全 志 科 技
Allwinner Technology

音频解码库 API 说明

文档履历

版本号	日期	制/修订人	内容描述
V1.0	2016-11-2	程衍	客户和调试人员使用的 Api 文档及其数据结构说明

目 录

1. 接口和流程设计

1.1. 音频解码库内部 API（仅供维护人员参考）

音频解码库内部 API 接口如下表所示。

开始解码前，应用程序首先调用 CreateAudioDecodeLib 函数创建一个解码器，然后调用 InitializeAudioDecodeLib 函数，将音频基本信息作为参数，初始化解码器。

音频解码库 APIs		
1	CreateAudioDecodeLib	创建一个音频解码库
2	DestroyAudioDecodeLib	销毁一个音频解码库，释放资源
3	InitializeAudioDecodeLib	根据音频码流信息（编码格式等）初始化音频解码库
4	DecodeAudioFrame	解码一笔音频码流
5	ParseRequestAudioBitstreamBuffer	获取码流 Buffer，用于填充码流数据
6	ParseUpdateAudioBitstreamData	填充完码流数据后，将数据提交给解码器
7	ParseAudioStreamDataSize	获取 parser buffer 数据长度
8	BitstreamQueryQuality	获取当前 bitstream buffer 数据信息:有效数据百分比，有效数据帧数
9	ParseBitstreamSeekSync	设置跳播相对时间

1.1.1. CreateAudioDecodeLib

函数原型	AudioDecoderLib* CreateAudioDecodeLib(void)
功能	创建一个音频解码库
参数	无
返回值	成功：音频解码库指针； 失败：返回 NULL；
调用说明	初始化函数，为整个 audio 接口的总指针开空间。

1.1.2. DestroyAudioDecodeLib

函数原型	int DestroyAudioDecodeLib(AudioDecoderLib* pDecoder)

功能	销毁一个音频解码库
参数	pDecoder 通过 CreateAudioDecodelib 函数创建的音频解码库指针
返回值	成功：0； 失败：<0
调用说明	无

1.1.3. InitializeAudioDecodeLib

函数原型	int InitializeAudioDecodeLib(AudioDecoderLib* pDecoder AudioStreamInfo* pAudioStreamInfo, BsInFor *pBsInFor)
功能	初始化音频解码库
参数	pDecoder: 通过 CreateAudioDecodeLib 函数创建的音频解码库指针 pAudioStreamInfo: 音频码流的基本信息，如编码格式、采样率等 pBsInFor：解码过程中交互信息
返回值	0: 表示成功； -1: 失败，不支持的编码格式或内存资源不足；
调用说明	AudioStreamInfo 中，不是所有信息都是必须的。对于 pcm,adpcm 需要的信息比较多，对于其它格式，需要的信息相对较少。

1.1.4. DecodeAudioFrame

函数原型	int DecodeAudioFrame(AudioDecoderLib* pDecoder, char* ppBuf, int* pBufSize)
功能	解码音频，解码库会对码流 Buffer 中的码流进行解码
参数	pDecoder: 通过 CreateAudioDecodeLib 函数创建的音频解码器指针； ppBuf: 返回用，解码后 pcm 数据首地址； pBufSize: 返回用，解码后 pcm 数据长度；
返回值	返回类型： AUDIO_DEC_RESULT
调用说明	无

1.1.5. ParseRequestAudioBitstreamBuffer

函数原型	int ParseRequestAudioBitstreamBuffer(AudioDecoderLib* pDecoder, int nRequireSize, unsigned char** ppBuf, int* pBufSize, unsigned char** ppRingBuf, int* pRingBufSize, int* pnoffset)
功能	向解码器请求存放码流 Buffer，用于存放数据
参数	pDecoder: 通过 CreateAudioDecodeLib 函数创建的音频解码器指针 nRequireSize: 请求 Buffer 的大小，以字节为单位；

	ppBuf: 输出参数，码流 Buffer 起始地址，等于 NULL 表示失败； pBufSize: 输出参数，码流 Buffer ppBuf 的大小； ppRingBuf: 输出参数，第二块 Buffer 的起始地址，等于 NULL 表示没有； pRingBufSize: 第二块 Buffer ppRingBuf 的大小； Pnoffset:当前文件读的位置（有多少数据已经从文件中读出）
返回值	0: 表示成功；-1: 失败；没有得到 buffer
调用说明	码流 Buffer 是一块循环 Buffer，当 Buffer 回头时，外部请求的 Buffer 被分成两段，ppBuf 和 pBufSize 返回第一段 Buffer 的地址和大小，ppRingBuf 和 pRingBufSize 返回第二段 Buffer 的地址和大小。 Buffer 没有回头时，ppRingBuf 和 pRingBufSize 返回 NULL。

1.1.6. ParseUpdateAudioBitstreamData

函数原型	int ParseUpdateAudioBitstreamData(AudioDecoderLib* pDecoder, int nFilledLen, int64_t nTimeStamp, int nOffset)
功能	向解码器提交码流数据
参数	pDecoder: 通过 CreateAudioDecodeLib 函数创建的音频解码器指针； nFilledLen: parser 释放填充的原始数据的长度，解码库可以用这段数据解码了。 nTimeStamp：当前帧时间戳 nOffset：当前读位置，为是否跳播用。
返回值	0: 表示成功；-1: 失败；
调用说明	无

1.1.7. ParseAudioStreamDataSize

函数原型	int ParseAudioStreamDataSize(AudioDecoderLib* pDecoder)
功能	获取 parser buffer 里面还有多少数据没有解码
参数	pDecoder: 通过 CreateAudioDecodeLib 函数创建的音频解码库指针 。
返回值	没有解码数据长度；
调用说明	无

1.1.8. BitstreamQueryQuality

函数原型	Void BitstreamQueryQuality(AudioDecoderLib* pDecoder, int* pValidPercent,int* vbv)
功能	获取没有解码数据的数据信息。
参数	pDecoder: 通过 CreateAudioDecoder 函数创建的音频解码器指针；

	pValidPercent：有效数据占用的%比，此值已经乘以 100 vbv：有效的帧数
返回值	无返回值
调用说明	无

1.1.9. ParseBitstreamSeekSync

函数原型	Void ParseBitstreamSeekSync(AudioDecoderLib* pDecoder, int64_t nSeekTime, int nGetAudioInfoFlag)
功能	为上层应用跳播用。
参数	pDecoder：通过 CreateAudioDecoder 函数创建的音频解码器指针； nSeekTime：当前播放要播放的时间（微秒） nGetAudioInfoFlag：是否获得解码完成标示符。1：还没有解码，0：已经完成解码并获得解码信息。
返回值	无返回值
调用说明	无

1.2. 音频解码库外部 API（供调用者参考）

音频解码库外部 API 接口如下表所示。

开始解码前，应用程序首先调用 CreateAudioDecoder 函数创建一个解码器，然后调用 InitializeAudioDecoder 函数，将音频基本信息作为参数，初始化解码器。

音频解码库 APIs		
1	CreateAudioDecoder	创建一个音频解码库实作句柄
2	DestroyAudioDecoder	销毁一个音频解码库实作句柄，释放资源
3	InitializeAudioDecoder	根据音频码流信息（编码格式等）初始化音频解码器
4	ResetAudioDecoder	重置 bitStreamManager 和 frameBufferManger 和解码器，刷掉当前 bitStreamManager 和 frameBufferManger 的缓冲
5	DecodeAudioStream	解码一笔音频码流
6	ParserRequestBsBuffer	从 bitStreamManager 获取码流 Buffer，用于填充码流数据
7	ParserUpdateBsBuffer	填充完码流数据后，将数据提交给 bitStreamManager
8	BsQueryQuality	从 bitStreamManager 获取当前 bitstream buffer 数据信息:有效数据百分比，有效数据帧数
9	AudioStreamDataSize	获取当前 bitStreamManager 的未消费 bitstream 长度

10	AudioPCMDataSize	从 frameBufferManger 获取解码出来的 PCM 数据库存总长度，供 render 参考
11	DecRequestPcmBuffer	向 frameBufferManger 发送解码 pcm 数据请求，获取 pcm buffer
12	DecUpdatePcmBuffer	向 frameBufferManger 上传解码出来的 pcm 数据
13	PlybkRequestPcmBuffer	向 frameBufferManger 发送 render 请求，获取 pcm buffer
14	PlybkUpdatePcmBuffer	从 frameBufferManger 拿走数据，并释放占用的 pcm buffer，供后续解码用
15	PlybkRequestPcmPts	获取当前播放时间
16	PcmQueryQuality	获取当前解码后 pcm buffer 信息：有效数据百分比，有效数据长度
17	AudioDecoderSeek	跳播，Cedar1.0 遗留功能，seek to specified point，abandon
18	AudioStreamBufferSize	获取 bitStreamManager buffer 总长（总帧数*帧长）
19	AudioStreamBufferMaxFrameNum	获取 bitStreamManager 总帧数
20	SetRawPlayParam	设置透传模式

初始化后，解码器可以开始解码音频流。

应用程序通过 ParserRequestBsBuffer 函数从解码器获取码流 Buffer，将数据填入后，通过 ParserUpdateBsBuffer 函数将码流提交给解码器。

应用程序通过调用 DecodeAudioStream 函数解码音频码流。

应用程序通过调用 DecRequestPcmBuffer 函数获取 pcm 空 buffer,为解码库填充 pcm 数据用，音频解码完毕后，应用程序调用 DecUpdatePcmBuffer 将 pcm Buffer 送到 plybk 模块。

音频解码库支持多线程操作，码流的传送、解码和音频的输出工作可以在不同的线程中进行。一般来说，播放器会有 Demux 线程、音频解码线程和音频渲染（Render）等三个线程处理音频相关的工作。Demux 线程不断调用 ParserRequestBsBuffer 函数和 ParserUpdateBsBuffer 函数传送数据；音频解码线程通过调用 DecodeAudioStream 函数解码音频流；音频渲染线程不断调用 PlybkRequestPcmBuffer 函数获取声音数据用于播放，调用 PlybkUpdatePcmBuffer 归还已经播放的空间。

1.2.1. CreateAudioDecoder

函数原型	AudioDecoder* CreateAudioDecoder(void)
功能	创建一个音频解码库实作句柄

参数	无
返回值	成功：音频解码器指针； 失败：返回 NULL；
调用说明	初始化函数，为整个 audio 接口的总指针开空间。

1.2.2. DestroyAudioDecoder

函数原型	int DestroyAudioDecoder(AudioDecoder* pDecoder)
功能	销毁一个音频解码库实作句柄，释放资源
参数	pDecoder 通过 CreateAudioDecoder 函数创建的音频解码器指针
返回值	成功：0； 失败：<0
调用说明	无

1.2.3. ResetAudioDecoder

函数原型	int ResetAudioDecoder(AudioDecoder* pDecoder int64_t nSeekTime)
功能	重置 bitStreamManager 和 frameBufferManger 和解码器，刷掉当前 bitStreamManager 和 frameBufferManger 的缓冲，目前跳播用到。
参数	pDecoder: 通过 CreateAudioDecoder 函数创建的音频解码器指针 nSeekTime: 要跳到的时间（绝对时间，单位，微秒）
返回值	0
调用说明	同 AudioDecoderSeek 功能一样

1.2.4. InitializeAudioDecoder

函数原型	int InitializeAudioDecoder(AudioDecoder* pDecoder AudioStreamInfo* pAudioStreamInfo, BsInFor_ *pBsInFor)
功能	根据音频码流信息（编码格式等）初始化音频解码器
参数	pDecoder: 通过 CreateAudioDecoder 函数创建的音频解码器指针 pAudioStreamInfo: 音频码流的基本信息，如编码格式、采样率等 pBsInFor：解码过程中交互信息
返回值	0: 表示成功； -1: 失败,不支持的编码格式或内存资源不足或者 dlopen fail,详看 dlerror
调用说明	AudioStreamInfo 中，不是所有信息都是必须的。对于 pcm,adpcm 需要的信息比较多，用于打头，对于其它格式，需要的信息相对较少。

1.2.5. DecodeAudioStream

函数原型	int DecodeVideostream(AudioDecoder* pDecoder, AudioStreamInfo*pAudioStreamInfo,

	char* ppBuf, int* pBufSize)
功能	解码音频，解码库会对码流 Buffer 中的码流进行解码
参数	pDecoder：通过 CreateAudioDecoder 函数创建的音频解码器指针； pAudioStreamInfo:音频初始化信息； ppBuf： 返回用，解码后 pcm 数据首地址； pBufSize：返回用，解码后 pcm 数据长度；
返回值	返回类型： AUDIO_DEC_RESULT
调用说明	无

1.2.6. ParserRequestBsBuffer

函数原型	int ParserRequestBsBuffer(AudioDecoder* pDecoder, int nRequireSize, unsigned char** ppBuf, int* pBufSize, unsigned char** ppRingBuf, int* pRingBufSize, int* nOffset)
功能	从 bitStreamManager 获取码流 Buffer，用于填充码流数据
参数	pDecoder：通过 CreateAudioDecoder 函数创建的音频解码器指针 nRequireSize：请求 Buffer 的大小，以字节为单位； ppBuf：输出参数，码流 Buffer 起始地址，等于 NULL 表示失败； pBufSize：输出参数，码流 Buffer ppBuf 的大小； ppRingBuf：输出参数，第二块 Buffer 的起始地址，等于 NULL 表示没有； pRingBufSize：第二块 Buffer ppRingBuf 的大小； nOffset:当前文件读的位置（有多少数据已经从文件中读出）
返回值	0： 表示成功；-1：失败；没有得到 buffer
调用说明	码流 Buffer 是一块循环 Buffer，当 Buffer 回头时，外部请求的 Buffer 被分成两段，ppBuf 和 pBufSize 返回第一段 Buffer 的地址和大小，ppRingBuf 和 pRingBufSize 返回第二段 Buffer 的地址和大小。 Buffer 没有回头时，ppRingBuf 和 pRingBufSize 返回 NULL。

1.2.7. ParserUpdateBsBuffer

函数原型	int ParserUpdateBsBuffer(AudioDecoder* pDecoder, int nFilledLen,INT64 nTimeStamp,int nOffset)
功能	填充完码流数据后，将数据提交给 bitStreamManager
参数	pDecoder：通过 CreateAudioDecoder 函数创建的音频解码器指针； nFilledLen： parser 释放填充的原始数据的长度，解码库可以用这段数据解码了。 nTimeStamp：当前帧时间戳 nOffset：当前读位置，为是否跳播用。
返回值	0： 表示成功；-1：失败；

调用说明	无

1.2.8. BsQueryQuality

函数原型	void BsQueryQuality(AudioDecoder* pDecoder, int* pValidPercent, int* vbv)
功能	从 bitStreamManager 获取当前 bitstream buffer 数据信息:有效数据百分比，有效数据帧数
参数	pDecoder: 通过 CreateAudioDecoder 函数创建的音频解码器指针; pValidPercent：有效数据占用的%比，此值已经乘以 100 vbv：有效的帧数
返回值	无
调用说明	无

1.2.9. AudioStreamDataSize

函数原型	int AudioStreamDataSize(AudioDecoder* pDecoder)
功能	获取当前 bitStreamManager 的未消费 bitstream 长度
参数	pDecoder: 通过 CreateAudioDecoder 函数创建的音频解码器指针; 。
返回值	获取 parser buffer 里面还有多少数据没有解码;
调用说明	无

1.2.10. AudioPCMDataSize

函数原型	int AudioPCMDataSize(AudioDecoder* pDecoder)
功能	从 frameBufferManger 获取解码出来的 PCM 数据库存总长度,供 render 参考
参数	pDecoder: 通过 CreateAudioDecoder 函数创建的音频解码器指针; 。
返回值	获取到 buffer 中含有的 pcm 数;
调用说明	无

1.2.11. DecRequestPcmBuffer

函数原型	int DecRequestPcmBuffer(AudioDecoder* pDecoder, char** pOutWritePtr)
功能	向 frameBufferManger 发送解码 pcm 数据请求，获取 pcm buffer 以存放 PCM 数据
参数	pDecoder: 通过 CreateAudioDecoder 函数创建的音频解码器指针; pOutWritePtr: 如果成功 buffer 首地址;
返回值	0: 表示成功; -1: 失败;
调用说明	Audio 解码每帧前申请 pcm 数据空间，空间大小有宏 MAX_AUDIO_FRAME_SIZE

	确定,，如果申请成功可以解码，如果不成功需要 sleep 后再次申请。 为 dec 线程 components 所调用

1.2.12. DecUpdatePcmBuffer

函数原型	int DecUpdatePcmBuffer(AudioDecoder* pDecoder,int nPcmOutSize)
功能	向 frameBufferManger 上传解码出来的 pcm 数据
参数	pDecoder：通过 CreateAudioDecoder 函数创建的音频解码器指针； nPcmOutSize：　需要更新的数据长度；
返回值	0：　表示成功； -1：失败；
调用说明	Audio 解码每帧后更新 pcm buffer 空间，使的 playback 可以播放 pcm，如果是硬件需要刷新一下 cache。 为 dec 线程 component 所调用

1.2.13. PlybkRequestPcmBuffer

函数原型	int PlybkRequestPcmBuffer(AudioDecoder* pDecoder, unsigned char **pOutReadPtr, int *psize)
功能	向 frameBufferManger 发送 render 请求，获取 pcm buffer 以播放
参数	pDecoder　：　通过 CreateAudioDecoder 函数创建的音频解码器指针； pOutReadPtr　：数据首地址 psize ：**调用时需要先赋值，需要的的数据长度，返回时函数内部会修改其值，为实际返回数据长度**
返回值	0：　表示成功； -1：失败；需要重新调用
调用说明	Playback 要 Audio 解码的 pcm 数据，如果有 pcm 数据，返回 pcm 首地址和长度，如果没有返回失败，为 Render 线程 component 所调用

1.2.14. PlybkUpdatePcmBuffer

函数原型	int PlybkUpdatePcmBuffer(AudioDecoder* pDecoder,int nPcmOutSize)
功能	从 frameBufferManger 拿走数据，并释放占用的 pcm buffer，供后续解码用
参数	pDecoder：通过 CreateAudioDecoder 函数创建的音频解码器指针； nPcmOutSize：　需要更新的数据长度；
返回值	0：　表示成功； -1：失败；
调用说明	Playback 释放 pcm buffer，playback 已经取走本段 pcm，解码库可以再次用本段空间；为 Render 线程 component 所调用

1.2.15. PlybkRequestPcmPts

函数原型	int PlybkRequestPcmPts(AudioDecoder* pDecoder)
功能	获取当前时间戳。
参数	pDecoder: 通过 CreateAudioDecoder 函数创建的音频解码器指针;
返回值	返回当前数据播放的起始时间
调用说明	获取 pcm 数据中要播放的起始时间，为视频同步用; 为 AudioDecRequstBuffer 所调用

1.2.16. PcmQueryQuality

函数原型	Void PcmQueryQuality(AudioDecoder* pDecoder, int* pValidPercent,int* vbv)
功能	获取没有解码后 pcm 数据的数据信息。
参数	pDecoder: 通过 CreateAudioDecoder 函数创建的音频解码器指针; pValidPercent：有效数据占用的%比，此值已经乘以 100 vbv：有效数据长度
返回值	无返回值
调用说明	获取 pcm 数据长度和 pcm buffer 占总空间的百分比; 为 AudioDecQueryBufferState 所调用

1.2.17. AudioDecoderSeek

函数原型	Void AudioDecoderSeek(AudioDecoder* pDecoder, int64_t nSeekTime)
功能	为上层应用跳播用。
参数	pDecoder: 通过 CreateAudioDecoder 函数创建的音频解码器指针; nSeekTime：当前播放跳播到的时间（微秒）
返回值	无返回值
调用说明	无

1.2.18. AudioStreamBufferSize

函数原型	int AudioStreamBufferSize(void)
功能	获取 bitStreamManager buffer 总长（总帧数*帧长)。
参数	无
返回值	bitStreamManager buffer 总长（总帧数*帧长）
调用说明	无

1.2.19. AudioStreamBufferMaxFrameNum

函数原型	int AudioStreamBufferMaxFrameNum(void)
功能	获取 bitStreamManager 总帧数。
参数	无
返回值	bitStreamManager 总帧数
调用说明	无

1.2.20. SetRawPlayParam

函数原型	#ifdef OS_LINUX
	void SetRawPlayParam(AudioDecoder* pDecoder,void *self,int flag);
	#else
	void SetRawPlayParam(AudioDecoder* pDecoder,void *self);
	#endif

功能	设置透传模式。
参数	pDecoder: 通过 CreateAudioDecoder 函数创建的音频解码器指针; self: 调用者自身指针，目前暂无用处，留作未来回调用 #ifdef OS_LINUX flag: linux 平台下，没有 property 这种全局外部信息交互手段来获取透传模式，只能通过外部传递 flag 来完成。 #else android 平台有通过 property 完成模式传递，故无需 flag #endif
返回值	无返回值
调用说明	#ifdef OS_LINUX Flag 0: 不透传 1: HDMI 透传 2: SPDIF 透传 #else 无 #endif

2. 数据结构设计

本章节结构体不允许轻易改动，如果非要改动，请在变量最后增加变量

2.1. AudioDecoderLib

名称	AudioDecoderLib		
功能描述	表示一个音频解码器，包含音频解码器的所有信息。参见结构体 AudioDecoderContextLib，对外声明为 void		
属性	类型	初始值	描述
	void	*	

2.2. AudioDecoder

名称	AudioDecoder		
功能描述	表示一个音频解码器，包含音频解码器的所有信息。参见结构体 AudioDecoderContext，对外声明为 AudioDecoder		
属性	类型	初始值	描述
nPrivFlag	int	0	数据流状态，0：play，1：reset，2：file end
nGetAudioInfoFlag	int	1	初始化标志，1：没有初始化成功，0：初始化成功
nAudioChannel	int	0	输出那个通路参见 CEDARX_AUDIO_CHANNEL_TYPE
volumegain	int	0	音量控制
mute	int	0	是否静音：0 不静音正常输出，1：静音左声道，2：静音右声道，3：两个声道全部静音。
nEnableResample	int	0	是否允许重采样标志，0：不重采样，1：可以重采样。
RawPlayFlag	int	0	暂时无用，设计之初用来传透传模式。考虑取消

2.3. AudioDecoderContext

名称	AudioDecoderContext		
功能描述	表示一个音频解码器，包含音频解码器的所有信息。对内结构体。		
属性	类型	初始值	描述
common	AudioDecoder	0	音频解码器对外信息
pBsInFor	BsInFor*	0	音频交互信息结构体
libHandle	Void*	0	打开.so 库句柄
pAudioDecodeLib	AudioDecoderLib*	0	解码库指针
mutex_audiodec_thread	pthread_mutex_t	0	互斥锁，为数据安全访问用
ADCedarCtxA	OutputPcmBuf	0	Pcm 数据 buffer 控制结构体
ADCedarCtx	ACedarContext	0	为处理解码的 p c m数据用
CdxPlaybkCfg	raw_data	0	播放参数 configure 结构体，详看超链接。用于和 render 模块交互普通播放或者透传信息
Void*	hself	0	调用者自身句柄，用来回调。透传初始版本遗留，暂无用
streamsize	(*func)	0	获取 parser buffer 里面还有多少数据没有解码
ParseRequestAudioBitstreamBuffer	(*func)	0	向解码器请求存放码流 Buffer，用于存放数据
ParseUpdateAudioBitstreamData	(*func)	0	向解码器提交码流数据
BitstreamQueryQuality	(*func)	0	获取没有解码数据的数据信息。
ParseSeekSync	(*func)	0	为上层应用跳播用。
InitializeAudioDecodeLib	(*func)	0	初始化音频解码库
DecodeAudioFrame	(*func)	0	解码音频，解码库会对码流 Buffer 中的码流进行解码
DestroyAudioDecodeLib	(*func)	0	销毁一个音频解码库
CreateAudioDecodeLib	(*func)	0	创建一个音频解码库

2.4.OutputPcmBuf

名称	OutputPcmBuf		
功能描述	pcm buffer 控制		
属性	类型	初始值	描述
pAOutBufferStart	Char*	0	Pcm buffer 首地址，空间大小（AOUT_BUFFER_SIZE 1024*192）
pAOutBufferEnd	Char*	0	Pcm buffer 最后一个字节的地址
nAOutBufferTotalSize	int	0	Pcm buffer 大小
pAOutBufferWritePtr	Char*	0	往 pcm buffer 写数据指针
pAOutBufferReadPtr	Char*	0	从 pcm buffer 读数据指针
pAOutPcmEndPtr	Char*	0	最终有效数据指针，为有效数据+1 指针。
nAOutBufferEmptySize	int	0	Pcm buffer 空闲空间长度
nAOutBufferValidSize	int	0	Pcm buffer 有效数据长度。
frmFifo	astream_fifo_t	0	外挂控制 pcm 码流结构体，长度，时间戳等，保证每次送出的时间戳准确（为视频用），同 parse 的时间戳管理机制匹配。

2.5.ACedarContext

名称	ACedarContext		
功能描述	处理 p c m数据结构体		
属性	类型	初始值	描述
nOriChannels	int	0	解码库解码原始通道数
nOriSampleRate	int	0	解码库解码原始采样率
pTempResampleBuffer	Char*	0	重采样临时 buffer，为 linux 用

2.6. BsInFor

黄色为外部只读，解码器写信息
内部使用：为解码器内部用，外部不用
蓝色字体为外部设置，内部使用

名称	BsInFor		
功能描述	解码器与外部通信接口		
属性	类型	初始值	描述
TotalplayTime	int	0	外部只读，总时间（毫秒）
NowPlayTime	int	0	外部只读，当前播放时间（毫秒）纯音频时间

Samplerate	int	0	外部只读，输出采样率
bitrate	int	0	外部只读，比特率
chan	int	0	外部只读，通道数
CBRflag	int	0	内部使用，Ac3,dts lib 库专用，1:cbr,0:vbr
framesize	int	0	内部使用，一帧数据有多少个 bytes（未解码数据），为计算 b i t r a t e 用
framecount	int	0	内部使用，帧数
oldfs	int	0	内部使用，Ac3,dts 解码库私用变量，初始采样率
oldbs	int	0	内部使用，Ac3,dts 解码库私用变量，初始比特率
firstflag	int	0	内部使用，初始化完成标志，1:完成初始化，0:没有初始化
framepcms	int	0	内部使用，解码出一帧含有 p c m的大小，b y t e s
modeflag	int	0	外部只读，输出数据格式，1：输出的为 r a w d a t a 数据，0：输出为 p c m数据
ulMode	int	0	外部设置标志，是否 raw data 输出及设备，内部只读，解码模式，0：pcm 输出，1:hdmi raw data out,2:spdif raw data out
bitpersample	int	0	采样数据精度
nBitStreamUnderFlow	int	0	外部只读，解码库结束标志，也为 playbk 退出标志，解码库数据读完，解码库接收到数据结束标志后置位，此标志告诉外部可以退出。
nShowBitsReturn	int	0	外部设置标志，parse 数据结束标志，1：没有数据，切换音轨和数据结束需要设置此值。强制退出
out_channels	int	0	外部设置，为需要改变通道数用，主要为解码库计算 buffer 时间用，变换通道需外部改变，参见 CEDARX_AUDIO_CHANNEL_TYPE
out_samplerate	int	0	外部设置，为需要改变采样率用，主要为解码库计算 buffer 时间用，变换采样率需要外部改变
nDecodeMode	int	0	外部设置标志，是否纯音频标志，1：纯音频封装格式文件，0：其它格式封装文件，例如视频，网络封装等。参见 CDX_DECODE_MODE
nDemuxType	int	0	外部设置标志，解码文件类型标志，为打头用，参见 CDX_MEDIA_FILE_FORMAT
nIsHwCodec	int	0	是否硬件解码标示，0:软件解码，1：硬件解码，为用什么方式分配空间用
NowPTSTime	INT64	0	视频解码时间戳，为外挂 pts 计算视频用,此时间戳单位为微秒

2.7. AudioStreamInfo

名称	AudioStreamInfo		
功能描述	音频解码库初始化信息。		
属性	类型	初始值	描述
eCodecFormat	EAUDIOCODECFORMAT	0	数据类型 EAUDIOCODECFORMAT，例如：MP3 等
eSubCodecFormat	int	0	低 16 位，数据子类型，为音频数据的具体类型，高 16 位：大小端标志，0：小端模式，1：大端模式参见 __amr_format_sub_type_t__ 和 WAVE_FORMAT
nChannelNum	int	0	通道数
nBitsPerSample	int	0	比特数
nSampleRate	int	0	采样数
nAvgBitrate	int	0	平均采样率
nMaxBitRate	int	0	最大采样率
nFileSize	int	0	文件总长度
eAudioBitstreamSource	int	0	文件类型，此变量以后将删除不用。文件类型：AVI,TS,等参见 CDX_MEDIA_FILE_FORMAT
eDataEncodeType	int	0	音轨字体编码类型 CEDARX_SUBTITLE_ENCODE_TYPE
strLang[ADECODER_MAX_LANG_CHAR_SIZE]	Unsigned char	0	音轨字幕，用于切换音轨时显示的音轨文字
nCodecSpecificDataLen	int	0	extra_data 长度，为传送除了本结构明确规定的变量外的其它数据，用于解码器需要
pCodecSpecificData	Char*	0	extra_data 首地址
nFlags	int	0	数据打头标志，0：原始码流中每帧数据没有帧头，需要解码库打头。1：原始码流中每帧数据已经包含了帧头，不需要解码库打头。
nBlockAlign	int	0	数据块大小，新增加类型

2.8. CdxPlaybkCfg

名称	CdxPlaybkCfg		
功能描述	播放参数 configure 信息。		
属性	类型	初始值	描述
nRoutine	Int	0	对应播放声卡。0：audiocodec 1:hdmi

			2:spdif　　以后可扩展，暂时用不上。
nNeedDirect	int	0	标志是否需要采用直通模式输出。 0： 否 1：是
nChannels	int	0	通道数
nSamplerate	int	0	采样数
nBitpersample	int	0	数据位宽
nDataType	enum AUIDO_RAW_DATA_TYPE	0	数据类型，0 or 1:linear pcm >1 ： non linear pcm（IEC 协议透传数据或者其他诸如 1 bit audio 数据流）

2.9. 内部数据枚举类型

枚举类型按照其性质没有赋值的按照枚举规则自动赋值

2.9.1. __AUDIO_DEC_RESULT

名称	__AUDIO_DEC_RESULT	
功能描述	解码库返回值类型	
属性	值	描述
ERR_AUDIO_DEC_EXIT	-4	exit
ERR_AUDIO_DEC_ENDINGCHKFAIL	-3	big ending or little ending check failed
ERR_AUDIO_DEC_ABSEND	-2	audio bitstream decode end
ERR_AUDIO_DEC_ONEFRMFAIL	-1	decode one frame failed, can try again
ERR_AUDIO_DEC_NONE	0	decode successed, no error
ERR_AUDIO_DEC_FFREVRETURN	1	eturn from fast-forward or fast-reverse
ERR_AUDIO_DEC_RETAPOINT	2	return from A point under A/B play
ERR_AUDIO_DEC_RETTAG	3	return from the first frame of tag play
ERR_AUDIO_DEC_VIDEOJUMP	4	0X88(video jump) maybe return 4 or 0
ERR_AUDIO_DEC_NO_BITSTREAM	5	No enough bitstream ,try again
ERR_AUDIO_DEC_		

2.9.2. CEDARX_AUDIO_CHANNEL_TYPE

名称	CEDARX_AUDIO_CHANNEL_TYPE	
功能描述	输出通道数控制	
属性	值	描述
CEDARX_AUDIO_CHANNEL_STEREO	0	正常模式，通道数不变化，正常输出

CEDARX_AUDIO_CHANNEL_LEFT		在双通路输出的情况下只输出左通路(左右通路均是左通路数据)
CEDARX_AUDIO_CHANNEL_RIGHT		在双通路输出的情况下只输出右通路(左右通路均是右通路数据)

2.9.3. CDX_DECODE_MODE

名称	CDX_DECODE_MODE	
功能描述	文件类型标志	
属性	值	描述
CDX_DECODER_MODE_NORMAL	0	音视频文件
CDX_DECODER_MODE_RAWMUSIC		纯音频文件

2.9.4. CDX_MEDIA_FILE_FORMAT

名称	CDX_MEDIA_FILE_FORMAT	
功能描述	文件格式	
属性	值	描述
CDX_MEDIA_FILE_FMT_UNSUPPORT = -1,	-1	
CDX_MEDIA_FILE_FMT_UNKOWN = 0,	0	
CDX_MEDIA_FILE_FMT_AVI,		
CDX_MEDIA_FILE_FMT_RM,		
CDX_MEDIA_FILE_FMT_RMVB,		
CDX_MEDIA_FILE_FMT_MP4,		
CDX_MEDIA_FILE_FMT_M4V,		
CDX_MEDIA_FILE_FMT_3GP,		
CDX_MEDIA_FILE_FMT_MOV,		
CDX_MEDIA_FILE_FMT_FLV,		
CDX_MEDIA_FILE_FMT_MPG,		
CDX_MEDIA_FILE_FMT_VOB,		
CDX_MEDIA_FILE_FMT_MOD,		
CDX_MEDIA_FILE_FMT_PMP,		
CDX_MEDIA_FILE_FMT_WMV,		
CDX_MEDIA_FILE_FMT_ASF,		
CDX_MEDIA_FILE_FMT_MKV,		
CDX_MEDIA_FILE_FMT_PSR,		
CDX_MEDIA_FILE_FMT_RAM,		
CDX_MEDIA_FILE_FMT_SCM,		
CDX_MEDIA_FILE_FMT_OGM,		
CDX_MEDIA_FILE_FMT_M4P,		
CDX_MEDIA_FILE_FMT_M4B,		
CDX_MEDIA_FILE_FMT_TP,		

CDX_MEDIA_FILE_FMT_TPR,		
CDX_MEDIA_FILE_FMT_TS,		
CDX_MEDIA_FILE_FMT_PVA,		
CDX_MEDIA_FILE_FMT_PSS,		
CDX_MEDIA_FILE_FMT_MPE,		
CDX_MEDIA_FILE_FMT_WV,		
CDX_MEDIA_FILE_FMT_M2TS,		
CDX_MEDIA_FILE_FMT_EVO,		
CDX_MEDIA_FILE_FMT_RPM,		
CDX_MEDIA_FILE_FMT_3GPP,		
CDX_MEDIA_FILE_FMT_3G2,		
CDX_MEDIA_FILE_FMT_3GP2,		
CDX_MEDIA_FILE_FMT_QT,		
CDX_MEDIA_FILE_FMT_WMP,		
CDX_MEDIA_FILE_FMT_WM,		
CDX_MEDIA_FILE_FMT_AMV,		
CDX_MEDIA_FILE_FMT_DSM,		
CDX_MEDIA_FILE_FMT_M1V,		
CDX_MEDIA_FILE_FMT_M2V,		
CDX_MEDIA_FILE_FMT_SMK,		
CDX_MEDIA_FILE_FMT_BIK,		
CDX_MEDIA_FILE_FMT_RAT,		
CDX_MEDIA_FILE_FMT_VG2,		
CDX_MEDIA_FILE_FMT_IVF,		
CDX_MEDIA_FILE_FMT_VP6,		
CDX_MEDIA_FILE_FMT_VP7,		
CDX_MEDIA_FILE_FMT_D2V,		
CDX_MEDIA_FILE_FMT_M2P,		
CDX_MEDIA_FILE_FMT_VID,		
CDX_MEDIA_FILE_FMT_PMP2,		
CDX_MEDIA_FILE_FMT_MTS,		
CDX_MEDIA_FILE_FMT_MP3,		
CDX_MEDIA_FILE_FMT_WAV,		
CDX_MEDIA_FILE_FMT_WMA,		
CDX_MEDIA_FILE_FMT_APE,		
CDX_MEDIA_FILE_FMT_FLAC,		
CDX_MEDIA_FILE_FMT_OGG,		
CDX_MEDIA_FILE_FMT_RA,		
CDX_MEDIA_FILE_FMT_MP1,		
CDX_MEDIA_FILE_FMT_MP2,		
CDX_MEDIA_FILE_FMT_AAC,		
CDX_MEDIA_FILE_FMT_AC3,		
CDX_MEDIA_FILE_FMT_DTS,		

CDX_MEDIA_FILE_FMT_AIF,		
CDX_MEDIA_FILE_FMT_AIFF,		
CDX_MEDIA_FILE_FMT_AIFC,		
CDX_MEDIA_FILE_FMT_AMR,		
CDX_MEDIA_FILE_FMT_MAC,		
CDX_MEDIA_FILE_FMT_TTA,		
CDX_MEDIA_FILE_FMT_M4A,		
CDX_MEDIA_FILE_FMT_CDA,		
CDX_MEDIA_FILE_FMT_AU,		
CDX_MEDIA_FILE_FMT_ACC,		
CDX_MEDIA_FILE_FMT_MIDI,		
CDX_MEDIA_FILE_FMT_MID,		
CDX_MEDIA_FILE_FMT_RMI,		
CDX_MEDIA_FILE_FMT_MP5,		
CDX_MEDIA_FILE_FMT_MPA,		
CDX_MEDIA_FILE_FMT_MPGA,		
CDX_MEDIA_FILE_FMT_ACT,		
CDX_MEDIA_FILE_FMT_ATRC,		
CDX_MEDIA_FILE_FMT_WEBM,		
CDX_MEDIA_FILE_FMT_M3U,		
CDX_MEDIA_FILE_FMT_AWTS,		
CDX_MEDIA_FILE_FMT_WVM,//87		
/*for test raw data */		
CDX_MEDIA_FILE_FMT_RAW,		
CDX_MEDIA_FILE_FMT_AUDIO = (1<<16),	(1<<16)	
CDX_MEDIA_FILE_FMT_NETWORK = (2<<16),	(2<<16)	
CDX_MEDIA_FILE_FMT_NETWORK_RTSP = (4<<16),	(4<<16)	
CDX_MEDIA_FILE_FMT_IDXSUB = (8<<16),	(8<<16)	
CDX_MEDIA_FILE_FMT_NETWORK_SFT = (16<<16),	(16<<16)	
CDX_MEDIA_FILE_FMT_WIFI_DISPLAY = (32<<16),	(32<<16)	
CDX_MEDIA_FILE_FMT_STREAMINGSRC = (64<<16),	(64<<16)	
CDX_MEDIA_FILE_FMT_NETWORK_OTHERS = (128<<16),	(128<<16)	

2.9.5. CDX_COMP_PRIV_FLAGS

名称	CDX_COMP_PRIV_FLAGS	
功能描述	解码状态	
属性	值	描述
CDX_COMP_PRIV_FLAGS_REINIT	1	重新初始化，例如音轨切换
CDX_COMP_PRIV_FLAGS_STREAMEOF	2	文件结束

2.9.6. CEDARXAUDIOFLAGSENUM

名称	CEDARXAUDIOFLAGSENUM	
功能描述	Aac 是否需要打头	
属性	值	描述
ADEC_DISABLE_AAC_PACKING	1	不需要打头。

2.9.7. CEDARX_SUBTITLE_ENCODE_TYPE

名称	CEDARX_SUBTITLE_ENCODE_TYPE	
功能描述	字幕编码格式	
属性	值	描述
CDX_SUBTITLE_ENCODE_UNKNOWN		SUB_CHARSET_UNKNOWN
CDX_SUBTITLE_ENCODE_NONE		SUB_CHARSET_UNKNOWN
CDX_SUBTITLE_ENCODE_BITMAP		SUB_CHARSET_BITMAP
CDX_SUBTITLE_ENCODE_UTF8		SUB_CHARSET_UTF_8
CDX_SUBTITLE_ENCODE_GB2312		SUB_CHARSET_HZ_GB_2312
CDX_SUBTITLE_ENCODE_UTF16LE		SUB_CHARSET_UTF_16LE
CDX_SUBTITLE_ENCODE_UTF16BE		SUB_CHARSET_UTF_16BE
CDX_SUBTITLE_ENCODE_UTF32LE		SUB_CHARSET_UTF_32LE
CDX_SUBTITLE_ENCODE_UTF32BE		SUB_CHARSET_UTF_32BE
CDX_SUBTITLE_ENCODE_BIG5		SUB_CHARSET_BIG5
CDX_SUBTITLE_ENCODE_GBK		SUB_CHARSET_GBK
CDX_SUBTITLE_ENCODE_ANSI		SUB_CHARSET_UTF_8
		具体见 enum SUB_CHARSET
CDX_SUBTITLE_ENCODE_		

2.9.8. SUB_CHARSET

名称	SUB_CHARSET	
功能描述	文字编码格式	
属性	值	描述
SUB_CHARSET_BITMAP = -2,		
SUB_CHARSET_UNKNOWN		
SUB_CHARSET_BIG5 = 0,		
SUB_CHARSET_BIG5_HKSCS = 1,		
SUB_CHARSET_BOCU_1 = 2,		

SUB_CHARSET_CESU_8	= 3,		
SUB_CHARSET_CP864	= 4,		
SUB_CHARSET_EUC_JP	= 5,		
SUB_CHARSET_EUC_KR	= 6,		
SUB_CHARSET_GB18030	= 7,		
SUB_CHARSET_GBK	= 8,		
SUB_CHARSET_HZ_GB_2312	= 9,		
SUB_CHARSET_ISO_2022_CN	= 10,		
SUB_CHARSET_ISO_2022_CN_EXT	= 11,		
SUB_CHARSET_ISO_2022_JP	= 12,		
SUB_CHARSET_ISO_2022_KR	= 13,		
SUB_CHARSET_ISO_8859_1	= 14,		
SUB_CHARSET_ISO_8859_10	= 15,		
SUB_CHARSET_ISO_8859_13	= 16,		
SUB_CHARSET_ISO_8859_14	= 17,		
SUB_CHARSET_ISO_8859_15	= 18,		
SUB_CHARSET_ISO_8859_16	= 19,		
SUB_CHARSET_ISO_8859_2	= 20,		
SUB_CHARSET_ISO_8859_3	= 21,		
SUB_CHARSET_ISO_8859_4	= 22,		
SUB_CHARSET_ISO_8859_5	= 23,		
SUB_CHARSET_ISO_8859_6	= 24,		
SUB_CHARSET_ISO_8859_7	= 25,		
SUB_CHARSET_ISO_8859_8	= 26,		
SUB_CHARSET_ISO_8859_9	= 27,		
SUB_CHARSET_KOI8_R	= 28,		
SUB_CHARSET_KOI8_U	= 29,		
SUB_CHARSET_MACINTOSH	= 30,		
SUB_CHARSET_SCSU	= 31,		
SUB_CHARSET_SHIFT_JIS	= 32,		
SUB_CHARSET_TIS_620	= 33,		
SUB_CHARSET_US_ASCII	= 34,		
SUB_CHARSET_UTF_16	= 35,		
SUB_CHARSET_UTF_16BE	= 36,		
SUB_CHARSET_UTF_16LE	= 37,		
SUB_CHARSET_UTF_32	= 38,		
SUB_CHARSET_UTF_32BE	= 39,		
SUB_CHARSET_UTF_32LE	= 40,		
SUB_CHARSET_UTF_7	= 41,		
SUB_CHARSET_UTF_8	= 42,		
SUB_CHARSET_WINDOWS_1250	= 43,		
SUB_CHARSET_WINDOWS_1251	= 44,		
SUB_CHARSET_WINDOWS_1252	= 45,		

SUB_CHARSET_WINDOWS_1253 = 46,		
SUB_CHARSET_WINDOWS_1254 = 47,		
SUB_CHARSET_WINDOWS_1255 = 48,		
SUB_CHARSET_WINDOWS_1256 = 49,		
SUB_CHARSET_WINDOWS_1257 = 50,		
SUB_CHARSET_WINDOWS_1258 = 51,		
SUB_CHARSET_X_DOCOMO_SHIFT_JIS_2007 = 52,		
SUB_CHARSET_X_GSM_03_38_2000 = 53,		
SUB_CHARSET_X_IBM_1383_P110_1999 = 54,		
SUB_CHARSET_X_IMAP_MAILBOX_NAME = 55,		
SUB_CHARSET_X_ISCII_BE = 56,		
SUB_CHARSET_X_ISCII_DE = 57,		
SUB_CHARSET_X_ISCII_GU = 58,		
SUB_CHARSET_X_ISCII_KA = 59,		
SUB_CHARSET_X_ISCII_MA = 60,		
SUB_CHARSET_X_ISCII_OR = 61,		
SUB_CHARSET_X_ISCII_PA = 62,		
SUB_CHARSET_X_ISCII_TA = 63,		
SUB_CHARSET_X_ISCII_TE = 64,		
SUB_CHARSET_X_ISO_8859_11_2001 = 65,		
SUB_CHARSET_X_JAVAUNICODE = 66,		
SUB_CHARSET_X_KDDI_SHIFT_JIS_2007 = 67,		
SUB_CHARSET_X_MAC_CYRILLIC = 68,		
SUB_CHARSET_X_SOFTBANK_SHIFT_JIS_2007 = 69,		
SUB_CHARSET_X_UNICODEBIG = 70,		
SUB_CHARSET_X_UTF_16LE_BOM = 71,		
SUB_CHARSET_X_UTF16_OPPOSITEENDIAN = 72,		
SUB_CHARSET_X_UTF16_PLATFORMENDIAN = 73,		
SUB_CHARSET_X_UTF32_OPPOSITEENDIAN = 74,		
SUB_CHARSET_X_UTF32_PLATFORMENDIAN = 75		

2.9.9. EAUDIOCODECFORMAT

名称	EAUDIOCODECFORMAT	
功能描述	音频编码格式	
属性	值	描述
AUDIO_CODEC_FORMAT_ = -1	-1	
AUDIO_CODEC_FORMAT_UNKNOWN = 0,	0	
AUDIO_CODEC_FORMAT_MP1,		
AUDIO_CODEC_FORMAT_MP2,		
AUDIO_CODEC_FORMAT_MP3,		
AUDIO_CODEC_FORMAT_MPEG_AAC_LC,		

AUDIO_CODEC_FORMAT_AC3		
AUDIO_CODEC_FORMAT_DTS,		
AUDIO_CODEC_FORMAT_LPCM_V,		
AUDIO_CODEC_FORMAT_LPCM_A,		
AUDIO_CODEC_FORMAT_ADPCM,		
AUDIO_CODEC_FORMAT_PCM,		
AUDIO_CODEC_FORMAT_WMA_STANDARD,		
AUDIO_CODEC_FORMAT_FLAC,		
AUDIO_CODEC_FORMAT_APE,		
AUDIO_CODEC_FORMAT_OGG,		
AUDIO_CODEC_FORMAT_RAAC,		
AUDIO_CODEC_FORMAT_COOK,		
AUDIO_CODEC_FORMAT_SIPR,		
AUDIO_CODEC_FORMAT_ATRC,		
AUDIO_CODEC_FORMAT_AMR,		
AUDIO_CODEC_FORMAT_RA,		
AUDIO_CODEC_FORMAT_ALAC		.caf .alac .m4a
AUDIO_CODEC_FORMAT_G729		.g729
AUDIO_CODEC_FORMAT_DSD		.dsd .dsf
AUDIO_CODEC_FORMAT_OPUS		.opus .ogg
CDX_AUDIO_MLP = CDX_AUDIO_AC3,		支持
CDX_AUDIO_PPCM = CDX_AUDIO_UNKNOWN,		不支持
CDX_AUDIO_WMA_LOSS = CDX_AUDIO_UNKNOWN,		不支持
CDX_AUDIO_WMA_PRO = CDX_AUDIO_UNKNOWN,		不支持
CDX_AUDIO_MP3_PRO = CDX_AUDIO_UNKNOWN,		不支持

2.9.10. WAVE_FORMAT

名称	WAVE_FORMAT	
功能描述	WAV 文件内部子格式	
属性	值	描述
WAVE_FORMAT_UNKNOWN	0x0000	/* Unknown Format */
WAVE_FORMAT_PCM	0x0001	/* PCM */
WAVE_FORMAT_ADPCM	0x0002	/* Microsoft ADPCM Format */
WAVE_FORMAT_IEEE_FLOAT	0x0003	/* IEEE Float */
WAVE_FORMAT_VSELP	0x0004	/* Compaq Computer's VSELP */
WAVE_FORMAT_IBM_CSVD	0x0005	/* IBM CVSD */
WAVE_FORMAT_ALAW	0x0006	/* ALAW */
WAVE_FORMAT_MULAW	0x0007	/* MULAW */
WAVE_FORMAT_OKI_ADPCM	0x0010	/* OKI ADPCM */
WAVE_FORMAT_DVI_ADPCM	0x0011	/* Intel's DVI ADPCM */
WAVE_FORMAT_MEDIASPACE_ADPCM	0x0012	/* Videologic's MediaSpace

		ADPCM*/
WAVE_FORMAT_SIERRA_ADPCM	0x0013	/* Sierra ADPCM */
WAVE_FORMAT_G723_ADPCM	0x0014	/* G.723 ADPCM */
WAVE_FORMAT_DIGISTD	0x0015	/* DSP Solution's DIGISTD */
WAVE_FORMAT_DIGIFIX	0x0016	/* DSP Solution's DIGIFIX */
WAVE_FORMAT_DIALOGIC_OKI_ADPCM	0x0017	/* Dialogic OKI ADPCM */
WAVE_FORMAT_MEDIAVISION_ADPCM	0x0018	/* MediaVision ADPCM */
WAVE_FORMAT_CU_CODEC	0x0019	/* HP CU */
WAVE_FORMAT_YAMAHA_ADPCM	0x0020	/* Yamaha ADPCM */
WAVE_FORMAT_SONARC	0x0021	/* Speech Compression's Sonarc */
WAVE_FORMAT_TRUESPEECH	0x0022	/* DSP Group's True Speech */
WAVE_FORMAT_ECHOSC1	0x0023	/* Echo Speech's EchoSC1 */
WAVE_FORMAT_AUDIOFILE_AF36	0x0024	/* Audiofile AF36 */
WAVE_FORMAT_APTX	0x0025	/* APTX */
WAVE_FORMAT_AUDIOFILE_AF10	0x0026	/* AudioFile AF10 */
WAVE_FORMAT_PROSODY_1612	0x0027	/* Prosody 1612 */
WAVE_FORMAT_LRC	0x0028	/* LRC */
WAVE_FORMAT_AC2	0x0030	/* Dolby AC2 */
WAVE_FORMAT_GSM610	0x0031	/* GSM610 */
WAVE_FORMAT_MSNAUDIO	0x0032	/* MSNAudio */
WAVE_FORMAT_ANTEX_ADPCME	0x0033	/* Antex ADPCME */
WAVE_FORMAT_CONTROL_RES_VQLPC	0x0034	/* Control Res VQLPC */
WAVE_FORMAT_DIGIREAL	0x0035	/* Digireal */
WAVE_FORMAT_DIGIADPCM	0x0036	/* DigiADPCM */
WAVE_FORMAT_CONTROL_RES_CR10	0x0037	/* Control Res CR10 */
WAVE_FORMAT_VBXADPCM	0x0038	/* NMS VBXADPCM */
WAVE_FORMAT_ROLAND_RDAC	0x0039	/* Roland RDAC */
WAVE_FORMAT_ECHOSC3	0x003A	/* EchoSC3 */
WAVE_FORMAT_ROCKWELL_ADPCM	0x003B	/* Rockwell ADPCM */
WAVE_FORMAT_ROCKWELL_DIGITALK	0x003C	/* Rockwell Digit LK */
WAVE_FORMAT_XEBEC	0x003D	/* Xebec */
WAVE_FORMAT_G721_ADPCM	0x0040	/* Antex Electronics G.721 */
WAVE_FORMAT_G728_CELP	0x0041	/* G.728 CELP */
WAVE_FORMAT_MSG723	0x0042	/* MSG723 */
WAVE_FORMAT_MPEG	0x0050	/* MPEG Layer 1,2 */
WAVE_FORMAT_RT24	0x0051	/* RT24 */
WAVE_FORMAT_PAC	0x0051	/* PAC */
WAVE_FORMAT_MPEGLAYER3	0x0055	/* MPEG Layer 3 */
WAVE_FORMAT_CIRRUS	0x0059	/* Cirrus */
WAVE_FORMAT_ESPCM	0x0061	/* ESPCM */
WAVE_FORMAT_VOXWARE	0x0062	/* Voxware (obsolete) */
WAVE_FORMAT_CANOPUS_ATRAC	0x0063	/* Canopus Atrac */

WAVE_FORMAT_G726_ADPCM	0x0064	/* G.726 ADPCM */
WAVE_FORMAT_G722_ADPCM	0x0065	/* G.722 ADPCM */
WAVE_FORMAT_DSAT	0x0066	/* DSAT */
WAVE_FORMAT_DSAT_DISPLAY	0x0067	/* DSAT Display */
WAVE_FORMAT_VOXWARE_BYTE_ALIGNED	0x0069	/* Voxware Byte Aligned (obsolete)
WAVE_FORMAT_VOXWARE_AC8	0x0070	/* Voxware AC8 (obsolete) */
WAVE_FORMAT_VOXWARE_AC10	0x0071	/* Voxware AC10 (obsolete) */
WAVE_FORMAT_VOXWARE_AC16	0x0072	/* Voxware AC16 (obsolete) */
WAVE_FORMAT_VOXWARE_AC20	0x0073	/* Voxware AC20 (obsolete) */
WAVE_FORMAT_VOXWARE_RT24	0x0074	/* Voxware MetaVoice (obsolete) */
WAVE_FORMAT_VOXWARE_RT29	0x0075	/* Voxware MetaSound (obsolete) */
WAVE_FORMAT_VOXWARE_RT29HW	0x0076	/* Voxware RT29HW (obsolete) */
WAVE_FORMAT_VOXWARE_VR12	0x0077	/* Voxware VR12 (obsolete) */
WAVE_FORMAT_VOXWARE_VR18	0x0078	/* Voxware VR18 (obsolete) */
WAVE_FORMAT_VOXWARE_TQ40	0x0079	/* Voxware TQ40 (obsolete) */
WAVE_FORMAT_SOFTSOUND	0x0080	/* Softsound */
WAVE_FORMAT_VOXWARE_TQ60	0x0081	/* Voxware TQ60 (obsolete) */
WAVE_FORMAT_MSRT24	0x0082	/* MSRT24 */
WAVE_FORMAT_G729A	0x0083	/* G.729A */
WAVE_FORMAT_MVI_MV12	0x0084	/* MVI MV12 */
WAVE_FORMAT_DF_G726	0x0085	/* DF G.726 */
WAVE_FORMAT_DF_GSM610	0x0086	/* DF GSM610 */
WAVE_FORMAT_ISIAUDIO	0x0088	/* ISIAudio */
WAVE_FORMAT_ONLIVE	0x0089	/* Onlive */
WAVE_FORMAT_SBC24	0x0091	/* SBC24 */
WAVE_FORMAT_DOLBY_AC3_SPDIF	0x0092	/* Dolby AC3 SPDIF */
WAVE_FORMAT_ZYXEL_ADPCM	0x0097	/* ZyXEL ADPCM */
WAVE_FORMAT_PHILIPS_LPCBB	0x0098	/* Philips LPCBB */
WAVE_FORMAT_PACKED	0x0099	/* Packed */
WAVE_FORMAT_RHETOREX_ADPCM	0x0100	/* Rhetorex ADPCM */
WAVE_FORMAT_IRAT	0x0101	/* BeCubed Software's IRAT */
WAVE_FORMAT_VIVO_G723	0x0111	/* Vivo G.723 */
WAVE_FORMAT_VIVO_SIREN	0x0112	/* Vivo Siren */
WAVE_FORMAT_DIGITAL_G723	0x0123	/* Digital G.723 */
WAVE_FORMAT_CREATIVE_ADPCM	0x0200	/* Creative ADPCM */
WAVE_FORMAT_CREATIVE_FASTSPEECH8	0x0202	/* Creative FastSpeech8 */
WAVE_FORMAT_CREATIVE_FASTSPEECH10	0x0203	/* Creative FastSpeech10 */
WAVE_FORMAT_QUARTERDECK	0x0220	/* Quarterdeck */
WAVE_FORMAT_FM_TOWNS_SND	0x0300	/* FM Towns Snd */
WAVE_FORMAT_BTV_DIGITAL	0x0400	/* BTV Digital */

WAVE_FORMAT_VME_VMPCM	0x0680	/* VME VMPCM */
WAVE_FORMAT_OLIGSM	0x1000	/* OLIGSM */
WAVE_FORMAT_OLIADPCM	0x1001	/* OLIADPCM */
WAVE_FORMAT_OLICELP	0x1002	/* OLICELP */
WAVE_FORMAT_OLISBC	0x1003	/* OLISBC */
WAVE_FORMAT_OLIOPR	0x1004	/* OLIOPR */
WAVE_FORMAT_LH_CODEC	0x1100	/* LH Codec */
WAVE_FORMAT_NORRIS	0x1400	/* Norris */
WAVE_FORMAT_SOUNDSPACE_MUSICOMPRESS	0x1500	/* Soundspace Music Compression */
WAVE_FORMAT_DVM	0x2000	/* DVM */
WAVE_FORMAT_EXTENSIBTSMIRACAST	0xFFFC	/* LSZHANG TS Miracast WIFI*/
WAVE_FORMAT_EXTENSIBTS	0xFFFD	/* LSZHANG TS */
WAVE_FORMAT_EXTENSIBLE	0xFFFE	/* SubFormat */
WAVE_FORMAT_DEVELOPMENT	0xFFFF	/* Development */
//user define adpcm codec type from video file		
ADPCM_CODEC_ID_IMA_QT	0xE000	
ADPCM_CODEC_ID_IMA_WAV	0xE001	/* from avi file format */
ADPCM_CODEC_ID_IMA_DK3	0xE002	
ADPCM_CODEC_ID_IMA_DK4	0xE003	
ADPCM_CODEC_ID_IMA_WS	0xE004	
ADPCM_CODEC_ID_IMA_SMJPEG	0xE005	
ADPCM_CODEC_ID_MS	0xE006	
ADPCM_CODEC_ID_4XM	0xE007	
ADPCM_CODEC_ID_XA	0xE008	
ADPCM_CODEC_ID_ADX	0xE009	
ADPCM_CODEC_ID_EA	0xE00A	
ADPCM_CODEC_ID_G726	0xE00B	
ADPCM_CODEC_ID_CT	0xE00C	
ADPCM_CODEC_ID_SWF	0xE00D	/* from flv/swf file format */
ADPCM_CODEC_ID_YAMAHA	0xE00E	
ADPCM_CODEC_ID_SBPRO_4	0xE00F	
ADPCM_CODEC_ID_SBPRO_3	0xE010	
ADPCM_CODEC_ID_SBPRO_2	0xE011	

2.9.11. AUIDO_RAW_DATA_TYPE

名称	AUDIO_RAW_DATA_TYPE
功能描述	IEC61937 协议规定的音频输出码流枚举类型，供 HDMI 驱动使用去配置传输方式，

	一般软解的情况选择AUDIO_RAW_DATA_PCM,需要透传或者硬解时才配置1以上的值	
属性	值	描述
AUDIO_RAW_DATA_UNKOWN	0	类型不明
AUDIO_RAW_DATA_PCM	1	Linear pcm audio
AUDIO_RAW_DATA_AC3	2	IEC61937 packed compressed ac3 audio, transfer over pcm
AUDIO_RAW_DATA_MPEG1	3	IEC61937 packed compressed mpeg1 audio, transfer over pcm
AUDIO_RAW_DATA_MP3	4	IEC61937 packed compressed mp3 audio, transfer over pcm
AUDIO_RAW_DATA_MPEG2	5	IEC61937 packed compressed mpeg2 audio, transfer over pcm
AUDIO_RAW_DATA_AAC	6	IEC61937 packed compressed aac audio, transfer over pcm
AUDIO_RAW_DATA_DTS	7	IEC61937 packed compressed dts audio, transfer over pcm
AUDIO_RAW_DATA_ATRAC	8	IEC61937 packed compressed atrac audio, transfer over pcm
AUDIO_RAW_DATA_ONE_BIT_AUDIO	9	IEC61937 packed compressed dsd audio, transfer over pcm
AUDIO_RAW_DATA_DOLBY_DIGITAL_PLUS	10	IEC61937 packed compressed dolby digital plus audio, transfer over pcm
AUDIO_RAW_DATA_DTS_HD	11	IEC61937 packed compressed dts-hd audio, transfer over pcm
AUDIO_RAW_DATA_MAT	12	IEC61937 packed compressed dolby hd audio, transfer over pcm
AUDIO_RAW_DATA_DST	13	IEC61937 packed compressed dst audio, transfer over pcm
AUDIO_RAW_DATA_WMAPRO	14	IEC61937 packed compressed wma-pro audio, transfer over pcm