9 音频

9.1 概述

AUDIO 模块包括音频输入、音频输出、音频编码、音频解码四个子模块。音频输入和输出模块通过对 Hi35xx 芯片音频接口的控制实现音频输入输出功能。音频编码和解码模块提供对 G711、G726、ADPCM 格式的音频编解码功能，并支持录制和播放 LPCM 格式的原始音频文件。

9.2 功能描述

9.2.1 音频输入和音频输出

9.2.1.1 音频接口和 AI、AO 设备

音频输入输出接口简称为 AIO（Audio Input/Output）接口，用于和 Audio Codec 对接，完成声音的录制和播放。AIO 接口分为两种类型：只支持输入或只支持输出，当为输入类型时，又称为 AIP，当为输出类型时，又称为 AOP。

软件中负责抽象音频接口输入功能的单元，称之为 AI 设备；负责抽象音频接口输出功能的单元，称之为 AO 设备。

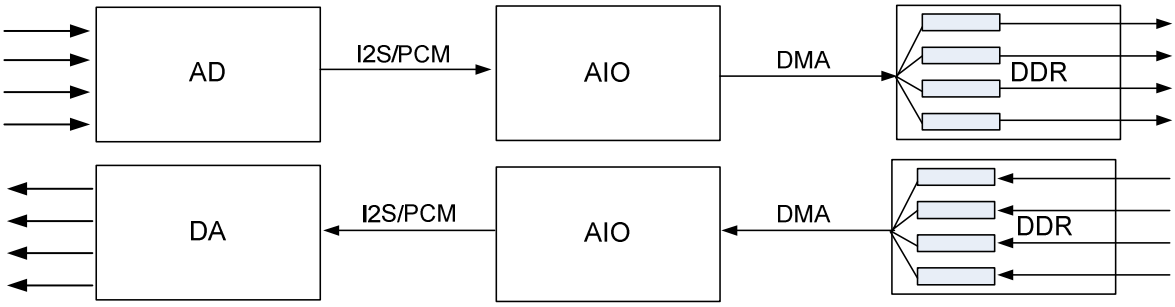
对每个输入输出接口，软件根据该接口支持的功能，分别与 AI 设备和 AO 设备建立映射关系。例如：AIP0 只支持音频输入，则 AIP0 映射为 AiDev0；AOP0 只支持音频输出，则 AOP0 映射为 AoDev0。

9.2.1.2 录音和播放原理

原始音频信号以模拟信号的形式给出后，通过 Audio Codec，按一定采样率和采样精度转换为数字信号。Audio Codec 以 I2S 时序或 PCM 时序的方式，将数字信号传输给 AI 设备。芯片利用 DMA 将 AI 设备中的音频数据搬移到内存中，完成录音操作。

播放和录音是基于同样的原理。芯片利用 DMA 将内存中的数据传输到 AO 设备。AO 设备通过 I2S 时序或 PCM 时序向 Audio Codec 发送数据。Audio Codec 完成数字信号到模拟信号的转换过程，并输出模拟信号。

图9-1 录音和播放示意图



9.2.1.3 音频接口时序和 AI、AO 通道排列

音频接口时序

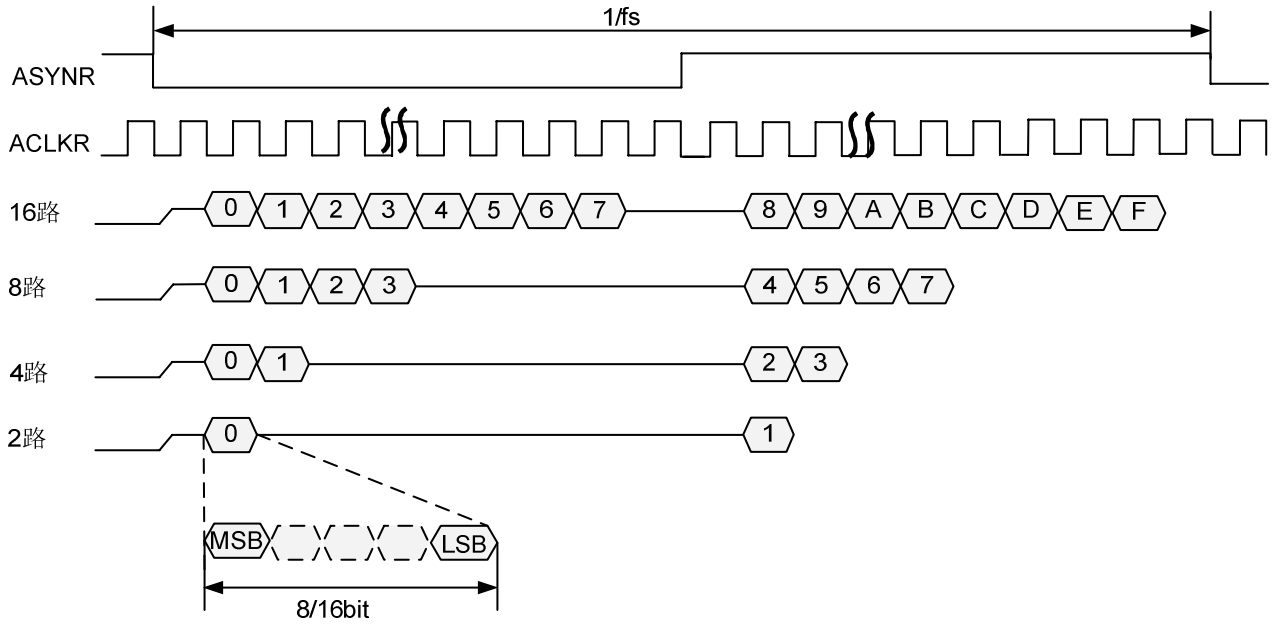
音频接口支持标准的 I^2S 接口时序模式和 PCM 接口时序模式，并提供灵活的配置以支持与多种 Audio Codec 对接。详细的时序支持情况请参考对应芯片的用户指南。

为实现成功对接，需要对 I^2S 或者 PCM 协议以及对接的 Audio Codec 时序支持情况有足够了解，这里只简单介绍下 I^2S 及 PCM 接口时序的几个特性：

- 按照标准 I^2S 或 PCM 协议，总是先传送最高有效位 MSB，后传送最低有效位 LSB，即按照从高位到低位的顺序传输串行数据。
- AI 设备支持扩展的多路接收的 I^2S 及 PCM 接口时序。对接时，Codec 的时序模式选择、同步时钟、采样位宽等配置必须与 AI 设备的配置保持一致，否则可能采集不到正确的数据。
- AI/AO 设备支持主模式和从模式，主模式即 AI/AO 设备提供时钟，从模式即 Audio Codec 提供时钟。主模式时，如果 AI 设备与 AO 设备同时对接同一个 Codec，则时钟供输入和输出共同使用，其他情况没有此限制。而从模式时的输入输出时钟可以分别由外围 Audio Codec 提供。
- 由于时序的问题，在 AI/AO 设备从模式下，建议用户先配置好对接的 Codec，再配置 AI 或 AO 设备；而在 AI/AO 设备主模式下，建议用户先配置好 AI 或 AO 设备，再配置对接的 Codec。
- AI/AO 设备选择主模式时，有些 AI/AO 设备只提供用于时序同步的帧同步时钟和位流时钟，不提供 MCLK，这时如果 Audio Codec 使用外接的晶振作为工作时钟，这样可能导致声音失真，因此推荐使用从模式或者使用位流时钟产生 Codec 内部工作主时钟。
- 当 AI/AO 设备为主模式时，对于向外提供了 MCLK 的 AI/AO 设备，MCLK 的设定为：
  - 采样率为 96k/48k/24k/12k 时，提供 12.288MHz 的主时钟。
  - 采样率为 32k/16k/8k（32k 采样位宽不是 256bits，8k 要求采样位宽不是 16bits）时，提供 12.288MHz 的主时钟。
  - 采样率为 64k/32k/16k/8k（32k 采样位宽为 256bits 或 8k 采样位宽为 16bits）时，提供 8.192MHz 的主时钟。
  - 采样率为 44.1k/22.05k/11.025 时，提供 11.2896MHz 的主时钟。

- AI/AO 设备支持标准 PCM 模式和自定义 PCM 模式。PCM 模式下只支持单声道模式，因此在 PCM 模式下，需要关闭对接 Codec 的右声道输出，否则声音会有杂音。根据 PCM 标准模式协议，当工作模式为标准 PCM 主模式或标准 PCM 从模式时，音频输入输出的数据相对帧同步信号延迟 1 个位流时钟（BCLK）周期。如果在标准 PCM 主模式或标准 PCM 从模式下对接外置的 Codec，需要确保外置 Codec 的数据相对帧同步信号延迟的位流时钟（BCLK）周期数大于音频输入输出数据相对帧同步信号延迟的位流时钟周期数（1 个 BCLK），否则声音会有杂音。

图9-2 I²S 1/2/4/8/16 路接收



AI、AO 通道

AI 和 AO 通道是软件层次的概念，下面通过举例的方式来说明通道的概念。例如当 AI 设备使用多路复用的 I²S 接收模式时，标准的 I²S 协议只有左右声道这个概念，AI 设备最大支持左右声道各接收 128bit 音频数据。此时假设 Codec 具有复用功能，可以将 16 路 16bit 采样精度的音频数据复用为 2 路 128bit 的 I²S 左右声道数据，那么 AI 设备可以解析这 16 路音频数据，并称这 16 路音频数据为 16 个音频通道。

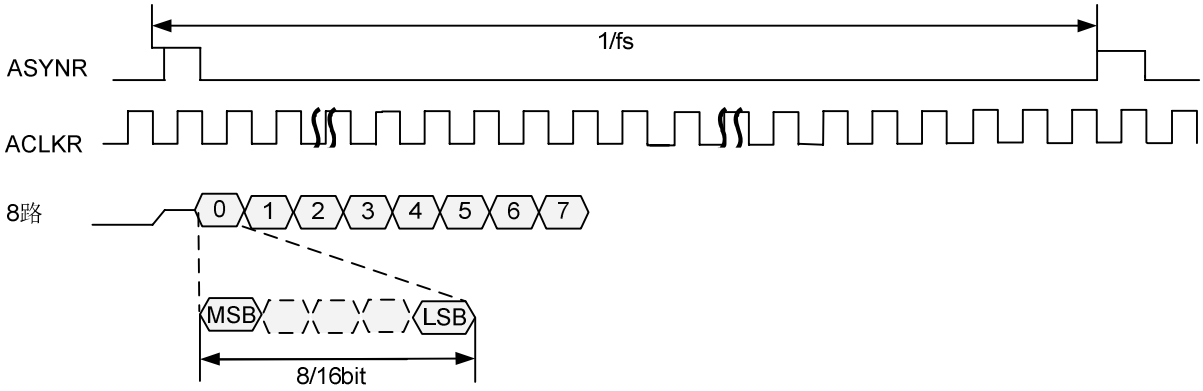
AIO 在不同协议时，支持的 AI、AO 多路复用模式有差异，如表 9-1 所示。

表9-1 AIO 最大支持 AI、AO 通道数

AIO 多路复用	最大支持	通道举例
I²S 时序接收	左右声道各 128bit	16bit16chn 等
PCM 时序接收	单声道 128bit	16bit8chn 等
I²S 时序发送	左右声道各 16bit	16bit2chn
PCM 时序发送	单声道 16bit	16bit1chn

AI、AO 可以在 AI/AO 设备最大支持的比特范围内，按采样精度拆分 AI 和 AO 通道，并按照时序上的顺序，依次视为 AiChn0、AiChn1 等或 AoChn0、AoChn1 等。只有 AI/AO 设备配置的 I^2S 或 PCM 时序与 Codec 配置的时序一致时，才能接收或传送正确的音频数据，将 AI/AO 设备的时序配置和 Codec 的时序配置调为一致的过程称作对接。例如配置 AI 为标准 PCM 从模式，8k 采样率，16bit 采样精度，4 通道；配置 Codec 提供的帧同步时钟为 8k，位流时钟为 8kx64bit，每个采样点为 64bit（Codec 支持将四个 16bit 的采样点复用为一个采样点）；PCM 时序上的通道排列如图 9-3 所示。

图9-3 PCM 时序发送示意图



Hi35xx 上音频 AI 支持的最大通道数 AI 为 16 通道，I2S 模式下 AO 支持的最大通道数为 2 通道，PCM 模式下 AO 支持的最大通道数为 1 通道(其中 Hi35xx 受内置 Codec 限制，对接内置 Codec 时，AI 和 AO 最大都只支持 2 通道)，且配置 AI 和 AO 设备属性时需要将通道配置为偶数或 1。多通道情况下，AI、AO 视通道排列中相对应的两通道为立体声输入输出，例如图 9-3 中，通道 0 和 4、1 和 5、2 和 6、3 和 7 视为立体声的左右声道，即共有四路 16bit 采样精度的立体声输入。这时只应对左声道 0~3 进行操作。

AIO 接口中的 AI 设备与 AO 设备是相互独立的，如果它们同时对接同一个 Codec，则 AI 和 AO 配置的工作模式必须一致，例如 AI 和 AO 均配置为 I2S 时序的从模式。当 AIO 工作在主模式时，AI 和 AO 的通道数与采样精度的乘积也必须相等，这样才能保证由 AIO 发送给 Audio Codec 的同步时钟，对 AI 和 AO 是一致的；如果他们没有同时对接同一个 Codec，则没有此限制。当 AIO 工作在从模式时，AI 和 AO 的通道数与采样精度的乘积可以不一致，此时可以由 Audio Codec 发送不同的同步时钟给 AI 和 AO。Hi35xx SDK 支持通过 SYS 模块绑定接口，建立 AI、AO 通道间的绑定关系，实现音频数据的实时播放。

9.2.1.4 重采样

Hi35xx 的音频输入和音频输出模块支持对音频数据实施重采样。如果启用 AI 重采样功能，则在 HI_MPI_AI_GetFrame 获取数据返回前，内部将会先执行重采样处理，再返回处理后的数据。如果启用了 AO 重采样功能，则音频数据在发送给 AO 之前，内部先执行重采样处理，处理完成后再发送给 AO 通道进行播放。

音频重采样支持任意两种不同采样率（64k、96k 除外）之间的重采样。重采样支持的输入输出采样率为：8kHz，11.025kHz，12kHz，16kHz，22.05kHz，24kHz，32kHz，

44.1kHz，48kHz；不支持的输入输出采样率：64kHz, 96kHz。重采样仅支持处理单声道。

- 如果是 AI 的重采样，则重采样的输入采样率与 AI 设备属性配置的采样率相同，重采样的输出采样率必须与 AI 设备属性配置的采样率不相同，用户只需要配置重采样的输出采样率。重采样之前的每帧采样点数目与 AI 设备属性配置的每帧采样点数目相同。
- 如果是 AO 的重采样，则重采样的输出采样率与 AO 设备属性配置的采样率相同，重采样的输入采样率必须与 AO 设备属性配置的采样率不相同，用户只需要配置重采样的输入采样率。重采样之后音频帧的每帧采样点数目与 AO 设备属性配置的每帧采样点数目相同。
- 如果 AI 的数据需要送到 AENC 进行编码且 AI 启动了重采样，则重采样后的音频帧长 Out PtNumPerFrm 必须满足：CPU 软件编码时 Out PtNumPerFrm 小于等于编码通道属性的 u32PtNumPerFrm；VOIE 编码时 Out PtNumPerFrm 必须是 80、160、240、320、480 中的一个值。Out PtNumPerFrm、重采样前的音频帧长 InPtNumPerFrm、重采样前的采样率 InSampleRate、重采样后的采样率 OutSampleRate 之间的换算关系为：
  OutPtNumPerFrm=OutSampleRate*InPtNumPerFrm/InSampleRate。

Hi3516A/Hi3518EV200/Hi3519V100 不支持 VOIE 编码。

注意

- **当 AI-AO 的数据传输方式为系统绑定方式时，AI 或 AO 的重采样无效。**
- **当 AI-AENC 的数据传输方式为系统绑定方式时，AI 的重采样有效。**
- **非系统绑定方式下，用户可以通过 HI_MPI_AI_GetFrame 接口获取重采样处理后的 AI 音频帧，并发送给 AENC/AO，以建立 AI-AENC 或 AI-AO 的数据传输，此时 AI 或 AO 的重采样有效。**
- **ADEC-AO 的数据传输方式无上述限制，当为系统绑定方式时，AO 的重采样仍有效。**
- **接口 HI_MPI_AI_EnableReSmp 和 HI_MPI_AI_DisableReSmp 使用的是声音质量增强功能中的 RES 模块。**

9.2.1.5 声音质量增强（VQE）

Hi35xx 的音频输入和输出模块支持对音频数据进行声音质量增强(Voice Quality Enhancement)处理。VQE 针对 AI 和 AO 两条通路的异同点，分别通过 UpVQE 和 DnVQE 两个调度逻辑来处理两个通路的数据。

- AI 上行通路的 VQE 功能包含回声抵消、语音降噪、自动增益、高通滤波、录音噪声消除、均衡器、动态压缩、参量均衡器、高动态范围九个处理模块，如图 9-4 所示。AI 上行通路的 VQE 功能在 HI_MPI_AI_GetFrame 接口内实现，并返回处理后的数据。

- AO 下行通路的 VQE 功能包含语音降噪、自动增益、高通滤波、均衡器四个处理模块，如图 9-5 所示。AO 下行通路的 VQE 功能在 HI_MPI_AO_SendFrame 接口内实现。

VQE 分为 2 个调度接口模块（UpVQE 和 DnVQE），11 个功能模块（AEC、ANR、AGC、RNR、EQ、HPF、GAIN、RES、HDR、DRC、PEQ），以及 1 个共用模块（COMMON）。两个调度逻辑使用 libdl 库的 dlopen 方式动态加载各个功能模块，通过统一功能模块的 API 接口格式，对功能模块进行统一调度。VQE 功能模块支持剪裁，用户可根据实际应用场景选择需要用到的功能模块动态库文件，如表 9-2 所示：

表9-2 VQE 库的应用关系

模块名	功能描述	库文件名称	AI 上行通路	AO 下行通路	是否可剪裁	依赖	互斥
UpVQE	AI 音效处理调度接口	libupvqe.a(.so)	有	无	否	无	无
DnVQE	AO 音效处理调度接口	libdnvqe.a(.so)	无	有	否	无	无
AEC	回声抵消	libhive_AEC.so	有	无	是	COMMON	RNR/DRC/PEQ
ANR	语音降噪	libhive_ANR.so	有	有	是	COMMON	RNR/ DRC/PEQ
AGC	自动增益控制	libhive_AGC.so	有	有	是	COMMON	RNR/DRC/PEQ
RNR	录音噪声消除	libhive_RNR.so	有	无	是	无	AEC/ANR/AGC/EQ
DRC	动态压缩控制	libhive_ DRC.so	有	无	是	无	AEC/ANR/AGC/EQ
PEQ	参量均衡器	libhive_PEQ.so	有	无	是	无	AEC/ANR/AGC/EQ
HDR	高动态范围	libhive_HDR.so	有	无	是	无	无
EQ	均衡处理器	libhive_EQ.so	有	有	是	AGC/COMMON	RNR/DRC/PEQ
HPF	高通滤波	libhive_HPF.so	有	有	是	无	无
GAIN	音量调节	libhive_GAIN.so	有	无	是	无	无
RES	重采样	libhive_RES.so	有	有	是	无	无

模块名	功能描述	库文件名称	AI 上行通路	AO 下行通路	是否可剪裁	依赖	互斥
COMMON	公共模块	libhive_common.so	有	有	是	无	无

注意

- AI 上行音效处理的相关接口为 HI_MPI_AI_SetVqeAttr、HI_MPI_AI_GetVqeAttr、HI_MPI_AI_SetHiFiVqeAttr、HI_MPI_AI_GetHiFiVqeAttr、HI_MPI_AI_SetTalkVqeAttr、HI_MPI_AI_GetTalkVqeAttr、HI_MPI_AI_EnableVqe、HI_MPI_AI_DisableVqe、HI_MPI_AI_SetVqeVolume（功能模块仅与 libhive_GAIN.so 相关）、HI_MPI_AI_GetVqeVolume（功能模块仅与 libhive_GAIN.so 相关）。其中，HI_MPI_AI_SetVqeAttr 和 HI_MPI_AI_GetVqeAttr、HI_MPI_AI_SetHiFiVqeAttr 和 HI_MPI_AI_GetHiFiVqeAttr、HI_MPI_AI_SetTalkVqeAttr 和 HI_MPI_AI_GetTalkVqeAttr 是三对接口，它们只能配对调用，例如：如果调用 HI_MPI_AI_SetVqeAttr 设置了参数，则只能使用 HI_MPI_AI_GetVqeAttr 获取设置的参数，调用 HI_MPI_AI_GetHiFiVqeAttr 或者 HI_MPI_AI_GetTalkVqeAttr 获取参数都会返回错误。同时，任意一对与其他两对不能同时使用，在调用其中一个 set 接口设置参数后，在调用 HI_MPI_AI_EnableVqe 使能 vqe 之前，可以调用另外两个 set 接口设置参数；调用 HI_MPI_AI_EnableVqe 后，如果想设置参数，则须先调用 HI_MPI_AI_DisableVqe 禁止使能 vqe，然后再调用其中的一个 set 接口设置参数。各芯片对这三对接口的支持情况表，如表 9-3 中分别用 Vqe、HiFiVqe 和 TalkVqe 代替这三对接口名字。
- AO 下行音效处理的相关接口为 HI_MPI_AO_SetVqeAttr、HI_MPI_AO_GetVqeAttr、HI_MPI_AO_EnableVqe、HI_MPI_AO_DisableVqe。
- 与 libhive_RES.so 有关联的接口还有 HI_MPI_AI_EnableReSmp、HI_MPI_AI_DisableReSmp、HI_MPI_AO_EnableReSmp、HI_MPI_AO_DisableReSmp。
- libhive_RES.so 仅在不使能重采样，且 ai 和 ao 设置的采样率与 vqe 工作采样率一致的情况下可剪裁。
- libhive_common.so 在不使用 AEC、AGC、ANR、EQ 功能模块时可剪裁。
- 使用音频动态库时，需在执行应用程序前指定音频动态库的路径或者将音频动态库拷贝到“/usr/lib”目录下。动态库是在运行时加载的，因此在编译应用程序时不需要链接动态库。

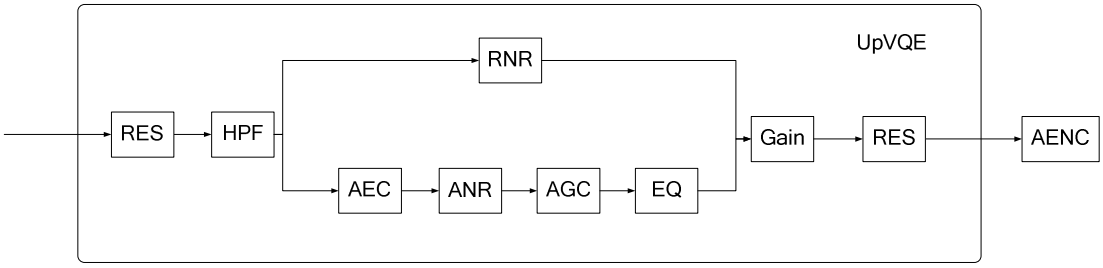
表9-3 Vqe、HiFiVqe 和 TalkVqe 接口支持情况表

接口	Vqe	HiFiVqe	TalkVqe
Hi3516A	支持	支持	不支持
Hi3518EV200	支持	不支持	不支持

Hi3519V100	不支持	支持	支持

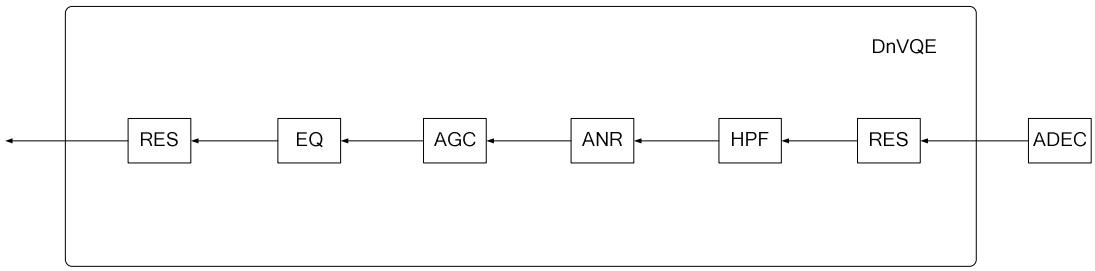
UpVQE 是负责对 AI 数据进行处理的调度逻辑。在上行 AI 通路中，UpVQE 的处理流程如图 9-4 所示：

图9-4 UpVQE 处理流程图



DnVQE 是负责对 AO 数据进行处理的调度逻辑。在 AO 下行通路中，DnVQE 的处理流程如图 9-5 所示：

图9-5 DnVQE 处理流程图



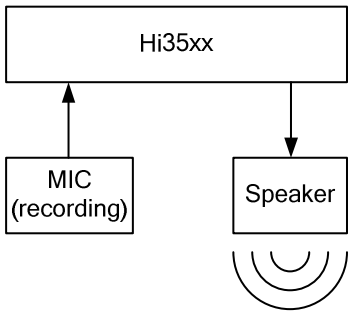
注意

- 当 AI- AO 的数据传输方式为系统绑定方式时，使能 AI 或 AO 的任何 VQE 功能均不起作用。
- 声音质量增强功能不支持立体声。
- 如果 AI-AENC 之间采取的是系统绑定方式传输数据，使能 AI 上行通路的 VQE 功能后，在 AENC 中先进行相应的 VQE 处理，再进行编码。
- 当 AI-AENC/AO 的数据传输方式为非系统绑定方式时，使能 AI 上行通路的 VQE 功能后，VQE 功能在 HI_MPI_AI_GetFrame 接口内部进行处理，用户可以通过该 MPI 接口获取 VQE 处理后的 AI 音频帧，然后发送给 AENC/AO，以建立 AI-AENC 或 AI-AO 的数据传输。
- VQE 支持对采样率为 8kHz，11.025kHz，12kHz，16kHz，22.05kHz，24kHz，32kHz，44.1kHz，48kHz 的数据做处理；不支持对采样率：64kHz, 96kHz 的数据做处理。

AEC

AEC 为回声抵消（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，远端语音数据在 AO 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 AO 设备播放的声音（回声）。

图9-6 回声抵消示意图



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）两路数据来进行算法处理，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。成功启用回声抵消需要具备一定条件：单声道模式，工作采样率为 8kHz、16kHz，且 MIC 采集语音数据的 AI 帧长和远程语音播放的 AO 配置帧长必须相同。以上条件 AI 和 AO 都必须满足。

ANR

ANR 为语音降噪（Audio Noise Reduction）模块，主要工作在需要去除外界噪声，保留语音输入的场景下。

与 RNR 算法比起来，ANR 更讲究噪声处理的干净程度。ANR 会滤除一些环境声音，主要保留语音数据，并会带来一定的细节丢失。所以 ANR 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。

注意

语音降噪功能仅支持 8kHz，16kHz 采样率，不支持立体声。

RNR

RNR 为录音噪声消除（Record Noise Reduction）模块，主要工作在需要去除环境噪声，但保留小信号输入的场景下。

与 ANR 算法比起来，RNR 更讲究细节输入（小信号）的保留度，RNR 会在降噪的同时保留小信号的输入，所以降噪力度会低一点，但能更多的保留现场声音，真实还原场景，适用于运动 DV 场景。

注意

录音噪声消除功能在 Vqe 接口中使用时支持 8kHz、16kHz、48kHz 采样率，在 HiFiVqe 接口中使用仅支持 48kHz 采样率，在 TalkVqe 接口中使用仅支持 8kHz、16kHz 采样率，不支持立体声。

DRC

DRC 为动态压缩控制（Dynamic Range Control）模块，负责控制输出电平，将输出增益控制在一个范围，主要工作在需要保证声音不至于过大或过小的场景下。

DRC 与 AGC 作用相似，但算法实现及调节力度不同。其配合 RNR 使用在运动 DV 场景，与 AEC/ANR/AGC/EQ 互斥。

注意

动态压缩控制功能仅支持 48kHz 采样率，不支持立体声。

PEQ

PEQ 为参量均衡器（Parameter Equalizer）模块，主要对音频数据进行均衡处理，以调节音频数据中各频段声音的增益。

PEQ 与 EQ 均为均衡处理器，但是 PEQ 调节方式更灵活，适用于运动 DV 场景。

注意

参量均衡器功能仅支持 48kHz 采样率，不支持立体声。

HDR

HDR 为高动态范围（High Dynamic Range）模块，主要用于 Codec 输入音量控制，通过动态调节 Codec 增益控制 Codec 音量在合理范围内，保证声音不至于过大或过小。

注意

高动态范围功能在 Vqe 接口中使用时支持 8kHz、16kHz、48kHz 采样率，在 HiFiVqe 接口中使用仅支持 48kHz 采样率，在 TalkVqe 接口中使用仅支持 8kHz、16kHz 采样率，不支持立体声。

HPF

HPF 为高通滤波（high-pass filte）模块，主要负责去除低频噪声。

低频噪声来源经常为硬件噪声或工频噪声，表现为轰轰轰类不舒适的声音。我们可以通过使用频谱分析单板在安静环境下录制的码流，来确定是否需要加入该模块。如果低频噪声不是非常明显，并且客户需要保留低频部分的音源，则不建议加入该模块。推荐配置参数为 AUDIO_HPF_FREQ_120 或 AUDIO_HPF_FREQ_150。

注意

高通滤波功能在 Vqe 接口中使用时支持 8kHz、16kHz、48kHz 采样率，在 HiFiVqe 接口中使用仅支持 48kHz 采样率，在 TalkVqe 接口中使用仅支持 8kHz、16kHz 采样率，不支持立体声。建议在使用 VQE 功能时，一直开启高通滤波功能。

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。AI 通路如果使能了 AGC 功能，那么将不再能够通过调节 AI 增益来控制输出声音大小，需要通过调用接口 HI_MPI_AI_SetVqeVolume 来控制声音输出大小。

注意

自动增益功能仅支持 8kHz，16kHz 采样率，不支持立体声。

EQ

EQ 模块为均衡处理器（Equalizer）模块，主要对音频数据进行均衡处理，以调节音频数据中各频段声音的增益。

注意

- **均衡处理器仅支持 8kHz，16kHz 采样率，不支持立体声。**
- **均衡器使用的前提是，自动增益功能开启。**

GAIN

GAIN 模块是音量调节模块，主要用于调节 AGC 开启后的音量大小。

AGC 负责对语音的音量进行动态增益控制，算法能够处理的 Sin 语音电平范围为 0dB 至-40dB，处理完毕后的语音电平最大值为-2dB，并且在语音最大增益方面做到了 30dB 的增益，这使得 sout 的语音电平能够做到-2dB 至-10dB，同时也使得通过调整 AI 增益来调整 sout 的音量成了一个很困难的事情，因此 GAIN 模块在 VQE 处理流程中位于 AGC 模块后端（如图 9-4 所示），以实现 AI 上行通路的音量调节。

RES

RES 模块为重采样（Resampler）模块。当 AI 上行或 AO 下行通路中开启 VQE 各功能模块时，在处理前后各存在一次重采样（如图 9-4 和图 9-5 所示），第一次的作用是将输入采样率下的音频数据转换成功能模块所支持的工作采样率（8kHz/16kHz/48kHz），第二次则将工作采样率下的数据转换成输出采样率。

9.2.2 音频编码和解码

9.2.2.1 音频编解码流程

Hi35xx SDK 音频的编码类型 G711、G726、ADPCM_DVI4、ADPCM_ORG_DVI4、ADPCM_IMA 均使用 CPU 软件编解码。所有的解码功能都基于独立封装的海思音频解码库，核心解码器工作在用户态，使用 CPU 软件解码。SDK 支持通过 SYS 模块的绑定接口，将一个 AI 通道绑定到 AENC 通道，实现录音编码功能；也可以将一个 ADEC 通道绑定到 AO 通道，实现解码播放功能。

9.2.2.2 音频编解码协议

Hi35xx 支持的音频编解码协议说明如表 9-4 所示。

表9-4 音频编解码协议说明

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	1 MHz	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。 G.711 提供 A 律与 μ 律压缩编码，适用于综合业务网和大多数数字电话链路。北美与日本通常采用 μ 律编码，欧洲和其他地区大都采用 A 律编码。
G726	8kHz	80/160/240/320/480	16、24、32、40（注：G726 编码是一种有损压缩方法，码率比较小时，压缩比和量化误差较大，可能会影响声音质量）	8～3.2	5 MHz	优点：算法简单；语音质量高，多次转换后语音质量有保证，能够在低码率上达到网络等级的话音质量。 缺点：压缩效率较低。 G726_16KBPS 与 MEDIA_G726_16KBPS 两种编码器区别在于编码输出的打包格式。G726_16KBPS 适用于网络传输；MEDIA_G726_16KBPS 适用于 ASF 存储。请参考 RFC3551.pdf。
ADPCM	8kHz	80/160/240/320/480 或 81/161/241/321/481	32	4	2 MHz	优点：算法简单；语音质量高，多次转换后语音质量有保证，能够在低码率上达到网络等级的话音质量。 缺点：压缩效率较低。 ADPCM_IMA4、ADPCM_ORG_DVI4 和

协议	采样率	帧长（采样点）	码率（kbps）	压缩率	CPU 消耗	描述
						ADPCM_DVI 是 ADPCM 编码算法的三种封包格式。IMA 封包格式以第一个采样点作为预测值，DVI 封包格式以上一帧作为预测值。即 IMA 编码每帧需要多输入一个采样点，输入采样点个数为 81/161/241/321/481。DVI 编码每帧输入采样点个数为 80/160/240/320/480。

注意

- “cpu 消耗”的结果值基于 ARM9 288MHz 环境，2 MHz 表示解码分别占有 2M CPU。
- G726 编码是一种有损压缩方法，码率比较小时，压缩比和量化误差较大，可能会影响声音质量。
- 由于 ADPCM_ORG_DVI4 没有传输预测值，这样在网络丢包时，可能会引起异常。因此在音频码流网络传输应用中，不推荐使用该协议编码。
- 当编码或解码协议选择 LPCM 时，音频编码和解码模块可以录制和播放 LPCM 格式的原始音频文件。

9.2.2.3 海思语音帧结构

使用海思语音编解码库进行 G711、G726、ADPCM 格式的编码，编码后的码流遵循以下表格中描述的帧结构，即在每帧码流数据的净荷数据之前填充有 4 个字节的帧头；使用语音编解码库进行以上格式的解码时，需要读取相应的帧头信息。

表9-5 海思语音帧结构

参数位置(单位：HI_S16)	参数比特位说明	参数含义
0	[15:8]	数据帧类型标志位。 01：语音帧； 其他：保留。
	[7:0]	保留。
1	[15:8]	帧循环计数器：0～255。
	[7:0]	数据净荷长度(单位：HI_S16)。
2	[15:0]	净荷数据。

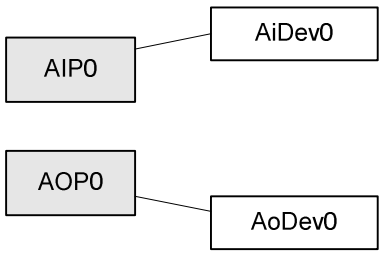
参数位置(单位：HI_S16)	参数比特位说明	参数含义
3	[15:0]	净荷数据。
……	[15:0]	净荷数据。
2+n-1	[15:0]	净荷数据。
2+n	[15:0]	净荷数据。

9.2.3 音频接口与设备的对应关系

芯片集成的 AIO 内部分两类：只支持音频输入的 AIP 和只支持音频输出的 AOP，相应的 AIO 规格详见对应芯片的用户指南。Hi3516A/Hi3518EV200/Hi3519V100 的 AIO 中 AIP/AOP 与 AI、AO 关系如图 9-7 所示。

AIO 接口中只支持音频输出的 AOP0 可以配置是否共用帧同步时钟和位流时钟给 AIP0 使用。共用时，AI 设备 0 和 AO 设备 0 的帧同步时钟与位流时钟必须相同，即采样精度乘以通道数目必须一致，并且采样频率也必须一致。

图9-7 Hi35XX AIO 中 AIP/AOP 与 AI、AO 关系示意图



说明

- AIP0 与 AOP0 同时对接同一个 Codec，可用于语音对讲。
- AIP0/AOP0 对接内置 Codec 时，只支持 I2S 主模式。

音频的输入输出设备号范围

芯片的音频输入输出设备号范围如表 9-6 所示。

表9-6 音频输入输出设备号

芯片类型	音频输入设备 ID	音频输出设备 ID
Hi3516A/Hi3518EV200/Hi3519V100	[0]	[0]

9.2.4 内置 Audio Codec

9.2.4.1 概述

Hi3516A/Hi3518EV200/Hi3519V100 提供一个内置的 Audio Codec（其中，Hi3516A/Hi3518EV200 采用 V500，Hi3519V100 采用 V600），用于对接 AIP0/AOP0 实现声音的播放和录制。AIP0/AOP0 接口可以选择对接内置的 Audio Codec 或外接的 Audio Codec，进行声音的播放及录制。因为内置 Audio Codec 不能发送同步时钟，所以 AIP0/AOP0 接口对接内置 Codec 时只能配置为 I^2S 时序的主模式，用户仍需要正确配置 AIP0/AOP0 和内置 Audio Codec 对接的时序才可接收或发送音频数据。

9.2.4.2 重要概念

内置 Audio Codec 分为模拟部分和数字部分。模拟部分可以通过模拟混音（MICPGA）选择由麦克风输入或 LINEIN 输入，模拟混音支持增益调节。数字部分有 ADC 和 DAC，完成模拟信号和数字信号之间的转换，并且可分别调节音量。用户在进行音量调节时，可综合模拟部分和数字部分的音量调节，建议优先调节模拟部分音量。

内置 Audio Codec 的 DAC 支持软静音和软撤销静音。软静音控制数字音频数据被逐渐衰减到 0，软撤销静音控制数字音频数据逐渐增大到配置的增益值，改变的速率由 mute_rate 控制。当软静音使能时，数字音频数据将会逐渐静音为 0；当软静音禁用时，数字音频数据恢复为配置的增益值。

内置 Audio Codec 的工作时钟由 AIP0/AOP0 提供，并由工作时钟分别产生模拟部分和数字部分的时钟，再由数字部分的时钟产生 ADC 的时钟和 DAC 的时钟。如果默认配置下声音不正常，可以配置数字部分时钟和模拟部分时钟反向，或 DAC 时钟和 ADC 时钟反向。内置 Audio Codec 配置默认采样精度为 16bit。

另外内置 Audio Codec 支持去加重滤波、pop 音抑制和高通滤波，并默认开启这些功能。

使用内置 CODEC 时，AI 设备 0 不能共用 AO 设备 0 的时钟。

9.2.4.3 ioctl 函数

内置 Audio Codec 的用户态接口以 ioctl 形式体现，其形式如下：

```
int ioctl (int fd,
           unsigned long cmd,
          ……
          );
```

该函数是 Linux 标准接口，具备可变参数特性。但在 Audio Codec 中，实际只需要 3 个参数。因此，其语法形式等同于：

```
int ioctl (int fd,
           unsigned long cmd,
           CMD_DATA_TYPE *cmddata);
```

其中，CMD_DATA_TYPE 随参数 cmd 的变化而变化。这 3 个参数的详细描述如表 9-7 所示。

表9-7 ioctl 函数的 3 个参数

参数名称	描述	输入/输出
fd	内置 Audio Codec 设备文件描述符，是调用 open 函数打开内置 Audio Codec 设备文件之后的返回值。	输入
cmd	主要的 cmd（命令控制字）如下： • ACODEC_SOFT_RESET_CTRL：将内置 Codec 恢复为默认设置 • ACODEC_SET_I2S1_FS：设置 I^2S1 接口采样率 • ACODEC_SET_INPUT_VOL：输入总音量控制 • ACODEC_GET_INPUT_VOL：获取输入总音量 • ACODEC_SET_OUTPUT_VOL：输出总音量控制 • ACODEC_GET_OUTPUT_VOL：获取输出总音量 • ACODEC_SET_MIXER_MIC：输入方式选择 • ACODEC_SET_GAIN_MICL：左声道输入的模拟增益控制 • ACODEC_SET_GAIN_MICR：右声道输入的模拟增益控制 • ACODEC_SET_DACL_VOL：左声道输出音量控制 • ACODEC_SET_DACR_VOL：右声道输出音量控制 • ACODEC_SET_ADCL_VOL：左声道输入音量控制 • ACODEC_SET_ADCR_VOL：右声道输入音量控制 • ACODEC_SET_MICL_MUTE：左声道输入静音控制 • ACODEC_SET_MICR_MUTE：右声道输入静音控制 • ACODEC_SET_DACL_MUTE：左声道输出静音控制 • ACODEC_SET_DACR_MUTE：右声道输出静音控制 • ACODEC_DAC_SOFT_MUTE：输出软静音控制 • ACODEC_DAC_SOFT_UNMUTE：输出软撤销静音控制 • ACODEC_ENABLE_BOOSTL：左声道 boost 模拟增益控制 • ACODEC_ENABLE_BOOSTR：右声道 boost 模拟增益控制 • ACODEC_GET_GAIN_MICL：获取模拟左声道输入的增益 • ACODEC_GET_GAIN_MICR：获取模拟右声道输入的增益 • ACODEC_GET_DACL_VOL：获取左声道输出的音量控制 • ACODEC_GET_DACR_VOL：获取右声道输出的音量控制 • ACODEC_GET_ADCL_VOL：获取左声道输入的音量控制 • ACODEC_GET_ADCR_VOL：获取右声道输入的音量控制 • ACODEC_SET_PD_DACL：左声道输出的下电控制 • ACODEC_SET_PD_DACR：右声道输出的下电控制 • ACODEC_SET_PD_ADCL：左声道输入的下电控制	输入

参数名称	描述	输入/输出
	• ACODEC_SET_PD_ADCR：右声道输入的下电控制 • ACODEC_SET_PD_LINEINL：左声道 LINEIN 输入的下电控制 • ACODEC_SET_PD_LINEINR：右声道 LINEIN 输入的下电控制 • ACODEC_SEL_DAC_CLK：设置 DAC 的时钟沿同沿或反沿 • ACODEC_SEL_ADC_CLK：设置 ADC 的时钟沿同沿或反沿 • ACODEC_SEL_ANA_MCLK：设置模拟部分和数字部分的时钟沿同沿或反沿 • ACODEC_DACL_SEL_TRACK：设置 DACL 选择左声道或右声道 • ACODEC_DACR_SEL_TRACK：设置 DACR 选择左声道或右声道 • ACODEC_ADCL_SEL_TRACK：设置 ADCL 选择左声道或右声道 • ACODEC_ADCR_SEL_TRACK：设置 ADCR 选择左声道或右声道 • ACODEC_SET_DAC_DE_EMPHASIS：DAC 的去加重滤波控制 • ACODEC_SET_ADC_HP_FILTER：ADC 的高通滤波控制 • ACODEC_DAC_POP_FREE：DAC 的去 POP 音控制 • ACODEC_DAC_SOFT_MUTE_RATE：DAC 软静音速率控制 • ACODEC_DAC_SEL_I2S：DAC I^2S 接口选择 • ACODEC_ADC_SEL_I2S：ADC I^2S 接口选择 • ACODEC_SET_I2S1_DATAWIDTH：I^2S1 数据接口位宽 • ACODEC_SET_I2S2_DATAWIDTH：I^2S2 数据接口位宽 • ACODEC_SET_I2S2_FS：设置 I^2S2 接口采样率 • ACODEC_SET_DACR2DACL_VOL：DACR 到 DACL 音量控制 • ACODEC_SET_DACL2DACR_VOL：DACL 到 DACR 音量控制 • ACODEC_SET_ADCL2DACL_VOL：ADCL 到 DACL 音量控制 • ACODEC_SET_ADCR2DACL_VOL：ADCR 到 DACL 音量控制 • ACODEC_SET_ADCL2DACR_VOL：DACR 到 DACL 音量控制 • ACODEC_SET_ADCR2DACR_VOL：ADCR 到 DACR 音量	

参数名称	描述	输入/输出
	控制	
cmddata	各 cmd 对应的数据指针	输入/输出

9.3 API 参考

9.3.1 音频输入

音频输入（AI）主要实现配置及启用音频输入设备、获取音频帧数据等功能。

该功能模块提供以下 MPI：

- HI_MPI_AI_SetPubAttr：设置 AI 设备属性。
- HI_MPI_AI_GetPubAttr：获取 AI 设备属性。
- HI_MPI_AI_Enable：启用 AI 设备。
- HI_MPI_AI_Disable：禁用 AI 设备。
- HI_MPI_AI_EnableChn：启用 AI 通道。
- HI_MPI_AI_DisableChn：禁用 AI 通道。
- HI_MPI_AI_GetFrame：获取音频帧。
- HI_MPI_AI_ReleaseFrame：释放音频帧。
- HI_MPI_AI_SetChnParam：设置 AI 通道参数。
- HI_MPI_AI_GetChnParam：获取 AI 通道参数。
- HI_MPI_AI_EnableReSmp：启用 AI 重采样。
- HI_MPI_AI_DisableReSmp：禁用 AI 重采样。
- HI_MPI_AI_SetVqeAttr：设置 AI 的声音质量增强功能相关属性。
- HI_MPI_AI_GetVqeAttr：获取 AI 的声音质量增强功能相关属性。
- HI_MPI_AI_SetHiFiVqeAttr：设置 AI 的声音质量增强功能（HiFi）相关属性。
- HI_MPI_AI_GetHiFiVqeAttr：获取 AI 的声音质量增强功能（HiFi）相关属性。
- HI_MPI_AI_SetTalkVqeAttr：设置 AI 的声音质量增强功能（Talk）相关属性。
- HI_MPI_AI_GetTalkVqeAttr：获取 AI 的声音质量增强功能（Talk）相关属性。
- HI_MPI_AI_EnableVqe：使能 AI 的声音质量增强功能。
- HI_MPI_AI_DisableVqe：禁用 AI 的声音质量增强功能。
- HI_MPI_AI_SetTrackMode：设置声道模式。
- HI_MPI_AI_GetTrackMode：获取声道模式。
- HI_MPI_AI_GetFd：获取 AI 通道对应设备文件句柄。
- HI_MPI_AI_ClrPubAttr：清除 AI 设备属性。

- HI_MPI_AI_SaveFile：开启音频输入保存文件功能。
- HI_MPI_AI_SetVqeVolume：设置声音质量增强功能中的音量大小。
- HI_MPI_AI_QueryFileStatus：查询音频输入通道是否处于存文件的状态。
- HI_MPI_AI_GetVqeVolume：获取声音质量增强功能中的音量大小。
- HI_MPI_AI_EnableAecRefFrame：在 AEC 不打开的情况下也使用户能获取到 AEC 参考帧。
- HI_MPI_AI_DisableAecRefFrame：在 AEC 不打开的情况下禁止获取 AEC 参考帧。

HI_MPI_AI_SetPubAttr

【描述】

设置 AI 设备属性。

【语法】

```
HI_S32 HI_MPI_AI_SetPubAttr(AUDIO_DEV AiDevId,const AIO_ATTR_S *pstAttr);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。 取值范围：请参见表 9-6。	输入
pstAttr	AI 设备属性指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a

【注意】

音频输入设备的属性决定了输入数据的格式，输入设备属性包括工作模式、采样率、采样精度、buffer 大小、每帧的采样点数、扩展标志、时钟选择和通道数目。这些属性应与对接 Codec 配置的时序一致，即能成功对接。

- 工作模式

音频输入输出目前支持 I²S 主模式、I²S 从模式、标准 PCM 主模式、标准 PCM 从模式、自定义 PCM 主模式以及自定义 PCM 从模式。

由于时序的问题，AI 设备在从模式下，建议用户先配置好对接的 Codec，再配置 AI 设备；而在主模式下，建议用户先配置好 AI 设备，再配置对接的 Codec。

- 采样率

  采样率指一秒中内的采样点数，采样率越高表明失真度越小，处理的数据量也就随之增加。一般来说语音使用 8k 采样率，音频使用 32k 或以上的采样率；设置时请确认对接的 Audio Codec 是否支持所要设定的采样率。

- 采样精度

  采样精度指某个通道的采样点数据宽度，同时决定整个设备的通道分布。采样精度可以设置为 8bit 或 16bit（Hi3516A/Hi3518EV200/Hi3519V100 只支持 16bit），实际应用中采样精度还受 Audio Codec 限制。

- buffer 大小

  AIO_ATTR_S 中的 u32FrmNum 项用于配置 AI 中用于接收音频数据的缓存的音频帧帧数，建议配置为 5 以上，否则可能出现采集丢帧等异常。

- 每帧的采样点数

  当音频采样率较高时，建议相应地增加每帧的采样点数目。如要将这些采集到的音频数据送编码，则应保证每帧的持续时长不少于 10ms（例如 16K 的采样频率下每帧的采样点数至少应设置为 160，如果声音有断断续续，可以适当增加每帧的采样点数，参数设置与具体芯片的性能有关），否则解码后声音可能有异常。

- 通道数目

  通道数目指当前输入设备的 AI 功能的通道数目，需与对接的 Audio Codec 的配置保持一致；支持 1 路、2 路、4 路、8 路和 16 路。

- 扩展标志对 AI 设备无效。

- 时钟选择

  AI 必须和 AD 配合起来才能正常工作，用户必须清楚 AD 采集的数据分布和通道的关系才能从正确的通道取得数据。AI 设备为主模式时，决定 AI 设备输出时钟的关键配置项是采样率、采样精度以及通道数目，采样精度乘以通道数目即为 AI 设备时序一次采样的位宽。

  AIO 接口中 AI 设备与 AO 设备是各自独立的，AI 设备 0 和 AO 设备 0 的帧同步时钟与位流时钟是否共用可以通过 u32ClkSel 来配置。当 u32ClkSel 配置为 1 时，表示 AI 设备 0 和 AO 设备 0 的帧同步时钟配置与位流时钟配置是共用的，此时要保证 AI 设备 0 和 AO 设备 0 的帧同步时钟与位流时钟是一致的，当 u32ClkSel 配置为 0 时，表示 AI 设备 0 和 AO 设备 0 的帧同步时钟与位流时钟是由不同的源提供的。当 u32ClkSel 为 1 时，AI 设备 0 和 AO 设备 0 的帧同步时钟与位流时钟必须相同，即采样精度乘以通道数目必须一致，并且采样频率也必须一致。

  对接内置 Codec 时，AI 设备 0 和 AO 设备 0 的帧同步时钟与位流时钟不能共用，u32ClkSel 需要配置为 0。

【举例】

下面的代码实现设置 AI 设备属性及启用 AI 设备。

```
HI_S32 s32ret;
AIO_ATTR_S stAttr;
```

```
AUDIO_DEV AiDevId = 0;

stAttr.enBitwidth = AUDIO_BIT_WIDTH_16;
stAttr.enSamplerate = AUDIO_SAMPLE_RATE_8000;
stAttr.enSoundmode = AUDIO_SOUND_MODE_MONO;
stAttr.enWorkmode = AIO_MODE_I2S_SLAVE;
stAttr.u32EXFlag = 0;
stAttr.u32FrmNum = 5;
stAttr.u32PtNumPerFrm = 160;
stAttr.u32ChnCnt = 2;
stAttr.u32ClkSel = 1;

/* set public attribute of AI device*/
s32ret = HI_MPI_AI_SetPubAttr(AiDevId, &stAttr);
if(HI_SUCCESS != s32ret)
{
    printf("set ai %d attr err:0x%x\n", AiDevId, s32ret);
    return s32ret;
}
/* enable AI device */
s32ret = HI_MPI_AI_Enable(AiDevId);
if(HI_SUCCESS != s32ret)
{
    printf("enable ai dev %d err:0x%x\n", AiDevId, s32ret);
    return s32ret;
}
```

HI_MPI_AI_GetPubAttr

【描述】

获取 AI 设备属性。

【语法】

```
HI_S32 HI_MPI_AI_GetPubAttr(AUDIO_DEV AiDevId, AIO_ATTR_S*pstAttr);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。 取值范围：请参见表 9-6。	输入
pstAttr	AI 设备属性指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a

【注意】

- 获取的属性为前一次配置的属性。
- 如果从来没有配置过属性，则返回失败。

【举例】

```
HI_S32 s32ret;
AUDIO_DEV AiDevId = 0;
AIO_ATTR_S stAttr;

s32ret = HI_MPI_AI_GetPubAttr(AiDevId, &stAttr);
if(HI_SUCCESS != s32ret)
{
    printf("get ai %d attr err:0x%x\n", AiDevId, s32ret);
    return s32ret;
}
```

HI_MPI_AI_Enable

【描述】

启用 AI 设备。

【语法】

```
HI_S32 HI_MPI_AI_Enable(AUDIO_DEV AiDevId);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。 取值范围：请参见表 9-6。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a

【注意】

- 必须在启用前配置 AI 设备属性，否则返回属性未配置错误。
- 如果 AI 设备已经处于启用状态，则直接返回成功。

【举例】

请参见 HI_MPI_AI_SetPubAttr 的举例。

HI_MPI_AI_Disable

【描述】

禁用 AI 设备。

【语法】

```
HI_S32 HI_MPI_AI_Disable(AUDIO_DEV AiDevId);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。 取值范围：请参见表 9-6。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a

【注意】

- 如果 AI 设备已经处于禁用状态，则直接返回成功。
- 禁用 AI 设备前必须先禁用该设备下已启用的所有 AI 通道。
- 要求在禁用 AI 设备之前，先禁用与之关联、使用 AI 的音频数据的 AENC 通道和 AO 设备，否则可能导致该接口调用失败。

【举例】

```
HI_S32 s32ret;
AUDIO_DEV AiDevId = 0;

s32ret = HI_MPI_AI_Disable(AiDevId);
if(HI_SUCCESS != s32ret)
{
    printf("disable ai %d err:0x%x\n", AiDevId);
    return s32ret;
}
```

HI_MPI_AI_EnableChn

【描述】

启用 AI 通道。

【语法】

```
HI_S32 HI_MPI_AI_EnableChn(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 支持的通道范围由 AI 设备属性中的最大通道个数 u32ChnCnt 与声道模式 enSoundmode 决定，详见 AUDIO_SOUND_MODE_E 定义的描述。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a

【注意】

启用 AI 通道前，必须先启用其所属的 AI 设备，否则返回设备未启动的错误码。

【举例】

无。

HI_MPI_AI_DisableChn

【描述】

禁用 AI 通道。

【语法】

```
HI_S32 HI_MPI_AI_DisableChn(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a

【注意】

无。

【举例】

无。

HI_MPI_AI_GetFrame

【描述】

获取音频帧。

【语法】

```
HI_S32 HI_MPI_AI_GetFrame(AUDIO_DEV AiDevId, AI_CHN AiChn, AUDIO_FRAME_S
*pstFrm, AEC_FRAME_S *pstAecFrm , HI_S32 s32MilliSec);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 支持的通道范围由 AI 设备属性中的最大通道个数 u32ChnCnt 与声道模式 enSoundmode 决定。	输入
pstFrm	音频帧结构体指针。	输出
pstAecFrm	回声抵消参考帧结构体指针。	输出
s32MilliSec	获取数据的超时时间 -1 表示阻塞模式，无数据时一直等待； 0 表示非阻塞模式，无数据时则报错返回； >0 表示阻塞 s32MilliSec 毫秒，超时则报错返回。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a

【注意】

- 如果 AI 的回声抵消功能已使能，pstAecFrm 不能是空指针；如果 AI 的回声抵消功能没有使能，pstAecFrm 可以置为空。

- AI 模块会缓存音频帧数据，用于用户态获取。缓存的深度通过 HI_MPI_AI_SetChnParam 接口设定，默认为 0。
- s32MilliSec 的值必须大于等于-1，等于-1 时采用阻塞模式获取数据，等于 0 时采用非阻塞模式获取数据，大于 0 时，阻塞 s32MilliSec 毫秒后，没有数据则返回超时并报错。
- 获取音频帧数据前，必须先使能对应的 AI 通道。
- VQE 使能的情况下，建议采用阻塞模式获取数据。

【举例】

无。

HI_MPI_AI_ReleaseFrame

【描述】

释放音频帧。

【语法】

```
HI_S32 HI_MPI_AI_ReleaseFrame(AUDIO_DEV AiDevId, AI_CHN AiChn,
AUDIO_FRAME_S *pstFrm, AEC_FRAME_S *pstAecFrm);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 支持的通道范围由 AI 设备属性中的最大通道个数 u32ChnCnt 与声道模式 enSoundmode 决定。	输入
pstFrm	音频帧结构体指针。	输入
pstAecFrm	回声抵消参考帧结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h

- 库文件：libmpi.a

【注意】

如果不需要释放回声抵消参考帧，pstAecFrm 置为 NULL 即可。

【举例】

无。

HI_MPI_AI_SetChnParam

【描述】

设置 AI 通道参数。

【语法】

```
HI_S32 HI_MPI_AI_SetChnParam(AUDIO_DEV AiDevId, AI_CHN AiChn,
AI_CHN_PARAM_S *pstChnParam);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 支持的通道范围由 AI 设备属性中的最大通道个数 u32ChnCnt 与声道模式 enSoundmode 决定。	输入
pstChnParam	音频通道参数。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a

【注意】

- 通道参数目前只有一个成员变量，用于设置用户获取音频帧的缓存深度，默认深度为 0。该成员变量的值不能大于 30。

- 建议先调用 HI_MPI_AI_GetChnParam 接口获取默认配置，再调用本接口修改配置，以便于后续扩展。

【举例】

无。

HI_MPI_AI_GetChnParam

【描述】

获取 AI 通道参数。

【语法】

```
HI_S32 HI_MPI_AI_GetChnParam(AUDIO_DEV AiDevId, AI_CHN AiChn,
AI_CHN_PARAM_S *pstChnParam);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 支持的通道范围由 AI 设备属性中的最大通道个数 u32ChnCnt 与声道模式 enSoundmode 决定。	输入
pstChnParam	音频通道参数。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a

【注意】

无。

【举例】

无。

HI_MPI_AI_EnableReSmp

【描述】

启用 AI 重采样。

【语法】

```
HI_S32 HI_MPI_AI_EnableReSmp(AUDIO_DEV AiDevId, AI_CHN AiChn,
AUDIO_SAMPLE_RATE_E enOutSampleRate);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 支持的通道范围由 AI 设备属性中的最大通道个数 u32ChnCnt 决定。	输入
enOutSampleRate	音频重采样的输出采样率。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a libupvqe.a libhive_RES.so libhive_common.so

【注意】

- 在启用 AI 通道之后，调用此接口启用重采样功能。
- 允许重复启用重采样功能，但必须保证后配置的属性与之前配置的属性一样。
- 在禁用 AI 通道之后，如果重新启用 AI 通道，并使用重采样功能，需调用此接口重新启用重采样。

【举例】

以 AI 从 32K 到 8K 的重采样为例，配置如下：

```
/* dev attr of ai */
AUDIO_SAMPLE_RATE_E enOutSampleRate;
```

```
    stAioAttr.u32ChnCnt = 2;
    stAioAttr.enBitwidth = AUDIO_BIT_WIDTH_16;
    stAioAttr.enSamplerate = AUDIO_SAMPLE_RATE_32000;
    stAioAttr.enSoundmode = AUDIO_SOUND_MODE_MONO;
    stAioAttr.u32EXFlag = 1;
    stAioAttr.u32FrmNum = 30;
    stAioAttr.u32PtNumPerFrm = 320*4;
    enOutSampleRate = AUDIO_SAMPLE_RATE_8000;
    s32Ret = HI_MPI_AI_EnableReSmp(AiDev, AiChn, enOutSampleRate);
    if (HI_SUCCESS != s32Ret)
    {
    printf("func(%s) line(%d): failed, s32Ret:0x%x\n", __FUNCTION__,
__LINE__, s32Ret);
            return s32Ret;
    }
```

HI_MPI_AI_DisableReSmp

【描述】

禁用 AI 重采样。

【语法】

```
HI_S32 HI_MPI_AI_DisableReSmp(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h

- 库文件：libmpi.a libupvqe.a libhive_RES.so libhive_common.so

【注意】

- 不再使用 AI 重采样功能的话，应该调用此接口将其禁用。
- 要求在调用此接口之前，先禁用使用该 AI 设备相应通道音频数据的 AENC 通道和 AO 通道，否则可能导致该接口调用失败。

HI_MPI_AI_SetVqeAttr

【描述】

设置 AI 的声音质量增强功能相关属性。

【语法】

```
HI_S32 HI_MPI_AI_SetVqeAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, AUDIO_DEV AoDevId, AO_CHN AoChn, AI_VQE_CONFIG_S *pstVqeConfig);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频输入设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入
AoDevId	用于回声抵消的 AO 设备号。 取值范围：请参见表 9-6。	输入
AoChn	用于回声抵消的 AO 通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入
pstVqeConfig	音频输入声音质量增强配置结构体指针	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h

- 库文件：libmpi.a libupvqe.a libhive_common.so libhive_RNR.so libhive_HPF.so libhive_GAIN.so libhive_EQ.so libhive_ANR.so libhive_AGC.so libhive_AEC.so libhive_HDR

【注意】

- 启用声音质量增强功能前必须先设置相对应 AI 通道的声音质量增强功能相关属性。
- 设置 AI 的声音质量增强功能相关属性前，必须先使能对应的 AI 通道。
- 相同 AI 通道的声音质量增强功能不支持动态设置属性，重新设置 AI 通道的声音质量增强功能相关属性时，需要先关闭 AI 通道的声音质量功能，再设置 AI 通道的声音质量增强功能相关属性。
- 在设置声音质量增强功能属性时，可通过配置相应的声音质量增强功能属性来选择使能其中的部分功能。录音噪声消除仅在运动 DV 场景下使用，语音降噪主要在 IPC 场景下使用，两者不可以同时使用。

【举例】

无。

HI_MPI_AI_GetVqeAttr

【描述】

获取 AI 的声音质量增强功能相关属性。

【语法】

```
HI_S32 HI_MPI_AI_GetVqeAttr(AUDIO_DEV AiDevId, AI_CHN AiChn,
AI_VQE_CONFIG_S *pstVqeConfig);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频输入设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入
pstVqeConfig	音频输入声音质量增强配置结构体指针	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a libupvqe.a libhive_common.so libhive_RNR.so libhive_HPF.so libhive_GAIN.so libhive_EQ.so libhive_ANR.so libhive_AGC.so libhive_AEC.so libhive_HDR.so

【注意】

获取声音质量增强功能相关属性前必须先设置相对应 AI 通道的声音质量增强功能相关属性。

【举例】

无。

HI_MPI_AI_SetHiFiVqeAttr

【描述】

设置 AI 的声音质量增强功能（HiFi）相关属性。

【语法】

```
HI_S32 HI_MPI_AI_SetHiFiVqeAttr(AUDIO_DEV AiDevId, AI_CHN AiChn,
AI_HIFIVQE_CONFIG_S *pstVqeConfig);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频输入设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入
pstVqeConfig	音频输入声音质量增强配置结构体指针	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a libupvqe.a libhive_common.so libhive_RNR.so libhive_HPF.so libhive_GAIN.so libhive_HDR.so libhive_PEQ.so libhive_DRC.so

【注意】

- 启用声音质量增强功能前必须先设置相对应 AI 通道的声音质量增强功能相关属性。
- 设置 AI 的声音质量增强功能相关属性前，必须先使能对应的 AI 通道。
- 相同 AI 通道的声音质量增强功能不支持动态设置属性，重新设置 AI 通道的声音质量增强功能相关属性时，需要先关闭 AI 通道的声音质量功能，再设置 AI 通道的声音质量增强功能相关属性。
- 在设置声音质量增强功能属性时，可通过配置相应的声音质量增强功能属性来选择使能其中的部分功能。HiFi Vqe 仅在运动 DV 场景下使用。

【举例】

无。

HI_MPI_AI_GetHiFiVqeAttr

【描述】

获取 AI 的声音质量增强功能（HiFi）相关属性。

【语法】

```
HI_S32 HI_MPI_AI_GetHiFiVqeAttr(AUDIO_DEV AiDevId, AI_CHN AiChn,
AI_HIFIVQE_CONFIG_S *pstVqeConfig);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频输入设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入
pstVqeConfig	音频输入声音质量增强配置结构体指针	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a libupvqe.a libhive_common.so libhive_RNR.so libhive_HPF.so libhive_GAIN.so libhive_HDR.so libhive_PEQ.so libhive_DRC.so

【注意】

获取声音质量增强功能相关属性前必须先设置相对应 AI 通道的声音质量增强功能相关属性。

【举例】

无。

HI_MPI_AI_SetTalkVqeAttr

【描述】

设置 AI 的声音质量增强功能（Talk）相关属性。

【语法】

```
HI_S32 HI_MPI_AI_SetTalkVqeAttr(AUDIO_DEV AiDevId, AI_CHN AiChn,
AUDIO_DEV AoDevId, AO_CHN AoChn, AI_TALKVQE_CONFIG_S *pstVqeConfig);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频输入设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入
AoDevId	用于回声抵消的 AO 设备号。 取值范围：请参见表 9-6。	输入
AoChn	用于回声抵消的 AO 通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入
pstVqeConfig	音频输入声音质量增强配置结构体指针	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a libupvqe.a libhive_common.so libhive_RNR.so libhive_HPF.so libhive_GAIN.so libhive_EQ.so libhive_ANR.so libhive_AGC.so libhive_AEC.so libhive_HDR.so

【注意】

- 启用声音质量增强功能前必须先设置相对应 AI 通道的声音质量增强功能相关属性。
- 设置 AI 的声音质量增强功能相关属性前，必须先使能对应的 AI 通道。
- 相同 AI 通道的声音质量增强功能不支持动态设置属性，重新设置 AI 通道的声音质量增强功能相关属性时，需要先关闭 AI 通道的声音质量功能，再设置 AI 通道的声音质量增强功能相关属性。
- 在设置声音质量增强功能属性时，可通过配置相应的声音质量增强功能属性来选择使能其中的部分功能。Talk Vqe 主要在 IPC 场景下使用。

【举例】

无。

HI_MPI_AI_GetTalkVqeAttr

【描述】

获取 AI 的声音质量增强功能（Talk）相关属性。

【语法】

```
HI_S32 HI_MPI_AI_GetTalkVqeAttr(AUDIO_DEV AiDevId, AI_CHN AiChn,
AI_TALKVQE_CONFIG_S *pstVqeConfig);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频输入设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入
pstVqeConfig	音频输入声音质量增强配置结构体指针	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a libupvqe.a libhive_common.so libhive_RNR.so libhive_HPF.so libhive_GAIN.so libhive_EQ.so libhive_ANR.so libhive_AGC.so libhive_AEC.so libhive_HDR.so

【注意】

获取声音质量增强功能相关属性前必须先设置相对应 AI 通道的声音质量增强功能相关属性。

【举例】

无。

HI_MPI_AI_EnableVqe

【描述】

使能 AI 的声音质量增强功能。

【语法】

```
HI_S32 HI_MPI_AI_EnableVqe(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频输入设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a libupvqe.a libhive_common.so libhive_RNR.so libhive_HPF.so libhive_GAIN.so libhive_EQ.so libhive_ANR.so libhive_AGC.so libhive_AEC.so libhive_HDR.so libhive_PEQ.so libhive_DRC.so

【注意】

- 启用声音质量增强功能前必须先启用相对应的 AI 通道。
- 多次使能相同 AI 通道的声音质量增强功能时，返回成功。
- 禁用 AI 通道后，如果重新启用 AI 通道，并使用声音质量增强功能，需调用此接口重新启用声音质量增强功能。

【举例】

无。

HI_MPI_AI_DisableVqe

【描述】

禁用 AI 的声音质量增强功能。

【语法】

```
HI_S32 HI_MPI_AI_DisableVqe(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频输入设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h

- 库文件：libmpi.a libupvqe.a libhive_common.so libhive_RNR.so libhive_HPF.so libhive_GAIN.so libhive_EQ.so libhive_ANR.so libhive_AGC.so libhive_AEC.so libhive_HDR.so libhive_PEQ.so libhive_DRC.so

【注意】

- 不再使用 AI 声音质量增强功能时，应该调用此接口将其禁用。
- 多次禁用相同 AI 通道的声音质量增强功能，返回成功。

【举例】

无。

HI_MPI_AI_SetTrackMode

【描述】

设置 AI 声道模式。

【语法】

```
HI_S32 HI_MPI_AI_SetTrackMode(AUDIO_DEV AiDevId, AUDIO_TRACK_MODE_E enTrackMode);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。 取值范围：请参见表 9-6。	输入
enTrackMode	音频输入声道模式。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a

【注意】

- 在 AI 设备成功启用后再调用此接口。
- AI 设备工作在 I^2S 模式时，支持设置声道模式，PCM 模式下不支持。

【举例】

```
HI_S32 s32Ret;
AUDIO_DEV AiDev = 0;
AUDIO_TRACK_MODE_E enTrackMode = AUDIO_TRACK_NORMAL;
AUDIO_TRACK_MODE_E temp;
s32Ret = HI_MPI_AI_SetTrackMode(AiDev, enTrackMode);
if (HI_SUCCESS != s32Ret)
{
printf("Ai set track mode failure! AiDev: %d, enTrackMode: %d, s32Ret:
0x%x.\n", AiDev, enTrackMode, s32Ret);
    return s32Ret;
 }
 s32Ret = HI_MPI_AI_GetTrackMode(AiDev, &temp);
 if (HI_SUCCESS != s32Ret)
 {
   printf("Ai get track mode failure! AiDev: %d, s32Ret: 0x%x.\n", AiDev,
s32Ret);
    return s32Ret;
  }
```

HI_MPI_AI_GetTrackMode

【描述】

获取 AI 声道模式。

【语法】

```
HI_S32 HI_MPI_AI_GetTrackMode(AUDIO_DEV AiDevId, AUDIO_TRACK_MODE_E
*penTrackMode);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。 取值范围：请参见表 9-6。	输入
penTrackMode	音频输入声道模式指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a

【注意】

- 在 AI 设备成功启用后再调用此接口。
- AI 设备工作在 I2S 模式时，支持获取声道模式，PCM 模式下不支持。

【举例】

无。

HI_MPI_AI_GetFd

【描述】

获取音频输入通道号对应的设备文件句柄。

【语法】

```
HI_S32 HI_MPI_AI_GetFd(AUDIO_DEV AiDevId, AI_CHN AiChn)
```

【参数】

参数名称	描述	输入/输出
AiDevId	AI 设备号。 取值范围：请参见表 9-6。	输入
AiChn	AI 通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
正数值	有效返回值。
非正数值	无效返回值。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a

【注意】

无。

【举例】

```
HI_S32 s32AiFd;
HI_S32 s32ret;
AUDIO_DEV AiDevId = 0;
AI_CHN AiChn = 0;

s32AiFd = HI_MPI_AI_GetFd(AiDevId, AiChn);
if(s32AiFd <= 0)
{
return HI_FAILURE;
}
```

HI_MPI_AI_ClrPubAttr

【描述】

清空 Pub 属性。

【语法】

```
HI_S32 HI_MPI_AI_ClrPubAttr(AUDIO_DEV AiDevId);
```

【参数】

参数名称	描述	输入/输出
AiDevId	AI 设备号。 取值范围：请参见表 9-6。	输入

【返回值】

返回值	描述
正数值	有效返回值。
非正数值	无效返回值。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a

【注意】

- 清除设备属性前，需要先停止设备。
- 当 AO 设备共用 AI 设备时钟，且两者设备均未使能时，AO 设备属性需要强行切换（修改后属性不满足共时钟要求：enSamplerate 相同，u32ClkChnCnt 与

enBitwidth 对应的采用位宽的乘积相等），建议切换前先使用该接口清除与 AI 的内部耦合关系。

【相关主题】

无。

HI_MPI_AI_SaveFile

【描述】

开启音频输入保存文件功能。

【语法】

```
HI_S32 HI_MPI_AI_SaveFile(AUDIO_DEV AiDevId, AI_CHN
AiChn,AUDIO_SAVE_FILE_INFO_S *pstSaveFileInfo);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入
pstSaveFileInfo	音频保存文件属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a

【注意】

此接口仅用于 AI-AENC、AI-AO 非系统绑定模式下 dump AI 中 VQE 处理前后的文件，没有使能 VQE 功能时或者是 AI-AENC/AI-AO 系统绑定模式下，使用该接口 dump AI 数据无效。调用后会在指定目录下写出三个指定大小文件。Sin.pcm 为 VQE 处理前的输入帧，Rin.pcm 为 VQE 处理前的回声抵消参考帧，Sou.pcm 为 VQE 处理后的输出帧。

【相关主题】

无。

HI_MPI_AI_SetVqeVolume

【描述】

设置 AI 设备音量大小。

【语法】

```
HI_S32 HI_MPI_AI_SetVqeVolume(AUDIO_DEV AiDevId, AI_CHN AiChn, HI_S32
s32VolumeDb);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 取值范围：[0, AIO_MAX_CHN_NUM]。	输入
s32VolumeDb	音频设备音量大小（以 Db 为单位）。 取值范围：[-20,10]	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a、libupvqe.a、libhive_common.so、libhive_GAIN.so

【注意】

在 AI VQE 使能后再调用此接口。

【举例】

无。

HI_MPI_AI_QueryFileStatus

【描述】

查询音频输入通道是否处于存文件的状态。

【语法】

```
HI_S32 HI_MPI_AI_QueryFileStatus(AUDIO_DEV AiDevId, AI_CHN AiChn,
AUDIO_FILE_STATUS_S* pstFileStatus);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入
pstFileStatus	状态属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a

【注意】

此接口用于查询音频输入通道是否处于存文件的状态，当用户调用 HI_MPI_AI_SaveFile 存储文件后，可调用此接口查询存储的文件是否达到了指定的大小，如果 pstFileStatus 的 bSaving 为 HI_TRUE，说明还没有达到指定大小，为 HI_FALSE 则已经达到指定大小。

【相关主题】

无。

HI_MPI_AI_GetVqeVolume

【描述】

获取 AI 设备音量大小。

【语法】

```
HI_S32 HI_MPI_AI_GetVqeVolume(AUDIO_DEV AiDevId, AI_CHN AiChn, HI_S32
*ps32VolumeDb);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AiChn	音频输入通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入
ps32VolumeDb	音频设备音量大小指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a libupvqe.a libhive_common.so libhive_GAIN.so

【注意】

在 AI VQE 使能后再调用此接口。

【举例】

无。

HI_MPI_AI_EnableAecRefFrame

【描述】

在 AEC 不打开的情况下也使用户能获取到 AEC 参考帧。

【语法】

```
HI_S32 HI_MPI_AI_EnableAecRefFrame(AUDIO_DEV AiDevId, AI_CHN AiChn,
AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【参数】

参数名称	描述	输入/输出
AiDevId	AI 设备号。 取值范围：请参见表 9-6。	输入
AiChn	AI 通道号。 支持的通道范围由 AI 设备属性中的最大通道个数 u32ChnCnt 决定。	输入
AoDevId	用于获取 AEC 参考帧的 AO 设备号。 取值范围：请参见表 9-6。	输入
AoChn	用于获取 AEC 参考帧的 AO 通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a

【注意】

- 此接口仅用于在 AEC 功能关闭的情况下使能获取 AEC 参考帧，AEC 参考帧通过接口 HI_MPI_AI_GetFrame 返回，供用户在上层自己做 AEC 处理。
- 如果在 AEC 功能打开的情况下调用此接口，则返回错误码 HI_ERR_AI_NOT_SUPPORT。
- 必须在 AI 通道使能的情况下才能调用此接口。
- 不支持立体声。

【相关主题】

无。

HI_MPI_AI_DisableAecRefFrame

【描述】

在 AEC 不打开的情况下禁止获取 AEC 参考帧。

【语法】

```
HI_S32 HI_MPI_AI_DisableAecRefFrame(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名称	描述	输入/输出
AiDevId	AI 设备号。 取值范围：请参见表 9-6。	输入
AiChn	AI 通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ai.h
- 库文件：libmpi.a

【注意】

重复调用本接口返回成功。

【举例】

无。

【相关主题】

无。

9.3.2 音频输出

音频输出（AO）主要实现启用音频输出设备、发送音频帧到输出通道等功能。

该功能模块提供以下 MPI：

- HI_MPI_AO_SetPubAttr：设置 AO 设备属性。
- HI_MPI_AO_GetPubAttr：获取 AO 设备属性。
- HI_MPI_AO_Enable：启用 AO 设备。
- HI_MPI_AO_Disable：禁用 AO 设备。
- HI_MPI_AO_EnableChn：启用 AO 通道。
- HI_MPI_AO_DisableChn：禁用 AO 通道。

- HI_MPI_AO_SendFrame：发送 AO 音频帧
- HI_MPI_AO_EnableReSmp：启用 AO 重采样。
- HI_MPI_AO_DisableReSmp：禁用 AO 重采样。
- HI_MPI_AO_PauseChn：暂停 AO 通道。
- HI_MPI_AO_ResumeChn：恢复 AO 通道。
- HI_MPI_AO_ClearChnBuf：清除 AO 通道中当前的音频数据缓存。
- HI_MPI_AO_QueryChnStat：查询 AO 通道中当前的音频数据缓存状态。
- HI_MPI_AO_SetTrackMode：设置 AO 设备声道模式
- HI_MPI_AO_GetTrackMode：获取 AO 设备声道模式
- HI_MPI_AO_SetVolume：设置 AO 设备音量大小
- HI_MPI_AO_GetVolume：获取 AO 设备音量大小
- HI_MPI_AO_SetMute：设置 AO 设备静音状态
- HI_MPI_AO_GetMute：获取 AO 设备静音状态
- HI_MPI_AO_GetFd：获取 AO 通道对应的设备文件句柄。
- HI_MPI_AO_SaveFile：开启音频输出保存文件功能。
- HI_MPI_AO_QueryFileStatus：查询音频输出通道是否处于存文件的状态。
- HI_MPI_AO_ClrPubAttr：清除 AO 设备属性。
- HI_MPI_AO_SetVqeAttr：设置 AO 的声音质量增强功能相关属性。
- HI_MPI_AO_GetVqeAttr：获取 AO 的声音质量增强功能相关属性。
- HI_MPI_AO_EnableVqe：使能 AO 的声音质量增强功能。
- HI_MPI_AO_DisableVqe：禁用 AO 的声音质量增强功能。

HI_MPI_AO_SetPubAttr

【描述】

设置 AO 设备属性。

【语法】

```
HI_S32 HI_MPI_AO_SetPubAttr(AUDIO_DEV AoDevId,const AIO_ATTR_S *pstAttr);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入
pstAttr	音频输出设备属性。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

- 在设置属性之前需要保证 AO 处于禁用状态，如果处于启用状态则需要首先禁用 AO 设备。
- AO 必须和 DA 配合起来才能正常工作，用户必须清楚 DA 发送的数据分布和通道的关系才能从正确的通道发送数据。
- 对接外置 Codec 时，由于时序的问题，在 AO 设备从模式下，建议用户先配置好对接的 Codec，再配置 AO 设备；而在 AO 设备主模式下，建议用户先配置好 AO 设备，再配置对接的 Codec。对接内置 Codec 时，都需要先配置内置 Codec，再配置 AO 设备。
- 对接内置 Codec 时，AI 设备 0 和 AO 设备 0 的帧同步时钟与位流时钟不能共用，u32ClkSel 需要配置为 0。
- AO 设备主模式时，决定 AO 设备输出时钟的关键配置项是采样率、采样精度以及通道数目，采样精度乘以通道数目即为 AO 设备时序一次采样的位宽。
- 扩展标志对 AO 设备无效。
- AO 设备属性结构体中其他项请参见 AI 模块中相关接口的描述。

【举例】

```
HI_S32 s32ret;
AIO_ATTR_S stAttr;
AUDIO_DEV AoDevId = 0;

stAttr.enBitwidth = AUDIO_BIT_WIDTH_16;
stAttr.enSamplerate = AUDIO_SAMPLE_RATE_8000;
stAttr.enSoundmode = AUDIO_SOUND_MODE_MONO;
stAttr.enWorkmode = AIO_MODE_I2S_SLAVE;
stAttr.u32EXFlag = 0;
stAttr.u32FrmNum = 5;
stAttr.u32PtNumPerFrm = 160;
stAttr.u32ChnCnt = 2;
stAttr.u32ClkSel = 0;

/* set ao public attr*/
s32ret = HI_MPI_AO_SetPubAttr(AoDevId, &stAttr);
```

```
if(HI_SUCCESS != s32ret)
{
    printf("set ao %d attr err:0x%x\n", AoDevId, s32ret);
    return s32ret;
}
/* enable ao device*/
s32ret = HI_MPI_AO_Enable(AoDevId);
if(HI_SUCCESS != s32ret)
{
    printf("enable ao dev %d err:0x%x\n", AoDevId, s32ret);
    return s32ret;
}
```

HI_MPI_AO_GetPubAttr

【描述】

获取 AO 设备属性。

【语法】

```
HI_S32 HI_MPI_AO_GetPubAttr(AUDIO_DEV AoDevId, AIO_ATTR_S *pstAttr);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入
pstAttr	音频输出设备属性指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

- 获取的属性为前一次配置的属性。

- 如果从未配置过属性，则返回属性未配置的错误。

【举例】

```
HI_S32 s32ret;
AUDIO_DEV AoDevId = 0;
AIO_ATTR_S stAttr;

/* first enable ao device*/

s32ret = HI_MPI_AO_GetPubAttr(AoDevId, &stAttr);
if(HI_SUCCESS != s32ret)
{
    printf("get ao %d attr err:0x%x\n", AoDevId,s32ret);
    return s32ret;
}
```

HI_MPI_AO_Enable

【描述】

启用 AO 设备。

【语法】

```
HI_S32 HI_MPI_AO_Enable(AUDIO_DEV AoDevId);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

- 要求在启用前配置 AO 设备属性，否则会返回属性未配置的错误。
- 如果 AO 设备已经启用，则直接返回成功。

【举例】

请参见 HI_MPI_AO_SetPubAttr 的举例。

HI_MPI_AO_Disable

【描述】

禁用 AO 设备。

【语法】

```
HI_S32 HI_MPI_AO_Disable(AUDIO_DEV AoDevId);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

- 如果 AO 设备已经禁用，则直接返回成功。
- 禁用 AO 设备前必须先禁用设备下所有 AO 通道。

【举例】

无。

HI_MPI_AO_EnableChn

【描述】

启用 AO 通道。

【语法】

```
HI_S32 HI_MPI_AO_EnableChn(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AoChn	音频输出通道号。 支持的通道范围由 AO 设备属性中的最大通道个数 u32ChnCnt 与声道模式 enSoundmode 决定。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

启用 AO 通道前，必须先启用其所属的 AO 设备，否则返回设备未启动的错误码。

【举例】

请参见 HI_MPI_AI_SetPubAttr 的举例。

HI_MPI_AO_DisableChn

【描述】

禁用 AO 通道。

【语法】

```
HI_S32 HI_MPI_AO_DisableChn(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AoChn	音频输出通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

无。

【举例】

无。

HI_MPI_AO_SendFrame

【描述】

发送 AO 音频帧。

【语法】

```
HI_S32 HI_MPI_AO_SendFrame(AUDIO_DEV AoDevId, AO_CHN AoChn, const
AUDIO_FRAME_S *pstData, HI_S32 s32MilliSec);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AoChn	音频输出通道号。 支持的通道范围由 AO 设备属性中的最大通道个数 u32ChnCnt 与声道模式 enSoundmode 决定。	输入

参数名称	描述	输入/输出
pstData	音频帧结构体指针。	输入
s32MilliSec	发送数据的超时时间 -1 表示阻塞模式； 0 表示非阻塞模式； >0 表示阻塞 s32MilliSec 毫秒，超时则报错返回。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

- 该接口用于用户主动发送音频帧至 AO 输出，如果 AO 通道已经通过系统绑定（HI_MPI_SYS_Bind）接口与 AI 或 ADEC 绑定，不需要也不建议调此接口。
- s32MilliSec 的值必须大于等于-1，等于-1 时采用阻塞模式发送数据，等于 0 时采用非阻塞模式发送数据，大于 0 时，阻塞 s32MilliSec 毫秒后，则返回超时并报错。
- 调用该接口发送音频帧到 AO 输出时，必须先使能对应的 AO 通道。

HI_MPI_AO_EnableReSmp

【描述】

启用 AO 重采样。

【语法】

```
HI_S32 HI_MPI_AO_EnableReSmp(AUDIO_DEV AoDevId, AO_CHN AoChn,
AUDIO_SAMPLE_RATE_E enInSampleRate);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入

参数名称	描述	输入/输出
AoChn	音频输出通道号。 支持的通道范围由 AO 设备属性中的最大通道个数 u32ChnCnt 决定。	输入
enInSampleRate	音频重采样的输入采样率。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a libupvqe.a libhive_RES.so libhive_common.so

【注意】

- 应该在启用 AO 通道之后，绑定 AO 通道之前，调用此接口启用重采样功能。
- 允许重复启用重采样功能，但必须保证后配置的重采样输入采样率与之前配置的重采样输入采样率一样。
- 在禁用 AO 通道后，如果重新启用 AO 通道，并使用重采样功能，需调用此接口重新启用重采样。
- AO 重采样的输入采样率必须与 AO 设备属性配置的采样率不相同。

【举例】

以 ADEC 到 AO 的解码回放 8K 到 32K 重采样为例，配置如下：

```
/* dev attr of ao */
    AUDIO_SAMPLE_RATE_E enInSampleRate;
    stAioAttr.u32ChnCnt = 2;
    stAioAttr.enBitwidth = AUDIO_BIT_WIDTH_16;
    stAioAttr.enSamplerate = AUDIO_SAMPLE_RATE_32000;
    stAioAttr.enSoundmode = AUDIO_SOUND_MODE_MONO;
    stAioAttr.u32EXFlag = 1;
    stAioAttr.u32FrmNum = 30;
    stAioAttr.u32PtNumPerFrm = 320*4;
    enInSampleRate = AUDIO_SAMPLE_RATE_8000;
    s32Ret = HI_MPI_AO_EnableReSmp(AoDev, AoChn, enInSampleRate);
    if (HI_SUCCESS != s32Ret)
    {
```

```
        printf("func(%s) line(%d): failed, s32Ret:0x%x\n", __FUNCTION__,
__LINE__, s32Ret);
        return s32Ret;
}
```

HI_MPI_AO_DisableReSmp

【描述】

禁用 AO 重采样。

【语法】

```
HI_S32 HI_MPI_AO_DisableReSmp(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AoChn	音频输出通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a libupvqe.a libhive_RES.so libhive_common.so

【注意】

不再使用 AO 重采样功能的话，应该调用此接口将其禁用。

【举例】

无。

HI_MPI_AO_PauseChn

【描述】

暂停 AO 通道。

【语法】

```
HI_S32 HI_MPI_AO_PauseChn(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AoChn	音频输出通道号。 支持的通道范围由 AO 设备属性中的最大通道个数 u32ChnCnt 与声道模式 enSoundmode 决定。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

- AO 通道暂停后，如果绑定的 ADEC 通道继续向此通道发送音频帧数据，发送的音频帧数据将会被阻塞；而如果绑定的 AI 通道继续向此通道发送音频帧数据，在通道缓冲未满的情况下则将音频帧放入缓冲区，在满的情况下则将音频帧丢弃。
- AO 通道为禁用状态时，不允许调用此接口暂停 AO 通道。

【举例】

无。

HI_MPI_AO_ResumeChn

【描述】

恢复 AO 通道。

【语法】

```
HI_S32 HI_MPI_AO_ResumeChn(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AoChn	音频输出通道号。 支持的通道范围由 AO 设备属性中的最大通道个数 u32ChnCnt 与声道模式 enSoundmode 决定。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

- AO 通道暂停后可以通过调用此接口重新恢复。
- AO 通道为暂停状态或使能状态下，调用此接口返回成功；否则调用将返回错误。

【举例】

无。

HI_MPI_AO_ClearChnBuf

【描述】

清除 AO 通道中当前的音频数据缓存。

【语法】

```
HI_S32 HI_MPI_AO_ClearChnBuf(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入

参数名称	描述	输入/输出
AoChn	AO 通道号。 支持的通道范围由 AO 设备属性中的最大通道个数 u32ChnCnt 与声道模式 enSoundmode 决定。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

- 在 AO 通道成功启用后再调用此接口。
- 为完全清除解码回放通路上所有缓存数据，此接口还应该与 HI_MPI_ADEC_ClearChnBuf 接口配合使用。

【举例】

无。

HI_MPI_AO_QueryChnStat

【描述】

查询 AO 通道中当前的音频数据缓存状态。

【语法】

```
HI_S32 HI_MPI_AO_QueryChnStat(AUDIO_DEV AoDevId, AO_CHN AoChn,
AO_CHN_STATE_S *pstStatus);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AoChn	AO 通道号。 支持的通道范围由 AO 设备属性中的最大通道个数	输入

参数名称	描述	输入/输出
	u32ChnCnt 与声道模式 enSoundmode 决定。	
pstStatus	缓存状态结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

在 AO 通道成功启用后再调用此接口。

【举例】

无。

HI_MPI_AO_SetTrackMode

【描述】

设置 AO 设备声道模式。

【语法】

```
HI_S32 HI_MPI_AO_SetTrackMode(AUDIO_DEV AoDevId, AUDIO_TRACK_MODE_E
enTrackMode);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入
enTrackMode	音频设备声道模式。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

- 在 AO 设备成功启用后再调用此接口。
- AO 设备工作在 I^2S 模式时，支持设置声道模式，PCM 模式下不支持。

【举例】

```
HI_S32 s32Ret;
AUDIO_DEV AoDev = 0;
AUDIO_TRACK_MODE_E enTrackMode = AUDIO_TRACK_NORMAL;
AUDIO_TRACK_MODE_E temp;
s32Ret = HI_MPI_AO_SetTrackMode(AoDev, enTrackMode);
if (HI_SUCCESS != s32Ret)
{
printf("Ao set track mode failure! AoDev: %d, enTrackMode: %d, s32Ret:
0x%x.\n", AoDev, enTrackMode, s32Ret);
    return s32Ret;
 }
 s32Ret = HI_MPI_AO_GetTrackMode(AoDev, &temp);
 if (HI_SUCCESS != s32Ret)
 {
   printf("Ao get track mode failure! AoDev: %d, s32Ret: 0x%x.\n", AoDev,
s32Ret);
    return s32Ret;
}
```

HI_MPI_AO_GetTrackMode

【描述】

获取 AO 设备声道模式。

【语法】

```
HI_S32 HI_MPI_AO_GetTrackMode(AUDIO_DEV AoDevId, AUDIO_TRACK_MODE_E
*penTrackMode);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入
penTrackMode	音频设备声道模式指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

- 在 AO 设备成功启用后再调用此接口。
- AO 设备工作在 I2S 模式时，支持获取声道模式，PCM 模式下不支持。

【举例】

无。

HI_MPI_AO_SetVolume

【描述】

设置 AO 设备音量大小。

【语法】

```
HI_S32 HI_MPI_AO_SetVolume(AUDIO_DEV AoDevId, HI_S32 s32VolumeDb);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入
s32VolumeDb	音频设备音量大小（以 Db 为单位）。 取值范围：[-121,6]	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

在 AO 设备成功启用后再调用此接口。

【举例】

无。

HI_MPI_AO_GetVolume

【描述】

获取 AO 设备音量大小。

【语法】

```
HI_S32 HI_MPI_AO_GetVolume(AUDIO_DEV AoDevId, HI_S32 *ps32VolumeDb);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入
ps32VolumeDb	音频设备音量大小指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

在 AO 设备成功启用后再调用此接口。

【举例】

无。

HI_MPI_AO_SetMute

【描述】

设置 AO 设备静音状态。

【语法】

```
HI_S32 HI_MPI_AO_SetMute(AUDIO_DEV AoDevId, HI_BOOL bEnable, AUDIO_FADE_S
*pstFade);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入
bEnable	音频设备是否启用静音。 HI_TRUE：启用静音功能； HI_FALSE：关闭静音功能。	输入
pstFade	淡入淡出结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

- 在 AO 设备成功启用后再调用此接口。

- 调用此接口时，用户可以选择是否使用淡入淡出功能，如果不使用淡入淡出则将结构体指针赋为空即可。

【举例】

无。

HI_MPI_AO_GetMute

【描述】

获取 AO 设备静音状态。

【语法】

```
HI_S32 HI_MPI_AO_GetMute(AUDIO_DEV AoDevId, HI_BOOL *pbEnable,
AUDIO_FADE_S *pstFade);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入
pbEnable	音频设备静音状态指针。	输出
pstFade	淡入淡出结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

在 AO 设备成功启用后再调用此接口。

【举例】

无。

HI_MPI_AO_GetFd

【描述】

获取音频输出通道号对应的设备文件句柄。

【语法】

```
HI_S32 HI_MPI_AO_GetFd(AUDIO_DEV AoDevId, AO_CHN AoChn)
```

【参数】

参数名称	描述	输入/输出
AoDevId	AO 设备号。 取值范围：请参见表 9-6。	输入
AoChn	AO 通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
正数值	有效返回值。
非正数值	无效返回值。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

无。

【举例】

```
HI_S32 s32AoFd;
HI_S32 s32ret;
AUDIO_DEV AoDevId = 0;
AO_CHN AoChn = 0;

/* first enable ao device */

s32AoFd = HI_MPI_AO_GetFd(AoDevId, AoChn);
if(s32AoFd <= 0)
{
    return HI_FAILURE;
```

```
}
```

HI_MPI_AO_SaveFile

【描述】

开启音频输出保存文件功能。

【语法】

```
HI_S32 HI_MPI_AO_SaveFile(AUDIO_DEV AoDevId, AO_CHN AoChn,
AUDIO_SAVE_FILE_INFO_S* pstSaveFileInfo);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AoChn	音频输入通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入
pstSaveFileInfo	音频保存文件属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

没有使能 VQE 功能时或者是 AI-AO 系统绑定模式下，使用该接口 dump AO 数据无效。调用后会在指定目录下写出两个指定大小文件。Sin.pcm 为 VQE 处理前的输入帧，Sou.pcm 为 VQE 处理后的输出帧。

【相关主题】

无。

HI_MPI_AO_QueryFileStatus

【描述】

查询音频输出通道是否处于存文件的状态。

【语法】

```
HI_S32 HI_MPI_AO_QueryFileStatus(AUDIO_DEV AoDevId, AO_CHN AoChn,
AUDIO_FILE_STATUS_S* pstFileStatus);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。 取值范围：请参见表 9-6。	输入
AoChn	音频输出通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入
pstFileStatus	状态属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

此接口用于查询音频输出通道是否处于存文件的状态，当用户调用 HI_MPI_AO_SaveFile 存储文件后，可调用此接口查询存储的文件是否达到了指定的大小。

- 如果 pstFileStatus 的 bSaving 为 HI_TRUE，则还没有达到指定大小。
- 如果 pstFileStatus 的 bSaving 为 HI_FALSE，则已经达到指定大小。

【相关主题】

无。

HI_MPI_AO_ClrPubAttr

【描述】

清除 AO 设备属性。

【语法】

```
HI_S32 HI_MPI_AO_ClrPubAttr(AUDIO_DEV AoDevId);
```

【参数】

参数名称	描述	输入/输出
AoDevId	AO 设备号。 取值范围：请参见表 9-6。	输入

【返回值】

返回值	描述
正数值	有效返回值。
非正数值	无效返回值。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a

【注意】

- 清除设备属性前，需要先停止设备。
- 当 AI 设备共用 AO 设备时钟，且两者设备均未使能时，AI 设备属性需要强行切换（修改后属性不满足共时钟要求：enSamplerate 相同，u32ClkChnCnt 与 enBitwidth 对应的采用位宽的乘积相等），建议切换前先使用该接口清除与 AO 的内部耦合关系。

【相关主题】

无。

HI_MPI_AO_SetVqeAttr

【描述】

设置 AO 的声音质量增强功能相关属性。

【语法】

```
HI_S32 HI_MPI_AO_SetVqeAttr(AUDIO_DEV AoDevId, AO_CHN AoChn,
AO_VQE_CONFIG_S *pstVqeConfig);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频输出设备号。 取值范围：请参见表 9-6。	输入
AoChn	音频输出通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入
pstVqeConfig	音频输出声音质量增强配置结构体指针	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a libdnvqe.a libhive_common.so libhive_HPF.so libhive_EQ.so libhive_ANR.so libhive_AGC.so

【注意】

- 启用声音质量增强功能前必须先设置相对应 AO 通道的声音质量增强功能相关属性。
- 设置 AO 的声音质量增强功能相关属性前，必须先使能对应的 AO 通道。
- 相同 AO 通道的声音质量增强功能不支持动态设置属性，重新设置 AO 通道的声音质量增强功能相关属性时，需要先关闭 AO 通道的声音质量功能，再设置 AO 通道的声音质量增强功能相关属性。
- 在设置声音质量增强功能属性时，可通过配置相应的声音质量增强功能属性来选择使能其中的部分功能。

【举例】

无。

HI_MPI_AO_GetVqeAttr

【描述】

获取 AO 的声音质量增强功能相关属性。

【语法】

```
HI_S32 HI_MPI_AO_GetVqeAttr(AUDIO_DEV AoDevId, AO_CHN AoChn,
AO_VQE_CONFIG_S *pstVqeConfig);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频输出设备号。 取值范围：请参见表 9-6。	输入
AoChn	音频输出通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入
pstVqeConfig	音频输出声音质量增强配置结构体指针	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a libdnvqe.a libhive_common.so libhive_HPF.so libhive_EQ.so libhive_ANR.so libhive_AGC.so

【注意】

获取声音质量增强功能相关属性前必须先设置相对应 AO 通道的声音质量增强功能相关属性。

【举例】

无。

HI_MPI_AO_EnableVqe

【描述】

使能 AO 的声音质量增强功能。

【语法】

```
HI_S32 HI_MPI_AO_EnableVqe(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频输出设备号。 取值范围：请参见表 9-6。	输入

参数名称	描述	输入/输出
AoChn	音频输出通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a libdnvqe.a libhive_common.so libhive_HPF.so libhive_EQ.so libhive_ANR.so libhive_AGC.so

【注意】

- 启用声音质量增强功能前必须先启用相对应的 AO 通道。
- 多次使能相同 AO 通道的声音质量增强功能时，返回成功。
- 禁用 AO 通道后，如果重新启用 AO 通道，并使用声音质量增强功能，需调用此接口重新启用声音质量增强功能。

【举例】

无。

HI_MPI_AO_DisableVqe

【描述】

禁用 AO 的声音质量增强功能。

【语法】

```
HI_S32 HI_MPI_AO_DisableVqe(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频输出设备号。 取值范围：请参见表 9-6。	输入
AoChn	音频输出通道号。 取值范围：[0, AIO_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_ao.h
- 库文件：libmpi.a libdnvqe.a libhive_common.so libhive_HPF.so libhive_EQ.so libhive_ANR.so libhive_AGC.so

【注意】

- 不再使用 AO 声音质量增强功能时，应该调用此接口将其禁用。
- 多次禁用相同 AO 通道的声音质量增强功能，返回成功。

【举例】

无。

9.3.3 音频编码

音频编码主要实现创建编码通道、发送音频帧编码及获取编码码流等功能。

该功能模块提供以下 MPI：

- HI_MPI_AENC_CreateChn：创建音频编码通道。
- HI_MPI_AENC_DestroyChn：销毁音频编码通道。
- HI_MPI_AENC_SendFrame：发送音频编码音频帧
- HI_MPI_AENC_GetStream：获取音频编码码流。
- HI_MPI_AENC_ReleaseStream：释放音频编码码流。
- HI_MPI_AENC_GetFd：获取音频编码通道号对应的设备文件句柄.
- HI_MPI_AENC_RegeisterEncoder：注册编码器。
- HI_MPI_AENC_UnRegisterEncoder：注销编码器。
- HI_MPI_AENC_SaveFile：开启音频编码之前通道存文件功能。
- HI_MPI_AENC_QueryFileStatus：查询音频编码通道是否处于存文件的状态。

HI_MPI_AENC_CreateChn

【描述】

创建音频编码通道。

【语法】

```
HI_S32 HI_MPI_AENC_CreateChn(AENC_CHN AeChn, const AENC_CHN_ATTR_S
*pstAttr);
```

【参数】

参数名称	描述	输入/输出
AeChn	通道号。 取值范围：[0, AENC_MAX_CHN_NUM)。	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，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、hi_comm_aenc.h、mpi_aenc.h
- 库文件：libmpi.a lib_VoiceEngine.a libupvqe.a

【注意】

- 协议类型指定该通道的编码协议，目前支持 G711、G726、ADPCM，具体内容如表 9-4 所示。
- 表 9-4 中列举的编解码协议只支持 16bit 线性 PCM 音频数据处理，如果输入的是 8bit 采样精度的数据，AENC 内部会将其扩展为 16bit；建议配置 AI 设备公共属性时将扩展标志置为 1，使得 AI 数据由 8bit 自动扩展到 16bit。
- 海思语音帧结构如表 9-5 所示。
- 音频编码的部分属性需要与输入的音频数据属性相匹配，例如采样率、帧长（每帧采样点数目）等。
- buffer 大小以帧为单位，取值范围是[2, MAX_AUDIO_FRAME_NUM]，建议配置为 10 以上，过小的 buffer 配置可能导致丢帧等异常。
- 在通道闲置时才能使用此接口，如果通道已经被创建，则返回通道已经创建的错误。

【举例】

```
HI_S32 s32ret;
AENC_CHN_ATTR_S stAencAttr;
ADEC_ATTR_ADPCM_S stAdpcmAenc;
AI_DEV AiDev = 0;
AI_CHN AiChn = 0;
```

```
AENC_CHN AencChn = 0;
AUDIO_FRAME_S stAudioFrm;
AUDIO_STREAM_S stAudioStream;
MPP_CHN_S stSrcChn, stDestChn;

stAencAttr.enType = PT_ADPCMA;/* ADPCM */
stAencAttr.u32BufSize = 8;
stAencAttr.pValue = &stAdpcmAenc;
stAdpcmAenc.enADPCMType = ADPCM_TYPE_DVI4;

/* create aenc chn*/
s32ret = HI_MPI_AENC_CreateChn(AencChn, &stAencAttr);
if (HI_SUCCESS != s32ret)
{
    printf("create aenc chn %d err:0x%x\n", AencChn,s32ret);
    return s32ret;
}

/* bind AENC to AI channel  */
stSrcChn.enModId   = HI_ID_AI;
stSrcChn.s32DevId  = AiDev;
stSrcChn.s32ChnId  = AiChn;

stDestChn.enModId  = HI_ID_AENC;
stDestChn.s32DevId = 0;
stDestChn.s32ChnId = AencChn;
s32Ret = HI_MPI_SYS_Bind(&stSrcChn, &stDestChn);
if (s32Ret != HI_SUCCESS)
{
    return s32Ret;
}

/* get stream from aenc chn */
s32ret = HI_MPI_AENC_GetStream(AencChn, &stAudioStream);
if (HI_SUCCESS != s32ret )
{
    printf("get stream from aenc chn %d fail \n", AencChn);
    return s32ret;
}

/* deal with audio stream */

/* release audio stream */
s32ret = HI_MPI_AENC_ReleaseStream(AencChn, &stAudioStream);
```

```
if (HI_SUCCESS != s32ret )
{
    return s32ret;
}

/* destroy aenc chn */
s32ret = HI_MPI_AENC_DestroyChn(AencChn);
if (HI_SUCCESS != s32ret )
{
    return s32ret;
}
```

HI_MPI_AENC_DestroyChn

【描述】

销毁音频编码通道。

【语法】

```
HI_S32 HI_MPI_AENC_DestroyChn(AENC_CHN AeChn);
```

【参数】

参数名称	描述	输入/输出
AeChn	通道号。 取值范围：[0, AENC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、hi_comm_aenc.h、mpi_aenc.h
- 库文件：libmpi.a lib_VoiceEngine.a libupvqe.a

【注意】

- 通道未创建的情况下调用此接口会返回成功。
- 如果正在获取/释放码流或者发送帧时销毁该通道，则会返回失败，用户同步处理时需要注意。

【举例】

请参见 HI_MPI_AENC_CreateChn 的举例。

HI_MPI_AENC_SendFrame

【描述】

发送音频编码音频帧。

【语法】

```
HI_S32 HI_MPI_AENC_SendFrame(AENC_CHN AeChn, const AUDIO_FRAME_S *pstFrm,
const AEC_FRAME_S *pstAecFrm);
```

【参数】

参数名称	描述	输入/输出
AeChn	通道号。 取值范围：[0, AENC_MAX_CHN_NUM)。	输入
pstFrm	音频帧结构体指针。	输入
pstAecFrm	回声抵消参考帧结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、hi_comm_aenc.h、mpi_aenc.h
- 库文件：libmpi.a libupvqe.a

【注意】

- 如果不需要回声抵消，pstAecFrm 可置为 NULL。
- 音频编码发送码流是非阻塞接口，如果音频码流缓存满，则直接返回失败。
- 该接口用于用户主动发送音频帧进行编码，如果 AENC 通道已经通过系统绑定（HI_MPI_SYS_Bind）接口与 AI 绑定，不需要也不建议调此接口。
- 调用该接口发送音频编码音频帧时，必须先创建对应的编码通道。

【举例】

无。

HI_MPI_AENC_GetStream

【描述】

获取编码后码流。

【语法】

```
HI_S32 HI_MPI_AENC_GetStream(AENC_CHN AeChn, AUDIO_STREAM_S *pstStream,
HI_S32 s32MilliSec);
```

【参数】

参数名称	描述	输入/输出
AeChn	通道号。 取值范围：[0, AENC_MAX_CHN_NUM)。	输入
pstStream	获取的音频码流。	输出
s32MilliSec	获取数据的超时时间 -1 表示阻塞模式，无数据时一直等待； 0 表示非阻塞模式，无数据时则报错返回； >0 表示阻塞 s32MilliSec 毫秒，超时则报错返回。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、hi_comm_aenc.h、mpi_aenc.h
- 库文件：libmpi.a

【注意】

- 必须创建通道后才可能获取码流，否则直接返回失败，如果在获取码流过程中销毁通道则会立刻返回失败。
- s32MilliSec 的值必须大于等于-1，等于-1 时采用阻塞模式获取数据，等于 0 时采用非阻塞模式获取数据，大于 0 时，阻塞 s32MilliSec 毫秒后，没有数据则返回超时并报错。
- 直接获取 AI 原始音频数据的方法

  创建一路 AENC 通道，编码协议类型设置为 PT_LPCM，绑定 AI 通道后，从此 AENC 通道获取的音频数据即 AI 原始数据。

【举例】

请参见 HI_MPI_AENC_CreateChn 的举例。

HI_MPI_AENC_ReleaseStream

【描述】

释放从音频编码通道获取的码流。

【语法】

```
HI_S32 HI_MPI_AENC_ReleaseStream(AENC_CHN AeChn, const AUDIO_STREAM_S
*pstStream);
```

【参数】

参数名称	描述	输入/输出
AeChn	通道号。 取值范围：[0, AENC_MAX_CHN_NUM)。	输入
pstStream	获取的码流指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、hi_comm_aenc.h、mpi_aenc.h
- 库文件：libmpi.a lib_VoiceEngine.a

【注意】

- 码流最好能够在使用完之后立即释放，如果不及时释放，会导致编码过程阻塞。
- 释放的码流必须是从该通道获取的码流，不得对码流信息结构体进行任何修改，否则会导致码流不能释放，使此码流 buffer 丢失，甚至导致程序异常。
- 释放码流时必须保证通道已经被创建，否则直接返回失败，如果在释放码流过程中销毁通道则会立刻返回失败。

【举例】

请参见 HI_MPI_AENC_CreateChn 的举例。

HI_MPI_AENC_GetFd

【描述】

获取音频编码通道号对应的设备文件句柄。

【语法】

```
HI_S32 HI_MPI_AENC_GetFd(AENC_CHN AeChn)
```

【参数】

参数名称	描述	输入/输出
AeChn	AENC 通道号。 取值范围：[0, AENC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
正数值	有效返回值。
非正数值	无效返回值。

【需求】

- 头文件：hi_comm_aio.h、hi_comm_aenc.h、mpi_aenc.h
- 库文件：libmpi.a

【注意】

无。

HI_MPI_AENC_RegeisterEncoder

【描述】

注册编码器。

【语法】

```
HI_S32 HI_MPI_AENC_RegeisterEncoder(HI_S32 *ps32Handle, 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，其值为错误码。

【需求】

- 头文件：hi_comm_aenc.h、mpi_aenc.h
- 库文件：libmpi.a

【注意】

- 用户通过传入编码器属性结构体，向 AENC 模块注册一个编码器，并返回注册句柄，用户可以最后通过注册句柄来注销该编码器。
- AENC 模块最大可注册 20 个编码器，且自身已注册 LPCM、G711.a、G711.u、G726、ADPCM 五个编码器。
- 同一种编码协议不允许重复注册编码器，例如假如已注册 G726 编码器，不允许另外再注册一个 G726 编码器。

【举例】

无。

HI_MPI_AENC_UnRegisterEncoder

【描述】

注销编码器。

【语法】

```
HI_S32 HI_MPI_AENC_UnRegisterEncoder(HI_S32 s32Handle)
```

【参数】

参数名称	描述	输入/输出
s32Handle	注册句柄（注册编码器时获得的句柄）。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aenc.h、mpi_aenc.h
- 库文件：libmpi.a

【注意】

通常不需要注销编码器。

【举例】

无。

HI_MPI_AENC_SaveFile

【描述】

开启音频编码之前通道存文件功能。

【语法】

```
HI_S32 HI_MPI_AENC_SaveFile(AENC_CHN AeChn, AUDIO_SAVE_FILE_INFO_S
*pstSaveFileInfo);
```

【参数】

参数名称	描述	输入/输出
AeChn	音频编码通道号。 取值范围：[0, AENC_MAX_CHN_NUM)。	输入
pstSaveFileInfo	音频保存文件属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_aenc.h
- 库文件：libmpi.a

【注意】

此接口仅用于 AI-AENC 系统绑定模式下 dump 编码通道在编码前的 VQE 处理前后的音频数据文件，AI-AENC 非系统绑定模式下或者是没有使能 VQE 功能时，使用该接

口 dump 编码通道数据，将不生效。调用后会在指定目录下写出三个指定大小文件。Sin.pcm 为 VQE 处理前的输入帧，Rin.pcm 为 VQE 处理前的回声抵消参考帧，Sou.pcm 为 VQE 处理后的输出帧。

【相关主题】

无。

HI_MPI_AENC_QueryFileStatus

【描述】

查询频编码通道是否处于存文件的状态。

【语法】

```
HI_S32 HI_MPI_AENC_QueryFileStatus(AENC_CHN AeChn, AUDIO_FILE_STATUS_S*
pstFileStatus);
```

【参数】

参数名称	描述	输入/输出
AeChn	音频编码通道号。 取值范围：[0, AENC_MAX_CHN_NUM)	输入
pstFileStatus	状态属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、mpi_aenc.h
- 库文件：libmpi.a

【注意】

此接口用于查询音频编码通道是否处于存文件的状态，当用户调用 HI_MPI_AENC_SaveFile 存储文件后，可调用此接口查询存储的文件是否达到了指定的大小。

- 如果 pstFileStatus 的 bSaving 为 HI_TRUE，则还没有达到指定大小。
- 如果 pstFileStatus 的 bSaving 为 HI_FALSE，则已经达到指定大小。

【相关主题】

无。

9.3.4 音频解码

音频解码主要实现创建解码通道、发送音频码流解码及获取解码后音频帧等功能。

该功能模块提供以下 MPI：

- HI_MPI_ADEC_CreateChn：创建音频解码通道。
- HI_MPI_ADEC_DestroyChn：销毁音频解码通道。
- HI_MPI_ADEC_SendStream：发送音频码流到音频解码通道。
- HI_MPI_ADEC_ClearChnBuf：清除 ADEC 通道中当前的音频数据缓存。
- HI_MPI_ADEC_RegeisterDecoder：注册解码器。
- HI_MPI_ADEC_UnRegisterDecoder：注销解码器。
- HI_MPI_ADEC_GetFrame：获取音频解码帧数据。
- HI_MPI_ADEC_ReleaseFrame：释放音频解码帧数据。
- HI_MPI_ADEC_SendEndOfStream：向解码器发送码流结束标识符，并清除码流 buffer。

HI_MPI_ADEC_CreateChn

【描述】

创建音频解码通道。

【语法】

```
HI_S32 HI_MPI_ADEC_CreateChn(ADEC_CHN AdChn, ADEC_CHN_ATTR_S *pstAttr);
```

【参数】

参数名称	描述	输入/输出
AdChn	通道号。 取值范围：[0, ADEC_MAX_CHN_NUM)。	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，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、hi_comm_adec.h、mpi_adec.h
- 库文件：libmpi.a lib_VoiceEngine.a

【注意】

- 协议类型指定了该通道的解码协议，目前支持 G711、G726、ADPCM。
- 音频解码的部分属性需要与输出设备属性相匹配，例如采样率、帧长（每帧采样点数目）等。
- buffer 大小以帧为单位，取值范围是[2, MAX_AUDIO_FRAME_NUM]，建议配置为 10 以上，过小的 buffer 配置可能导致丢帧等异常。
- 在通道未创建前（或销毁后）才能使用此接口，如果通道已经被创建，则返回通道已经创建。

【举例】

```
HI_S32 s32ret;
ADEC_CHN_ATTR_S stAdecAttr;
ADEC_ATTR_ADPCM_S stAdpcm;
ADEC_CHN AdChn = 0;
AUDIO_STREAM_S stAudioStream;
AUDIO_DEV AoDev = 0;
AO_CHN AoChn = 0;
MPP_CHN_S stSrcChn;
MPP_CHN_S stDestChn;

stAdecAttr.enType = PT_ADPCMA;
stAdecAttr.u32BufSize = 8;
stAdecAttr.enMode = ADEC_MODE_STREAM;
stAdecAttr.pValue = &stAdpcm;
stAdpcm.enADPCMType = ADPCM_TYPE_DVI4;

/* create adec chn*/
s32ret = HI_MPI_ADEC_CreateChn(AdChn, &stAdecAttr);
if (s32ret)
{
    printf("create adec chn %d err:0x%x\n", AdChn,s32ret);
    return s32ret;
}

/* bind ADEC to AO channel*/
stSrcChn.enModId = HI_ID_ADEC;
stSrcChn.s32DevId = 0;
stSrcChn.s32ChnId = AdChn;
stDestChn.enModId = HI_ID_AO;
stDestChn.s32DevId = AoDev;
stDestChn.s32ChnId = AoChn;
```

```
s32ret = HI_MPI_SYS_Bind(&stSrcChn, &stDestChn);
if (s32ret)
{
    printf("bind adec chn %d to ao(%d, %d) error:0x%x\n", AdChn, AoDev,
AoChn, s32ret);
    return s32ret;
}

/* get audio stream from network or file*/

/* send audio stream to adec chn */
s32ret = HI_MPI_ADEC_SendStream(AdChn, &stAudioStream);
if (s32ret)
{
    printf("send stream to adec fail\n");
    return s32ret;
}
/* destroy adec chn */
s32ret = HI_MPI_ADEC_DestroyChn(AdChn);
if (HI_SUCCESS != s32ret )
{
    return s32ret;
}
```

HI_MPI_ADEC_DestroyChn

【描述】

销毁音频解码通道。

【语法】

```
HI_S32 HI_MPI_ADEC_DestroyChn(ADEC_CHN AdChn);
```

【参数】

参数名称	描述	输入/输出
AdChn	通道号。 取值范围：[0, ADEC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、hi_comm_adec.h、mpi_adec.h
- 库文件：libmpi.a lib_VoiceEngine.a

【注意】

- 通道未创建的情况下调用此接口会返回成功。
- 如果正在获取/释放码流或者发送帧，销毁该通道则这些操作都会立即返回失败。

【举例】

请参见 HI_MPI_ADEC_CreateChn 的举例。

【相关主题】

无。

HI_MPI_ADEC_SendStream

【描述】

向音频解码通道发送码流。

【语法】

```
HI_S32 HI_MPI_ADEC_SendStream(ADEC_CHN AdChn, const AUDIO_STREAM_S
*pstStream, HI_BOOL bBlock);
```

【参数】

参数名称	描述	输入/输出
AdChn	通道号。 取值范围：[0, ADEC_MAX_CHN_NUM)。	输入
pstStream	音频码流。	输入
bBlock	阻塞标识。 HI_TRUE：阻塞。 HI_FALSE：非阻塞。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、hi_comm_adec.h、mpi_adec.h
- 库文件：libmpi.a lib_VoiceEngine.a

【注意】

- 创建解码通道时可以指定解码方式为 pack 方式或 stream 方式。
  - pack 方式用于确定码流包为一帧的情况下，比如从 AENC 直接获取的码流，从文件读取确切知道一帧边界（语音编码码流长度固定，很容易确定边界）的码流，效率较高。
  - stream 方式用于不确定码流包为一帧的情况下，效率较低，且可能会有延迟。
- LPCM 解码只支持 pack 方式；其他音频格式的解码两种方式都支持。
- 发送数据时必须保证通道已经被创建，否则直接返回失败，如果在送数据过程中销毁通道则会立刻返回失败。
- 支持阻塞或非阻塞方式发送码流。
- 当阻塞方式发送码流时，如果用于缓存解码后的音频帧的 Buffer 满，则此接口调用会被阻塞，直至解码后的音频帧数据被取走，或 ADEC 通道被销毁。
- 确保发送给 ADEC 通道的码流数据的正确性，否则可能引起解码器异常退出。

【举例】

请参见 HI_MPI_ADEC_CreateChn 的举例。

【相关主题】

无。

HI_MPI_ADEC_ClearChnBuf

【描述】

清除 ADEC 通道中当前的音频数据缓存。

【语法】

```
HI_S32 HI_MPI_ADEC_ClearChnBuf(ADEC_CHN AdChn);
```

【参数】

参数名称	描述	输入/输出
AdChn	通道号。 取值范围：[0, ADEC_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_aio.h、hi_comm_adec.h、mpi_adec.h
- 库文件：libmpi.a

【注意】

- 要求解码通道已经被创建，如果通道未被创建则返回通道不存在错误码。
- 使用本接口时，不建议使用 stream 方式解码。使用 stream 方式解码进行清除缓存操作时，用户需要确保清除完缓存后，发送给解码器的数据必须是完整的一帧码流，否则可能导致解码器不能正常工作。
- 无论是否使用 stream 方式解码，都要确保送数据解码的操作和清除缓存的操作之间的同步。

【举例】

无。

HI_MPI_ADEC_RegeisterDecoder

【描述】

注册解码器。

【语法】

```
HI_S32 HI_MPI_ADEC_RegeisterDecoder(HI_S32 *ps32Handle, 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，其值为错误码。

【需求】

- 头文件：hi_comm_adec.h、mpi_adec.h
- 库文件：libmpi.a

【注意】

- 用户通过传入解码器属性结构体，向 ADEC 模块注册一个解码器，并返回注册句柄，用户可以最后通过注册句柄来注销该解码器。
- ADEC 模块最大可注册 20 个解码器，且自身已注册 LPCM、G711a、G711u、G726、ADPCM 五个解码器。
- 同一种解码协议不允许重复注册解码器，例如假如已注册 G726 解码器，不允许另外再注册一个 G726 解码器。

【举例】

无。

HI_MPI_ADEC_UnRegisterDecoder

【描述】

注销解码器。

【语法】

```
HI_S32 HI_MPI_ADEC_UnRegisterDecoder(HI_S32 s32Handle);
```

【参数】

参数名称	描述	输入/输出
s32Handle	注册句柄（注册解码器时获得的句柄）。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，其值为错误码。

【需求】

- 头文件：hi_comm_adec.h、mpi_adec.h
- 库文件：libmpi.a

【注意】

通常不需要注销解码器。

【举例】

无

HI_MPI_ADEC_GetFrame

【描述】

获取音频解码帧数据。

【语法】

```
HI_S32 HI_MPI_ADEC_GetFrame(ADEC_CHN AdChn, AUDIO_FRAME_INFO_S
*pstFrmInfo, HI_BOOL bBlock);
```

【参数】

参数名称	描述	输入/输出
AdChn	音频解码通道。	输入
pstFrmInfo	音频帧数据结构体。	输出
bBlock	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，其值为错误码。

【需求】

- 头文件：hi_comm_adec.h、mpi_adec.h
- 库文件：libmpi.a

【注意】

- 必须在 ADEC 通道创建之后调用。
- 使用本接口获取解码帧数据时，建议发送码流时按帧发送。
- 使用本接口获取音频帧数据时，如果发送码流按流发送，请务必保证获取解码帧数据的及时性，否则会有异常。

- 使用本接口获取音频数据时，请解除 ADEC 与 AO 的绑定关系，否则获取到的帧是不连续的。

【举例】

无。

【相关主题】

无。

HI_MPI_ADEC_ReleaseFrame

【描述】

释放获取到的音频解码帧数据。

【语法】

```
HI_S32 HI_MPI_ADEC_ReleaseFrame(ADEC_CHN AdChn, AUDIO_FRAME_INFO_S
*pstFrmInfo);
```

【参数】

参数名称	描述	输入/输出
AdChn	音频解码通道。	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，其值为错误码。

【需求】

- 头文件：hi_comm_adec.h、mpi_adec.h
- 库文件：libmpi.a

【注意】

- 必须在 ADEC 通道创建之后调用。
- 本接口必须与接口 HI_MPI_ADEC_GetFrame 配合使用。

【举例】

无。

【相关主题】

无。

HI_MPI_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】

```
HI_S32 HI_MPI_ADEC_SendEndOfStream (ADEC_CHN AdChn, HI_BOOL bInstant);
```

【参数】

参数名称	描述	输入/输出
AdChn	解码通道号。	输入
bInstant	是否立即清除解码器内部的缓存数据。 取值范围： HI_FALSE：延时清除。不会立即清除解码器内部的缓存数据，解码会继续进行，直到剩余 buffer 不足一帧数据时进行清除操作。 HI_TRUE：立即清除解码器内部缓存数据。	输入

【返回值】

返回值	描述
0	清除码流 buffer 成功。
非 0	返回清除码流 buffer 失败的错误码。

【需求】

- 头文件：mpi_adec.h
- 库文件：libmpi.a

【注意】

无

【举例】

无

9.3.5 内置 Audio Codec

内置 Audio Codec 主要通过 ioctl 提供对硬件设备的操作。在提供的 ioctl 的 cmd 中，有些 cmd 用户可以不需要调用，直接使用模块加载时的默认配置即可。ioctl 调用实现的是对内置 Audio Codec 寄存器的读写。

9.3.5.1 内置 Audio Codec 标准功能 cmd

- ACODEC_SOFT_RESET_CTRL：将 Codec 恢复为默认设置。
- ACODEC_SET_I2S1_FS：设置 I^2S1 接口采样率。
- ACODEC_SET_INPUT_VOL：输入总音量控制。
- ACODEC_GET_INPUT_VOL：获取输入总音量。
- ACODEC_SET_OUTPUT_VOL：输出总音量控制。
- ACODEC_GET_OUTPUT_VOL：获取输出总音量。
- ACODEC_SET_MIXER_MIC：输入声道选择。
- ACODEC_SET_GAIN_MICL：左声道输入的模拟增益控制。
- ACODEC_SET_GAIN_MICR：右声道输入的模拟增益控制。
- ACODEC_SET_DACL_VOL：左声道输出音量控制。
- ACODEC_SET_DACR_VOL：右声道输出音量控制。
- ACODEC_SET_ADCL_VOL：左声道输入音量控制。
- ACODEC_SET_ADCR_VOL：右声道输入音量控制。
- ACODEC_SET_MICL_MUTE：左声道输入静音控制。
- ACODEC_SET_MICR_MUTE：右声道输入静音控制。
- ACODEC_SET_DACL_MUTE：左声道输出静音控制。
- ACODEC_SET_DACR_MUTE：右声道输出静音控制。
- ACODEC_DAC_SOFT_MUTE：输出软静音控制。
- ACODEC_DAC_SOFT_UNMUTE：输出软撤销静音控制。
- ACODEC_ENABLE_BOOSTL：左声道 boost 模拟增益控制。
- ACODEC_ENABLE_BOOSTR：右声道 boost 模拟增益控制。
- ACODEC_GET_GAIN_MICL：获取模拟左声道输入的增益。
- ACODEC_GET_GAIN_MICR：获取模拟右声道输入的增益。
- ACODEC_GET_DACL_VOL：获取左声道输出的音量控制。
- ACODEC_GET_DACR_VOL：获取右声道输出的音量控制。
- ACODEC_GET_ADCL_VOL：获取左声道输入的音量控制。
- ACODEC_GET_ADCR_VOL：获取右声道输入的音量控制。
- ACODEC_SET_PD_DACL：左声道输出的下电控制。
- ACODEC_SET_PD_DACR：右声道输出的下电控制。
- ACODEC_SET_PD_ADCL：左声道输入的下电控制。
- ACODEC_SET_PD_ADCR：右声道输入的下电控制。
- ACODEC_SET_PD_LINEINL：左声道 LINEIN 输入的下电控制。

- ACODEC_SET_PD_LINEINR：右声道 LINEIN 输入的下电控制。

ACODEC_SOFT_RESET_CTRL

【描述】

将 Codec 恢复为默认设置。

【语法】

```
int ioctl (int fd,
           ACODEC_SOFT_RESET_CTRL);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SOFT_RESET_CTRL	ioctl 号	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

在 AI/AO 设备都禁用时再调用该接口。

【举例】

```
if (ioctl(s32Acodec_Fd, ACODEC_SOFT_RESET_CTRL))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_I2S1_FS

【描述】

设置 I^2S1 接口采样率。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_I2S1_FS,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_I2S1_FS	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

无

【举例】

```
ACODEC_FS_E i2s_fs_sel;
i2s_fs_sel = ACODEC_FS_48000;
if (ioctl(s32Acodec_Fd, ACODEC_SET_I2S1_FS, &i2s_fs_sel))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_INPUT_VOL

【描述】

输入总音量控制。

【语法】

```
int ioctl (int fd, ACODEC_SET_INPUT_VOL, unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_INPUT_VOL	ioctl 号	输入
arg	有符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

- Hi3516A 输入音量范围为[-87, 86]，包括模拟增益和数字增益，赋值越大，音量越大。赋值为 86 时，音量最大，为 86dB，赋值为-87 时，音量最小，为静音。调节时左右声道一起生效。建议调节范围限制为[-10, 56]，此范围只调节模拟增益，这样引入的噪声最小，能够更好的保证声音质量。
- Hi3518EV200 输入音量范围为[-72, 86]，建议调节范围限制为[26, 56]。
- Hi3519V100 输入音量范围为[-78, 80]，建议调节范围限制为[19, 50]。

【举例】

```
int iVol;
iVol = 20;
if (ioctl(s32Acodec_Fd, ACODEC_SET_INPUT_VOL, &iVol))
{
    printf("ioctl err!\n");
}
```

ACODEC_GET_INPUT_VOL

【描述】

获取输入总音量。

【语法】

```
int ioctl (int fd, ACODEC_GET_INPUT_VOL, unsigned int  *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_GET_INPUT_VOL	ioctl 号	输入
arg	有符号整型指针	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

无。

【举例】

```
int iVol;
if (ioctl(s32Acodec_Fd, ACODEC_GET_INPUT_VOL, &iVol))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_OUTPUT_VOL

【描述】

输出总音量控制。

【语法】

```
int ioctl (int fd, ACODEC_SET_OUTPUT_VOL, unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_OUTPUT_VOL	ioctl 号	输入
arg	有符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

输出音量范围为[-121, 6]，赋值越大，音量越大。赋值为 6 时，音量最大，为 6dB，赋值为-121 时，音量最小，为静音。调节时左右声道一起生效。此接口调整的为数字增益，建议不要赋值太大，以免引入噪声。

【举例】

```
int iVol;
iVol = 0;
if (ioctl(s32Acodec_Fd, ACODEC_SET_OUTPUT_VOL, &iVol))
{
    printf("ioctl err!\n");
}
```

ACODEC_GET_OUTPUT_VOL

【描述】

获取输出总音量。

【语法】

```
int ioctl (int fd, ACODEC_GET_OUTPUT_VOL, unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_GET_OUTPUT_VOL	ioctl 号	输入
arg	有符号整型指针	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

无。

【举例】

```
int iVol;
if (ioctl(s32Acodec_Fd,  ACODEC_GET_OUTPUT_VOL, &iVol))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_MIXER_MIC

【描述】

输入方式选择。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_MIXER_MIC,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_MIXER_MIC	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

- Hi3516A 的 arg 取值范围[0, 1]，默认选择 ACODEC_MIXER_LINEIN。
- Hi3518EV200 的 arg 取值范围[3, 4]，默认选择 ACODEC_MIXER_IN。
- Hi3519V100 的 arg 取值范围[0, 2]，默认选择 IN0 单端输入。在使用无源音频信号输入时，需适当的增大模拟增益。

【芯片差异】

芯片	描述
Hi3516A	分为 MIC 输入和 LINEIN 输入两种方式， ACODEC_MIXER_MICIN 为 MIC 输入方式， ACODEC_MIXER_LINEIN 为 LINEIN 输入方式
Hi3518EV200	分为单端输入和差分输入两种方式， ACODEC_MIXER_IN 为单端输入方式 ACODEC_MIXER_IN_D 为差分输入方式。
Hi3519V100	分为 IN0 单端输入、IN1 单端输入、IN_D 差分输入。

【举例】

```
ACODEC_MIXER_E mixer_mic_ctrl;
mixer_mic_ctrl = ACODEC_MIXER_IN0;
if (ioctl(s32Acodec_Fd, ACODEC_SET_MIXER_MIC, &mixer_mic_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_GAIN_MICL

【描述】

左声道输入的模拟增益控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_GAIN_MICL,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_GAIN_MICL	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

- 在使用无源音频信号输入时，需适当的增大模拟增益。
- Hi3516A 的 arg 取值范围为[0, 31]，按 1.5db 递增，0 对应最小增益-16.5db，31 对应最大增益 30db。
- Hi3518EV200/Hi3519V100 的模拟增益范围为[0, 16]，其中 0 到 15 按 2db 递增，0 对应 0db，15 对应最大增益 30db，而 16 对应的是-1.5db。

【举例】

```
unsigned int gain_mac;
gain_mic = 0x18;
if (ioctl(s32Acodec_Fd,  ACODEC_SET_GAIN_MICL, &gain_mic))
{
   printf("ioctl err!\n");
}
```

ACODEC_SET_GAIN_MICR

【描述】

右声道输入的模拟增益控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_GAIN_MICR,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_GAIN_MICR	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

- 在使用无源音频信号输入时，需适当的增大模拟增益。
- Hi3516A 的 arg 取值范围为[0, 31]，按 1.5db 递增，0 对应最小增益-16.5db，31 对应最大增益 30db。
- Hi3518EV200/Hi3519V100 的模拟增益范围为[0, 16]，其中 0 到 15 按 2db 递增，0 对应 0db，15 对应最大增益 30db，而 16 对应的是-1.5db。

【举例】

```
unsigned int gain_mic;
gain_mic = 0x18;
if (ioctl(s32Acodec_Fd, ACODEC_SET_GAIN_MICR, &gain_mic))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_DACL_VOL

【描述】

左声道输出音量控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_DACL_VOL,
           ACODEC_VOL_CTRL *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_DACL_VOL	ioctl 号	输入
arg	ACODEC_VOL_CTRL 结构体指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

左声道输出音量范围：[0, 127]，赋值越大，音量越小。赋值为 0 时，音量最大，为 6db，赋值为 127 时，音量最小，为静音。

【举例】

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_DACL_VOL, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_DACR_VOL

【描述】

右声道输出音量控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_DACR_VOL,
           ACODEC_VOL_CTRL *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_DACL_VOL	ioctl 号	输入
arg	ACODEC_VOL_CTRL 结构体指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

右声道输出音量范围：[0, 127]，赋值越大，音量越小。赋值为 0 时，音量最大，为 6db，赋值为 127 时，音量最小，为静音。

【举例】

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_DACR_VOL, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_ADCL_VOL

【描述】

左声道输入音量控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_ADCL_VOL,
           ACODEC_VOL_CTRL *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_ADCL_VOL	ioctl 号	输入
arg	ACODEC_VOL_CTRL 结构体指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

左声道输入音量范围：[0, 127]，赋值越大，音量越小。赋值为 0 时，音量最大，为 30db，赋值为 127 时，音量最小，为-97db。

【举例】

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_ADCL_VOL, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_ADCR_VOL

【描述】

右声道输入音量控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_ADCR_VOL,
           ACODEC_VOL_CTRL *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_ADCR_VOL	ioctl 号	输入
arg	ACODEC_VOL_CTRL 结构体指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

右声道输入音量范围：[0, 127]，赋值越大，音量越小。赋值为 0 时，音量最大，为 30db，赋值为 127 时，音量最小，为-97db。

【举例】

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_ADCR_VOL, &vol_ctrl))
{
   printf("ioctl err!\n");
}
```

ACODEC_SET_MICL_MUTE

【描述】

左声道输入静音控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_MICL_MUTE,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_MICL_MUTE	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

arg 参数范围[0, 1]。

【举例】

```
unsigned int mute_ctrl;
mute_ctrl = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_SET_MICL_MUTE, &mute_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_MICR_MUTE

【描述】

右声道输入静音控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_MICR_MUTE,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入

参数名称	描述	输入/输出
ACODEC_SET_MICR_MUTE	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

arg 参数范围[0, 1]。

【举例】

```
unsigned int mute_ctrl;
ute_ctrl = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_SET_MICR_MUTE, &mute_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_DACL_MUTE

【描述】

左声道输出静音控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_DACL_MUTE,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_DACL_MUTE	ioctl 号	输入

参数名称	描述	输入/输出
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

arg 参数范围[0, 1]。

【举例】

```
unsigned int mute_ctrl;
mute_ctrl = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_SET_DACL_MUTE, &mute_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_DACR_MUTE

【描述】

右声道输出静音控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_DACR_MUTE,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_DACR_MUTE	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

arg 参数范围[0, 1]。

【举例】

```
unsigned int  mute_ctrl;
mute_ctrl = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_SET_DACR_MUTE, &mute_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_DAC_SOFT_MUTE

【描述】

输出软静音控制。

【语法】

```
int ioctl (int fd,
           ACODEC_DAC_SOFT_MUTE,
           unsigned int  *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_DAC_SOFT_MUTE	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

- arg 参数范围[0, 1]。
- Hi3518EV200 和 Hi3519V100 不支持该接口。

【举例】

```
unsigned int  soft_mute_ctrl;
soft_mute_ctrl = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_DAC_SOFT_MUTE, &soft_mute_ctrl))
{
   printf("ioctl err!\n");
}
```

ACODEC_DAC_SOFT_UNMUTE

【描述】

输出软撤销静音控制。

【语法】

```
int ioctl (int fd,
           ACODEC_DAC_SOFT_UNMUTE,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_DAC_SOFT_UNMUTE	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

- arg 参数范围[0, 1]。
- Hi3518EV200 和 Hi3519V100 不支持该接口。

【举例】

```
unsigned int  soft_mute_ctrl;
soft_nute_ctrl = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_DAC_SOFT_UNMUTE, &soft_mute_ctrl))
{
   printf("ioctl err!\n");
}
```

ACODEC_ENABLE_BOOSTL

【描述】

左声道 boost 模拟增益控制。

【语法】

```
int ioctl (int fd,
           ACODEC_ENABLE_BOOSTL,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_ENABLE_BOOSTL	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

- arg 参数范围[0, 1]，取 0 时模拟增益增加 0db，取 1 时模拟增益增加 20db。
- Hi3516A 和 Hi3518EV200 不支持该接口。

【举例】

```
unsigned int  enable_boostl;
enable_boostl = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_ENABLE_BOOSTL, &enable_boostl))
{
   printf("ioctl err!\n");
}
```

ACODEC_ENABLE_BOOSTR

【描述】

右声道 boost 模拟增益控制。

【语法】

```
int ioctl (int fd,
           ACODEC_ENABLE_BOOSTR,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_ ENABLE_BOOSTR	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

- arg 参数范围[0, 1]，取 0 时模拟增益增加 0db，取 1 时模拟增益增加 20db。
- Hi3516A 和 Hi3518EV200 不支持该接口。

【举例】

```
unsigned int  enable_boostr;
enable_boostr = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_ENABLE_BOOSTR, &enable_boostr))
{
   printf("ioctl err!\n");
}
```

ACODEC_GET_GAIN_MICL

【描述】

获取模拟左声道输入的增益。

【语法】

```
int ioctl (int fd,
           ACODEC_GET_GAIN_MICL,
           unsigned int  *arg);
```

【参数】

参数名称	描述	输入/输出
Fd	Audio Codec 设备文件描述符	输入
ACODEC_GET_GAIN_MICL	ioctl 号	输入
arg	无符号整型指针	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

无。

【举例】

```
unsigned int gain_mic;
if (ioctl(s32Acodec_Fd,  ACODEC_GET_GAIN_MICL, &gain_mic))
{
    printf("ioctl err!\n");
}
```

ACODEC_GET_GAIN_MICR

【描述】

获取模拟右声道输入的增益。

【语法】

```
int ioctl (int fd,
           ACODEC_GET_GAIN_MICR,
           unsigned int  *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_GET_GAIN_MICR	ioctl 号	输入
arg	无符号整型指针	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

无。

【举例】

```
unsigned int gain_mic;
if (ioctl(s32Acodec_Fd, ACODEC_GET_GAIN_MICR, &gain_mic))
{
    printf("ioctl err!\n");
}
```

ACODEC_GET_DACL_VOL

【描述】

获取左声道输出的音量控制。

【语法】

```
int ioctl (int fd,
           ACODEC_GET_DACL_VOL,
           ACODEC_VOL_CTRL *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_GET_DACL_VOL	ioctl 号	输入
arg	ACODEC_VOL_CTRL 结构体指针	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

无。

【举例】

```
ACODEC_VOL_CTRL vol_ctrl;
if (ioctl(s32Acodec_Fd, ACODEC_GET_DACL_VOL, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_GET_DACR_VOL

【描述】

获取右声道输出的音量控制。

【语法】

```
int ioctl (int fd,
           ACODEC_GET_DACR_VOL,
           ACODEC_VOL_CTRL *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_GET_DACL_VOL	ioctl 号	输入
arg	ACODEC_VOL_CTRL 结构体指针	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

无。

【举例】

```
ACODEC_VOD_CTRL vol_ctrl;
if (ioctl(s32Acodec_Fd, ACODEC_GET_DACR_VOL, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_GET_ADCL_VOL

【描述】

获取左声道输入的音量控制。

【语法】

```
int ioctl (int fd,
           ACODEC_GET_ ADCL_VOL,
           ACODEC_VOL_CTRL *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_GET_ADCL_VOL	ioctl 号	输入
arg	ACODEC_VOL_CTRL 结构体指针	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

无。

【举例】

```
ACODEC_VOL_CTRL vol_ctrl;
if (ioctl(s32Acodec_Fd, ACODEC_GET_ADCL_VOL, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_GET_ADCR_VOL

【描述】

获取右声道输入的音量控制。

【语法】

```
int ioctl (int fd,
           ACODEC_GET ADCR_VOL,
           ACODEC_VOL_CTRL *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_GET_ADCR_VOL	ioctl 号	输入
arg	ACODEC_CTRL 结构体指针	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

无。

【举例】

```
ACODEC_VOL_CTRL vol_ctrl;
if (ioctl(s32Acodec_Fd, ACODEC_GET_ADCR_VOL, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_PD_DACL

【描述】

左声道输出的下电控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_PD_DACL,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_PD_DACL	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

不使用 DAC 时，可以调用此接口，将 DACL 下电。arg 参数范围[0, 1]。

【举例】

```
unsigned int pd_ctrl;
pd_ctrl = 0x01;
if (ioctl(s32Acodec_Fd, ACODEC_SET_PD_DACL, &pd_ctrl))
{
   printf("ioctl err!\n");
}
```

ACODEC_SET_PD_DACR

【描述】

右声道输出的下电控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_PD_DACR,
```

```
            unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_PD_DACR	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

不使用 DAC 时，可以调用此接口，将 DACR 下电。arg 参数范围[0, 1]。

【举例】

```
unsigned int pd_ctrl;
pd_ctrl = 0x01
if (ioctl(s32Acodec_Fd, ACODEC_SET_PD_DACR, &pd_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_PD_ADCL

【描述】

左声道输入的下电控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_PD_ADCL,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_PD_ADCL	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

不使用 ADC 时，可以调用此接口，将 ADCL 下电。arg 参数范围[0, 1]。

【举例】

```
unsigned int pd_ctrl;
pd_ctrl = 0x01;
if (ioctl(s32Acodec_Fd, ACODEC_SET_PD_ADCL, &pd_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_PD_ADCR

【描述】

右声道输入的下电控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_PD_ADCR,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入

参数名称	描述	输入/输出
ACODEC_SET_PD_ADCR	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

不使用 ADC 时，可以调用此接口，将 ADCR 下电。arg 参数范围[0, 1]。

【举例】

```
unsigned int pd_ctrl;
pd_ctl = 0x01;
if (ioctl(s32Acodec_Fd, ACODEC_SET_PD_ADCR, &pd_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_PD_LINEINL

【描述】

左声道 LINEIN 输入的下电控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_PD_LINEINL,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_PD_LINEINL	ioctl 号	输入

参数名称	描述	输入/输出
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

- 不使用 LINEIN 输入时，可以调用此接口，将 LINEINL 下电。arg 参数范围[0, 1]。
- Hi3516A 和 Hi3518EV200 不支持该接口。

【举例】

```
unsigned int lineinl_pd_ctrl;
lineinl_pd_ctl = 0x01;
if (ioctl(s32Acodec_Fd, ACODEC_SET_PD_LINEINL, &lineinl_pd_ctrl))
{
   printf("ioctl err!\n");
}
```

ACODEC_SET_PD_LINEINR

【描述】

右声道 LINEIN 输入的下电控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_PD_LINEINR,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_PD_LINEINR	ioctl 号	输入

参数名称	描述	输入/输出
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

- 不使用 LINEIN 输入时，可以调用此接口，将 LINEINR 下电。arg 参数范围[0, 1]。
- Hi3516A 和 Hi3518EV200 不支持该接口。

【举例】

```
unsigned int lineinr_pd_ctrl;
lineinr_pd_ctl = 0x01;
if (ioctl(s32Acodec_Fd, ACODEC_SET_PD_LINEINR, &lineinr_pd_ctrl))
{
    printf("ioctl err!\n");
}
```

9.3.5.2 扩展功能 cmd

- ACODEC_SEL_DAC_CLK：设置 DAC 的时钟沿同沿或反沿。
- ACODEC_SEL_ADC_CLK：设置 ADC 的时钟沿同沿或反沿。
- ACODEC_SEL_ANA_MCLK：设置模拟部分和数字部分的时钟沿同沿或反沿。
- ACODEC_DACL_SEL_TRACK：设置 DACL 选择左声道或右声道。
- ACODEC_DACR_SEL_TRACK：设置 DACR 选择左声道或右声道。
- ACODEC_ADCL_SEL_TRACK：设置 ADCL 选择左声道或右声道。
- ACODEC_ADCR_SEL_TRACK：设置 ADCR 选择左声道或右声道。
- ACODEC_SET_DAC_DE_EMPHASIS：DAC 的去加重滤波控制。
- ACODEC_SET_ADC_HP_FILTER：ADC 的高通滤波控制。
- ACODEC_DAC_POP_FREE：DAC 的去 POP 音控制。
- ACODEC_DAC_SOFT_MUTE_RATE：DAC 软静音速率控制。
- ACODEC_DAC_SEL_I2S：DAC I2S 接口选择。

- ACODEC_ADC_SEL_I2S：ADC I2S 接口选择。
- ACODEC_SET_I2S1_DATAWIDTH：I2S1 数据接口位宽。
- ACODEC_SET_I2S2_DATAWIDTH：I2S2 数据接口位宽。
- ACODEC_SET_I2S2_FS：设置 I2S2 接口采样率。
- ACODEC_SET_DACR2DACL_VOL：DACR 到 DACL 音量控制。
- ACODEC_SET_DACL2DACR_VOL：DACL 到 DACR 音量控制。
- ACODEC_SET_ADCL2DACL_VOL：ADCL 到 DACL 音量控制。
- ACODEC_SET_ADCR2DACL_VOL：ADCR 到 DACL 音量控制。
- ACODEC_SET_ADCL2DACR_VOL：ADCL 到 DACR 音量控制。
- ACODEC_SET_ADCR2DACR_VOL：ADCR 到 DACR 音量控制。

ACODEC_SEL_DAC_CLK

【描述】

设置 DAC 的时钟沿同沿或反沿。

【语法】

```
int ioctl (int fd,
           ACODEC_SEL_DAC_CLK,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SEL_DAC_CLK	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

arg 参数范围[0, 1]。

【举例】

```
unsigned int clk_sel;
clk_sel = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_SEL_DAC_CLK, &clk_sel))
{
    printf("ioctl err!\n");
}
```

ACODEC_SEL_ADC_CLK

【描述】

设置 ADC 的时钟沿同沿或反沿。

【语法】

```
int ioctl (int fd,
           ACODEC_SEL_ADC_CLK,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SEL_ADC_CLK	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

- arg 参数范围[0, 1]。
- Hi3516A 和 Hi3518EV200 不支持该接口。

【举例】

```
unsigned int clk_sel;
```

```
clk_sel = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_SEL_ADC_CLK, &clk_sel))
{
    printf("ioctl err!\n");
}
```

ACODEC_SEL_ANA_MCLK

【描述】

设置模拟部分和数字部分的时钟沿同沿或反沿。

【语法】

```
int ioctl (int fd,
           ACODEC_SEL_ANA_MCLK,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_SEL_ANA_MCLK	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

arg 参数范围[0, 1]。

【举例】

```
unsigned int clk_sel;
clk_sel = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_SEL_ANA_MCLK, &clk_sel))
{
    printf("ioctl err!\n");
```

```
}
```

ACODEC_DACL_SEL_TRACK

【描述】

设置 DACL 选择左声道或右声道。

【语法】

```
int ioctl (int fd,
           ACODEC_DACL_SEL_TRACK,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_DACL_SEL_TRACK	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

Hi3518EV200 和 Hi3519V100 不支持该接口。

【举例】

```
unsigned int track_select;
track_select = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_DACL_SEL_TRACK, &track_select))
{
    printf("ioctl err!\n");
}
```

ACODEC_DACR_SEL_TRACK

【描述】

设置 DACR 选择左声道或右声道。

【语法】

```
int ioctl (int fd,
           ACODEC_DACR_SEL_TRACK,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_DACR_SEL_TRACK	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

Hi3518EV200 和 Hi3519V100 不支持该接口。

【举例】

```
unsigned int track_select;
track_select = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_DACR_SEL_TRACK, &track_select)
{
    printf("ioctl err!\n");
}
```

ACODEC_ADCL_SEL_TRACK

【描述】

设置 ADCL 选择左声道或右声道。

【语法】

```
int ioctl (int fd,
           ACODEC_ADCL_SEL_TRACK,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_ADCL_SEL_TRACK	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

Hi3518EV200 和 Hi3519V100 不支持该接口。

【举例】

```
unsigned int track_select;
track_select = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_ADCL_SEL_TRACK, &track_select))
{
    printf("ioctl err!\n");
}
```

ACODEC_ADCR_SEL_TRACK

【描述】

设置 ADCR 选择左声道或右声道。

【语法】

```
int ioctl (int fd,
```

```
            ACODEC_ADCR_SEL_TRACK,
            unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述符	输入
ACODEC_ADCR_SEL_TRACK	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

Hi3518EV200 和 Hi3519V100 不支持该接口。

【举例】

```
unsigned int track_select;
track_select = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_ADCR_SEL_TRACK, &track_select))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_DAC_DE_EMPHASIS

【描述】

DAC 的去加重滤波控制。

【语法】

```
int ioctl (int fd,
            ACODEC_SET_DAC_DE_EMPHASIS,
            unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
Fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_DAC_DE_EMPHASIS	ioctl 号	输入
Arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

arg 参数范围[0, 3]。

【举例】

```
unsigned int dac_deemphasis;
dac_deemphasis = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_SET_DAC_DE_EMPHASIS, &dac_deemphasis))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_ADC_HP_FILTER

【描述】

ADC 的高通滤波控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_ADC_HP_FILTER,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
Fd	Audio Codec 设备文件描述符	输入

参数名称	描述	输入/输出
ACODEC_SET_ADC_HP_FILTER	ioctl 号	输入
Arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

arg 参数范围[0, 1]。

【举例】

```
unsigned int adc_hpf;
adc_hpf = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_SET_ADC_HP_FILTER, &adc_hpf))
{
    printf("ioctl err!\n");
}
```

ACODEC_DAC_POP_FREE

【描述】

DAC 的去 POP 音控制。

【语法】

```
int ioctl (int fd,
           ACODEC_DAC_POP_FREE,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
Fd	Audio Codec 设备文件描述符	输入
ACODEC_DAC_POP_FREE	ioctl 号	输入

参数名称	描述	输入/输出
Arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

- arg 参数范围[0, 1]。
- Hi3518EV200 和 Hi3519V100 不支持该接口。

【举例】

```
unsigned int dac_pop_free;
dac_pop_free = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_DAC_POP_FREE, &dac_pop_free))
{
   printf("ioctl err!\n");
}
```

ACODEC_DAC_SOFT_MUTE_RATE

【描述】

DAC 软静音速率控制。

【语法】

```
int ioctl (int fd,
           ACODEC_DAC_POP_FREE,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
Fd	Audio Codec 设备文件描述符	输入
ACODEC_DAC_SOFT_MUTE_RATE	ioctl 号	输入

参数名称	描述	输入/输出
Arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

- arg 参数范围[0, 3]。
- Hi3518EV200 和 Hi3519V100 不支持该接口。

【举例】

```
unsigned int soft_mute_rate;
soft_mute_rate = 0x1;
if (ioctl(s32Acodec_Fd, ACODEC_DAC_SOFT_MUTE_RATE, &soft_mute_rate))
{
   printf("ioctl err!\n");
}
```

ACODEC_DAC_SEL_I2S

【描述】

DAC I^2S 接口选择。

【语法】

```
int ioctl (int fd,
           ACODEC_DAC_SEL_I2S,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
Fd	Audio Codec 设备文件描述符	输入
ACODEC_DAC_SEL_I2S	ioctl 号	输入

参数名称	描述	输入/输出
Arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

- arg 参数范围[0, 1]。
- Hi3518EV200 和 Hi3519V100 不支持该接口。

【举例】

```
unsigned int i2s_sel;
i2s_sel = 0x0;
if (ioctl(s32Acodec_Fd, ACODEC_DAC_SOFT_MUTE_RATE, &i2s_sel))
{
   printf("ioctl err!\n");
}
```

ACODEC_ADC_SEL_I2S

【描述】

ADC I^2S 接口选择。

【语法】

```
int ioctl (int fd,
           ACODEC_ADC_SEL_I2S,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
Fd	Audio Codec 设备文件描述符	输入
ACODEC_ADC_SEL_I2S	ioctl 号	输入

参数名称	描述	输入/输出
Arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

- arg 参数范围[0, 1]。
- Hi3518EV200 和 Hi3519V100 不支持该接口。

【举例】

```
unsigned int i2s_sel;
i2s_sel = 0x0;
if (ioctl(s32Acodec_Fd, ACODEC_ADC_SEL_I2S, &i2s_sel))
{
   printf("ioctl err!\n");
}
```

ACODEC_SET_I2S1_DATAWIDTH

【描述】

I^2S1 数据接口位宽。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_I2S1_DATAWIDTH,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
Fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_I2S1_DATAWIDTH	ioctl 号	输入

参数名称	描述	输入/输出
Arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

arg 参数范围[0, 3]。

【举例】

```
unsigned int i2s_datawidth;
i2s_datawidth = 0x0;
if (ioctl(s32Acodec_Fd, ACODEC_SET_I2S1_DATAWIDTH, &i2s_datawidth))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_I2S2_DATAWIDTH

【描述】

I^2S2 数据接口位宽。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_I2S2_DATAWIDTH,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
Fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_I2S2_DATAWIDTH	ioctl 号	输入
Arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

- arg 参数范围[0, 3]。
- Hi3518EV200 和 Hi3519V100 不支持该接口。

【举例】

```
unsigned int i2s_datawidth;
i2s_datawidth = 0x0;
if (ioctl(s32Acodec_Fd, ACODEC_SET_I2S2_DATAWIDTH, &i2s_datawidth))
{
   printf("ioctl err!\n");
}
```

ACODEC_SET_I2S2_FS

【描述】

设置 I^2S2 接口采样率。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_I2S2_FS,
           unsigned int *arg);
```

【参数】

参数名称	描述	输入/输出
Fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_I2S2_FS	ioctl 号	输入
Arg	无符号整型指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

- arg 参数范围[0, 31]。
- Hi3518EV200 和 Hi3519V100 不支持该接口。

【举例】

```
ACODEC_FS_E i2s_fs_sel;
i2s_fs_sel = ACODEC_FS_48000;
if (ioctl(s32Acodec_Fd, ACODEC_SET_I2S2_FS, &i2s_fs_sel))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_DACR2DACL_VOL

【描述】

DACR 到 DACL 音量控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_DACR2DACL_VOL,
           ACODEC_VOL_CTRL *arg);
```

【参数】

参数名称	描述	输入/输出
Fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_DACR2DACL_VOL	ioctl 号	输入
Arg	ACODEC_VOL_CTRL 指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

Hi3518EV200 和 Hi3519V100 不支持该接口。

【举例】

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_DACR2DACL, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_DACL2DACR_VOL

【描述】

DACL 到 DACR 音量控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_DACL2DACR_VOL,
           ACODEC_VOL_CTRL *arg);
```

【参数】

参数名称	描述	输入/输出
Fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_DACL2DACR_VOL	ioctl 号	输入
Arg	ACODEC_VOL_CTRL 指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

Hi3518EV200 和 Hi3519V100 不支持该接口。

【举例】

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_DACL2DACR, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_ADCL2DACL_VOL

【描述】

ADCL 到 DACL 音量控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_ADCL2DACL_VOL,
           ACODEC_VOL_CTRL *arg);
```

【参数】

参数名称	描述	输入/输出
Fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_ADCL2DACL_VOL	ioctl 号	输入
Arg	ACODEC_VOL_CTRL 指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

Hi3518EV200 和 Hi3519V100 不支持该接口。

【举例】

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_ADCL2DACL, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_ADCR2DACL_VOL

【描述】

ADCR 到 DACL 音量控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_ADCR2DACL_VOL,
           ACODEC_VOL_CTRL *arg);
```

【参数】

参数名称	描述	输入/输出
Fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_ADCR2DACL_VOL	ioctl 号	输入
Arg	ACODEC_VOL_CTRL 指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

Hi3518EV200 和 Hi3519V100 不支持该接口。

【举例】

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_ADCR2DACL, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_ADCL2DACR_VOL

【描述】

ADCL 到 DACR 音量控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_ADCL2DACR_VOL,
           ACODEC_VOL_CTRL *arg);
```

【参数】

参数名称	描述	输入/输出
Fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_ADCL2DACR_VOL	ioctl 号	输入
Arg	ACODEC_VOL_CTRL 指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

Hi3518EV200 和 Hi3519V100 不支持该接口。

【举例】

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_ADCL2DACR, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

ACODEC_SET_ADCR2DACR_VOL

【描述】

ADCR 到 DACR 音量控制。

【语法】

```
int ioctl (int fd,
           ACODEC_SET_ADCR2DACR_VOL,
           ACODEC_VOL_CTRL *arg);
```

【参数】

参数名称	描述	输入/输出
Fd	Audio Codec 设备文件描述符	输入
ACODEC_SET_ADCR2DACR_VOL	ioctl 号	输入
Arg	ACODEC_VOL_CTRL 指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

acodec.h

【注意】

Hi3518EV200 和 Hi3519V100 不支持该接口。

【举例】

```
ACODEC_VOL_CTRL vol_ctrl;
vol_ctrl. vol_ctrl_mute = 0x0;
vol_ctrl. vol_ctrl = 0x06;
if (ioctl(s32Acodec_Fd, ACODEC_SET_ADCR2DACR, &vol_ctrl))
{
    printf("ioctl err!\n");
}
```

9.4 数据类型

9.4.1 音频输入输出

音频输入输出相关数据类型、数据结构定义如下：

- AIO_MAX_NUM：定义音频输入/输出设备的最大个数。
- AIO_MAX_CHN_NUM：定义音频输入/输出设备的最大通道数。
- AI_HIFIVQE_MASK_HPF：HiFi Vqe HPF 功能的 Mask。
- AI_HIFIVQE_MASK_RNR：HiFi Vqe RNR 功能的 Mask。
- AI_HIFIVQE_MASK_HDR：HiFi Vqe HDR 功能的 Mask。
- AI_HIFIVQE_MASK_DRC：HiFi Vqe DRC 功能的 Mask。
- AI_HIFIVQE_MASK_PEQ：HiFi Vqe PEQ 功能的 Mask。
- AI_TALKVQE_MASK_HPF：Talk Vqe HPF 功能的 Mask。
- AI_TALKVQE_MASK_AEC：Talk Vqe AEC 功能的 Mask。
- AI_TALKVQE_MASK_HDR：Talk Vqe HDR 功能的 Mask。
- AI_TALKVQE_MASK_AGC：Talk Vqe AGC 功能的 Mask。
- AI_TALKVQE_MASK_EQ：Talk Vqe EQ 功能的 Mask。
- AI_TALKVQE_MASK_ANR：Talk Vqe ANR 功能的 Mask。
- AO_VQE_MASK_HPF：AO Vqe HPF 功能的 Mask。

- AO_VQE_MASK_ANR：AO Vqe ANR 功能的 Mask。
- AO_VQE_MASK_AGC：AO Vqe AGC 功能的 Mask。
- AO_VQE_MASK_EQ：AO Vqe EQ 功能的 Mask。
- MAX_AUDIO_FILE_PATH_LEN：音频保存文件的路径的最大长度限制。
- MAX_AUDIO_FILE_NAME_LEN：音频保存文件的名称的最大长度限制。
- AUDIO_SAMPLE_RATE_E：定义音频采样率。
- AUDIO_BIT_WIDTH_E：定义音频采样精度。
- AIO_MODE_E：定义音频输入输出工作模式。
- AUDIO_SOUND_MODE_E：定义音频声道模式。
- AIO_ATTR_S：定义音频输入输出设备属性结构体。
- AI_CHN_PARAM_S：定义通道参数结构体。
- AUDIO_FRAME_S：定义音频帧数据结构体。
- AEC_FRAME_S：定义回声抵消参考帧信息结构体。
- AUDIO_AGC_CONFIG_S：定义音频自动增益控制配置信息结构体。
- AI_AEC_CONFIG_S：定义音频回声抵消配置信息结构体。
- AUDIO_ANR_CONFIG_S：定义音频语音降噪功能配置信息结构体。
- AUDIO_HPF_FREQ_E：定义音频高通滤波截止频率。
- AUDIO_HPF_CONFIG_S：定义音频高通滤波功能配置信息结构体。
- AI_RNR_CONFIG_S：定义音频录音噪声消除功能配置信息结构体。
- VQE_WORKSTATE_E：定义声音质量增强的工作模式。
- VQE_EQ_BAND_NUM：定义 EQ 功能可调节的频段数。
- AUDIO_EQ_CONFIG_S：定义音频均衡器功能配置信息结构体。
- AI_HDR_CONFIG_S：定义音频高动态范围功能配置信息结构体。
- VQE_DRC_SECNUM：定义 DRC 动态曲线可配置级数。
- AI_DRC_CONFIG_S：定义音频动态压缩控制功能配置信息结构体。
- VQE_PEQ_BAND_NUM：定义 PEQ 功能可调节的最大频段数。
- AI_PEQ_CONFIG_S：定义音频参量均衡器配置信息结构体。
- AI_VQE_CONFIG_S：定义音频输入声音质量增强配置信息结构体。
- AI_HIFIVQE_CONFIG_S：定义音频输入声音质量增强（HiFi）配置信息结构体。
- AI_TALKVQE_CONFIG_S：定义音频输入声音质量增强（Talk）配置信息结构体。
- AO_VQE_CONFIG_S：定义音频输出声音质量增强配置信息结构体。
- AUDIO_STREAM_S：定义音频码流结构体。
- AO_CHN_STATE_S：音频输出通道的数据缓存状态结构体。
- AUDIO_TRACK_MODE_E：音频设备声道模式类型。
- AUDIO_FADE_RATE_E：音频设备淡入淡出速率类型。
- AUDIO_FADE_S：音频设备淡入淡出设置结构体。
- G726_BPS_E：定义 G.726 编解码协议速率。

- ADPCM_TYPE_E：定义 ADPCM 编解码协议类型。
- AUDIO_SAVE_FILE_INFO_S：定义音频保存文件功能配置信息结构体。
- AUDIO_FILE_STATUS_S：定义音频文件保存状态结构体。

AIO_MAX_NUM

【说明】

定义音频输入/输出设备的最大个数。

【定义】

```
#define AIO_MAX_NUM        1
```

【注意事项】

无。

【相关数据类型及接口】

无。

AIO_MAX_CHN_NUM

【说明】

定义音频输入/输出设备的最大通道数。

【定义】

```
#define AIO_MAX_CHN_NUM     16
```

【注意事项】

无。

【相关数据类型及接口】

无。

AI_HIFIVQE_MASK_HPF

【说明】

定义 HiFi Vqe HPF 功能的 Mask。

【定义】

```
#define AI_HIFIVQE_MASK_HPF    0x1
```

【注意事项】

Hi3518EV200 无此宏定义。

【相关数据类型及接口】

赋值给 AI_HIFIVQE_CONFIG_S 结构体成员 u32OpenMask 表示开启 HPF 功能。如 u32OpenMask = AI_HIFIVQE_MASK_RNR | AI_HIFIVQE_MASK_HPF;表示开启 RNR 和 HPF 功能。

AI_HIFIVQE_MASK_RNR

【说明】

定义 HiFi Vqe RNR 功能的 Mask。

【定义】

```
#define AI_HIFIVQE_MASK_RNR    0x2
```

【注意事项】

Hi3518EV200 无此宏定义。

【相关数据类型及接口】

赋值给 AI_HIFIVQE_CONFIG_S 结构体成员 u32OpenMask 表示开启 RNR 功能。如 u32OpenMask = AI_HIFIVQE_MASK_RNR | AI_HIFIVQE_MASK_HPF;表示开启 RNR 和 HPF 功能。

AI_HIFIVQE_MASK_HDR

【说明】

定义 HiFi Vqe　HDR 功能的 Mask。

【定义】

```
#define AI_HIFIVQE_MASK_HDR    0x4
```

【注意事项】

Hi3518EV200 无此宏定义。

【相关数据类型及接口】

赋值给 AI_HIFIVQE_CONFIG_S 结构体成员 u32OpenMask 表示开启 HDR 功能。如 u32OpenMask = AI_HIFIVQE_MASK_RNR | AI_HIFIVQE_MASK_HDR;表示开启 RNR 和 HDR 功能。

AI_HIFIVQE_MASK_DRC

【说明】

定义 HiFi Vqe DRC 功能的 Mask。

【定义】

```
#define AI_HIFIVQE_MASK_DRC    0x8
```

【注意事项】

Hi3518EV200 无此宏定义。

【相关数据类型及接口】

赋值给 AI_HIFIVQE_CONFIG_S 结构体成员 u32OpenMask 表示开启 DRC 功能。如 u32OpenMask = AI_HIFIVQE_MASK_RNR | AI_HIFIVQE_MASK_DRC;表示开启 RNR 和 DRC 功能。

AI_HIFIVQE_MASK_PEQ

【说明】

定义 HiFi Vqe PEQ 功能的 Mask。

【定义】

```
#define AI_HIFIVQE_MASK_PEQ    0x10
```

【注意事项】

Hi3518EV200 无此宏定义。

【相关数据类型及接口】

赋值给 AI_HIFIVQE_CONFIG_S 结构体成员 u32OpenMask 表示开启 PEQ 功能。如 u32OpenMask = AI_HIFIVQE_MASK_RNR | AI_HIFIVQE_MASK_PEQ;表示开启 RNR 和 PEQ 功能。

AI_TALKVQE_MASK_HPF

【说明】

定义 Talk Vqe　HPF 功能的 Mask。

【定义】

```
#define AI_TALKVQE_MASK_HPF    0x1
```

【注意事项】

Hi3518EV200、Hi3516A 无此宏定义。

【相关数据类型及接口】

赋值给 AI_TALKVQE_CONFIG_S 结构体成员 u32OpenMask 表示开启 HPF 功能。如 u32OpenMask = AI_TALKVQE_MASK_AEC | AI_TALKVQE_MASK_HPF;表示开启 AEC 和 HPF 功能。

AI_TALKVQE_MASK_AEC

【说明】

定义 Talk Vqe　AEC 功能的 Mask。

【定义】

```
#define AI_TALKVQE_MASK_AEC    0x2
```

【注意事项】

Hi3518EV200、Hi3516A 无此宏定义。

【相关数据类型及接口】

赋值给 AI_TALKVQE_CONFIG_S 结构体成员 u32OpenMask 表示开启 AEC 功能。如 u32OpenMask =AI_TALKVQE_MASK_AEC | AI_TALKVQE_MASK_HPF;表示开启 AEC 和 HPF 功能。

AI_TALKVQE_MASK_HDR

【说明】

定义 Talk Vqe　HDR 功能的 Mask。

【定义】

```
#define AI_TALKVQE_MASK_HDR    0x4
```

【注意事项】

Hi3518EV200、Hi3516A 无此宏定义。

【相关数据类型及接口】

赋值给 AI_TALKVQE_CONFIG_S 结构体成员 u32OpenMask 表示开启 HDR 功能。如 u32OpenMask = AI_TALKVQE_MASK_AEC | AI_TALKVQE_MASK_HDR;表示开启 AEC 和 HDR 功能。

AI_TALKVQE_MASK_AGC

【说明】

定义 Talk Vqe　AGC 功能的 Mask。

【定义】

```
#define AI_TALKVQE_MASK_AGC    0x8
```

【注意事项】

Hi3518EV200、Hi3516A 无此宏定义。

【相关数据类型及接口】

赋值给 AI_TALKVQE_CONFIG_S 结构体成员 u32OpenMask 表示开启 AGC 功能。如 u32OpenMask = AI_TALKVQE_MASK_AEC | AI_TALKVQE_MASK_AGC;表示开启 AEC 和 AGC 功能。

AI_TALKVQE_MASK_EQ

【说明】

定义 Talk Vqe　EQ 功能的 Mask。

【定义】

```
#define AI_TALKVQE_MASK_EQ     0x10
```

【注意事项】

Hi3518EV200、Hi3516A 无此宏定义。

【相关数据类型及接口】

赋值给 AI_TALKVQE_CONFIG_S 结构体成员 u32OpenMask 表示开启 EQ 功能。如 u32OpenMask = AI_TALKVQE_MASK_AEC | AI_TALKVQE_MASK_EQ;表示开启 AEC 和 EQ 功能。

AI_TALKVQE_MASK_ANR

【说明】

定义 Talk Vqe　ANR 功能的 Mask。

【定义】

```
#define AI_TALKVQE_MASK_ANR    0x20
```

【注意事项】

Hi3518EV200、Hi3516A 无此宏定义。

【相关数据类型及接口】

赋值给 AI_TALKVQE_CONFIG_S 结构体成员 u32OpenMask 表示开启 ANR 功能。如 u32OpenMask = AI_TALKVQE_MASK_AEC | AI_TALKVQE_MASK_ANR;表示开启 AEC 和 ANR 功能。

AO_VQE_MASK_HPF

【说明】

定义 AO Vqe HPF 功能的 Mask。

【定义】

```
#define AO_VQE_MASK_HPF    0x1
```

【注意事项】

Hi3518EV200、Hi3516A 无此宏定义。

【相关数据类型及接口】

赋值给 AO_VQE_CONFIG_S 结构体成员 u32OpenMask 表示开启 HPF 功能。如 u32OpenMask = AO_VQE_MASK_AGC | AO_VQE_MASK_HPF;表示开启 AGC 和 HPF 功能。

AO_VQE_MASK_ANR

【说明】

定义 AO Vqe ANR 功能的 Mask。

【定义】

```
#define AI_TALKVQE_MASK_ANR    0x2
```

【注意事项】

Hi3518EV200、Hi3516A 无此宏定义。

【相关数据类型及接口】

赋值给 AO_VQE_CONFIG_S 结构体成员 u32OpenMask 表示开启 ANR 功能。如 u32OpenMask = AO_VQE_MASK_AGC | AO_VQE_MASK_ANR;表示开启 AGC 和 ANR 功能。

AO_VQE_MASK_AGC

【说明】

定义 AO Vqe AGC 功能的 Mask。

【定义】

```
#define AO_VQE_MASK_AGC    0x4
```

【注意事项】

Hi3518EV200、Hi3516A 无此宏定义。

【相关数据类型及接口】

赋值给 AO_VQE_CONFIG_S 结构体成员 u32OpenMask 表示开启 AGC 功能。如 u32OpenMask = AO_VQE_MASK_AGC | AO_VQE_MASK_EQ;表示开启 AGC 和 EQ 功能。

AO_VQE_MASK_EQ

【说明】

定义 AO Vqe EQ 功能的 Mask。

【定义】

```
#define AO_VQE_MASK_EQ    0x8
```

【注意事项】

Hi3518EV200、Hi3516A 无此宏定义。

【相关数据类型及接口】

赋值给 AO_VQE_CONFIG_S 结构体成员 u32OpenMask 表示开启 EQ 功能。如 u32OpenMask = AO_VQE_MASK_AGC | AO_VQE_MASK_EQ;表示开启 AGC 和 EQ 功能。

MAX_AUDIO_FILE_PATH_LEN

【说明】

定义音频保存文件的路径的最大长度限制。

【定义】

```
#define MAX_AUDIO_FILE_PATH_LEN    256
```

【注意事项】

无。

【相关数据类型及接口】

AUDIO_SAVE_FILE_INFO_S

MAX_AUDIO_FILE_NAME_LEN

【说明】

定义音频保存文件的名称的最大长度限制。

【定义】

```
#define MAX_AUDIO_FILE_NAME_LEN    256
```

【注意事项】

无。

【相关数据类型及接口】

AUDIO_SAVE_FILE_INFO_S

AUDIO_SAMPLE_RATE_E

【说明】

定义音频采样率。

【定义】

```
typedef enum hiAUDIO_SAMPLE_RATE_E
{
    AUDIO_SAMPLE_RATE_8000 =8000,      /* 8kHz sampling rate */
    AUDIO_SAMPLE_RATE_12000=12000,     /* 12kHz sampling rate */
    AUDIO_SAMPLE_RATE_11025=11025,     /* 11.025kHz sampling rate */
    AUDIO_SAMPLE_RATE_16000=16000,     /* 16kHz sampling rate */
    AUDIO_SAMPLE_RATE_22050=22050,     /* 22.050kHz sampling rate */
    AUDIO_SAMPLE_RATE_24000=24000,     /* 24kHz sampling rate */
    AUDIO_SAMPLE_RATE_32000=32000,     /* 32kHz sampling rate */
    AUDIO_SAMPLE_RATE_44100=44100,     /* 44.1kHz sampling rate */
    AUDIO_SAMPLE_RATE_48000=48000,     /* 48kHz sampling rate */
   AUDIO_SAMPLE_RATE_64000=64000,      /* 64K samplerate*/
   AUDIO_SAMPLE_RATE_96000=96000,      /* 96K samplerate*/
   AUDIO_SAMPLE_RATE_BUTT,
}AUDIO_SAMPLE_RATE_E;
```

【成员】

成员名称	描述
AUDIO_SAMPLE_RATE_8000	8kHz 采样率。
AUDIO_SAMPLE_RATE_12000	12kHz 采样率。
AUDIO_SAMPLE_RATE_11025	11.025kHz 采样率。
AUDIO_SAMPLE_RATE_16000	16kHz 采样率。
AUDIO_SAMPLE_RATE_22050	22.050kHz 采样率。
AUDIO_SAMPLE_RATE_24000	24kHz 采样率。
AUDIO_SAMPLE_RATE_32000	32kHz 采样率。
AUDIO_SAMPLE_RATE_44100	44.1kHz 采样率。
AUDIO_SAMPLE_RATE_48000	48kHz 采样率。
AUDIO_SAMPLE_RATE_64000	64kHz 采样率。
AUDIO_SAMPLE_RATE_96000	96kHz 采样率。

【注意事项】

- 这里枚举值不是从 0 开始，而是与实际的采样率值相同。
- 64kHz、96kHz 的采样率仅适用于 AI 采集、AO 播放，不支持对 64kHz、96kHz 采样率的数据做重采样、VQE 等处理。

【相关数据类型及接口】

AIO_ATTR_S

AUDIO_BIT_WIDTH_E

【说明】

定义音频采样精度。

【定义】

```
typedef enum hiAUDIO_BIT_WIDTH_E
{
    AUDIO_BIT_WIDTH_8   =0,     /* 8bit width */
    AUDIO_BIT_WIDTH_16  =1,     /* 16bit width */
    AUDIO_BIT_WIDTH_24  =2,     /* 24bit width */
    AUDIO_BIT_WIDTH_BUTT,
}AUDIO_BIT_WIDTH_E;
```

【成员】

成员名称	描述
AUDIO_BIT_WIDTH_8	采样精度为 8bit 位宽。
AUDIO_BIT_WIDTH_16	采样精度为 16bit 位宽。
AUDIO_BIT_WIDTH_24	采样精度为 24bit 位宽。

【注意事项】

目前只支持 16bit 位宽。

【相关数据类型及接口】

AIO_ATTR_S

AIO_MODE_E

【说明】

定义音频输入输出设备工作模式。

【定义】

```
typedef enum hiAIO_MODE_E
{
    AIO_MODE_I2S_MASTER = 0,        /* I2S master mode */
    AIO_MODE_I2S_SLAVE  = 1,        /* I2S slave mode */
    AIO_MODE_PCM_SLAVE_STD,         /* SIO PCM slave standard mode */
    AIO_MODE_PCM_SLAVE_NSTD,        /* SIO PCM slave non-standard mode */
    AIO_MODE_PCM_MASTER_STD,       /* SIO PCM master standard mode */
    AIO_MODE_PCM_MASTER_NSTD,      /* SIO PCM master non-standard mode */
    AIO_MODE_BUTT
}AIO_MODE_E;
```

【成员】

成员名称	描述
AIO_MODE_I2S_MASTER	I^2S 主模式。
AIO_MODE_I2S_SLAVE	I^2S 从模式。
AIO_MODE_PCM_SLAVE_STD	PCM 从模式（标准协议）
AIO_MODE_PCM_SLAVE_NSTD	PCM 从模式（自定义协议）
AIO_MODE_PCM_MASTER_STD	PCM 主模式（标准协议）
AIO_MODE_PCM_MASTER_NSTD	PCM 主模式（自定义协议）

【注意事项】

Hi35xx 对接内置 Audio Codec 时，只支持 I^2S 主模式。

【相关数据类型及接口】

AIO_ATTR_S

AUDIO_SOUND_MODE_E

【说明】

定义音频声道模式。

【定义】

```
typedef enum hiAIO_SOUND_MODE_E
{
    AUDIO_SOUND_MODE_MONO   =0,     /*mono*/
    AUDIO_SOUND_MODE_STEREO =1,     /*stereo*/
    AUDIO_SOUND_MODE_BUTT
}AUDIO_SOUND_MODE_E;
```

【成员】

成员名称	描述
AUDIO_SOUND_MODE_MONO	单声道。
AUDIO_SOUND_MODE_STEREO	双声道。

【注意事项】

对于双声道模式，只应对左声道（即编号小于设备属性中通道数 u32ChnCnt 一半的通道）进行操作，SDK 内部会自动对右声道也进行相应的操作。

【相关数据类型及接口】

AIO_ATTR_S

AIO_ATTR_S

【说明】

定义音频输入输出设备属性结构体。

【定义】

```
typedef struct hiAIO_ATTR_S
{
    AUDIO_SAMPLE_RATE_E enSamplerate;       /*sample rate*/
    AUDIO_BIT_WIDTH_E   enBitwidth;         /*bitwidth*/
    AIO_MODE_E          enWorkmode;         /*master or slave mode*/
```

```
    AUDIO_SOUND_MODE_E  enSoundmode;        /*momo or steror*/
    HI_U32              u32EXFlag;          /*expand 8bit to 16bit */
    HI_U32              u32FrmNum;          /*frame num in buffer*/
    HI_U32              u32PtNumPerFrm;     /*number of samples*/
    HI_U32              u32ChnCnt;
    HI_U32              u32ClkSel;
}AIO_ATTR_S;
```

【成员】

成员名称	描述
enSamplerate	音频采样率（从模式下，此参数不起作用）。 静态属性。
enBitwidth	音频采样精度（从模式下，此参数必须和音频 AD/DA 的采样精度匹配）。 静态属性。
enWorkmode	音频输入输出工作模式。 静态属性。
enSoundmode	音频声道模式。 静态属性。
u32EXFlag	取值范围：{0, 1, 2}。 • 0：不扩展。 • 1：扩展成 16 位，8bit 到 16bit 扩展标志（只对 AI 采样精度为 8bit 时有效）。 • 2：24 位裁剪成 16 位，在外置 Codec 的场景下可能用到。 静态属性，保留参数，一般设置成 1 即可。
u32FrmNum	缓存帧数目。 取值范围：[2, MAX_AUDIO_FRAME_NUM]。 静态属性。
u32PtNumPerFrm	每帧的采样点个数。 取值范围：G711、G726、ADPCM_DVI4 编码时取值为 80、160、240、320、480；ADPCM_IMA 编码时取值为 81、161、241、321、481。 AI 取值范围为：[80, 2048]，AO 取值范围为：[80, 4096]。 静态属性。
u32ChnCnt	支持的通道数目。 取值：1、2、4、8、16。 （输入最多支持 16 个通道，输出最多支持 2 个通道）

成员名称	描述
u32ClkSel	配置 AI 设备 0 是否复用 AO 设备 0 的帧同步时钟及位流时钟。 取值：0、1。 0：不复用； 1：复用。

【注意事项】

无。

【相关数据类型及接口】

HI_MPI_AI_SetPubAttr

AI_CHN_PARAM_S

【说明】

定义通道参数结构体。

【定义】

```
typedef struct hiAI_CHN_PARAM_S
{
    HI_U32 u32UsrFrmDepth;
} AI_CHN_PARAM_S;
```

【成员】

成员名称	描述
u32UsrFrmDepth	音频帧缓存深度。

【注意事项】

无。

【相关数据类型及接口】

无。

AUDIO_FRAME_S

【说明】

定义音频帧结构体。

【定义】

```
typedef struct hiAUDIO_FRAME_S
{
    AUDIO_BIT_WIDTH_E   enBitwidth;  /*audio frame bitwidth*/
    AUDIO_SOUND_MODE_E  enSoundmode; /*audio frame momo or stereo mode*/
    HI_VOID             *pVirAddr[2];
    HI_U32              u32PhyAddr[2];
    HI_U64              u64TimeStamp;/*audio frame timestamp*/
    HI_U32              u32Seq;      /*audio frame seq*/
    HI_U32              u32Len;      /*data lenth per channel in frame*/
    HI_U32              u32PoolId[2];
}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。
pVirAddr[2]	音频帧数据虚拟地址。
u32PhyAddr[2]	音频帧数据物理地址。
u64TimeStamp	音频帧时间戳。 以 μs 为单位。
u32Seq	音频帧序号。
u32Len	音频帧长度。 以 byte 为单位。
u32PoolId[2]	音频帧缓存池 ID。

【注意事项】

- u32Len（音频帧长度）指单个声道的数据长度。
- 单声道数据直接存放，采样点数为 ptnum，长度为 len；立体声数据按左右声道分开存放，先存放采样点为 ptnum、长度为 len 的左声道数据，然后存放采样点为 ptnum，长度为 len 的右声道数据。

【相关数据类型及接口】

无。

AEC_FRAME_S

【说明】

定义音频回声抵消参考帧信息结构体。

【定义】

```
typedef struct hiAEC_FRAME_S
{
    AUDIO_FRAME_S   stRefFrame;     /* aec reference audio frame */
    HI_BOOL         bValid;         /* whether frame is valid */
    HI_BOOL        bSysBind;      /* whether is sysbind */
}AEC_FRAME_S;
```

【成员】

成员名称	描述
stRefFrame	回声抵消参考帧结构体。
bValid	参考帧有效的标志。 取值范围： HI_TRUE：参考帧有效。 HI_FALSE：参考帧无效，无效时不能使用此参考帧进行回声抵消。
bSysBind	AI 和 AENC 是否系统绑定。

【注意事项】

无。

【相关数据类型及接口】

无。

AUDIO_AGC_CONFIG_S

【说明】

定义音频自动增益控制配置信息结构体。

【定义】

```
typedef struct hiAUDIO_AGC_CONFIG_S
{
    HI_BOOL bUsrMode;
    HI_S8 s8TargetLevel;
    HI_S8 s8NoiseFloor;
    HI_S8 s8MaxGain;
    HI_S8 s8AdjustSpeed;
    HI_S8 s8ImproveSNR;
    HI_S8 s8UseHighPassFilt;
    HI_S8 s8OutputMode;
    HI_S16 s16NoiseSupSwitch;
```

```
    HI_S32 s32Reserved;
} AUDIO_AGC_CONFIG_S;
```

【成员】

成员名称	描述
bUsrMode	是否采用用户模式： 0：自动模式 1：用户模式 默认为 0 关闭
s8TargetLevel	目标电平，该值为经过 AGC 处理后的最大电平门限；范围：[-40 ~ -1]dB；调整步长 1dB
s8NoiseFloor	噪声底线；范围：[-65 ~ -20]dB；调整步长 1dB。 注：噪声底线需要开启输出模式才能生效。
s8MaxGain	最大增益；范围：[0 ~ 30]dB；调整步长 1dB
s8AdjustSpeed	调整速度；范围：[0 ~ 10]dB/s；调整步长 1dB/s
s8ImproveSNR	提高信噪比开关；范围：[0:不提升, 1:上限 3dB, 2:上限 6dB]
s8UseHighPassFilt	打开高通滤波标志，该值为配置 AGC 内置高通滤波截止频率；范围：[0:关闭, 1:80Hz, 2:120Hz, 3:150Hz, 4:300Hz, 5:500Hz]
s8OutputMode	输出模式,低于 NoiseFloor 的信号输出静音；范围：[0:关闭, 1:打开]。 注：输出模式需要配合噪声底线使用，配置为 1 后，低于噪声底线的信号将会静音输出。
s16NoiseSupSwitch	噪声抑制开关；范围{0, 1}，0 表示关闭，1 表示开启
s32Reserved	保留。

【注意事项】

- 当用户模式开启时，以上各个高级参数才生效，否则按照 AI_VQE_CONFIG_S/ AI_TALKVQE_CONFIG_S/AO_VQE_CONFIG_S 中的工作模式 enWorkstate 对应的默认值来配置。
- 配置参数时，只有在用户模式开启时，才会对高级参数做正确性检查，只有正确的高级参数才能配置成功。

【相关数据类型及接口】

- AI_VQE_CONFIG_S
- AO_VQE_CONFIG_S

AI_AEC_CONFIG_S

【说明】

定义音频回声抵消配置信息结构体。

【定义】

```
typedef struct hiAI_AEC_CONFIG_S
{
    HI_BOOL bUsrMode;
    HI_S8   s8CngMode;     /* cosy-noisy mode:0 close,1 open, default 1*/
    HI_S8 s8NearAllPassEnergy;
    HI_S8 s8NearCleanSupEnergy;
    HI_S16 s16DTHnlSortQTh;
    HI_S16 s16EchoBandLow;
    HI_S16 s16EchoBandHigh;
    HI_S16 s16EchoBandLow2;
    HI_S16 s16EchoBandHigh2;
    HI_S16 s16ERLBand[6];
    HI_S16 s16ERL[7];
    HI_S16 s16ViocеProtectFreqL;
    HI_S16 s16ViocеProtectFreqL1;
    HI_S32 s32Reserved;
} AI_AEC_CONFIG_S;
```

【成员】

成员名称	描述
bUsrMode	是否采用用户模式： 0：自动模式 1：用户模式 默认为 0 关闭
s8CngMode	是否开启舒适噪声模式： 0：关闭 1：开启
s8NearAllPassEnergy	判断近端是否透传的远端能量阈值，默认为 1，取值:【0:-59dBm0, 1:-49dBm0, 2:-39dBm0】
s8NearCleanSupEnergy	近端信号强制清零的能量阈值，默认为 2，取值:【0:12dB, 1:15dB, 2:18dB】
s16DTHnlSortQTh	单双讲判断门限值，标为 Q15，默认取 16384, [0, 32767]
s16EchoBandLow	语音处理频段 1，低频参数，8k 下配置参数[1, 63), 16k 下配置参数[1, 127)，默认 10

成员名称	描述
s16EchoBandHigh	语音处理频段 1，高频参数, 8k 下配置参数(s16EchoBandLow, 63], 16k 下配置参数(s16EchoBandLow, 127], 默认 41
s16EchoBandLow2	语音处理频段 2,低频参数，8k 下配置参数[1, 63)，16k 下配置参数[1, 127)，默认 47
s16EchoBandHigh2	语音处理频段 2,高频参数，8k 下配置参数(s16EchoBandLow2, 63]，默认 63，16k 下配置参数(s16EchoBandLow2, 127]，默认 72
s16ERLBand[6]	ERL(回波衰减路径)保护频段数组参数，8k 下配置参数[1, 63], 16k 下配置参数[1, 127]，其中，配置频段参数必须每段递增，即 s16ERLBand[n+1] > s16ERLBand[n]。建议配置参数{4, 6, 36, 49, 50, 51}；
s16ERL[7]	ERL(回波衰减路径)频段保护值数组参数, 配合参数 s16ERLBand [6]使用，s16ERLBand 将 s16ERL 值分频段设置，0 至 s16ERLBand[0]频带对应值为 s16ERL[0]，s16ERLBand[0] 至 s16ERLBand[1]频带对应值为 s16ERL[1]，以此类推，s16ERLBand[5]至 MaxBand 频带对应值为 s16ERL[6]，其中 MaxBand 在 8k 下为 65，在 16k 下为 129。该参数对应取值越小，保护力度越大，参数范围[0, 18]，建议配置参数{7, 10, 16, 10, 18, 18, 18}
s16VioceProtectFreqL	近端低频保护区域频点参数, 8k 下配置参数[1, 63), 16k 下配置参数[1, 127), 默认为 3
s16VioceProtectFreqL1	近端低频保护区域频点参数 1,8k 下配置参数(s16VioceProtectFreqL, 63], 16k 下配置参数(s16VioceProtectFreqL, 127], 默认为 6
s32Reserved	保留，未使用。

【注意事项】

- 当用户模式开启时，其他参数才生效，否则按照 AI_VQE_CONFIG_S/ AI_TALKVQE_CONFIG_S 中的工作模式 enWorkstate 对应的默认值来配置。
- 配置参数时，只有在用户模式开启时，才会对高级参数做正确性检查，只有正确的高级参数才能配置成功。

【相关数据类型及接口】

AI_VQE_CONFIG_S

AUDIO_ANR_CONFIG_S

【说明】

定义音频语音降噪功能配置信息结构体。

【定义】

```
typedef struct hiAUDIO_ANR_CONFIG_S
{
    HI_BOOL bUsrMode;
    HI_S16 s16NrIntensity;
    HI_S16 s16NoiseDbThr;
    HI_S8  s8SpProSwitch;
    HI_S32 s32Reserved;
} AUDIO_ANR_CONFIG_S;
```

【成员】

成员名称	描述
bUsrMode	是否采用用户模式： 0：自动模式 1：用户模式 默认为 0 关闭。
s16NrIntensity	降噪力度配置，取值为[0, 25]，配置值越大降噪力度越高，但同时也会带来细节音的丢失/损伤。
s16NoiseDbThr	噪声门限配置，取值为[30,60]，配置值越大，检测力度越弱，声音更平滑
s8SpProSwitch	音乐检测开关，取值为[0,1],开启后增加对音乐细节的检测，喧闹场景下不建议开启
s32Reserved	保留，未使用。

【注意事项】

- 当用户模式开启时，以上各个高级参数才生效，否则按照 AI_VQE_CONFIG_S/AI_TALKVQE_CONFIG_S/AO_VQE_CONFIG_S 中的工作模式 enWorkstate 对应的默认值来配置。
- 配置参数时，只有在用户模式开启时，才会对高级参数做正确性检查，只有正确的高级参数才能配置成功。

【相关数据类型及接口】

- AI_VQE_CONFIG_S
- AO_VQE_CONFIG_S

AUDIO_HPF_FREQ_E

【说明】

定义音频高通滤波截止频率。

【定义】

```
typedef enum hiAUDIO_HPF_FREQ_E
{
    AUDIO_HPF_FREQ_80   = 80,    /* 80Hz */
    AUDIO_HPF_FREQ_120  = 120,   /* 120Hz */
    AUDIO_HPF_FREQ_150  = 150,   /* 150Hz */
    AUDIO_HPF_FREQ_BUTT,
} AUDIO_HPF_FREQ_E;
```

【成员】

成员名称	描述
AUDIO_HPF_FREQ_80	截止频率为 80Hz。
AUDIO_HPF_FREQ_120	截止频率为 120Hz。
AUDIO_HPF_FREQ_150	截止频率为 150Hz。

【注意事项】

默认配置为 150Hz。

【相关数据类型及接口】

无。

AUDIO_HPF_CONFIG_S

【说明】

定义音频高通滤波功能配置信息结构体。

【定义】

```
typedef struct hiAUDIO_HPF_CONFIG_S
{
    HI_BOOL bUsrMode;
    AUDIO_HPF_FREQ_E enHpfFreq; /*freq to be processed*/
} AUDIO_HPF_CONFIG_S;
```

【成员】

成员名称	描述
bUsrMode	是否采用用户模式： 0：自动模式 1：用户模式 默认为 0 关闭。

成员名称	描述
enHpfFreq	高通滤波截止频率选择： 80：截止频率为 80Hz 120：截止频率为 120Hz 150：截止频率为 150Hz

【注意事项】

- 当用户模式开启时，高级参数才生效，否则按照 AI_VQE_CONFIG_S/AI_TALKVQE_CONFIG_S/AI_HIFIVQE_CONFIG_S/AO_VQE_CONFIG_S 中的工作模式 enWorkstate 对应的默认值来配置。
- 配置参数时，只有在用户模式开启时，才会对高级参数做正确性检查，只有正确的高级参数才能配置成功。

【相关数据类型及接口】

- AI_VQE_CONFIG_S
- AO_VQE_CONFIG_S

AI_RNR_CONFIG_S

【说明】

定义音频录音噪声消除功能配置信息结构体。

【定义】

```
typedef struct hiAI_RNR_CONFIG_S
{
    HI_BOOL bUsrMode;
    HI_S32  s32NrMode;
    HI_S32 s32MaxNrLevel;
    HI_S32  s32NoiseThresh;
} AI_RNR_CONFIG_S;
```

【成员】

成员名称	描述
bUsrMode	是否采用用户模式： 0：自动模式 1：用户模式 默认为 0 关闭。
s32NrMode	降噪模式。 0：降底噪；

成员名称	描述
	1：降环境噪声。 默认值：0。
s32MaxNrLevel	最大降噪能力；范围：[2，20]dB;调整步长 1dB。 默认值：15。
s32NoiseThresh	噪声阈值；范围: [-80，-20]。默认值：-55。

【注意事项】

- 当用户模式开启时，高级参数才生效，否则按照 AI_VQE_CONFIG_S/ AI_HIFIVQE_CONFIG_S 中的工作模式 enWorkstate 对应的默认值来配置。
- 配置参数时，只有在用户模式开启时，才会对高级参数做正确性检查，只有正确的高级参数才能配置成功。
- 当 s32NrMode 为 1 时，不支持 48k 工作采样率（s32WorkSampleRate）。

【相关数据类型及接口】

AI_VQE_CONFIG_S

VQE_WORKSTATE_E

【说明】

定义声音质量增强的工作模式。

【定义】

```
typedef enum hiVQE_WORKSTATE_E
{
    VQE_WORKSTATE_COMMON  = 0,
    VQE_WORKSTATE_MUSIC   = 1,
    VQE_WORKSTATE_NOISY   = 2
} VQE_WORKSTATE_E;
```

【成员】

成员名称	描述
VQE_WORKSTATE_COMMON	一般模式。
VQE_WORKSTATE_MUSIC	音乐模式。
VQE_WORKSTATE_NOISY	噪声模式。

【注意事项】

无。

【相关数据类型及接口】

- AI_VQE_CONFIG_S
- AO_VQE_CONFIG_S

VQE_EQ_BAND_NUM

【说明】

定义 EQ 功能可调节的频段数。

【定义】

```
#define VQE_EQ_BAND_NUM 10
```

【注意事项】

无。

【相关数据类型及接口】

AUDIO_EQ_CONFIG_S

AUDIO_EQ_CONFIG_S

【说明】

定义音频均衡器功能配置信息结构体。

【定义】

```
typedef struct hiAUDIO_EQ_CONFIG_S
{
    HI_S8  s8GaindB[VQE_EQ_BAND_NUM];
    HI_S32 s32Reserved;
} AUDIO_EQ_CONFIG_S;
```

【成员】

成员名称	描述
s8GaindB	EQ 频段增益调节，频段依次为 100Hz,200Hz, 250Hz, 350Hz, 500Hz, 800Hz, 1.2kHz, 2.5kHz, 4kHz, 8kHz.其中，8kHz 只有在工作采样率（s32WorkSampleRate）配置为 16k 时有效。每个频段取值范围：[-100, 20]dB；调整步长 1dB。
s32Reserved	保留。

【注意事项】

该功能没有自动模式，需通过高级参数的配置生效。

【相关数据类型及接口】

无。

AI_HDR_CONFIG_S

【说明】

定义音频高动态范围功能配置信息结构体。

【定义】

```
typedef struct hiAI_HDR_CONFIG_S
{
    HI_BOOL bUsrMode;
    HI_S32 s32MinGaindB;
    HI_S32 s32MaxGaindB;
    HI_S32 s32MicGaindB;
    HI_S32 s32MicGainStepdB;
    pFuncGainCallBack pcallback;
} AI_HDR_CONFIG_S;
```

【成员】

成员名称	描述
bUsrMode	是否采用用户模式。 0：自动模式； 1：用户模式。 默认为 0，表示采用默认的内置 audio codec 方式调节。
s32MinGaindB	CODEC 允许最小配置增益，取值范围[0, 120]
s32MaxGaindB	CODEC 允许最大配置增益，取值范围[0, 120]
s32MicGaindB	CODEC 当前配置增益，取值范围[s32MinGaindB, s32MaxGaindB]
s32MicGainStepdB	增益调整步长,默认为 2，取值范围[1,3]
pcallback	修改 CODEC 增益的函数指针

【注意】

- 该功能需要用户根据 CODEC 配置相应的参数。
- 当用户模式开启时，高级参数才生效，否则按照 AI_VQE_CONFIG_S/ AI_TALKVQE_CONFIG_S/AI_HIFIVQE_CONFIG_S 中的工作模式 enWorkstate 对应的默认值来配置。
- 配置参数时，只有在用户模式开启时，才会对高级参数做正确性检查，只有正确的高级参数才能配置成功

【相关数据类型及接口】

无

VQE_DRC_SECNUM

【说明】

定义 DRC 动态曲线可配置级数。

【定义】

```
#define VQE_DRC_SECNUM 5
```

【注意事项】

无。

【相关数据类型及接口】

AI_DRC_CONFIG_S

AI_DRC_CONFIG_S

【说明】

定义音频动态压缩控制功能配置信息结构体。

【定义】

```
typedef struct hiAI_DRC_CONFIG_S
{
    HI_BOOL bUsrMode;
    HI_S16  s16AttackTime;
    HI_S16  s16ReleaseTime;
    HI_S16 s16OldLevDb[VQE_DRC_SECNUM];
    HI_S16 s16NewLevDb[VQE_DRC_SECNUM];
} AI_DRC_CONFIG_S;
```

【成员】

成员名称	描述
bUsrMode	是否采用用户模式。 0：自动模式； 1：用户模式。 默认为 0 关闭。
s16AttackTime	信号从大变小的时间(ms)，取值范围[10, 250]
s16ReleaseTime	信号从小变大的时间(ms)，取值范围[10, 250]
s16OldLevDb	动态曲线调整前拐点电平 Q4，取值范围[-1440,0]， 从大往小存储，即 s16OldLevDb[n]>= s16OldLevDb[n+1]；默认值[0, -472, -792, -960, -1280]。同时必须满足

成员名称	描述
	s16NewLevDb[n]>= s16OldLevDb[n]。
s16NewLevDb	动态曲线调整后拐点电平 Q4，取值范围[-1440,0]， 从大往小存储 ，即 s16NewLevDb [n]>= s16NewLevDb [n+1]；默认值[0, -174, -410, -608, -1021]。同时必须满足 s16NewLevDb[n]>= s16OldLevDb[n]。

【注意】

- 当用户模式开启时，高级参数才生效，否则按照 AI_HIFIVQE_CONFIG_S 中的工作模式 enWorkstate 对应的默认值来配置。
- 配置参数时，只有在用户模式开启时，才会对高级参数做正确性检查，只有正确的高级参数才能配置成功
- 该功能只支持 48KHz 工作采样率。

【相关数据类型及接口】

无

VQE_PEQ_BAND_NUM

【说明】

定义 PEQ 功能可调节的最大频段数。

【定义】

```
#define VQE_PEQ_BAND_NUM 10
```

【注意事项】

无。

【相关数据类型及接口】

AUDIO_EQ_CONFIG_S

AI_PEQ_CONFIG_S

【说明】

定义音频参量均衡器配置信息结构体。

【定义】

```
typedef struct hiAI_PEQ_CONFIG_S
{
    HI_BOOL bUsrMode;
    HI_U32 u32BandNum;
    HI_U8  u8FilterType[VQE_PEQ_BAND_NUM];
    HI_S8  s8GaindB[VQE_PEQ_BAND_NUM];
```

```
    HI_U16 u16Frequency[VQE_PEQ_BAND_NUM];
    HI_U16 u16Q[VQE_PEQ_BAND_NUM];
} AI_PEQ_CONFIG_S;
```

【成员】

成员名称	描述
bUsrMode	是否采用用户模式。 0：自动模式； 1：用户模式。 默认为 0 关闭。
u32BandNum	调节频段数，取值范围(0, 10]
u8FilterType	滤波器类型，包括高通滤波（HP）、低通滤波(LP)、高通搁架滤波(HS)、低通搁架滤波(LS)、峰值滤波(PK)，范围:[0: HP, 1: LS, 2: PK, 3: HS 4: LP]
s8GaindB	PEQ 频段增益(db)，HP,LP 滤波器增益为 0dB，其他类型的滤波器范围[-15, 15]
u16Frequency	PEQ 频段中心频率(Hz), HP /LS 对应范围: [20, 4000], PK 对应范围[20, 22000],HS/LP 对应范围[4000, 22000]
u16Q	Q 值，HS/LS 对应范围[7, 10], PK 对应范围[5, 100], HP/LP 为 7

【注意】

- 当用户模式开启时，高级参数才生效，否则按照 AI_HIFIVQE_CONFIG_S 中的工作模式 enWorkstate 对应的默认值来配置。
- 配置参数时，只有在用户模式开启时，才会对高级参数做正确性检查，只有正确的高级参数才能配置成功。
- 该功能只支持 48KHz 工作采样率。

【相关数据类型及接口】

无

AI_VQE_CONFIG_S

【说明】

定义音频输入声音质量增强配置信息结构体。

【定义】

```
typedef struct hiAI_VQE_CONFIG_S
{
    HI_S32            bHpfOpen;
```

```
    HI_S32            bAecOpen;
    HI_S32            bAnrOpen;
    HI_S32            bRnrOpen;
    HI_S32            bAgcOpen;
    HI_S32            bEqOpen;
    HI_S32            bHdrOpen;
    HI_S32            s32WorkSampleRate;
    HI_S32            s32FrameSample;
    VQE_WORKSTATE_E     enWorkstate;
    AUDIO_HPF_CONFIG_S  stHpfCfg;
    AI_AEC_CONFIG_S     stAecCfg;
    AUDIO_ANR_CONFIG_S stAnrCfg;
    AI_RNR_CONFIG_S     stRnrCfg;
    AUDIO_AGC_CONFIG_S  stAgcCfg;
    AUDIO_EQ_CONFIG_S   stEqCfg;
    AI_HDR_CONFIG_S     stHdrCfg;
} AI_VQE_CONFIG_S;
```

【成员】

成员名称	描述
bHpfOpen	高通滤波功能是否使能标志。
bAecOpen	回声抵消功能是否使能标志。
bAnrOpen	语音降噪功能是否使能标志。
bRnrOpen	录音噪声消除功能是否使能标志。
bAgcOpen	自动增益控制功能是否使能标志
bEqOpen	均衡器功能是否使能标志
bHdrOpen	高动态范围功能是否使能标志
s32WorkSampleRate	工作采样频率。该参数为内部功能算法工作采样率。 取值范围：8KHz/16KHz/48KHz。默认值为 8KHz。(Rnr/Hpf/Hdr 支持 8KHz/16KHz/48KHz，其余支持 8KHz/16KHz)。
s32FrameSample	VQE 的帧长，即采样点数目。 支持范围[80, 4096]。
enWorkstate	工作模式。
stHpfCfg	高通滤波功能相关配置信息。
stAecCfg	回声抵消功能相关配置信息。
stAnrCfg	语音降噪功能相关配置信息。

成员名称	描述
stRnrCfg	录音噪声消除功能相关配置信息。
stAgcCfg	自动增益控制相关配置信息。
stEqCfg	均衡器相关配置信息。
stHdrCfg	高动态范围功能相关配置信息。

【注意事项】

- Vqe 各个场景模式下的默认参数配置如表 9-8 所示：

表9-8 Vqe 各个场景模式下的默认参数配置表

参数		场景模式		
		COMMON	MUSIC	NOISY
HPF	enHpfFreq	AUDIO_HPF_FREQ_120	AUDIO_HPF_FREQ_120	AUDIO_HPF_FREQ_120
AEC	s8CngMode	1	1	1
ANR	s16NrIntensity	15	8	15
	s16NoiseDbThr	45	60	45
	s8SpProSwitch	1	1	0
RNR	s32NrMode	1	0	1
	s32MaxNrLevel	15	15	15
	s32NoiseThresh	50	50	50
AGC	s8TargetLevel	-2	-2	-2
	s8NoiseFloor	-40	-40	-40
	s8MaxGain	15	10	15
	s8AdjustSpeed	10	5	10
	s8ImproveSNR	2	0	2
	s8UseHighPassFilt	0	0	0
	s8OutputMode	0	0	0
	s16NoiseSupSwitch	1	0	1

- Hi3518EV200 不支持 RNR、HDR 功能。
- Hi3519V100 不支持此数据结构。

- HDR 需要在有内置 audio codec 的前提下，才能配置为自动模式。

【相关数据类型及接口】

无。

AI_HIFIVQE_CONFIG_S

【说明】

定义音频输入声音质量增强（HiFi）配置信息结构体。

【定义】

```
typedef struct hiAI_HIFIVQE_CONFIG_S
{
    HI_U32              u32OpenMask;
    HI_S32            s32WorkSampleRate;
    HI_S32            s32FrameSample;
    VQE_WORKSTATE_E     enWorkstate;
    AUDIO_HPF_CONFIG_S  stHpfCfg;
    AI_RNR_CONFIG_S     stRnrCfg;
    AI_HDR_CONFIG_S     stHdrCfg;
    AI_DRC_CONFIG_S     stDrcCfg;
    AI_PEQ_CONFIG_S     stPeqCfg;
} AI_HIFIVQE_CONFIG_S;
```

【成员】

成员名称	描述
u32OpenMask	HiFi Vqe 的各功能使能的 Mask 值，u32OpenMask 不能为 0。
s32WorkSampleRate	工作采样频率。该参数为内部功能算法工作采样率。 取值范围：48KHz。
s32FrameSample	VQE 的帧长，即采样点数目。 支持范围：[80, 4096]。
enWorkstate	工作模式。
stHpfCfg	高通滤波功能相关配置信息。
stRnrCfg	录音噪声消除功能相关配置信息。
stHdrCfg	高动态范围功能相关配置信息。
stDrcCfg	动态压缩控制功能配置信息。
stPeqCfg	参量均衡器相关配置信息。

【注意事项】

- HiFi Vqe 各个场景模式下的默认参数配置如表 9-9 所示：

表9-9 HiFi Vqe 各个场景模式下的默认参数配置表

参数		场景模式		
		COMMON	MUSIC	NOISY
HPF	enHpfFreq	AUDIO_HPF_FREQ_120	AUDIO_HPF_FREQ_120	AUDIO_HPF_FREQ_120
RNR	s32NrMode	1	0	1
	s32MaxNrLevel	15	15	15
	s32NoiseThresh	50	50	50
DRC	s16AttackTime	24	24	24
	s16ReleaseTime	100	100	100
	s16OldLevDb	{0, -472, -792, -960, -1280}	{0, -472, -792, -960, -1280}	{0, -472, -792, -960, -1280}
	s16NewLevDb	{0, -174, -410, -608, -1021}	{0, -174, -410, -608, -1021}	{0, -174, -410, -608, -1021}

- Hi3518EV200 不支持 HiFi Vqe。
- PEQ 在自动模式下（bUsrMode 为假），功能不生效。
- HDR 需要在有内置 audio codec 的前提下，才能配置为自动模式。

【相关数据类型及接口】

无。

AI_TALKVQE_CONFIG_S

【说明】

定义音频输入声音质量增强（Talk）配置信息结构体。

【定义】

```
typedef struct hiAI_TALKVQE_CONFIG_S
{
    HI_U32              u32OpenMask;
    HI_S32            s32WorkSampleRate;
    HI_S32            s32FrameSample;
    VQE_WORKSTATE_E     enWorkstate;
    AUDIO_HPF_CONFIG_S  stHpfCfg;
    AI_AEC_CONFIG_S     stAecCfg;
    AUDIO_ANR_CONFIG_S  stAnrCfg;
```

```
    AUDIO_AGC_CONFIG_S  stAgcCfg;
    AUDIO_EQ_CONFIG_S   stEqCfg;
    AI_HDR_CONFIG_S     stHdrCfg;
} AI_TALKVQE_CONFIG_S
```

【成员】

成员名称	描述
u32OpenMask	Talk Vqe 的各功能使能的 Mask 值，u32OpenMask 不能为 0。
s32WorkSampleRate	工作采样频率。该参数为内部功能算法工作采样率。 取值范围：8KHz/16KHz。默认值为 8KHz。
s32FrameSample	VQE 的帧长，即采样点数目。 支持范围[80, 4096]。
enWorkstate	工作模式。
stHpfCfg	高通滤波功能相关配置信息。
stAecCfg	回声抵消功能相关配置信息。
stAnrCfg	语音降噪功能相关配置信息。
stAgcCfg	自动增益控制相关配置信息。
stEqCfg	均衡器相关配置信息。
stHdrCfg	高动态范围功能相关配置信息。

【注意事项】

- Talk Vqe 各个场景模式下的默认参数配置如表 9-10 所示：

表9-10 Talk Vqe 各个场景模式下的默认参数配置表

参数		场景模式		
		COMMON	MUSIC	NOISY
HPF	enHpfFreq	AUDIO_HPF_FREQ_120	AUDIO_HPF_FREQ_120	AUDIO_HPF_FREQ_120
AEC	s8CngMode	1	1	1
ANR	s16NrIntensity	15	8	15
	s16NoiseDbThr	45	60	45
	s8SpProSwitch	1	1	0
AGC	s8TargetLevel	-2	-2	-2

参数		场景模式		
		COMMON	MUSIC	NOISY
	s8NoiseFloor	-40	-40	-40
	s8MaxGain	15	10	15
	s8AdjustSpeed	10	5	10
	s8ImproveSNR	2	0	2
	s8UseHighPassFilt	0	0	0
	s8OutputMode	0	0	0
	s16NoiseSupSwitch	1	0	1

- Hi3518EV200、Hi3516A 不支持 Talk Vqe。
- HDR 需要在有内置 audio codec 的前提下，才能配置为自动模式。

【相关数据类型及接口】

无。

AO_VQE_CONFIG_S

【说明】

定义音频输出声音质量增强配置信息结构体。此结构体在 Hi3518EV200、Hi3516A 上的定义与 Hi3519V100 上有少许差异。

【定义】

Hi3518EV200、Hi3516A 上的定义：

```
typedef struct hiAO_VQE_CONFIG_S
{
    HI_S32            bHpfOpen;
    HI_S32            bAnrOpen;
    HI_S32            bAgcOpen;
    HI_S32            bEqOpen;
    HI_S32            s32WorkSampleRate;
    HI_S32            s32FrameSample;
    VQE_WORKSTATE_E     enWorkstate;
    AUDIO_HPF_CONFIG_S  stHpfCfg;
    AUDIO_ANR_CONFIG_S  stAnrCfg;
    AUDIO_AGC_CONFIG_S  stAgcCfg;
    AUDIO_EQ_CONFIG_S   stEqCfg;
} AO_VQE_CONFIG_S;
```

Hi3519V100 上的定义：

```
typedef struct hiAO_VQE_CONFIG_S
{
    HI_U32              u32OpenMask;
    HI_S32            s32WorkSampleRate;
    HI_S32            s32FrameSample;
    VQE_WORKSTATE_E     enWorkstate;
    AUDIO_HPF_CONFIG_S  stHpfCfg;
    AUDIO_ANR_CONFIG_S  stAnrCfg;
    AUDIO_AGC_CONFIG_S  stAgcCfg;
    AUDIO_EQ_CONFIG_S   stEqCfg;
} AO_VQE_CONFIG_S;
```

【成员】

项目名	成员名称	描述
Hi3518EV200, Hi3516A	bHpfOpen	高通滤波功能是否使能标志。
	bAnrOpen	语音降噪功能是否使能标志。
	bAgcOpen	自动增益控制功能是否使能标志
	bEqOpen	均衡器功能是否使能标志
	s32WorkSampleRate	工作采样频率。该参数为内部功能算法工作采样率。 取值范围：8KHz/16KHz/48KHz。默认值为8KHz。(仅 Hpf 支持 48KHz)
	s32FrameSample	VQE 的帧长，即采样点数目。 支持范围：[80, 4096]。
	enWorkstate	工作模式。
	stHpfCfg	高通滤波功能相关配置信息。
	stAnrCfg	语音降噪功能相关配置信息。
	stAgcCfg	自动增益控制相关配置信息。
	stEqCfg	均衡器相关配置信息。
Hi3519V100	u32OpenMask	AO Vqe 的各功能使能的 Mask 值。 u32OpenMask 不能为 0。
	s32WorkSampleRate	工作采样频率。该参数为内部功能算法工作采样率。 取值范围：8KHz/16KHz/48KHz。默认值为8KHz。(仅 Hpf 支持 48KHz)
	s32FrameSample	VQE 的帧长，即采样点数目。

项目名	成员名称	描述
		支持范围：[80, 4096]。
	enWorkstate	工作模式。
	stHpfCfg	高通滤波功能相关配置信息。
	stAnrCfg	语音降噪功能相关配置信息。
	stAgcCfg	自动增益控制相关配置信息。
	stEqCfg	均衡器相关配置信息。

【注意事项】

DnVQE 各个场景模式下的默认参数配置如表 9-11 所示：

表9-11 DnVQE 各个场景模式下的默认参数配置表

参数		场景模式		
		COMMON	MUSIC	NOISY
HPF	enHpfFreq	AUDIO_HPF_FREQ_120	AUDIO_HPF_FREQ_120	AUDIO_HPF_FREQ_120
ANR	s16NrIntensity	15	8	15
	s16NoiseDbThr	45	60	45
	s8SpProSwitch	1	1	0
AGC	s8TargetLevel	-2	-2	-2
	s8NoiseFloor	-40	-40	-40
	s8MaxGain	15	10	15
	s8AdjustSpeed	10	5	10
	s8ImproveSNR	2	0	2
	s8UseHighPassFilt	0	0	0
	s8OutputMode	0	0	0
	s16NoiseSupSwitch	1	0	1

【相关数据类型及接口】

无。

AUDIO_STREAM_S

【说明】

定义音频码流结构体。

【定义】

```
typedef struct hiAUDIO_STREAM_S
{
    HI_U8 *pStream;        /* the virtual address of stream */
    HI_U32 u32PhyAddr;     /* the physics address of stream */
    HI_U32 u32Len;         /* stream lenth, by bytes */
    HI_U64 u64TimeStamp;   /* frame time stamp*/
    HI_U32 u32Seq;         /* frame seq,if stream is not a valid frame,
u32Seq is 0*/
} AUDIO_STREAM_S;
```

【成员】

成员名称	描述
pStream	音频码流数据指针。
u32PhyAddr	音频码流的物理地址。
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。

【注意事项】

无。

【相关数据类型及接口】

HI_MPI_AENC_GetStream

AO_CHN_STATE_S

【说明】

音频输出通道的数据缓存状态结构体。

【定义】

```
typedef struct hiAO_CHN_STATE_S
{
    HI_U32              u32ChnTotalNum;
    HI_U32              u32ChnFreeNum;
    HI_U32              u32ChnBusyNum;
} AO_CHN_STATE_S;
```

【成员】

成员名称	描述
u32ChnTotalNum	输出通道总的缓存块数。
u32ChnFreeNum	可用的空闲缓存块数。
u32ChnBusyNum	被占用缓存块数。

【注意事项】

无。

【相关数据类型及接口】

HI_MPI_AO_QueryChnStat

AUDIO_TRACK_MODE_E

【说明】

定义音频设备声道模式类型。

【定义】

```
typedef enum hiAUDIO_TRACK_MODE_E
{
    AUDIO_TRACK_NORMAL      = 0,
    AUDIO_TRACK_BOTH_LEFT   = 1,
    AUDIO_TRACK_BOTH_RIGHT  = 2,
    AUDIO_TRACK_EXCHANGE    = 3,
    AUDIO_TRACK_MIX         = 4,
    AUDIO_TRACK_LEFT_MUTE   = 5,
    AUDIO_TRACK_RIGHT_MUTE  = 6,
    AUDIO_TRACK_BOTH_MUTE   = 7,
    AUDIO_TRACK_BUTT
} AUDIO_TRACK_MODE_E;
```

【成员】

成员名称	描述
AUDIO_TRACK_NORMAL	正常模式，不做处理
AUDIO_TRACK_BOTH_LEFT	两个声道全部为左声道声音
AUDIO_TRACK_BOTH_RIGHT	两个声道全部为右声道声音
AUDIO_TRACK_EXCHANGE	左右声道数据互换，左声道为右声道声音，右声道为左声道声音
AUDIO_TRACK_MIX	左右两个声道输出为左右声道相加（混音）

成员名称	描述
AUDIO_TRACK_LEFT_MUTE	左声道静音，右声道播放原右声道声音
AUDIO_TRACK_RIGHT_MUTE	右声道静音，左声道播放原左声道声音
AUDIO_TRACK_BOTH_MUTE	左右声道均静音

【注意事项】

无。

【相关数据类型及接口】

- HI_MPI_AI_SetTrackMode
- HI_MPI_AO_SetTrackMode

AUDIO_FADE_RATE_E

【说明】

定义音频输出设备淡入淡出速度类型。

【定义】

```
typedef enum hiAUDIO_FADE_RATE_E
{
    AUDIO_FADE_RATE_1   = 0,
    AUDIO_FADE_RATE_2   = 1,
    AUDIO_FADE_RATE_4   = 2,
    AUDIO_FADE_RATE_8   = 3,
    AUDIO_FADE_RATE_16  = 4,
    AUDIO_FADE_RATE_32  = 5,
    AUDIO_FADE_RATE_64  = 6,
    AUDIO_FADE_RATE_128 = 7,
    AUDIO_FADE_RATE_BUTT
} AUDIO_FADE_RATE_E;
```

【成员】

成员名称	描述
AUDIO_FADE_RATE_1	1 个采样点改变一次
AUDIO_FADE_RATE_2	2 个采样点改变一次
AUDIO_FADE_RATE_4	4 个采样点改变一次
AUDIO_FADE_RATE_8	8 个采样点改变一次
AUDIO_FADE_RATE_16	16 个采样点改变一次

成员名称	描述
AUDIO_FADE_RATE_32	32 个采样点改变一次
AUDIO_FADE_RATE_64	64 个采样点改变一次
AUDIO_FADE_RATE_128	128 个采样点改变一次

【注意事项】

无。

【相关数据类型及接口】

无。

AUDIO_FADE_S

【说明】

音频输出设备淡入淡出配置结构体。

【定义】

```
typedef struct hiAUDIO_FADE_S
{
    HI_BOOL           bFade;
    AUDIO_FADE_RATE_E enFadeInRate;
    AUDIO_FADE_RATE_E enFadeOutRate;
} AUDIO_FADE_S;
```

【成员】

成员名称	描述
bFade	是否开启淡入淡出功能。 HI_TRUE：开启淡入淡出功能。 HI_FALSE：关闭淡入淡出功能。
enFadeInRate	音频输出设备音量淡入速度。
enFadeOutRate	音频输出设备音量淡出速度。

【注意事项】

无。

【相关数据类型及接口】

HI_MPI_AO_SetMute

G726_BPS_E

【说明】

定义 G.726 编解码协议速率。

【定义】

```
typedef enum hiG726_BPS_E
{
    G726_16K = 0,
    G726_24K,
    G726_32K,
    G726_40K,
    MEDIA_G726_16K,     /* G726 16kbit/s for ASF */
    MEDIA_G726_24K,     /* G726 24kbit/s for ASF */
    MEDIA_G726_32K,     /* G726 32kbit/s for ASF */
    MEDIA_G726_40K,     /* G726 40kbit/s for ASF */
    G726_BUTT,
} G726_BPS_E;
```

【成员】

成员名称	描述
G726_16K	16kbit/s G.726，请参见“RFC3551 文档 4.5.4 G72616”。
G726_24K	24kbit/s G.726，请参见“RFC3551 文档 4.5.4 G72624”。
G726_32K	32kbit/s G.726，请参见“RFC3551 文档 4.5.4 G72632”。
G726_40K	40kbit/s G.726，请参见“RFC3551 文档 4.5.4 G72640”。
MEDIA_G726_16K	G726 16kbit/s for ASF。
MEDIA_G726_24K	G726 24kbit/s for ASF。
MEDIA_G726_32K	G726 32kbit/s for ASF。
MEDIA_G726_40K	G726 40kbit/s for ASF。

【注意事项】

无。

【相关数据类型及接口】

无。

ADPCM_TYPE_E

【说明】

定义 ADPCM 编解码协议类型。

【定义】

```
typedef enum hiADPCM_TYPE_E
{
    ADPCM_TYPE_DVI4 = 0,
    ADPCM_TYPE_IMA,
    ADPCM_TYPE_ORG_DVI4,
    ADPCM_TYPE_BUTT,
} ADPCM_TYPE_E;
```

【成员】

成员名称	描述
ADPCM_TYPE_DVI4	32kbit/s ADPCM(DVI4)。
ADPCM_TYPE_IMA	32kbit/s ADPCM(IMA)。
ADPCM_TYPE_ORG_DVI4	32kbit/s ADPCM(ORG_DVI4)。

【注意事项】

无。

【相关数据类型及接口】

无。

AUDIO_SAVE_FILE_INFO_S

【说明】

定义音频保存文件功能配置信息结构体。

【定义】

```
typedef struct hiAUDIO_SAVE_FILE_INFO_S
{
    HI_BOOL  bCfg;
    HI_CHAR     aFilePath[MAX_AUDIO_FILE_PATH_LEN];
    HI_CHAR     aFileName[MAX_AUDIO_FILE_NAME_LEN];
    HI_U32   u32FileSize;  /*in KB*/
} AUDIO_SAVE_FILE_INFO_S;
```

【成员】

成员名称	描述
bCfg	配置使能开关。

成员名称	描述
aFilePath[MAX_AUDIO_FILE_PATH_LEN]	音频文件保存路径
aFileName[MAX_AUDIO_FILE_NAME_LEN]	音频文件保存名称
u32FileSize	文件大小，取值范围[1, 10240]KB。

【注意事项】

无。

【相关数据类型及接口】

无。

AUDIO_FILE_STATUS_S

【说明】

定义音频文件保存状态结构体。

【定义】

```
typedef struct hiAUDIO_FILE_STATUS_S
{
    HI_BOOL     bSaving;
} AUDIO_FILE_STATUS_S;
```

【成员】

成员名称	描述
bSaving	是否处于存文件状态。 • HI_TRUE：处于存文件状态； • HI_FALSE：不处于存文件状态。

【注意事项】

无。

【相关数据类型及接口】

无。

9.4.2 音频编码

音频编码相关数据类型、数据结构定义如下：

- AENC_MAX_CHN_NUM：定义音频编码通道的最大个数。
- AENC_ATTR_G711_S：定义 G.711 编码协议属性结构体。

- AENC_ATTR_G726_S：定义 G.726 编码协议属性结构体。
- AENC_ATTR_ADPCM_S：定义 ADPCM 编码协议属性结构体。
- AENC_ATTR_LPCM_S：定义 LPCM 编码协议属性结构体。
- AENC_CHN_ATTR_S：定义音频编码通道属性结构体。
- AENC_ENCODER_S：定义编码器属性结构体。

AENC_MAX_CHN_NUM

【说明】

定义音频编码通道的最大个数。

【定义】

```
#define AENC_MAX_CHN_NUM        32
```

【注意事项】

无。

【相关数据类型及接口】

无

AENC_ATTR_G711_S

【说明】

定义 G.711 编码协议属性结构体。

【定义】

```
typedef struct hiAENC_ATTR_G711_S
{
    HI_U32 resv;
}AENC_ATTR_G711_S;
```

【成员】

成员名称	描述
resv	待扩展用（目前暂未使用）。

【注意事项】

无。

【相关数据类型及接口】

无。

AENC_ATTR_G726_S

【说明】

定义 G.726 编码协议属性结构体。

【定义】

```
typedef struct hiAENC_ATTR_G726_S
{
    G726_BPS_E  enG726bps;
}AENC_ATTR_G726_S;
```

【成员】

成员名称	描述
enG726bps	G.726 协议码率。

【注意事项】

无。

【相关数据类型及接口】

G726_BPS_E

AENC_ATTR_ADPCM_S

【说明】

定义 ADPCM 编码协议属性结构体。

【定义】

```
typedef struct hiAENC_ATTR_ADPCM_S
{
    ADPCM_TYPE_E    enADPCMType;
}AENC_ATTR_ADPCM_S;
```

【成员】

成员名称	描述
enADPCMType	ADPCM 类型。

【注意事项】

无。

【相关数据类型及接口】

ADPCM_TYPE_E

AENC_ATTR_LPCM_S

【说明】

定义 LPCM 编码协议属性结构体。

【定义】

```
typedef struct hiAENC_ATTR_LPCM_S
{
    HI_U32 resv;          /*reserve item*/
}AENC_ATTR_LPCM_S;
```

【成员】

成员名称	描述
resv	待扩展用（目前暂未使用）。

【注意事项】

无。

【相关数据类型及接口】

无。

AENC_CHN_ATTR_S

【说明】

定义音频编码通道属性结构体。

【定义】

```
typedef struct hiAENC_CHN_ATTR_S
{
    PAYLOAD_TYPE_E  enType;
    HI_U32  u32PtNumPerFrm;
    HI_U32  u32BufSize; /*buffer size，以帧为单位，[2～MAX_AUDIO_FRAME_NUM]*/
    HI_VOID *pValue;
}AENC_CHN_ATTR_S;
```

【成员】

成员名称	描述
enType	音频编码协议类型。 静态属性。

成员名称	描述
u32PtNumPerFrm	音频编码协议对应的帧长（编码时收到的音频帧长小于等于该帧长都可以进行编码）。
u32BufSize	音频编码缓存大小。 取值范围：[2, MAX_AUDIO_FRAME_NUM]，以帧为单位。 静态属性。
pValue	具体协议属性指针。 静态属性。

【注意事项】

无。

【相关数据类型及接口】

无。

AENC_ENCODER_S

【说明】

定义编码器属性结构体。

【定义】

```
typedef struct hiAENC_ENCODER_S
{
    PAYLOAD_TYPE_E  enType;
    HI_U32     u32MaxFrmLen;
    HI_CHAR    aszName[16];
    HI_S32     (*pfnOpenEncoder)(HI_VOID *pEncoderAttr, HI_VOID
**ppEncoder);
    HI_S32     (*pfnEncodeFrm)(HI_VOID *pEncoder, const AUDIO_FRAME_S
*pstData, HI_U8 *pu8Outbuf,HI_U32 *pu32OutLen);
    HI_S32     (*pfnCloseEncoder)(HI_VOID *pEncoder);
}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。

成员名称	描述
pfnOpenEncoder	打开编码器的函数指针。
pfnEncodeFrm	进行编码的函数指针。
pfnCloseEncoder	关闭编码器的函数指针。

【注意事项】

请参见《音频组件 API 参考》。

【相关数据类型及接口】

HI_MPI_AENC_RegeisterEncoder

9.4.3 音频解码

音频解码相关数据类型、数据结构定义如下：

- MAX_AUDIO_FRAME_NUM：定义最大音频解码缓存帧数。
- ADEC_MAX_CHN_NUM：定义音频解码通道的最大个数。
- ADEC_ATTR_G711_S：定义 G.711 解码协议属性结构体。
- ADEC_ATTR_G726_S：定义 G.726 解码协议属性结构体。
- ADEC_ATTR_ADPCM_S：定义 ADPCM 解码协议属性结构体。
- ADEC_ATTR_LPCM_S：定义 LPCM 解码协议属性结构体。
- ADEC_MODE_E：定义解码方式。
- ADEC_CHN_ATTR_S：定义解码通道属性结构体。
- ADEC_DECODER_S：定义解码器属性结构体。
- AUDIO_FRAME_INFO_S：定义解码后的音频帧信息结构体。

MAX_AUDIO_FRAME_NUM

【说明】

定义最大音频解码缓存帧数。

【定义】

```
#define MAX_AUDIO_FRAME_NUM       300
```

【注意事项】

无。

【相关数据类型及接口】

无。

ADEC_MAX_CHN_NUM

【说明】

定义音频解码通道的最大个数。

【定义】

```
#define ADEC_MAX_CHN_NUM        32
```

【注意事项】

无。

【相关数据类型及接口】

无

ADEC_ATTR_G711_S

【说明】

定义 G.711 解码协议属性结构体。

【定义】

```
typedef struct hiADEC_ATTR_G711_S
{
    HI_U32  resv;
}ADEC_ATTR_G711_S;
```

【成员】

成员名称	描述
resv	待扩展用（目前暂未使用）。

【注意事项】

无。

【相关数据类型及接口】

无。

ADEC_ATTR_G726_S

【说明】

定义 G.726 解码协议属性结构体。

【定义】

```
typedef struct hiADEC_ATTR_G726_S
{
```

```
    G726_BPS_E  enG726bps;
}ADEC_ATTR_G726_S;
```

【成员】

成员名称	描述
enG726bps	G.726 协议码率。

【注意事项】

无。

【相关数据类型及接口】

G726_BPS_E

ADEC_ATTR_ADPCM_S

【说明】

定义 ADPCM 解码协议属性结构体。

【定义】

```
typedef struct hiADEC_ATTR_ADPCM_S
{
    ADPCM_TYPE_E    enADPCMType;
}ADEC_ATTR_ADPCM_S;
```

【成员】

成员名称	描述
enADPCMType	ADPCM 类型。

【注意事项】

无。

【相关数据类型及接口】

ADPCM_TYPE_E

ADEC_ATTR_LPCM_S

【说明】

定义 LPCM 解码协议属性结构体。

【定义】

```
typedef struct hiADEC_ATTR_LPCM_S
```

```
{
    HI_U32 resv;
}ADEC_ATTR_LPCM_S;
```

【成员】

成员名称	描述
resv	待扩展用（目前暂未使用）。

【注意事项】

无。

【相关数据类型及接口】

无。

ADEC_MODE_E

【说明】

定义解码方式。

【定义】

```
typedef enum hiADEC_MODE_E
{
    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*/
    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*/
    ADEC_MODE_BUTT
}ADEC_MODE_E;
```

【成员】

成员名称	描述
ADEC_MODE_PACK	Pack 方式解码。
ADEC_MODE_STREAM	stream 方式解码。

【注意事项】

- pack 方式用于用户确认当前码流包为一帧数据编码结果的情况下，解码器会直接进行对其解码，如果不是一帧，解码器会出错。这种模式的效率比较高，在使用 AENC 模块编码的码流包如果没有破坏，均可以使用此方式解码。
- stream 方式用于用户不能确认当前码流包是不是一帧数据的情况下，解码器需要对码流进行判断并缓存，此工作方式的效率低下，一般用于读文件码流送解码或者不确定码流包边界的情况。当然由于语音编码码流长度固定，很容易确定在码流中的帧边界，推荐使用 pack 方式解码。

【相关数据类型及接口】

无。

ADEC_CHN_ATTR_S

【说明】

定义解码通道属性结构体。

【定义】

```
typedef struct hiADEC_CH_ATTR_S
{
    PAYLOAD_TYPE_E  enType;
    HI_U32          u32BufSize;
    ADEC_MODE_E     enMode;
    HI_VOID         *pValue;
}ADEC_CHN_ATTR_S;
```

【成员】

成员名称	描述
enType	音频解码协议类型。 静态属性。
u32BufSize	音频解码缓存大小。 取值范围：[2, MAX_AUDIO_FRAME_NUM]，以帧为单位。 静态属性。
enMode	解码方式。 静态属性。
pValue	具体协议属性指针。

【注意事项】

无。

【相关数据类型及接口】

无。

ADEC_DECODER_S

【说明】

定义解码器属性结构体。

【定义】

```
typedef struct hiADEC_DECODER_S
{
    PAYLOAD_TYPE_E  enType;
    HI_CHAR         aszName[16];
    HI_S32          (*pfnOpenDecoder)(HI_VOID *pDecoderAttr, HI_VOID
**ppDecoder);
    HI_S32          (*pfnDecodeFrm)(HI_VOID *pDecoder, HI_U8
**pu8Inbuf,HI_S32 *ps32LeftByte, HI_U16 *pu16Outbuf,HI_U32
*pu32OutLen,HI_U32 *pu32Chns);
    HI_S32          (*pfnGetFrmInfo)(HI_VOID *pDecoder, HI_VOID *pInfo);
    HI_S32          (*pfnCloseDecoder)(HI_VOID *pDecoder);
    HI_S32          (*pfnResetDecoder)(HI_VOID *pDecoder);
} 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。
pfnResetDecoder	清空缓存 buffer，复位解码器。

【注意事项】

请参见《音频组件 API 参考》。

【相关数据类型及接口】

HI_MPI_ADEC_RegeisterDecoder

AUDIO_FRAME_INFO_S

【说明】

定义解码后的音频帧信息结构体。

【定义】

```
typedef struct hiAUDIO_FRAME_INFO_S
{
    AUDIO_FRAME_S *pstFrame;
    HI_U32        u32Id;
} AUDIO_FRAME_INFO_S;
```

【成员】

成员名称	描述
pstFrame	音频帧指针。
u32Id	音频帧的索引，范围[0, 49]。

【注意事项】

请参见《音频组件 API 参考》。

【相关数据类型及接口】

- HI_MPI_ADEC_GetFrame
- HI_MPI_ADEC_ReleaseFrame

9.4.4 内置 Audio Codec

内置 Audio Codec 相关数据类型、数据结构定义如下：

- ACODEC_FS_E：定义 I^2S 接口的采样率选择。
- ACODEC_MIXER_E：定义内置 Audio Codec 输入选择。
- ACODEC_VOL_CTRL：定义内置 Audio Codec 音量控制结构体。

ACODEC_FS_E

【说明】

定义 I^2S 接口的采样率选择。

【定义】

```
typedef enum hiACODEC_FS_E {
    ACODEC_FS_48000 =   0x1a,
    ACODEC_FS_24000 =   0x19,
    ACODEC_FS_12000 =   0x18,
    ACODEC_FS_44100 =   0x1a,
```

```
    ACODEC_FS_22050 =   0x19,
    ACODEC_FS_11025 =   0x18,
    ACODEC_FS_32000 =   0x1a,
    ACODEC_FS_16000 =   0x19,
    ACODEC_FS_8000  =   0x18,
    ACODEC_FS_64000 =   0x1b,
    ACODEC_FS_96000 =   0x1b,
    ACODEC_FS_BUTT = 0x1c,
} ACODEC_FS_E;
```

【成员】

成员名称	描述
ACODEC_FS_48000	48K 采样率。
ACODEC_FS_24000	24K 采样率。
ACODEC_FS_12000	12K 采样率。
ACODEC_FS_44100	44.1K 采样率。
ACODEC_FS_22050	22.05K 采样率。
ACODEC_FS_11025	11.025K 采样率。
ACODEC_FS_32000	32K 采样率。
ACODEC_FS_16000	16K 采样率。
ACODEC_FS_8000	8K 采样率。
ACODEC_FS_64000	64K 采样率。
ACODEC_FS_96000	96K 采样率。

【注意事项】

无。

【相关数据类型及接口】

无。

ACODEC_MIXER_E

【说明】

定义内置 Audio Codec 输入选择。

【定义】

For Hi3516A:

```
typedef enum hiACODEC_MIXER_E {
    /*select MICIN or LINEIN*/
    ACODEC_MIXER_LINEIN   = 0x0,
    ACODEC_MIXER_MICIN    = 0x1,
    ACODEC_MIXER_BUTT,
} ACODEC_MIXER_E;
```

For Hi3518EV200:

```
typedef enum hiACODEC_MIXER_E {
    /*select IN or IN_D*/
    ACODEC_MIXER_IN    = 0x3,
    ACODEC_MIXER_IN_D  = 0x4,
    ACODEC_MIXER_BUTT,
} ACODEC_MIXER_E;
```

For Hi3519V100:

```
typedef enum hiACODEC_MIXER_E {
    ACODEC_MIXER_IN0   = 0x0,
    ACODEC_MIXER_IN1   = 0x1,
    ACODEC_MIXER_IN_D = 0x2,
    ACODEC_MIXER_BUTT,
} ACODEC_MIXER_E
```

【成员】

For Hi3516A:

成员名称	描述
ACODEC_MIXER_LINEIN	MICPGA 选择 LINEIN 输入。
ACODEC_MIXER_MICIN	MICPGA 选择 MICIN 输入。

For Hi3518EV200:

成员名称	描述
ACODEC_MIXER_IN	MICPGA 选择单端输入。
ACODEC_MIXER_IN_D	MICPGA 选择差分输入。

For Hi3519V100:

成员名称	描述
ACODEC_MIXER_IN0	MICPGA 选择 IN0 单端输入

成员名称	描述
ACODEC_MIXER_IN1	MICPGA 选择 IN1 单端输入。
ACODEC_MIXER_IN_D	MPCPGA 选择 IN_D 差分输入

【注意事项】

目前 Hi3519V100 的音频内置 codec 输入接口分两组，编号为 0 和 1（每组包含一对左右声道），使用单端输入时，可任选其中一组，使用差分输入时，0 和 1 组各输入一路差分信号（分别作为左右声道）。

【相关数据类型及接口】

无。

ACODEC_VOL_CTRL

【说明】

定义内置 Audio Codec 音量控制结构体。

【定义】

```
typedef struct {
    /*volume control, 0x00~0x7e, 0x7F:mute*/
    unsigned int vol_ctrl;
    /*adc/dac mute control, 1:mute, 0:unmute*/
    unsigned int vol_ctrl_mute;
} ACODEC_VOL_CTRL;
```

【成员】

成员名称	描述
vol_ctrl	音量大小。
vol_ctrl_mute	静音控制。

【注意事项】

无。

【相关数据类型及接口】

无。

9.5 错误码

音频输入错误码

音频输入 API 错误码如表 9-12 所示。

表9-12 音频输入 API 错误码

错误代码	宏定义	描述
0xA0158001	HI_ERR_AI_INVALID_DEVID	音频输入设备号无效
0xA0158002	HI_ERR_AI_INVALID_CHNID	音频输入通道号无效
0xA0158003	HI_ERR_AI_ILLEGAL_PARAM	音频输入参数设置无效
0xA0158005	HI_ERR_AI_NOT_ENABLED	音频输入设备或通道没有使能
0xA0158006	HI_ERR_AI_NULL_PTR	输入参数空指针错误
0xA0158007	HI_ERR_AI_NOT_CONFIG	音频输入设备属性未设置
0xA0158008	HI_ERR_AI_NOT_SUPPORT	操作不支持
0xA0158009	HI_ERR_AI_NOT_PERM	操作不允许
0xA015800C	HI_ERR_AI_NOMEM	分配内存失败
0xA015800D	HI_ERR_AI_NOBUF	音频输入缓存不足
0xA015800E	HI_ERR_AI_BUF_EMPTY	音频输入缓存为空
0xA015800F	HI_ERR_AI_BUF_FULL	音频输入缓存为满
0xA0158010	HI_ERR_AI_SYS_NOTREADY	音频输入系统未初始化
0xA0158012	HI_ERR_AI_BUSY	音频输入系统忙
0xA0158041	HI_ERR_AI_VQE_ERR	AI VQE 处理错误

音频输出错误码

音频输出 API 错误码如表 9-13 所示。

表9-13 音频输出 API 错误码

错误代码	宏定义	描述
0xA0168001	HI_ERR_AO_INVALID_DEVID	音频输出设备号无效
0xA0168002	HI_ERR_AO_INVALID_CHNID	音频输出通道号无效
0xA0168003	HI_ERR_AO_ILLEGAL_PARAM	音频输出参数设置无效

错误代码	宏定义	描述
0xA0168005	HI_ERR_AO_NOT_ENABLED	音频输出设备或通道没使能
0xA0168006	HI_ERR_AO_NULL_PTR	输出空指针错误
0xA0168007	HI_ERR_AO_NOT_CONFIG	音频输出设备属性未设置
0xA0168008	HI_ERR_AO_NOT_SUPPORT	操作不被支持
0xA0168009	HI_ERR_AO_NOT_PERM	操作不允许
0xA016800C	HI_ERR_AO_NOMEM	系统内存不足
0xA016800D	HI_ERR_AO_NOBUF	音频输出缓存不足
0xA016800E	HI_ERR_AO_BUF_EMPTY	音频输出缓存为空
0xA016800F	HI_ERR_AO_BUF_FULL	音频输出缓存为满
0xA0168010	HI_ERR_AO_SYS_NOTREADY	音频输出系统未初始化
0xA0168012	HI_ERR_AO_BUSY	音频输出系统忙
0xA0168041	HI_ERR_AO_VQE_ERR	AO VQE 处理错误

音频编码错误码

音频编码 API 错误码如表 9-14 所示。

表9-14 音频编码 API 错误码

错误代码	宏定义	描述
0xA0178001	HI_ERR_AENC_INVALID_DEVID	音频设备号无效
0xA0178002	HI_ERR_AENC_INVALID_CHNID	音频编码通道号无效
0xA0178003	HI_ERR_AENC_ILLEGAL_PARAM	音频编码参数设置无效
0xA0178004	HI_ERR_AENC_EXIST	音频编码通道已经创建
0xA0178005	HI_ERR_AENC_UNEXIST	音频编码通道未创建
0xA0178006	HI_ERR_AENC_NULL_PTR	输入参数空指针错误
0xA0178007	HI_ERR_AENC_NOT_CONFIG	编码通道未配置
0xA0178008	HI_ERR_AENC_NOT_SUPPORT	操作不被支持
0xA0178009	HI_ERR_AENC_NOT_PERM	操作不允许
0xA017800C	HI_ERR_AENC_NOMEM	系统内存不足
0xA017800D	HI_ERR_AENC_NOBUF	编码通道缓存分配失败

错误代码	宏定义	描述
0xA017800E	HI_ERR_AENC_BUF_EMPTY	编码通道缓存空
0xA017800F	HI_ERR_AENC_BUF_FULL	编码通道缓存满
0xA0178010	HI_ERR_AENC_SYS_NOTREADY	系统没有初始化
0xA0178040	HI_ERR_AENC_ENCODER_ERR	音频编码数据错误
0xA0178041	HI_ERR_AENC_VQE_ERR	AENC VQE 处理错误

音频解码错误码

音频解码 API 错误码如表 9-15 所示。

表9-15 音频解码 API 错误码

错误代码	宏定义	描述
0xA0188001	HI_ERR_ADEC_INVALID_DEVID	音频解码设备号无效
0xA0188002	HI_ERR_ADEC_INVALID_CHNID	音频解码通道号无效
0xA0188003	HI_ERR_ADEC_ILLEGAL_PARAM	音频解码参数设置无效
0xA0188004	HI_ERR_ADEC_EXIST	音频解码通道已经创建
0xA0188005	HI_ERR_ADEC_UNEXIST	音频解码通道未创建
0xA0188006	HI_ERR_ADEC_NULL_PTR	输入参数空指针错误
0xA0188007	HI_ERR_ADEC_NOT_CONFIG	解码通道属性未配置
0xA0188008	HI_ERR_ADEC_NOT_SUPPORT	操作不被支持
0xA0188009	HI_ERR_ADEC_NOT_PERM	操作不允许
0xA018800C	HI_ERR_ADEC_NOMEM	系统内存不足
0xA018800D	HI_ERR_ADEC_NOBUF	解码通道缓存分配失败
0xA018800E	HI_ERR_ADEC_BUF_EMPTY	解码通道缓存空
0xA018800F	HI_ERR_ADEC_BUF_FULL	解码通道缓存满
0xA0188010	HI_ERR_ADEC_SYS_NOTREADY	系统没有初始化
0xA0188040	HI_ERR_ADEC_DECODER_ERR	音频解码数据错误
0xA0188041	HI_ERR_ADEC_BUF_LACK	解码输入缓存空间不够