述
正值	成功
负值	失败

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

AXERA CONFIDENTIAL FOR Sipeed

## pcm\_readi

### 【描述】

从 PCM 设备读取音频数据。

### 【语法】

```
int pcm_readi(struct pcm *pcm, void *data, unsigned int frame_count)
```

### 【参数】

参数名称	描述	输入/输出
pcm	pcm 句柄	输入
data	音频数据指针	输入
frame_count	帧数	输入

### 【返回值】

返回值	描述
正值	成功
负值	失败

### 【需求】

- 头文件：pcm.h
- 库文件：libtinyalsa.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

AXERA CONFIDENTIAL FOR Sipeed

2.2 声音质量增强

- AX\_AUDIO\_PROCESS\_Init: 初始化声音增强模块。
- AX\_AUDIO\_PROCESS\_DeInit: 反初始化声音增强模块。
- AX\_AUDIO\_PROCESS\_Proc: 声音增强音频处理。
- AX\_AUDIO\_InterleavedToNoninterleaved16: 交错转非交错（16bit）。
- AX\_AUDIO\_MonoToStereo16: 单声道转立体声（16bit）。

AX\_AUDIO\_PROCESS\_Init

【描述】

初始化声音增强模块。

【语法】

```
AX_S32 AX_AUDIO_PROCESS_Init(const AUDIO_PROCESS_ATTR_S *pstAudioProcessAttr)
```

【参数】

参数名称	描述	输入/输出
pstAudioProcessAttr	声音增强属性指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: ax\_audio\_process.h
- 库文件: libax\_audio\_3a.so

【注意】



无

**【举例】**

无

**【相关主题】**

无

## AX\_AUDIO\_PROCESS\_DeInit

**【描述】**

反初始化声音增强模块。

**【语法】**

```
AX_S32 AX_AUDIO_PROCESS_DeInit()
```

**【参数】**

无

**【返回值】**

返回值	描述
0	成功
非 0	失败

**【需求】**

- 头文件：ax\_audio\_process.h
- 库文件：libax\_audio\_3a.so

**【注意】**

无

**【举例】**

无

#### 【相关主题】

无

## AX\_AUDIO\_PROCESS\_Proc

#### 【描述】

声音增强音频处理。

#### 【语法】

```
AX_S32 AX_AUDIO_PROCESS_Proc(AX_VOID *in_data, AX_VOID *ref_data, AX_VOID *out_data)
```

#### 【参数】

参数名称	描述	输入/输出
in_data	输入音频	输入
ref_data	参考音频	输入
out_data	输出音频	输出

#### 【返回值】

返回值	描述
0	成功
非 0	失败

#### 【需求】

- 头文件：ax\_audio\_process.h
- 库文件：libax\_audio\_3a.so

#### 【注意】

无

**【举例】**

无

**【相关主题】**

无

**AX\_AUDIO\_InterleavedToNoninterleaved16****【描述】**

交错转非交错（16bit）。

**【语法】**

```
AX_VOID AX_AUDIO_InterleavedToNoninterleaved16(AX_S16 *interleaved, AX_U32
frameCount, AX_S16 *left, AX_S16 *right)
```

**【参数】**

参数名称	描述	输入/输出
interleaved	交错格式音频	输入
frameCount	帧数	输入
left	非交错格式左声道	输出
right	非交错格式右声道	输出

**【返回值】**

无

**【需求】**

- 头文件：ax\_audio\_process.h
- 库文件：libax\_audio\_3a.so

**【注意】**

无

**【举例】**

无

**【相关主题】**

无

**AX\_AUDIO\_MonoToStereo16****【描述】**

单声道转立体声（16bit）。

**【语法】**

```
AX_VOID AX_AUDIO_MonoToStereo16(AX_S16 *mono, AX_U32 frameCount, AX_S16
*stereo)
```

**【参数】**

参数名称	描述	输入/输出
mono	单声道音频	输入
frameCount	帧数	输入
stereo	立体声音频	输出

**【返回值】**

无

**【需求】**

- 头文件：ax\_audio\_process.h
- 库文件：libax\_audio\_3a.so

**【注意】**

无

**【举例】**

无

【相关主题】

无

## 2.3 音频编码

- AX\_AENC\_Init: 初始化编码器。
- AX\_AENC\_DeInit: 反初始化编码器。
- AX\_AENC\_CreateChn: 创建音频编码通道。
- AX\_AENC\_DestroyChn: 销毁音频编码通道。
- AX\_AENC\_SendFrame: 发送音频编码音频帧。
- AX\_AENC\_GetStream: 获取音频编码码流。
- AX\_AENC\_ReleaseStream: 释放音频编码码流。
- AX\_AENC\_RegisterEncoder: 注册音频编码器。
- AX\_AENC\_UnRegisterEncoder: 注销音频编码器。

## AX\_AENC\_Init

### 【描述】

初始化编码器。

### 【语法】

```
AX_S32 AX_AENC_Init()
```

### 【参数】

无

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## AX\_AENC\_DeInit

### 【描述】

反初始化编码器。

### 【语法】

```
AX_S32 AX_AENC_DeInit()
```

### 【参数】

无

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## AX\_AENC\_CreateChn

### 【描述】

创建音频编码通道。

### 【语法】

```
AX_S32 AX_AENC_CreateChn(AX_AENC_CHN aeChn, const AX_AENC_CHN_ATTR_S
*pstAttr)
```

### 【参数】

参数名称	描述	输入/输出
aeChn	通道号	输入
pstAttr	音频编码通道属性指针	输入

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无



AXERA CONFIDENTIAL FOR Sipeed

## AX\_AENC\_DestroyChn

### 【描述】

销毁音频编码通道。

### 【语法】

```
AX_S32 AX_AENC_DestroyChn(AX_AENC_CHN aeChn)
```

### 【参数】

参数名称	描述	输入/输出
aeChn	通道号	输入

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## AX\_AENC\_SendFrame

### 【描述】

发送音频编码音频帧。

### 【语法】

```
AX_S32 AX_AENC_SendFrame(AX_AENC_CHN aeChn, const AX_AUDIO_FRAME_S *pstFrm)
```

### 【参数】

参数名称	描述	输入/输出
aeChn	通道号	输入
pstFrm	音频帧结构体指针	输入

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## AX\_AENC\_GetStream

### 【描述】

获取编码后码流。

### 【语法】

```
AX_S32 AX_AENC_GetStream(AX_AENC_CHN aeChn, AX_AUDIO_STREAM_S *pstStream,  
AX_S32 s32MilliSec)
```

### 【参数】

参数名称	描述	输入/输出
aeChn	通道号	输入
pstStream	获取的音频码流	输出
s32MilliSec	获取数据的超时时间	输入

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

AXERA CONFIDENTIAL FOR Sipeed

## AX\_AENC\_ReleaseStream

### 【描述】

释放编码后码流。

### 【语法】

```
AX_S32 AX_AENC_ReleaseStream(AX_AENC_CHN aeChn, const AX_AUDIO_STREAM_S
*pstStream)
```

### 【参数】

参数名称	描述	输入/输出
aeChn	通道号	输入
pstStream	获取的音频码流	输入

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

AXERA CONFIDENTIAL FOR Sipeed

## AX\_AENC\_RegisterEncoder

### 【描述】

注册编码器。

### 【语法】

```
AX_S32 AX_AENC_RegisterEncoder(AX_S32 *ps32Handle, const AX_AENC_ENCODER_S
*pstEncoder)
```

### 【参数】

参数名称	描述	输入/输出
ps32Handle	注册句柄	输出
pstEncoder	编码器属性结构体	输入

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无



AXERA CONFIDENTIAL FOR Sipeed

## AX\_AENC\_UnRegisterEncoder

### 【描述】

注销编码器。

### 【语法】

```
AX_S32 AX_AENC_UnRegisterEncoder(AX_S32 s32Handle)
```

### 【参数】

参数名称	描述	输入/输出
s32Handle	注册句柄	输入

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## 2.4 音频解码

- `AX_ADEC_Init`: 初始化解码器。
- `AX_ADEC_DeInit`: 反初始化解码器。
- `AX_ADEC_CreateChn`: 创建音频解码通道。
- `AX_ADEC_DestroyChn`: 销毁音频解码通道。
- `AX_ADEC_SendStream`: 发送音频码流到音频解码通道。
- `AX_ADEC_ClearChnBuf`: 清除 ADEC 通道中当前的音频数据缓存。
- `AX_ADEC_GetFrame`: 获取音频解码帧数据。
- `AX_ADEC_ReleaseFrame`: 释放音频解码帧数据。
- `AX_ADEC_SendEndOfStream`: 向解码器发送码流结束标识符，并清除码流 buffer。
- `AX_ADEC_RegisterDecoder`: 注册解码器。
- `AX_ADEC_UnRegisterDecoder`: 注销解码器。

## AX\_ADEC\_Init

### 【描述】

初始化解码器。

### 【语法】

```
AX_S32 AX_ADEC_Init()
```

### 【参数】

无

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## AX\_ADEC\_DeInit

### 【描述】

反初始化解码器。

### 【语法】

```
AX_S32 AX_ADEC_DeInit()
```

### 【参数】

无

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## AX\_ADEC\_CreateChn

### 【描述】

创建音频解码通道。

### 【语法】

```
AX_S32 AX_ADEC_CreateChn(AX_ADEC_CHN adChn, const AX_ADEC_CHN_ATTR_S
*pstAttr)
```

### 【参数】

参数名称	描述	输入/输出
adChn	通道号	输入
pstAttr	通道属性指针	输入

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

AXERA CONFIDENTIAL FOR Sipeed

## AX\_ADEC\_DestroyChn

### 【描述】

销毁音频解码通道。

### 【语法】

```
AX_S32 AX_ADEC_DestroyChn(AX_ADEC_CHN adChn)
```

### 【参数】

参数名称	描述	输入/输出
adChn	通道号	输入

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无



## AX\_ADEC\_SendStream

### 【描述】

向音频解码通道发送码流。

### 【语法】

```
AX_S32 AX_ADEC_SendStream(AX_ADEC_CHN adChn, const AX_AUDIO_STREAM_S
*pstStream, AX_BOOL bBlock)
```

### 【参数】

参数名称	描述	输入/输出
adChn	通道号	输入
pstStream	音频码流	输入
bBlock	阻塞标志	输入

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

AXERA CONFIDENTIAL FOR Sipeed

## AX\_ADEC\_ClearChnBuf

### 【描述】

清除 ADEC 通道中当前的音频数据缓存。

### 【语法】

```
AX_S32 AX_ADEC_ClearChnBuf(AX_ADEC_CHN adChn)
```

### 【参数】

参数名称	描述	输入/输出
adChn	通道号	输入

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

## AX\_ADEC\_GetFrame

### 【描述】

获取音频解码帧数据。

### 【语法】

```
AX_S32 AX_ADEC_GetFrame(AX_ADEC_CHN adChn, AX_AUDIO_FRAME_S *pstFrmInfo,  
AX_BOOL bBlock)
```

### 【参数】

参数名称	描述	输入/输出
adChn	通道号	输入
pstFrmInfo	音频帧数据结构体	输出
s32MilliSec	是否以阻塞方式获取	输入

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

AXERA CONFIDENTIAL FOR Sipeed

## AX\_ADEC\_ReleaseFrame

### 【描述】

释放音频解码帧数据。

### 【语法】

```
AX_S32 AX_ADEC_ReleaseFrame(AX_ADEC_CHN adChn, const AX_AUDIO_FRAME_S
*pstFrmInfo)
```

### 【参数】

参数名称	描述	输入/输出
adChn	通道号	输入
pstFrmInfo	音频帧数据结构体	输入

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

AXERA CONFIDENTIAL FOR Sipeed

## AX\_ADEC\_SendEndOfStream

### 【描述】

向解码器发送码流结束标识符，并清除码流 buffer。

### 【语法】

```
AX_S32 AX_ADEC_SendEndOfStream(AX_ADEC_CHN adChn, AX_BOOL bInstant)
```

### 【参数】

参数名称	描述	输入/输出
adChn	通道号	输入
bInstant	是否立即清除解码器内部的缓存数据	输入

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无



## AX\_ADEC\_RegisterDecoder

### 【描述】

注册解码器。

### 【语法】

```
AX_S32 AX_ADEC_RegisterDecoder(AX_S32 *ps32Handle, const AX_ADEC_DECODER_S
*pstDecoder)
```

### 【参数】

参数名称	描述	输入/输出
ps32Handle	注册句柄	输出
pstDecoder	解码器属性结构体	输入

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

AXERA CONFIDENTIAL FOR Sipeed

## AX\_ADEC\_UnRegisterDecoder

### 【描述】

注销解码器。

### 【语法】

```
AX_S32 AX_ADEC_UnRegisterDecoder(AX_S32 s32Handle)
```

### 【参数】

参数名称	描述	输入/输出
s32Handle	注册句柄	输入

### 【返回值】

返回值	描述
0	成功
非 0	失败

### 【需求】

- 头文件：ax\_aenc\_api.h
- 库文件：libax\_audio.so

### 【注意】

无

### 【举例】

无

### 【相关主题】

无

本章节包含：

### 3 数据结构

3.1 Tinyalsa 音频输入输出

3.2 声音质量增强

#### AEC\_MODE\_E

【说明】

回声消除模式。

【定义】

```
typedef enum axAEC_MODE_E {  
  
    AEC_MODE_DISABLE = 0,  
  
    AEC_MODE_FLOAT,  
  
    AEC_MODE_FIXED,  
  
} AEC_MODE_E;
```

【成员】

成员名称	描述
AEC_MODE_DISABLE	禁止
AEC_MODE_FLOAT	浮点模式
AEC_MODE_FIXED	整形模式

【注意】

无

【相关数据类型及接口】

无

SUPPRESSION\_LEVEL\_E

【说明】

回声消除抑制级别。

【定义】

```
typedef enum axSUPPRESSION_LEVEL_E {  
  
    SUPPRESSION_LEVEL_LOW = 0,  
  
    SUPPRESSION_LEVEL_MODERATE,  
  
    SUPPRESSION_LEVEL_HIGH  
} SUPPRESSION_LEVEL_E;
```

【成员】

成员名称	描述
SUPPRESSION_LEVEL_LOW	低抑制级别
SUPPRESSION_LEVEL_MODERATE	中等抑制级别
SUPPRESSION_LEVEL_HIGH	高抑制级别

【注意】

无

【相关数据类型及接口】

无

AEC\_FLOAT\_CONFIG\_S

【说明】

回声消除浮点模式属性结构体。

**【定义】**

```
typedef struct axAEC_FLOAT_CONFIG_S {  
  
    SUPPRESSION_LEVEL_E enSuppressionLevel;  
  
} AEC_FLOAT_CONFIG_S;
```

**【成员】**

成员名称	描述
enSuppressionLevel	回声消除抑制级别

**【注意】**

无

**【相关数据类型及接口】**

无

**ROUTING\_MODE\_E****【说明】**

回声消除路由模式。

**【定义】**

```
typedef enum axROUTING_MODE_E {  
  
    ROUTING_MODE_QUIET_EARPIECE_OR_HEADSET = 0,  
  
    ROUTING_MODE_EARPIECE,  
  
    ROUTING_MODE_LOUD_EARPIECE,  
  
    ROUTING_MODE_SPEAKERPHONE,  
  
    ROUTING_MODE_LOUD_SPEAKERPHONE
```

```
} ROUTING_MODE_E;
```

**【成员】**

成员名称	描述
ROUTING_MODE_QUIET_EARPIECE_OR_HEADSET	耳机
ROUTING_MODE_EARPIECE	头戴式耳机
ROUTING_MODE_LOUD_EARPIECE	扩音式耳机
ROUTING_MODE_SPEAKERPHONE	扬声器
ROUTING_MODE_LOUD_SPEAKERPHONE	扩音器

**【注意】**

无

**【相关数据类型及接口】**

无

**AEC\_FIXED\_CONFIG\_S****【说明】**

回声消除整形模式属性结构体。

**【定义】**

```
typedef struct axAEC_FIXED_CONFIG_S {  
    ROUTING_MODE_E eRoutingMode;  
}  
AEC_FIXED_CONFIG_S;
```

**【成员】**

成员名称	描述
eRoutingMode	回声消除路由模式。

**【注意】**

无

【相关数据类型及接口】

无

AEC\_CONFIG\_S

【说明】

回声消除属性结构体。

【定义】

```
typedef struct axAEC_CONFIG_S {  
  
    AEC_MODE_E enAecMode;  
  
    union {  
  
        /*0 ~ 2 default 0*/  
  
        AEC_FLOAT_CONFIG_S stAecFloatCfg;  
  
        /*0 ~ 4 default 3*/  
  
        AEC_FIXED_CONFIG_S stAecFixedCfg;  
  
    };  
} AEC_CONFIG_S;
```

【成员】

成员名称	描述
enAecMode	回声消除模式。
stAecFloatCfg	回声消除浮点模式属性结构体。
stAecFixedCfg	回声消除整形模式属性结构体。

【注意】



无

【相关数据类型及接口】

无

AGGRESSIVENESS\_LEVEL\_E

【说明】

降噪力度级别。

【定义】

```
typedef enum axAGGRESSIVENESS_LEVEL {  
  
    AGGRESSIVENESS_LEVEL_LOW = 0,  
  
    AGGRESSIVENESS_LEVEL_MODERATE,  
  
    AGGRESSIVENESS_LEVEL_HIGH,  
  
    AGGRESSIVENESS_LEVEL_VERYHIGH  
} AGGRESSIVENESS_LEVEL_E;
```

【成员】

成员名称	描述
AGGRESSIVENESS_LEVEL_LOW	低力度。
AGGRESSIVENESS_LEVEL_MODERATE	中等力度。
AGGRESSIVENESS_LEVEL_HIGH	高力度。
AGGRESSIVENESS_LEVEL_VERYHIGH	超高力度。

【注意】

无

【相关数据类型及接口】

无

## NS\_CONFIG\_S

### 【说明】

噪声抑制属性结构体。

### 【定义】

```
typedef struct axNS_CONFIG_S {  
  
    AX_BOOL bNsEnable;  
  
    /*0 ~ 3 default 2*/  
  
    AGGRESSIVENESS_LEVEL_E enAggressivenessLevel;  
  
} NS_CONFIG_S;
```

### 【成员】

成员名称	描述
bNsEnable	使能开关。
enAggressivenessLevel	降噪力度级别。

### 【注意】

无

### 【相关数据类型及接口】

无

## AGC\_MODE\_E

### 【说明】

自动增益控制模式。

**【定义】**

```
typedef enum axAGC_MODE_E {  
  
    AGC_MODE_ADAPTIVE_ANALOG = 0,  
  
    AGC_MODE_ADAPTIVE_DIGITAL,  
  
    AGC_MODE_FIXED_DIGITAL  
  
} AGC_MODE_E;
```

**【成员】**

成员名称	描述
AGC_MODE_ADAPTIVE_ANALOG	自适应模拟模式。（暂不支持）
AGC_MODE_ADAPTIVE_DIGITAL	自适应数字模式。（暂不支持）
AGC_MODE_FIXED_DIGITAL	固定数字模式。

**【注意】**

无

**【相关数据类型及接口】**

无

**AGC\_CONFIG\_S****【说明】**

自动增益控制属性结构体。

**【定义】**

```
typedef struct axAGC_CONFIG_S {  
  
    AX_BOOL bAgcEnable;  
  
    AGC_MODE_E enAgcMode;  
  
}
```

```
/*-31 ~ 0 default -3*/  
  
AX_S16 s16TargetLevel;  
  
/*0 ~ 90 default 9*/  
  
AX_S16 s16Gain;  
  
} AGC_CONFIG_S;
```

### 【成员】

成员名称	描述
bAgcEnable	使能开关。
enAgcMode	自动增益控制模式。
s16TargetLevel	目标电平。
s16Gain	最大增益。

### 【注意】

无

### 【相关数据类型及接口】

无

## AUDIO\_PROCESS\_ATTR\_S

### 【说明】

声音增强音频处理属性结构体。

### 【定义】

```
typedef struct axAUDIO_PROCESS_ATTR_S {  
  
    AX_S32                s32SampleRate;  
  
    AX_U32                u32FrameSamples;
```

```
AEC_CONFIG_S          stAecCfg;

NS_CONFIG_S           stNsCfg;

AGC_CONFIG_S           stAgcCfg;

} AUDIO_PROCESS_ATTR_S;
```

**【成员】**

成员名称	描述
s32SampleRate	采样率。
u32FrameSamples	算法处理帧数。
stAecCfg	回声消除属性结构体。
stNsCfg	噪声抑制属性结构体。
stAgcCfg	自动增益控制属性结构体。

**【注意】**

无

**【相关数据类型及接口】**

无

音频

### 3.4 音频解码

AXERA CONFIDENTIAL FOR Sipeed

## 3.1 Tinyalsa 音频输入输出

### pcm\_format

#### 【说明】

音频样本格式。

#### 【定义】

```
enum pcm_format {  
  
    /* Note: This section must stay in the same  
     * order for binary compatibility with older  
     * versions of TinyALSA. */  
  
    PCM_FORMAT_INVALID = -1,  
  
    /** Signed 16-bit, little endian */  
    PCM_FORMAT_S16_LE = 0,  
  
    /** Signed, 32-bit, little endian */  
    PCM_FORMAT_S32_LE,  
  
    /** Signed, 8-bit */  
    PCM_FORMAT_S8,  
  
    /** Signed, 24-bit (32-bit in memory), little endian */  
    PCM_FORMAT_S24_LE,  
  
    /** Signed, 24-bit, little endian */  
    PCM_FORMAT_S24_3LE,
```

```
/* End of compatibility section. */

/** Signed, 16-bit, big endian */
PCM_FORMAT_S16_BE,

/** Signed, 24-bit (32-bit in memory), big endian */
PCM_FORMAT_S24_BE,

/** Signed, 24-bit, big endian */
PCM_FORMAT_S24_3BE,

/** Signed, 32-bit, big endian */
PCM_FORMAT_S32_BE,

/** Max of the enumeration list, not an actual format. */
PCM_FORMAT_MAX
};
```

### 【成员】

成员名称	描述
PCM_FORMAT_INVALID	无效格式
PCM_FORMAT_S16_LE	16 位小端
PCM_FORMAT_S32_LE	32 位小端
PCM_FORMAT_S8	8 位
PCM_FORMAT_S24_LE	24 位（32-bit in memory）小端
PCM_FORMAT_S24_3LE	24 位小端
PCM_FORMAT_S16_BE	16 位大端
PCM_FORMAT_S24_BE	24 位（32-bit in memory）大端



PCM_FORMAT_S24_3BE	24 位大端
PCM_FORMAT_S32_BE	32 位大端

**【注意】**

无

**【相关数据类型及接口】**

无

AXERA CONFIDENTIAL FOR Sipeed

## pcm\_mask

### 【说明】

PCM 设备参数掩码。

### 【定义】

```
struct pcm_mask {  
    /** bits of the bit mask */  
    unsigned int bits[32 / sizeof(unsigned int)];  
};
```

### 【成员】

成员名称	描述
bits	256 位硬件参数掩码

### 【注意】

无

### 【相关数据类型及接口】

无

## pcm\_config

### 【说明】

PCM 设备参数结构体定义。

### 【定义】

```
struct pcm_config {  
    /** The number of channels in a frame */  
    unsigned int channels;  
  
    /** The number of frames per second */  
    unsigned int rate;  
  
    /** The number of frames in a period */  
    unsigned int period_size;  
  
    /** The number of periods in a PCM */  
    unsigned int period_count;  
  
    /** The sample format of a PCM */  
    enum pcm_format format;  
  
    /* Values to use for the ALSA start, stop and silence thresholds. Setting  
    * any one of these values to 0 will cause the default tinyalsa values to  
    be  
    * used instead. Tinyalsa defaults are as follows.  
    *  
    * start_threshold    : period_count * period_size  
    * stop_threshold    : period_count * period_size  
    * silence_threshold : 0  
*/  
};
```

```
*/  
  
/** The minimum number of frames required to start the PCM */  
  
unsigned int start_threshold;  
  
/** The minimum number of frames required to stop the PCM */  
  
unsigned int stop_threshold;  
  
/** The minimum number of frames to silence the PCM */  
  
unsigned int silence_threshold;  
  
};
```

### 【成员】

成员名称	描述
channels	通道
rate	采样率
period_size	每次传输的数据长度
period_count	缓冲区个数
format	数据格式，如位宽、大小端
start_threshold	启动数据传输阈值
stop_threshold	停止数据传输阈值
silence_threshold	静音阈值

### 【注意】

无

### 【相关数据类型及接口】

无

## pcm\_param

### 【说明】

PCM 设备参数枚举。

### 【定义】

```
enum pcm_param
{
    /** A mask that represents the type of read or write method available
    (e.g. interleaved, mmap). */
    PCM_PARAM_ACCESS,

    /** A mask that represents the @ref pcm_format available (e.g. @ref
    PCM_FORMAT_S32_LE) */
    PCM_PARAM_FORMAT,

    /** A mask that represents the subformat available */
    PCM_PARAM_SUBFORMAT,

    /** An interval representing the range of sample bits available (e.g. 8 to
    32) */
    PCM_PARAM_SAMPLE_BITS,

    /** An interval representing the range of frame bits available (e.g. 8 to
    64) */
    PCM_PARAM_FRAME_BITS,

    /** An interval representing the range of channels available (e.g. 1 to 2)
    */
    PCM_PARAM_CHANNELS,

    /** An interval representing the range of rates available (e.g. 44100 to
```

```
192000) */

PCM_PARAM_RATE,

PCM_PARAM_PERIOD_TIME,

/** The number of frames in a period */

PCM_PARAM_PERIOD_SIZE,

/** The number of bytes in a period */

PCM_PARAM_PERIOD_BYTES,

/** The number of periods for a PCM */

PCM_PARAM_PERIODS,

PCM_PARAM_BUFFER_TIME,

PCM_PARAM_BUFFER_SIZE,

PCM_PARAM_BUFFER_BYTES,

PCM_PARAM_TICK_TIME,

};
```

### 【成员】

成员名称	描述
PCM_PARAM_ACCESS	访问属性
PCM_PARAM_FORMAT	数据格式属性
PCM_PARAM_SUBFORMAT	数据子格式属性
PCM_PARAM_SAMPLE_BITS	样本位宽属性
PCM_PARAM_FRAME_BITS	帧位宽属性
PCM_PARAM_CHANNELS	通道属性
PCM_PARAM_RATE	采样率属性
PCM_PARAM_PERIOD_TIME	周期时间属性

PCM_PARAM_PERIOD_SIZE	周期帧数属性
PCM_PARAM_PERIOD_BYTES	周期字节数属性
PCM_PARAM_PERIODS	周期数属性
PCM_PARAM_BUFFER_TIME	缓冲区时间属性
PCM_PARAM_BUFFER_SIZE	缓冲区大小属性
PCM_PARAM_BUFFER_BYTES	缓冲区字节数属性
PCM_PARAM_TICK_TIME	计时器属性

**【注意】**

无

**【相关数据类型及接口】**

无

## 3.2 声音质量增强

### AEC\_MODE\_E

#### 【说明】

回声消除模式。

#### 【定义】

```
typedef enum axAEC_MODE_E {  
  
    AEC_MODE_DISABLE = 0,  
  
    AEC_MODE_FLOAT,  
  
    AEC_MODE_FIXED,  
  
} AEC_MODE_E;
```

#### 【成员】

成员名称	描述
AEC_MODE_DISABLE	禁止
AEC_MODE_FLOAT	浮点模式
AEC_MODE_FIXED	整形模式

#### 【注意】

无

#### 【相关数据类型及接口】

无

### SUPPRESSION\_LEVEL\_E

#### 【说明】



回声消除抑制级别。

### 【定义】

```
typedef enum axSUPPRESSION_LEVEL_E {  
  
    SUPPRESSION_LEVEL_LOW = 0,  
  
    SUPPRESSION_LEVEL_MODERATE,  
  
    SUPPRESSION_LEVEL_HIGH  
  
} SUPPRESSION_LEVEL_E;
```

### 【成员】

成员名称	描述
SUPPRESSION_LEVEL_LOW	低抑制级别
SUPPRESSION_LEVEL_MODERATE	中等抑制级别
SUPPRESSION_LEVEL_HIGH	高抑制级别

### 【注意】

无

### 【相关数据类型及接口】

无

## AEC\_FLOAT\_CONFIG\_S

### 【说明】

回声消除浮点模式属性结构体。

### 【定义】

```
typedef struct axAEC_FLOAT_CONFIG_S {  
  
    SUPPRESSION_LEVEL_E enSuppressionLevel;
```

```
} AEC_FLOAT_CONFIG_S;
```

**【成员】**

成员名称	描述
enSuppressionLevel	回声消除抑制级别

**【注意】**

无

**【相关数据类型及接口】**

无

## ROUTING\_MODE\_E

**【说明】**

回声消除路由模式。

**【定义】**

```
typedef enum axROUTING_MODE_E {  
  
    ROUTING_MODE_QUITE_EARPIECE_OR_HEADSET = 0,  
  
    ROUTING_MODE_EARPIECE,  
  
    ROUTING_MODE_LOUD_EARPIECE,  
  
    ROUTING_MODE_SPEAKERPHONE,  
  
    ROUTING_MODE_LOUD_SPEAKERPHONE  
  
} ROUTING_MODE_E;
```

**【成员】**

成员名称	描述
ROUTING_MODE_QUITE_EARPIECE_OR_HEADSET	耳机

ROUTING_MODE_EARPIECE	头戴式耳机
ROUTING_MODE_LOUD_EARPIECE	扩音式耳机
ROUTING_MODE_SPEAKERPHONE	扬声器
ROUTING_MODE_LOUD_SPEAKERPHONE	扩音器

**【注意】**

无

**【相关数据类型及接口】**

无

**AEC\_FIXED\_CONFIG\_S****【说明】**

回声消除整形模式属性结构体。

**【定义】**

```
typedef struct axAEC_FIXED_CONFIG_S {  
  
    ROUTING_MODE_E eRoutingMode;  
  
} AEC_FIXED_CONFIG_S;
```

**【成员】**

成员名称	描述
eRoutingMode	回声消除路由模式。

**【注意】**

无

**【相关数据类型及接口】**

无

## AEC\_CONFIG\_S

### 【说明】

回声消除属性结构体。

### 【定义】

```
typedef struct axAEC_CONFIG_S {  
  
    AEC_MODE_E enAecMode;  
  
    union {  
  
        /*0 ~ 2 default 0*/  
  
        AEC_FLOAT_CONFIG_S stAecFloatCfg;  
  
        /*0 ~ 4 default 3*/  
  
        AEC_FIXED_CONFIG_S stAecFixedCfg;  
  
    };  
} AEC_CONFIG_S;
```

### 【成员】

成员名称	描述
enAecMode	回声消除模式。
stAecFloatCfg	回声消除浮点模式属性结构体。
stAecFixedCfg	回声消除整形模式属性结构体。

### 【注意】

无

### 【相关数据类型及接口】

无

## AGGRESSIVENESS\_LEVEL\_E

### 【说明】

降噪力度级别。

### 【定义】

```
typedef enum axAGGRESSIVENESS_LEVEL {  
  
    AGGRESSIVENESS_LEVEL_LOW = 0,  
  
    AGGRESSIVENESS_LEVEL_MODERATE,  
  
    AGGRESSIVENESS_LEVEL_HIGH,  
  
    AGGRESSIVENESS_LEVEL_VERYHIGH  
  
} AGGRESSIVENESS_LEVEL_E;
```

### 【成员】

成员名称	描述
AGGRESSIVENESS_LEVEL_LOW	低力度。
AGGRESSIVENESS_LEVEL_MODERATE	中等力度。
AGGRESSIVENESS_LEVEL_HIGH	高力度。
AGGRESSIVENESS_LEVEL_VERYHIGH	超高力度。

### 【注意】

无

### 【相关数据类型及接口】

无

## NS\_CONFIG\_S

### 【说明】

噪声抑制属性结构体。

【定义】

```
typedef struct axNS_CONFIG_S {  
  
    AX_BOOL bNsEnable;  
  
    /*0 ~ 3 default 2*/  
  
    AGGRESSIVENESS_LEVEL_E enAggressivenessLevel;  
  
} NS_CONFIG_S;
```

【成员】

成员名称	描述
bNsEnable	使能开关。
enAggressivenessLevel	降噪力度级别。

【注意】

无

【相关数据类型及接口】

无

AGC\_MODE\_E

【说明】

自动增益控制模式。

【定义】

```
typedef enum axAGC_MODE_E {  
  
    AGC_MODE_ADAPTIVE_ANALOG = 0,  
  
    AGC_MODE_ADAPTIVE_DIGITAL,
```

```
AGC_MODE_FIXED_DIGITAL
```

```
} AGC_MODE_E;
```

### 【成员】

成员名称	描述
AGC_MODE_ADAPTIVE_ANALOG	自适应模拟模式。（暂不支持）
AGC_MODE_ADAPTIVE_DIGITAL	自适应数字模式。（暂不支持）
AGC_MODE_FIXED_DIGITAL	固定数字模式。

### 【注意】

无

### 【相关数据类型及接口】

无

## AGC\_CONFIG\_S

### 【说明】

自动增益控制属性结构体。

### 【定义】

```
typedef struct axAGC_CONFIG_S {  
  
    AX_BOOL bAgcEnable;  
  
    AGC_MODE_E enAgcMode;  
  
    /*-31 ~ 0 default -3*/  
  
    AX_S16 s16TargetLevel;  
  
    /*0 ~ 90 default 9*/  
  
    AX_S16 s16Gain;  
}
```

```
} AGC_CONFIG_S;
```

**【成员】**

成员名称	描述
bAgcEnable	使能开关。
enAgcMode	自动增益控制模式。
s16TargetLevel	目标电平。
s16Gain	最大增益。

**【注意】**

无

**【相关数据类型及接口】**

无

**AUDIO\_PROCESS\_ATTR\_S****【说明】**

声音增强音频处理属性结构体。

**【定义】**

```
typedef struct axAUDIO_PROCESS_ATTR_S {  
  
    AX_S32                s32SampleRate;  
  
    AX_U32                u32FrameSamples;  
  
    AEC_CONFIG_S          stAecCfg;  
  
    NS_CONFIG_S           stNsCfg;  
  
    AGC_CONFIG_S          stAgcCfg;  
  
} AUDIO_PROCESS_ATTR_S;
```



【成员】

成员名称	描述
s32SampleRate	采样率。
u32FrameSamples	算法处理帧数。
stAecCfg	回声消除属性结构体。
stNsCfg	噪声抑制属性结构体。
stAgcCfg	自动增益控制属性结构体。

【注意】

无

【相关数据类型及接口】

无

3.3 音频编码

AX\_AENC\_CHN\_ATTR\_S

【说明】

音频编码通道属性结构体。

【定义】

```
typedef struct axAENC_CHN_ATTR_S
{
    AX_PAYLOAD_TYPE_E    enType;           /*payload type */
    AX_U32                u32PtNumPerFrm;
    AX_U32                u32BufSize;      /*buf size [2~MAX_AUDIO_FRAME_NUM] */
    AX_VOID               *pValue;        /*point to attribute of definite audio
encoder*/
} AX_AENC_CHN_ATTR_S;
```

【成员】

成员名称	描述
enType	音频编码协议类型
u32PtNumPerFrm	音频编码协议对应帧长
u32BufSize	音频编码缓存大小
pValue	暂未使用

【注意】

无

【相关数据类型及接口】

无

AXERA CONFIDENTIAL FOR Sipeed

AX\_AUDIO\_BIT\_WIDTH\_E

【说明】

音频采样位宽枚举。

【定义】

```
typedef enum axAUDIO_BIT_WIDTH_E
{
    AX_AUDIO_BIT_WIDTH_8    = 0,    /* 8bit width */
    AX_AUDIO_BIT_WIDTH_16   = 1,    /* 16bit width*/
    AX_AUDIO_BIT_WIDTH_24   = 2,    /* 24bit width*/
    AX_AUDIO_BIT_WIDTH_BUTT,
} AX_AUDIO_BIT_WIDTH_E;
```

【成员】

成员名称	描述
AX_AUDIO_BIT_WIDTH_8	8 位
AX_AUDIO_BIT_WIDTH_16	16 位
AX_AUDIO_BIT_WIDTH_24	24 位

【注意】

无

【相关数据类型及接口】

无

## AX\_AUDIO\_SOUND\_MODE\_E

### 【说明】

音频声道枚举。

### 【定义】

```
typedef enum axAUDIO_SOUND_MODE_E
{
    AX_AUDIO_SOUND_MODE_MONO    =0, /*mono*/
    AX_AUDIO_SOUND_MODE_STEREO =1, /*stereo*/
    AX_AUDIO_SOUND_MODE_BUTT
} AX_AUDIO_SOUND_MODE_E;
```

### 【成员】

成员名称	描述
AX_AUDIO_SOUND_MODE_MONO	单声道
AX_AUDIO_SOUND_MODE_STEREO	立体声

### 【注意】

无

### 【相关数据类型及接口】

无

## AX\_AUDIO\_FRAME\_S

### 【说明】

音频帧结构体定义。

### 【定义】

```
typedef struct axAUDIO_FRAME_S
{
    AX_AUDIO_BIT_WIDTH_E  enBitwidth;    /*audio frame bitwidth*/
    AX_AUDIO_SOUND_MODE_E enSoundmode;    /*audio frame momo or stereo mode*/
    AX_U8*  u64VirAddr;
    AX_U64  u64PhyAddr;
    AX_U64  u64TimeStamp;    /*audio frame timestamp*/
    AX_U32  u32Seq;          /*audio frame seq*/
    AX_U32  u32Len;          /*data lenth in frame*/
    AX_U32  u32PoolId[2];
} AX_AUDIO_FRAME_S;
```

### 【成员】

成员名称	描述
enBitwidth	音频采样精度
enSoundmode	音频声道模式
u64VirAddr	音频帧数据虚拟地址
u64PhyAddr	暂未使用
u64TimeStamp	音频帧时间戳
enBitwidth	音频采样精度

成员名称	描述
enSoundmode	音频声道模式
u64VirAddr	音频帧数据虚拟地址

**【注意】**

无

**【相关数据类型及接口】**

无

AXERA CONFIDENTIAL FOR Sipeed

## AX\_AENC\_ENCODER\_S

### 【说明】

定义编码器属性结构体。

### 【定义】

```
typedef struct axAENC_ENCODER_S
{
    AX_PAYLOAD_TYPE_E  enType;

    AX_U32              u32MaxFrmLen;

    AX_S8              aszName[17];    /* encoder type, be used to print proc
information */

    AX_S32              (*pfnOpenEncoder) (AX_VOID *pEncoderAttr, AX_VOID
**ppEncoder); /* pEncoder is the handle to control the encoder */

    AX_S32              (*pfnEncodeFrm) (AX_VOID *pEncoder, const AX_AUDIO_FRAME_S
*pstData,

                                      AX_U8 *pu8Outbuf, AX_U32 *pu32OutLen);

    AX_S32              (*pfnCloseEncoder) (AX_VOID *pEncoder);
} AX_AENC_ENCODER_S;
```

### 【成员】

成员名称	描述
enType	编码协议类型
u32MaxFrmLen	最大码流长度
aszName	编码器名称
pfnOpenEncoder	打开编码器的函数指针
pfnEncodeFrm	进行编码的函数指针



成员名称	描述
pfnCloseEncoder	关闭编码器的函数指针

**【注意】**

无

**【相关数据类型及接口】**

无

AXERA CONFIDENTIAL FOR Sipeed

## AX\_AUDIO\_STREAM\_S

### 【说明】

定义音频码流结构体。

### 【定义】

```
typedef struct axAUDIO_STREAM_S
{
    AX_U8 *pStream;          /* the virtual address of stream */
    AX_U64 u64PhyAddr;       /* the physics address of stream */
    AX_U32 u32Len;           /* stream lenth, by bytes */
    AX_U64 u64TimeStamp;     /* frame time stamp*/
    AX_U32 u32Seq;           /* frame seq,if stream is not a valid frame,u32Seq
is 0*/
} AX_AUDIO_STREAM_S;
```

### 【成员】

成员名称	描述
pStream	音频码流数据指针
u64PhyAddr	暂未使用
u32Len	音频码流长度，以 byte 为单位
u64TimeStamp	音频码流时间戳
u32Seq	音频码流序号

### 【注意】

无

### 【相关数据类型及接口】

无

AXERA CONFIDENTIAL FOR Sipeed

3.4 音频解码

AX\_ADEC\_MODE\_E

【说明】

定义解码方式。

【定义】

```
typedef enum axADEC_MODE_E
{
    AX_ADEC_MODE_PACK = 0, /*require input is valid dec pack(a
                             complete frame encode result),
                             e.g.the stream get from AENC is a
                             valid dec pack, the stream know actually
                             pack len from file is also a dec pack.
                             this mode is high-performative*/
    AX_ADEC_MODE_STREAM, /*input is stream, low-performative,
                           if you couldn't find out whether a stream is
                           vaild dec pack, you could use
                           this mode*/
    AX_ADEC_MODE_BUTT
}AX_ADEC_MODE_E;
```

【成员】

成员名称	描述
AX_ADEC_MODE_PACK	Pack 方式解码

成员名称	描述
AX_ADEC_MODE_STREAM	stream 方式解码

**【注意】**

无

**【相关数据类型及接口】**

无

AXERA CONFIDENTIAL FOR Sipeed

AX\_ADEC\_CHN\_ATTR\_S

【说明】

定义解码通道属性结构体。

【定义】

```
typedef struct axADEC_CH_ATTR_S
{
    AX_PAYLOAD_TYPE_E enType;

    AX_U32          u32BufSize; /*buf size[2~MAX_AUDIO_FRAME_NUM]*/

    AX_ADEC_MODE_E  enMode;     /*decode mode*/

    AX_VOID          *pValue;

}AX_ADEC_CHN_ATTR_S;
```

【成员】

成员名称	描述
enType	音频解码协议类型
u32BufSize	音频解码缓存大小
enMode	解码属性
pValue	具体协议属性指针

【注意】

无

【相关数据类型及接口】

无

## AX\_ADEC\_DECODER\_S

### 【说明】

定义解码器属性结构体。

### 【定义】

```
typedef struct axADEC_DECODER_S
{
    AX_PAYLOAD_TYPE_E  enType;

    AX_S8               aszName[17];

    AX_S32              (*pfnOpenDecoder)(AX_VOID *pDecoderAttr, AX_VOID
**ppDecoder);

    AX_S32              (*pfnDecodeFrm)(AX_VOID *pDecoder, AX_U8 **pu8Inbuf, AX_S32
*ps32LeftByte, AX_U16 *pul6Outbuf, AX_U32 *pu32OutLen, AX_U32 *pu32Chns);

    AX_S32              (*pfnGetFrmInfo)(AX_VOID *pDecoder, AX_VOID *pInfo);

    AX_S32              (*pfnCloseDecoder)(AX_VOID *pDecoder);

    AX_S32              (*pfnResetDecoder)(AX_VOID *pDecoder);

} AX_ADEC_DECODER_S;
```

### 【成员】

成员名称	描述
enType	解码协议类型
aszName	解码器名称
pfnOpenDecoder	打开解码器的函数指针
pfnDecodeFrm	进行解码的函数指针
pfnGetFrmInfo	获取音频帧信息的函数指针
pfnCloseDecoder	关闭解码器的函数指针

成员名称	描述
pfnResetDecoder	清空缓存 buffer，复位解码器

**【注意】**

无

**【相关数据类型及接口】**

无

AXERA CONFIDENTIAL FOR Sipeed



本章节包含：

[4.1 音频编码错误码](#)

[4.2 音频解码错误码](#)

## 4 错误码

AXERA CONFIDENTIAL FOR Sipeed

## 4.1 音频编码错误码

音频编码 API 错误码如下表所示：

表4-1 音频编码 API 错误码

错误代码	宏定义	描述
0x800c0004	AX_ERR_AENC_INVALID_CHNID	音频编码通道号无效
0x800c000b	AX_ERR_AENC_NULL_PTR	输入参数空指针错误
0x800c0014	AX_ERR_AENC_NOT_SUPPORT	操作不被支持
0x800c0015	AX_ERR_AENC_NOT_PERM	操作不允许
0x800c0016	AX_ERR_AENC_EXIST	音频编码通道已经创建
0x800c0017	AX_ERR_AENC_UNEXIST	音频编码通道未创建
0x800c0018	AX_ERR_AENC_NOMEM	系统内存不足
0x800c0019	AX_ERR_AENC_NOBUF	编码通道缓存分配失败
0x800c0020	AX_ERR_AENC_BUF_EMPTY	编码通道缓存空
0x800c0021	AX_ERR_AENC_BUF_FULL	编码通道缓存满

## 4.2 音频解码错误码

音频解码 API 错误码如下表所示：

表4-2 音频解码 API 错误码

错误代码	宏定义	描述
0x800f0002	AX_ERR_ADEC_INVALID_DEVID	音频解码设备号无效
0x800f0004	AX_ERR_ADEC_INVALID_CHNID	音频解码通道号无效
0x800f000a	AX_ERR_ADEC_ILLEGAL_PARAM	音频解码参数设置无效
0x800f000b	AX_ERR_ADEC_NULL_PTR	输入参数空指针错误
0x800f0010	AX_ERR_ADEC_SYS_NOTREADY	系统没有初始化

错误代码	宏定义	描述
0x800f0013	AX_ERR_ADEC_NOT_CONFIG	解码通道属性未配置
0x800f0014	AX_ERR_ADEC_NOT_SUPPORT	操作不被支持
0x800f0015	AX_ERR_ADEC_NOT_PERM	操作不允许
0x800f0016	AX_ERR_ADEC_EXIST	音频解码通道已经创建
0x800f0017	AX_ERR_ADEC_UNEXIST	音频解码通道未创建
0x800f0018	AX_ERR_ADEC_NOMEM	系统内存不足
0x800f0019	AX_ERR_ADEC_NOBUF	解码通道缓存分配失败
0x800f0020	AX_ERR_ADEC_BUF_EMPTY	解码通道缓存空
0x800f0021	AX_ERR_ADEC_BUF_FULL	解码通道缓存满
0x800f0080	AX_ERR_ADEC_DECODER_ERR	音频解码数据错误
0x800f0081	AX_ERR_ADEC_BUF_LACK	解码输入缓存空间不够
0x800f0082	AX_ERR_ADEC_END_OF_STREAM	解码通道码流结束

待补充。

## 5 调试信息

AXERA CONFIDENTIAL FOR Sipeed