全志科技
Allwinner Technology

音视频解码库项目

播控模块概要设计报告

文档履历

版本号	日期	制/修订人	内容描述
V0.1	2014-4-15		建立初稿
V0.2	2015-5-28		修改接口、数据结构体，修正病句
V1.0	2015-5-28		Release 版

目 录

1. 概述

1.1.编写目的

设计播控模块的基本框架、内/外部接口、主要数据结构和流程。指导播控模块的开发、使用和后续维护。

1.2.适用范围

A80/A83/H3/H8 等各个芯片平台的 Android 系统 SDK 和 Linux SDK。

1.3.相关人员

开发和维护播控模块的相关人员。

2. 模块介绍

2.1. 功能介绍

播控模块类似于中间件的作用，与音视频解码库、demux 对接，提供上层 player 的 API，方便上层实现 player 的功能。协调各组件，如音频解码器、视频解码器之间的运作，提高解码播放性能，实现音视频的同步播放。

2.2. 相关术语介绍

Player: 播放控制模块
PTS：时间戳，表示图像的显示时间，presentation time stamp 的缩写

3. 播控模块 API 说明

3.1. 控制类 API

3.1.1. PlayerCreate

函数原型	Player* PlayerCreate(void)
功能	创建一个 player
参数	无
返回值	成功：player 指针 失败：返回 NULL
调用说明	无

3.1.2. PlayerDestroy

函数原型	void PlayerDestroy(Player* pl)
功能	销毁一个 player
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	无
调用说明	无

3.1.3. PlayerSetCallback

函数原型	int PlayerSetCallback(Player* pl, PlayerCallback callback, void* pUserData)
功能	设置 player 的回调函数，player 通过此接口将一些状态及时回调到上层
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 Callback：回调函数接口 pUserData：自定义数据
返回值	0：表示成功
调用说明	在 PlayerStart 之前调用此接口

3.1.4. PlayerStart

函数原型	int PlayerStart(Player* pl)
功能	启动 player
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：启动正常 -1：调用此接口的时机不对或相关的资源尚未准备好

调用说明	在 PlayerSetVideoStreamInfo 和 PlayerSetAudioStreamInfo 之后调用此接口

3.1.5. PlayerStop

函数原型	int PlayerStop(Player* pl)
功能	将 player 置为停止状态
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：设置成功 -1：表示 player 已经处于 stop 状态，操作无效
调用说明	无

3.1.6. PlayerPause

函数原型	int PlayerPause(Player* pl)
功能	将 player 置为暂停状态
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：设置成功 -1：player 不是处于 start 状态，设置失败
调用说明	在 player 处于 start 状态时调用此接口

3.1.7. PlayerReset

函数原型	int PlayerReset(Player* pl, int64_t nSeekTimeUs)
功能	重置 player，用于跳播操作
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 nSeekTimeUs：跳播的播放时间点
返回值	0：表示重置成功 -1：调用此接口时，player 处于 stop 状态，重置失败；
调用说明	此重置接口用于 seek 即跳播，调用此接口前，必须先调用 PlayerPause，使 player 处于暂停状态，否则调用无效

3.1.8. PlayerClear

函数原型	int PlayerClear(Player* pl)
功能	清空 player，销毁各模块，如 AudioRender，VideoRender，AudioDecCom，VideoDecCom 等
参数	Pl：通过 PlayerCreate 函数创建的 player 指针

返回值	0：表示成功清空 player -1：表示调用此接口时，player 不处于 stop 状态，操作失败
调用说明	调用此接口前，必须先调用 PlayerStop，使 player 处于 stop 的状态进行清空操作，否则操作无效

3.1.9. PlayerFast

函数原型	int PlayerFast(Player* pl)
功能	设置 player 进入快进模式
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：设置成功
调用说明	无

3.1.10. PlayerStopFast

函数原型	int PlayerStopFast(Player* pl)
功能	设置 player 退出快进播放模式，恢复到正常播放模式
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：设置成功
调用说明	无

3.1.11. PlayerGetStatus

函数原型	enum EPLAYERSTATUS PlayerGetStatus(Player* pl)
功能	获取 player 的状态
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	返回 player 的状态值，即枚举结构体 EPLAYERSTATUS 的数值： PLAYER_STATUS_STOPPED ：停止状态 PLAYER_STATUS_STARTED ：开始状态 PLAYER_STATUS_PAUSED ：暂停状态
调用说明	无

3.1.12. PlayerGetPositionCMCC

函数原型	int64_t PlayerGetPositionCMCC(Player* pl)
功能	获取 player 当前的播放时间点，即播放位置
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	返回当前的播放时间点，单位为微秒 us
调用说明	此函数为 CMCC 场景专用，CMCC 要求返回的播放时间跟普通的播放场景不同，故用此函数进行区分。

3.1.13. PlayerGetPosition

函数原型	int64_t PlayerGetPosition(Player* pl)
功能	获取 player 当前的播放时间点，即播放位置
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	返回当前的播放时间点，单位为微秒 us
调用说明	无

3.1.14. PlayerGetPts

函数原型	int64_t PlayerGetPts(Player* pl)
功能	获取 player 当前播放的时间戳 pts
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	返回当前播放的 pts，单位为微秒 us
调用说明	无

3.1.15. PlayerSetDiscardAudio

函数原型	int64_t PlayerSetDiscardAudio(Player* pl， int f)
功能	设置静音
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 F：是否丢掉音频标志
返回值	0：设置成功
调用说明	用于倍速播放不输出音频

3.2.控制数据 API

3.2.1. PlayerRequestStreamBuffer

函数原型	int PlayerRequestStreamBuffer(Player*　　pl, int　　nRequireSize, void**　　ppBuf, int*　　pBufSize, void**　　ppRingBuf, int*　　pRingBufSize, enum EMEDIATYPE eMediaType, int　　nStreamIndex)

功能	获取视频或音频码流的 buffer，用于填充码流
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 nRequireSize：请求码流 buffer 的大小 ppBuf：用于存放请求到的第一笔 buffer 指针 pBufSize：请求到的第一笔 buffer 的大小 ppRingBuf：请求到的第二笔 buffer 的指针 pRingBufSize：请求到的第二笔 buffer 的大小 eMediaType：码流类型 MEDIA_TYPE_VIDEO ：视频流 MEDIA_TYPE_AUDIO ：音频流 MEDIA_TYPE_SUBTITLE ：字幕流 nStreamIndex：第几路码流，视频、音频、字幕都会存在多路码流的情况，通过此参数指明播放的是哪一路码流
返回值	0：表示请求 buffer 成功，但不一定请求到的 buffer 大小为 nRequireSize，实际请求到的大小为 pBufSize + pRingBufSize -1：表示请求失败，buffer 已满
调用说明	调用此接口会请求到两笔 buffer，是因为 audio 和 video 用于存放码流的 buffer 采用的是循环 buffer 机制，当 buffer 回头时，就会出现有两笔 buffer； 调用此接口填充完码流数据后，应马上调用 PlayerSubmitStreamData，将数据提交到 player，结束一笔码流数据的填充

3.2.2. PlayerSubmitStreamData

函数原型	int PlayerSubmitStreamData(Player* pl, MediaStreamDataInfo* pDataInfo, enum EMEDIATYPE eMediaType, int nStreamIndex)
功能	填充完码流数据后，将数据提交给 player
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 pDataInfo：将要提交的数据的基本信息，如数据长度 nLength、时间戳 npts eMediaType：码流类型 MEDIA_TYPE_VIDEO ：视频流 MEDIA_TYPE_AUDIO ：音频流 MEDIA_TYPE_SUBTITLE ：字幕流 nStreamIndex：第几路码流，视频、音频、字幕都会存在多路码流的情况，通过此参数指明播放的是哪一路码流
返回值	0：表示成功提交码流数据 -1：表示提交失败
调用说明	调用此接口前先调用 PlayerRequestStreamBuffer，两者配套调用

3.2.3. PlayerSetEos

函数原型	int PlayerSetEos(Player* pl)
功能	向 player 设置码流结束信息 Eos，告知 player 已经播放到码流的结束位置
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：表示设置成功
调用说明	无

3.2.4. PlayerSetFirstPts

函数原型	int PlayerSetFirstPts(Player* pl, int64_t nFirstPts)
功能	设置第一个 video 或者 audio 的 pts(取两者的最小值),用于计算字幕的 pts。在 ts parser 中，大多数情况下第一个 video 和 audio 的 pts 都不是 0，而外挂的 pts 往往都是从 0 开始的，在这种场景下，必须基于 nFirstPts 计算字幕的 pts。
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 nFirstPts：第一个 video 或者 audio 的 pts
返回值	0：表示设置成功
调用说明	无

3.2.5. PlayerGetVideoBitrate

函数原型	int PlayerGetVideoBitrate(Player* pl)
功能	获取视频的码率
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：获取失败 正常数值：视频的码率值
调用说明	无

3.2.6. PlayerGetVideoFrameRate

函数原型	int PlayerGetVideoFrameRate(Player* pl)
功能	获取视频的帧率
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：获取失败 正常值：帧率大小
调用说明	无

3.2.7. PlayerGetVideoFrameDuration

函数原型	int PlayerGetVideoFrameDuration(Player* pl)
功能	获取视频的帧间隔
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：获取失败 正常值：帧间隔大小
调用说明	无

3.2.8. PlayerGetVideoStreamDataSize

函数原型	int PlayerGetVideoStreamDataSize(Player* pl)
功能	获取视频解码器持有的码流数据的大小
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：获取失败 正常值：码流数据的大小值
调用说明	无

3.2.9. PlayerGetVideoStreamFrameNum

函数原型	int PlayerGetVideoStreamFrameNum(Player* pl)
功能	获取视频解码器持有的码流数据的数目（码流是一笔一笔数据传下来的）
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：获取失败 正常值：码流数据的数目值
调用说明	无

3.2.10. PlayerGetValidPictureNum

函数原型	int PlayerGetVideoStreamDataSize(Player* pl)
功能	获取视频解码器已经解出来的并准备用于显示的视频帧的数目
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：获取失败 正常值：视频帧的数目值
调用说明	无

3.2.11. PlayerGetAudioBitrate

函数原型	int PlayerGetAudioBitrate(Player* pl)
功能	获取音频的码率
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：获取失败 正常数值：音频的码率值
调用说明	无

3.2.12. PlayerGetAudioParam

函数原型	int PlayerGetAudioParam(Player* pl, int* pSampleRate, int* pChannelCount, int* pBitsPerSample)
功能	获取音频的采样率、轨道数、pcm 数据的码率参数
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 pSampleRate：用于存放采样率参数的变量的指针 pChannelCount：用于存放轨道数的变量的指针 pBitsPerSample：用于存放 pcm 数据的码率的变量的指针
返回值	0：获取成功 -1：获取失败
调用说明	无

3.2.13. PlayerGetAudioStreamDataSize

函数原型	int PlayerGetAudioStreamDataSize(Player* pl)
功能	获取音频解码器持有的码流数据的大小
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：获取失败 正常值：码流数据的大小
调用说明	无

3.2.14. PlayerGetAudioStreamFrameNum

函数原型	int PlayerGetAudioStreamFrameNum(Player* pl)
功能	获取音频解码器持有的码流数据的帧数
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：获取失败 正常值：码流帧数的大小
调用说明	无

3.2.15. PlayerGetAudioPcmDataSize

函数原型	int PlayerGetAudioPcmDataSize(Player* pl)
功能	获取音频解码器已经解出来的 pcm 数据的大小
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：获取失败 正常值：pcm 数据的大小
调用说明	无

3.2.16. PlayerGetAudioCacheTimeInSoundDevice

函数原型	int PlayerGetAudioCacheTimeInSoundDevice(Player* pl)
功能	获取音频设备缓冲的 pcm 数据的时间长度
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：获取失败 正常值：音频设备缓冲的 pcm 数据的时间长度
调用说明	无

3.3. 视频相关 API

3.3.1. PlayerSetVideoStreamInfo

函数原型	int PlayerSetVideoStreamInfo(Player* pl, VideoStreamInfo* pStreamInfo)
功能	设置视频码流信息，player 根据视频码流信息和配置信息初始化视频解码器
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 pStreamInfo：视频码流的基本信息，如分辨率、帧率、视频宽高等
返回值	0：设置成功 -1：设置失败，设置的时机不对或内存资源不足
调用说明	在 PlayerStart 之前调用此接口 VideoStreamInfo 中，不是所有信息都是必须的。 对于 H264 和 MPEG2，可以只填写编码格式信息（format），其他视频一般还需要正确填写视频分辨率信息（width 和 height，由于码流中没有该信息）。某些视频解码前需要初始化数据（initData）。

3.3.2. PlayerCanSupportVideoStream

函数原型	int PlayerCanSupportVideoStream(Player* pl , VideoStreamInfo* pStreamInfo)
功能	检测是否支持该视频格式
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	
调用说明	暂无用

3.3.3. PlayerHasVideo

函数原型	int PlayerHasVideo(Player* pl)

功能	检测 player 有无视频
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：表示无视频 1：表示有视频
调用说明	无

3.3.4. PlayerConfigVideoScaleDownRatio

函数原型	int PlayerConfigVideoScaleDownRatio(Player* pl, int nHorizonRatio, int nVerticalRatio)
功能	设置视频画面缩放比例
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 nHorizonRatio：水平缩放比例值 nVerticalRatio：垂直缩放比例值
返回值	0：表示设置成功 -1：表示设置失败，调用此接口时，player 不处于 stop 状态
调用说明	在 PlayerSetVideoStreamInfo 之前调用此接口，否则会导致设置失败或者无效

3.3.5. PlayerConfigVideoRotateDegree

函数原型	int PlayerConfigVideoRotateDegree(Player* pl, int nDegree)
功能	设置视频画面的旋转角度
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 nDegree：旋转角度值
返回值	0：表示设置成功 -1：表示设置失败，调用此接口时，player 不处于 stop 状态
调用说明	在 PlayerSetVideoStreamInfo 之前调用此接口，否则会导致设置失败或者无效

3.3.6. PlayerConfigVideoDeinterlace

函数原型	int PlayerConfigVideoDeinterlace(Player* pl, int bDeinterlace)
功能	设置视频的反交错功能
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 bDeinterlace：0 为不开启此功能，1 为开启此功能
返回值	0：表示设置成功 -1：表示设置失败，调用此接口时，player 不处于 stop 状态
调用说明	在 PlayerSetVideoStreamInfo 之前调用此接口，否则会导致设置失败或者无

	效； 暂无用

3.3.7. PlayerConfigDispErrorFrame

函数原型	int PlayerConfigDispErrorFrame(Player* pl, int bDispErrorFrame)
功能	设置是否显示错误帧
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 bDispErrorFrame：0 表示不显示错误帧，1 表示显示错误帧
返回值	0：设置成功 -1：设置失败
调用说明	无

3.3.8. PlayerConfigDropLaytedFrame

函数原型	int PlayerConfigDropLaytedFrame(Player* pl, int bDropLaytedFrame)
功能	设置是否使解码器丢弃过时的帧
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 bDropLaytedFrame：0 表示不丢弃过时的帧，1 表示丢弃过时的帧
返回值	0：设置成功 -1：设置失败
调用说明	无

3.3.9. PlayerConfigSetMemoryThresh

函数原型	int PlayerConfigSetMemoryThresh(Player* pl, int nMemoryThresh)
功能	设置解码器可申请的物理连续内存的最大值
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 nMemoryThresh：解码器可申请的物理连续内存的最大值
返回值	0：设置成功 -1：设置失败
调用说明	在某些方案上，如 H3 512M 方案，内存空间有限，此时可调用此函数设置解码器可申请物理连续内存的上限。在内存空间足够大的平台上不需要调用此设置函数。（暂无用）

3.3.10. PlayerConfigTvStreamFlag

函数原型	Int PlayerConfigTvStreamFlag(Player* pl, int bFlag)
功能	设置是否为数字电视播放
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 nMemoryThresh：解码器可申请的物理连续内存的最大值
返回值	0：设置成功 -1：设置失败
调用说明	在某些方案上，如 H3 512M 方案，内存空间有限，此时可调用此函数设置解

	码器可申请物理连续内存的上限。在内存空间足够大的平台上不需要调用此设置函数。（暂无用）

3.4.音频相关 API

3.4.1. PlayerSetAudioStreamInfo

函数原型	int PlayerSetAudioStreamInfo(Player* pl, AudioStreamInfo* pStreamInfo, int nStreamNum, int nDefaultStream)
功能	设置音频码流信息，player 根据音频码流信息和配置信息初始化音频解码器
参数	P1：通过 PlayerCreate 函数创建的 player 指针 pStreamInfo：音频码流的基本信息； nStreamNum ：音频所包含的音轨数目； nDefaultStream：player 默认播放的第几条音轨
返回值	0：音轨数目为 0 或成功设置音频信息； -1：调用此函数的时机不对或内存资源不足； -2：音频解码库不支持默认的音轨，可条用 SwitchAudio 选择其他音频进行播放
调用说明	在 PlayerStart 之前调用此接口

3.4.2. PlayerAddAudioStream

函数原型	int PlayerAddAudioStream(Player* pl, AudioStreamInfo* pStreamInfo)
功能	添加音频新的音轨
参数	P1：通过 PlayerCreate 函数创建的 player 指针； pStreamInfo：音轨的码流信息
返回值	0：表示添加成功： -1：表示添加失败，内存资源不足
调用说明	无

3.4.3. PlayerGetAudioStreamCnt

函数原型	int PlayerGetAudioStreamCnt(Player* pl)
功能	获取音频所包含的音轨的数目
参数	P1：通过 PlayerCreate 函数创建的 player 指针
返回值	返回音轨的数量
调用说明	无

3.4.4. PlayerHasAudio

函数原型	int PlayerHasAudio(Player* pl)
功能	检测 player 有无音频
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：表示无音频 1：表示有音频
调用说明	无

3.4.5. PlayerStartAudio

函数原型	int PlayerStartAudio(Player* pl)
功能	启动音频的播放
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：调用成功 -1：调用失败
调用说明	IPTV 场景专用

3.4.6. PlayerSwitchAudio

函数原型	int PlayerSwitchAudio(Player* pl, int nStreamIndex)
功能	切换音频轨道
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 nStreamIndex：音频轨道的索引值
返回值	0：调用成功 -1：调用失败
调用说明	无

3.4.7. PlayerGetAudioStreamInfo

函数原型	int PlayerGetAudioStreamInfo(Player* pl, int* pStreamNum, AudioStreamInfo** ppStreamInfo)
功能	获取音频流的信息
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 * pStreamNum： 用于存放音频轨道数目的指针 ** ppStreamInfo：用于存放音频轨道信息的指针
返回值	0：获取成功 -1：获取失败
调用说明	无

3.5.字幕相关 API

3.5.1. PlayerSetSubtitleStreamInfo

函数原型	int PlayerSetSubtitleStreamInfo(Player* pl, SubtitleStreamInfo* pStreamInfo, int nStreamNum, int nDefaultStream)
功能	设置字幕流的信息
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 pStreamInfo：存放字幕流信息的指针 nStreamNum ：字幕轨道数目 nDefaultStream：默认播放字幕轨道的 index
返回值	0：设置成功 -1：设置失败
调用说明	无

3.5.2. PlayerAddSubtitleStream

函数原型	int PlayerAddSubtitleStream(Player* pl, SubtitleStreamInfo* pStreamInfo)
功能	添加字幕轨道
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 pStreamInfo：存放字幕流信息的信息
返回值	0：添加成功 -1：添加失败
调用说明	无

3.5.3. PlayerGetSubtitleStreamCnt

函数原型	int PlayerGetSubtitleStreamCnt(Player* pl)
功能	获取字幕轨道数目
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	正常值：字幕轨道数值
调用说明	无

3.5.4. PlayerGetSubtitleStreamInfo

函数原型	int PlayerGetSubtitleStreamInfo(Player* pl, int* pStreamNum, SubtitleStreamInfo** ppStreamInfo)
功能	获取字幕轨道信息
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 pStreamNum：用于存放字幕轨道数目的指针 ppStreamInfo：用于存放字幕轨道信息的指针
返回值	0：获取成功

	-1：获取失败
调用说明	无

3.5.5. PlayerProbeSubtitleStreamInfo

函数原型	int PlayerProbeSubtitleStreamInfo(const char* strFileName, int* pStreamNum, SubtitleStreamInfo** ppStreamInfo)
功能	探测外挂字幕的信息
参数	strFileName： 字幕文件的路径名 pStreamNum： 用于存放字幕轨道数目的指针 SubtitleStreamInfo：用于存放字幕轨道信息的指针
返回值	0：探测成功 -1：探测失败
调用说明	无

3.5.6. PlayerProbeSubtitleStreamInfo

函数原型	int PlayerProbeSubtitleStreamInfoFd(int fd, int offset, int len, int* pStreamNum, SubtitleStreamInfo** ppStreamInfo)
功能	探测外挂字幕的信息
参数	Fd：字幕文件句柄 Offset：字幕文件有效数据的偏移值 Len：字幕文件有效数据的长度 pStreamNum： 用于存放字幕轨道数目的指针 SubtitleStreamInfo：用于存放字幕轨道信息的指针
返回值	0：探测成功 -1：探测失败
调用说明	无

3.5.7. PlayerSwitchSubtitle

函数原型	int PlayerSwitchSubtitle(Player* pl, int nStreamIndex)
功能	切换字幕轨道
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 nStreamIndex：将要播放的字幕轨道的 index
返回值	0：切换成功 -1：切换失败
调用说明	无

3.5.8. PlayerSetSubtitleShowTimeAdjustment

函数原型	int PlayerSetSubtitleShowTimeAdjustment(Player* pl, int nTimeMs)
功能	设置提前或延迟显示字幕

参数	Pl：通过 PlayerCreate 函数创建的 player 指针 nTimeMs： 若为负值则会提前显示，若为正值则会延迟显示
返回值	0：设置成功 -1：设置失败
调用说明	无

3.5.9. PlayerGetSubtitleShowTimeAdjustment

函数原型	int PlayerGetSubtitleShowTimeAdjustment(Player* pl)
功能	获取提前或延迟显示字幕的时间大小
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	其他值：获取成功 -1：获取失败
调用说明	无

3.5.10. PlayerSetSubCtrl

函数原型	int PlayerSetSubCtrl(Player* pl, SubCtrl* pSubCtrl)
功能	设置字幕显示输出接口
参数	Pl：通过 PlayerCreate 函数创建的 player 指针； pSubCtrl：字幕输出接口
返回值	其他值：获取成功 -1：获取失败
调用说明	无

3.6. 显示输出控制相关 API

3.6.1. PlayerSetWindow

函数原型	int PlayerSetWindow(Player* pl, LayerCtrl* pNativeWindow)
功能	设置视频输出 layerControl 接口，用于视频帧的显示
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 pNativeWindow：layerControl 接口
返回值	0：表示设置成功 -1：表示设置出现异常
调用说明	在 PlayerStart 之前调用此接口

3.6.2. PlayerSet3DMode

函数原型	int PlayerSet3DMode(Player* pl, enum EPICTURE3DMODE ePicture3DMode,

	enum EDISPLAY3DMODE eDisplay3DMode)
功能	设置 3D 播放模式
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 ePicture3DMode：3D 图像模式参数 PICTURE_3D_MODE_NONE , PICTURE_3D_MODE_TWO_SEPERATED_PICTURE, PICTURE_3D_MODE_SIDE_BY_SIDE, PICTURE_3D_MODE_TOP_TO_BOTTOM, PICTURE_3D_MODE_LINE_INTERLEAVE, PICTURE_3D_MODE_COLUME_INTERLEAVE eDisplay3DMode：3D 显示模式参数 DISPLAY_3D_MODE_2D, DISPLAY_3D_MODE_3D, DISPLAY_3D_MODE_HALF_PICTURE
返回值	0：设置成功 -1：设置失败
调用说明	无

3.6.3. PlayerGet3DMode

函数原型	int PlayerGet3DMode(Player* pl, enum EPICTURE3DMODE* ePicture3DMode, enum EDISPLAY3DMODE* eDisplay3DMode)
功能	获取 3D 播放模式
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 ePicture3DMode：用于存放 3D 图像模式的指针 eDisplay3DMode：用于存放 3D 显示模式的指针
返回值	0：获取成功 -1：获取失败
调用说明	无

3.6.4. PlayerGetPictureSize

函数原型	int PlayerGetPictureSize(Player* pl, int* pWidth, int* pHeight)
功能	获取视频的尺寸大小
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 pWidth：用于存放视频宽度的指针 pHeight：用于存放视频高度的指针
返回值	0：获取成功 -1：获取失败
调用说明	无

3.6.5. PlayerGetPictureCrop

函数原型	int PlayerGetPictureCrop(Player* pl, int* pLeftOff, int* pTopOff, int* pCropWidth, int* pCropHeight)
功能	获取视频的显示区域
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 pLeftOff：用于存放左边距的指针 pTopOff：用于存放上边距的指针 pCropWidth：用于存放显示区域宽度的指针 pCropHeight：用于存放显示区域高度的指针
返回值	0：获取成功 -1：获取失败
调用说明	无

3.6.6. PlayerVideoShow

函数原型	int PlayerVideoShow(Player* pl)
功能	打开视频的显示层，使视频正常显示
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：设置成功 -1：设置失败
调用说明	无

3.6.7. PlayerVideoHide

函数原型	int PlayerVideoHide(Player* pl)
功能	隐藏视频的显示层，不显示视频，进入黑屏状态
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：设置成功 -1：设置失败
调用说明	无

3.6.8. PlayerSetHoldLastPicture

函数原型	int PlayerSetHoldLastPicture(Player* pl, int bHold)
功能	设置当 playerstop 时，是否一直显示最后一帧视频图像（默认为是）
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：设置成功 -1：设置失败
调用说明	无

3.6.9. PlayerSetHoldLastPicture

函数原型	int PlayerSetHoldLastPicture(Player* pl, int bHold)
功能	设置当 playerstop 时，是否一直显示最后一帧视频图像（默认为是）
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：设置成功 -1：设置失败
调用说明	无

3.6.10. PlayerSetHoldLastPicture

函数原型	int PlayerSetHoldLastPicture(Player* pl, int bHold)
功能	设置当 playerstop 时，是否一直显示最后一帧视频图像（默认为是）
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：设置成功 -1：设置失败
调用说明	无

3.6.11. PlayerSetHoldLastPicture

函数原型	int PlayerSetHoldLastPicture(Player* pl, int bHold)
功能	设置当 playerstop 时，是否一直显示最后一帧视频图像（默认为是）
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：设置成功 -1：设置失败
调用说明	无

3.6.12. PlayerSetDeinterlace

函数原型	int PlayerSetDeinterlace(Player* pl, Deinterlace* pDi)
功能	设置当 playerstop 时，是否一直显示最后一帧视频图像（默认为是）
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：设置成功 -1：设置失败
调用说明	无

3.6.13. PlayerSetVideoRegion（暂无用）

函数原型	int PlayerSetVideoRegion(Player* pl, int x, int y, int nWidth, int nHeight)
功能	

参数	
返回值	0：设置成功 -1：设置失败
调用说明	暂无用

3.6.14. PlayerSetDisplayRatio（暂无用）

函数原型	int PlayerSetDisplayRatio(Player* pl, enum EDISPLAYRATIO eDisplayRatio)
功能	
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：设置成功 -1：设置失败
调用说明	暂无用

3.7. 音频输出控制相关 API

3.7.1. PlayerSetAudioSink

函数原型	int PlayerSetAudioSink(Player* pl, SoundCtrl* pAudioSink)
功能	设置音频输出接口指针，用于播放音频
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 pAudioSink：音频输出接口指针
返回值	0：表示设置成功 -1：表示设置出现异常
调用说明	在 PlayerStart 之前调用此接口

3.7.2. PlayerGetAudioBalance

函数原型	int PlayerGetAudioBalance(Player* pl)
功能	获取音频当前的声道（左声道、右声道、立体声）
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	1：左声道 2：右声道 3：立体声
调用说明	无

3.7.3. PlayerSetAudioBalance

函数原型	int PlayerSetAudioBalance(Player* pl, int nAudioBalance)

功能	设置音频的左右声道
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 nAudioBalance：设置参数，1 为左声道，2 为右声道，3 为立体声
返回值	0：设置成功 -1：设置失败
调用说明	无

3.7.4. PlayerSetAudioMute（暂无用）

函数原型	int PlayerSetAudioMute(Player* pl, int bMute)
功能	设置是否屏蔽音频的输出
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 bMute： 0 为不屏蔽音频的输出，1 为屏蔽音频的输出
返回值	0：设置成功 -1：设置失败
调用说明	无

3.7.5. PlayerGetAudioMuteFlag（暂无用）

函数原型	int PlayerGetAudioMuteFlag(Player* pl)
功能	获取屏蔽音频的标志位
参数	Pl：通过 PlayerCreate 函数创建的 player 指针
返回值	0：获取成功，标志位为 0 1：获取成功，标志位为 0 -1：获取失败
调用说明	无

3.7.6. PlayerSetAudioForceWriteToDeviceFlag（暂无用）

函数原型	int PlayerSetAudioForceWriteToDeviceFlag(Player* pl, int bForceFlag)
功能	设置是否屏蔽音频的输出
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 bForceFlag： 0 为不屏蔽音频的输出，1 为屏蔽音频的输出
返回值	0：设置成功 -1：设置失败
调用说明	与 PlayerSetAudioMute 函数的作用一致，后续会将此接口删除掉，只保留 PlayerSetAudioMute 接口。

3.7.7. PlayerSetVolume（暂不用）

函数原型	int PlayerSetVolume(Player* pl, float volume)

功能	设置音频的播放音量
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 Volume：音量值
返回值	0：设置成功 -1：设置失败
调用说明	无

3.7.8. PlayerGetVolume（暂不用）

函数原型	int PlayerGetVolume(Player* pl, float *volume)
功能	获取音频的播放音量
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 Volume：用于存放音量值的指针
返回值	0：获取成功 -1：获取失败
调用说明	无

3.7.9. PlayerConfigDispSubTitleInner（暂不用）

函数原型	int PlayerConfigDispSubTitleInner(Player* pl, int bEnable)
功能	获取音频的播放音量
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 Volume：用于存放音量值的指针
返回值	0：获取成功 -1：获取失败
调用说明	无

3.8. 其他 API

3.8.1. PlayerSetPlayBackSettings

函数原型	int PlayerSetPlayBackSettings(Player* pl,const struct XAudioPlaybackRate* rate)
功能	设置播放速率（android5.0 之后支持）
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 rate：播放速率
返回值	0：获取成功 -1：获取失败

调用说明	无

3.8.2. PlayerGetPlayBackSettings

函数原型	int PlayerGetPlayBackSettings(Player* pl,XAudioPlaybackRate* rate)
功能	获取播放速率（android5.0 之后支持）
参数	Pl：通过 PlayerCreate 函数创建的 player 指针 rate：播放速率
返回值	0：获取成功 -1：获取失败
调用说明	无

4. 数据结构设计

4.1. PlayerContext

名称	PlayerContext	
功能描述	表示一个播控模块，包含播控模块的所有信息	
属性	类型	描述
pVideoStreamInfo	VideoStreamInfo*	保存视频码流基本信息的结构体指针，如编码格式、宽、高、帧率等；
pAudioStreamInfo	AudioStreamInfo*	保存音频码流基本信息的结构体数组，数组的个数由音轨的数目决定
pSubtitleStreamInfo	SubtitleStreamInfo*	保存字幕码流基本信息的结构体数组，数组的个数由字幕轨道的数目决定
nAudioStreamNum	int	音频的音轨数目
nAudioStreamSelected	int	当前播放的音频轨的索引值
nSubtitleStreamNum	int	字幕轨的数目
nSubtitleStreamSelected	int	当前播放的字幕轨的索引值
pNativeWindow	LayerCtrl*	视频输出接口句柄
pAudioSink	SoundCtrl*	音频输出接口句柄
pSubCtrl	SubCtrl*	字幕输出接口句柄
pDeinterlace	Deinterlace*	Di 设备接口句柄
pVideoRender	VideoRenderComp* (VideoRenderCompContext*)	指向视频接收模块的指针
pAudioRender	AudioRenderComp* (AudioRenderCompContext*)	指向音频接收模块的指针
pSubtitleRender	SubtitleRenderComp*	指向字幕接收模块的指针
pVideoDecComp	VideoDecComp* (VideoDecCompContext*)	指向视频解码模块的指针
pAudioDecComp	AudioDecComp* (AudioDecCompContext*)	指向音频解码模块的指针
pSubtitleDecComp	SubtitleDecComp*	指向字幕解码模块的指针
pAvTimer	AvTimer* (AvTimerContext*)	指向时钟模块的指针
pVideoBitrateEstimater	BitrateEstimater*	指向视频码率估算模块的指针

pAudioBitrateEstimater	BitrateEstimater*	指向音频码率估算模块的指针
pVideoFramerateEstimater	FramerateEstimater*	指针视频帧率估算模块的指针
nPlayRate	XAudioPlaybackRate	速率播放结构体
eStatus	enum EPLAYERSTATUS	播控模块的状态值
callback	PlayerCallback	应用程序的回调函数
pUserData	void*	应用程序的自定义数据
pAudioStreamBuffer	void*	用于存放音频码流数据的临时 buffer（disable audio 时起作用）
nAudioStreamBufferSize	int	存放音频码流数据的临时 buffer 的大小（disable audio 时起作用）
pVideoStreamBuffer	Void*	用于存放视频码流数据的临时 buffer（disable video 时起作用）
nVideoStreamBufferSize	int	存放视频码流数据的临时 buffer 的大小（disable video 时起作用）
pSubtitleStreamBuffer	Void*	用于存放字幕码流数据的临时 buffer（disable subtitle 时起作用）
nSubtitleStreamBufferSize	int	存放字幕码流数据的临时 buffer 的大小（disable subtitle 时起作用）
bProcessingCommand	int	表示播控模块是否处于正在处理命令中（即状态的设置），1 为时，0 为否
nFirstVideoRenderPts	int64_t	第一帧视频显示帧的 pts
nFirstAudioRenderPts	int64_t	第一笔 pcm 音频数据的 pts
nTimeBase	int64_t	系统时钟的基准时间
nTimeOffset	int64_t	播放时间偏移量
timerMutex	pthread_mutex_t	互斥锁，Avtimer 模块支持多线程操作，互斥锁同步多个线程的操作，保护内部变量。
nLastTimeTimerAdjusted	int64_t	记录上一次调整系统时钟的基准时间

nPreVideoPts	int64_t	前一帧显示视频的 pts
nPreAudioPts	int64_t	前一帧播放的音频数据的 pts
nUnsyncVideoFrameCnt	int	没有作同步处理的视频帧的帧数
nLastInputPts	int64_t	Parser 端传到 player 模块最近一笔码流数据的 pts 值
nFirstPts	int64_t	从 parser 端传入的 Video 或 audio 的第一个 pts，取两者的最小值
bStreamEosSet	int	表示是否已经设置了码流结束标志
bVideoRenderEosReceived	int	表示视频接收模块是否已经接收到 eos
bAudioRenderEosReceived	int	表示音频接收模块是否已经接收到 eos
eosMutex	pthread_mutex_t	互斥锁，bVideoRenderEosReceived 和 bAudioRenderEosReceived 变量支持多线程操作，互斥锁同步多个线程的操作，保护这两个变量。
bInFastMode	int	表示播控模块是否处于快进模式
sVideoConfig	VConfig	保存配置视频解码器的配置信息的结构体变量
sVideoCropWindow[4]	int	存放视频显示区域的数组
audioDecoderMutex	pthread_mutex_t	pAudioDecComp 的同步锁
subtitleDecoderMutex	pthread_mutex_t	pSubtitleDecComp 的同步锁
bVideoCrash	int	视频解码器发生崩溃的标志位
bAudioCrash	int	音频解码器发生崩溃的标志位
bSubtitleCrash	int	字幕解码器发生崩溃的标志位
bUnSurpportVideoFlag	int	不支持视频播放标志位
pUnSurpportVideoBuffer	Void*	临时 buffer
nUnSurpportVideoBufferSize	int	临时 buffer 的大小

volume	float	音量值大小

4.2. AudioDecCompContext

名称	AudioDecCompContext	
功能描述	表示音频解码模块，包含音频解码模块的所有信息	
属性	类型	描述
mq	AwMessageQueue*	消息队列结构体指针
base	BaseCompCtx	基础组件结构体。
streamDataSem	sem_t	音频码流数据同步信号量，当音频解码器没有码流时，线程会一直阻塞到有码流数据为止
frameBufferSem	sem_t	Pcm buffer 的同步信号量，当音频解码器返回的是没有 pcm buffer 时，线程会一直阻塞到有 buffer 为止
nStartReply	int	线程响应 start 命令的返回值，返回值为 0 时说明响应成功，为-1 时说明响应失败
nStopReply	int	线程响应 stop 命令的返回值，返回值为 0 时说明响应成功，为-1 时说明响应失败
nPauseReply	int	线程响应 pause 命令的返回值，返回值为 0 时说明响应成功，为-1 时说明响应失败
nResetReply	int	线程响应 reset 命令的返回值，返回值为 0 时说明响应成功，为-1 时说明响应失败
sDecodeThread	pthread_t	此模块的线程，此线程会不断地调用 DecodeAudioStream 函数进行解码音频码流，并响应命令操作
eStatus	enum EPLAYERSTATUS	音频解码模块的状态值
pAvTimer	AvTimer*	系统时钟模块的操作指针
callback	PlayerCallback	播控模块的回调接口
pUserData	void*	播控模块传下来的自定义数据
bEosFlag	int	Eos 标志变量
nOffset	int	音频解码器码流 buffer 读位置的偏移量
bsInfo	BsInFor	存放音频解码器的基本信息的结构体
pDecoder	audio_sw_device_t*	指向音频解码器的指针

nStreamCount	int	音轨数目
nStreamSelected	int	当前播放的第几条音轨
pStreamInfoArr	AudioStreamInfo*	用于存放音频码流信息的结构体数组
pStreamManagerArr	StreamManager**	用于管理多音轨码流 buffer 的结构体数组
streamManagerMutex	pthread_mutex_t	互斥锁，StreamManager 结构体数组变量支持多线程操作，互斥锁同步多个线程的操作，保护成员变量。
decoderDestroyMutex	pthread_mutex_t	互斥锁
bCrashFlag	int	创建 audio decoder 失败的标志位

4.3. AudioRenderCompContext

名称	AudioRenderCompContext	
功能描述	表示音频接收模块，包含音频接收模块的所有信息	
属性	类型	描述
mq	AwMessageQueue*	消息队列结构体指针
base	BaseCompCtx	基础组件结构体。
sRenderThread	pthread_t	音频接收模块的主线程，线程不断地调用 AudioDecCompRequestPcmData 向解码器请求 pcm 数据，将请求的 pcm 数据传送到 audioSink 或 audioTrack 播放，并响应各个命令操作
eStatus	enum EPLAYERSTATUS	音频接收模块的状态值
pAudioSink	MediaPlayerBase::AudioSink*	音频播放设备操作指针
pSoundCtrl	SoundCtrl* (SoundCtrlContext*)	指针音频播放设备管理模块的指针
pDecComp	AudioDecComp* (AudioDecCompContext*)	指向音频解码模块的指针
pAvTimer	AvTimer* (AvTimerContext*)	指向系统时钟模块的指针
callback	PlayerCallback	播控模块的回调函数
pUserData	void*	播控模块传下来的自定义数据
bEosFlag	int	Eos 标志位
nConfigOutputBalance	int	音频声道设置参数，1 为左声道，2 为右声道，3 为立体声
bMute	int	屏蔽音频输出标志位
bForceWriteToDeviceFlag	int	屏蔽音频输出标志位

volume	float	音频大小值

4.4. VideoDecCompContext

名称	VideoDecCompContext	
功能描述	表示视频解码模块，包含视频解码模块的所有信息	
属性	类型	描述
mq	AwMessageQueue*	消息队列结构体指针
base	BaseCompCtx	基础组件结构体。
streamDataSem	sem_t	视频码流数据信息量，当视频解码器返回的是 no stream data 时，线程会一直阻塞到有数据为止
frameBufferSem	sem_t	视频帧 buffer 信号量，当视频解码器返回的是 no frame buffer 时，线程会一直阻塞到有 buffer 为止
nStartReply	int	线程响应 start 命令的返回值，返回值为 0 时说明响应成功，为-1 时说明响应失败
nStopReply	int	线程响应 stop 命令的返回值，返回值为 0 时说明响应成功，为-1 时说明响应失败
nPauseReply	int	线程响应 pause 命令的返回值，返回值为 0 时说明响应成功，为-1 时说明响应失败
nResetReply	int	线程响应 reset 命令的返回值，返回值为 0 时说明响应成功，为-1 时说明响应失败
sDecodeThread	pthread_t	视频解码模块的主线程，不断调用 DecodeVideoStream 函数解码视频码流
pDecoder	VideoDecoder*	视频解码器的操作指针
eStatus	enum EPLAYERSTATUS	视频解码模块的状态值
pAvTimer	AvTimer* (AvTimerContext*)	指向系统时钟模块的指针
callback	PlayerCallback	播控模块的回调函数
pUserData	void*	播控模块传下来的自定义数据
bEosFlag	int	Eos 标志位
bConfigDecodeKeyFrameOnly	int	只解关键帧标志位
bConfigDropDelayFrame	int	是否解 B 帧标志位

s		
bResolutionChange	int	分辨率发生变化的标志位
bCrashFlag	int	解码出错标志位
vconfig	VConfig	配置解码器的结构体
videoStreamInfo	VideoStreamInfo	存放视频流信息的结构体
nSelectStreamIndex	int	当前视频流的 index
pSecureBuf	char*	用于存放安全 buffer 的指针
nGpuAlignStride	int	Gpu 对齐方式
bUseNewDisplayFlag	int	是否走新显示框架的标志位

4.5. VideoRenderCompContext

名称	VideoRenderCompContext	
功能描述	表示视频接收模块，包含所有视频接收模块的信息	
属性	类型	描述
mq	AwMessageQueue*	消息队列结构体指针
base	BaseCompCtx	基础组件结构体。
sRenderThread	pthread_t	视频接收模块的主线程，线程不断调用 VideoDecCompRequestPicture 函数向解码器请求视频帧，将请求到的视频帧传送到显示层 layerControl 进行显示
eStatus	enum EPLAYERSTATUS	视频接收模块的状态值
pNativeWindow	void*	本地显示窗口的操作指针
pLayerCtrl	LayerCtrl*	指向显示控制接口的指针
pDecComp	VideoDecComp* (VideoDecCompContext*)	指向视频解码模块的指针
ePicture3DMode	enum EPICTURE3DMODE	3D 图像模式的设置值
eDisplay3DMode	enum EDISPLAY3DMODE	3D 显示模式的设置值
pAvTimer	AvTimer* (AvTimerContext*)	指向系统时钟模块的指针
callback	PlayerCallback	播控模块的回调函数
pUserData	void*	播控模块传下来的自定义数据
bEosFlag	int	Eos 标志位
nRotationAngle	int	旋转角度
nRequsetPictureNum	int	向解码器要的视频帧的数目
di	Deinterlace*	Deinterlace 处理模块的句柄
pDeinterlacePrePicture	VideoPicture*	用于做 deinterlace 的前一帧图像

nGpuYAlign	int	Gpu 关于 Y 分量的对齐方式
nGpuCAlign	int	Gpu 关于 C 分量的对齐方式
bSyncFirstPictureFlag	int	第一帧图像是否要做同步的标志位

4.6. AvTimerContext

名称	AvTimerContext	
功能描述	表示系统时钟模块，包含系统时钟模块的所有信息	
属性	类型	描述
sAvTimer	AvTimer	系统时钟模块的操作接口结构体
nSpeed	int	时钟速度
nStartTime	int64_t	系统时钟模块的基准时间
eStatus	int	时钟模块的状态值
startOsTime	struct timeval	当时钟模块启动时的系统时间
lastOsTime	struct timeval	当时钟模块被操作时的系统时间
mutex	pthread_mutex_t	互斥锁，时钟模块支持多线程操作，互斥锁同步多个线程的操作，保护内部变量。
nStartPts	int64_t	第一帧视频帧的 pts
nStartSystemTime	int64_t	显示第一帧视频时的系统时间

4.7. StreamManager

名称	StreamManager	
功能描述	表示一个音轨码流 buffer 管理模块，包含模块的所有信息	
属性	类型	描述
mutex	pthread_mutex_t	互斥锁，音轨码流 buffer 管理模块支持多线程操作，互斥锁同步多个线程的操作，保护内部变量。
nMaxBufferSize	int	
nValidDataSize	int	写入到码流 buffer 的有效数据的大小
nStreamID	int	音轨码流的 ID
streamFifo	StreamFifo	先进先出数据管理结构体变量
pMem	char*	用于临时存放 malloc 出来的 buffer
nMemSize	int	Malloc 的 buffer 大小

4.8. StreamFifo

名称	StreamFifo	
功能描述	先进先出数据管理结构体	
属性	类型	描述

pFrames	StreamFrame*	存放每一帧数据的信息的结构体数组
nMaxFrameNum	int	可存放的最大的帧数
nValidFrameNum	int	有效的帧数
nReadPos	int	读的位置(在 pFrames 数组中的位置)
nWritePos	int	写的位置(在 pFrames 数组中的位置)
nFlushPos	int	冲刷的位置（在 pFrames 数组中的位置)

5. Declaration

The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This datasheet neither states nor implies warranty of any kind, including fitness for any particular application.