全志科技
Allwinner Technology

视频解码库 API 文档

文档履历

版本号	日期	制/修订人	内容描述
V1.0	2016-11-01	刘小燕	初稿
V1.1	2017-08-20	王喜望	更新文档

目 录

1. 概述

1.1. 编写目的

设计视频解码库的对外 API 接口及相关的数据结构。指导基于视频解码库的开发和使用。

1.2. 适用范围

适用于公司带有 VE 模块的各个芯片平台的 Android 系统 SDK 和 Linux SDK。

1.3. 相关人员

基于视频解码库开发和使用的相关人员。

2. 模块介绍

2.1. 功能介绍

视频解码库是一个提供视频解码功能的库，编译输出的库文件为 libvdecoder.so。基于视频解码库，应用程序可以在全志公司的各个 IC 平台上实现高效的、多种视频格式的视频解码功能，所支持的视频格式为：H265、H264、MJPEG、MPEG2、MPEG1、MSMPEG4V1、MSMPEG4V2、DIVX3、DIVX4、DIVX5、XVID、H263、SORENSSON_H263、WMV1、WMV2、WMV3、VP6、VP8、VP9。

3. 接口说明

3.1. 接口函数

视频解码库 APIs		
NO.1	AddVDPlugin	加载所有的子解码库
NO.2	CreateVideoDecoder	创建一个视频解码器
NO.3	DestroyVideoDecoder	销毁视频解码器
NO.4	InitializeVideoDecoder	初始化视频解码器
NO.5	ResetVideoDecoder	重置视频解码器
NO.6	ReopenVideoEngine	在遇到分辨率发生改变的情况时，解码器返回 VDECODE_RESULT_RESOLUTION_CHANGE 通知中间件，对分辨率变化后的码流调用解码之前，需要先调用 ReopenVideoEngine 重新启动解码器
NO.7	DecodeVideoStream	视频码流解码函数
NO.8	RequestVideoStreamBuffer	从视频解码器获取传输视频码流的 buffer
NO.9	SubmitVideoStreamData	将视频码流数据传输到解码器中
NO.10	NextPictureInfo	获取下一个视频解码图像的信息
NO.11	RequestPicture	获取视频解码图像
NO.12	ReturnPicture	还回视频解码图像
NO.13	RotatePicture	对解码后的视频图像调用软件接口进行旋转
NO.14	RotatePictureHw	对解码后的视频图像调用硬件接口进行旋转
NO.15	GetVideoStreamInfo	获取 VideoStreamInfo 的信息
NO.16	VideoStreamBufferSize	获取视频解码器开辟的视频 buffer size
NO.17	VideoStreamDataSize	获取视频解码器中有效的数据 size
NO.18	VideoStreamFrameNum	获取视频解码器中还未解码的数据笔数
NO.19	VideoStreamDataInfoPointer	获取下一笔要解码的视频数据信息所在的地址
NO.20	TotalPictureBufferNum	获取解码器开辟的图像 buffer 个数
NO.21	EmptyPictureBufferNum	获取解码器空闲的图像 buffer 个数
NO.22	ValidPictureNum	获取解码器已经解完，但还未显示的图像个数
NO.23	AllocatePictureBuffer	申请物理地址连续的视频帧 buffer
NO.24	FreePictureBuffer	释放调用 AllocatePictureBuffer 申请的视频帧 buffer
NO.25	VideoDecoderPallocIonBuf	申请 ion buffer
NO.26	VideoDecoderFreeIonBuf	释放 ion buffer
NO.27	GetVideoFbmBufInfo	获取解码器申请的 fbm buffer 的信息
NO.28	SetVideoFbmBufAddress	将中间件申请的视频图像 buffer 地址设置到解码器
NO.29	SetVideoFbmBufRelease	在新显架构下，nativeWindow 发生变化时，在旧的 nativewindow 中申请的视频帧 buffer 将逐渐被在新 nativewindow 中申请的视频帧 buffer 所替代，通过调用此函数，解码器将原来的 buffer 全部设置为 release 状态
NO.30	RequestReleasePicture	中间件从解码器调用标记为 release 状态的视频帧

		图像
NO.31	ReturnRelasePicture	还回标记为 release 状态的图像，此图像对应的地址 buffer 已经被重新分配
NO.32	ConfigExtraScaleInfo	若中间件在调用解码器 InitializeVideoDecoder 函数时，无法确定 VE 模块缩放的参数时， 可以在调用完 InitializeVideoDecoder 后，在调用 DecodeVideoStream 函数前， 通过调用 ConfigExtraScaleInfo 设置 VE 的相关缩放参数
NO.33	DecoderSetSpecialData	设置特殊参数
NO.34	SetDecodePerformCmd	设置解码器开始或停止统计丢帧信息的命令
NO.35	GetDecodePerformInfo	从解码器中获取丢帧相关的信息

3.1.1. AddVDPlugin

函数原型	void AddVDPlugin(void);
功能	加载所有的子解码库，如 libawh264.so，libawh264.so 等。
参数	
返回值	
调用说明	上层调用者可自行设计按需加载子解码库的函数，如可设计只加载 libawh264.so 这个子解码库的函数；若上层没有按需加载子解码库的需求，则可以调用此函数。

3.1.2. CreateVideoDecoder

函数原型	VideoDecoder* CreateVideoDecoder(void)
功能	创建一个视频解码器
参数	
返回值	成功：视频解码器指针； 失败：返回 NULL；
调用说明	

3.1.3. DestroyVideoDecoder

函数原型	void DestroyVideoDecoder(VideoDecoder* pDecoder)
功能	销毁视频解码器
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针
返回值	无
调用说明	无

3.1.4. InitializeVideoDecoder

函数原型	int InitializeVideoDecoder(VideoDecoder* pDecoder, VideoStreamInfo* pVideoInfo, VConfig* pVconfig)
功能	初始化视频解码器
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针 pVconfig：解码器初始化信息 pVideoInfo：视频片源的相关信息
返回值	成功：返回 0； 失败：返回-1，
调用说明	pConfig：编码器基本初始化信息； 1. nInputWidth：输入图像帧的宽度，以像素为单位； 2. nInputHeight：输入图像帧的高度，以像素为单位； 3. nDstWidth：编码前对输入图像做 scale 后的宽度，以像素为单位； 如果不需要做 scale，nDstWidth 的值保持和 nInputWidth 一致； 4. nDstHeight：编码前对输入图像做 scale 后的高度，以像素为单位； 如果不需要做 scale，nDstHeight 的值保持和 nInputHeight 一致； 5. eInputFormat：输入的颜色格式； 6. nStride：输入图像帧在内存中的行宽，以像素为单位，编码器要求 nStride 必须 16 对齐； 7. Memops：编码器内部内存管理的数据结构，该数据结构由调用者初始化，其定义在 memory 模块中，具体请参看 memory 相关文档；

3.1.5. ResetVideoDecoder

函数原型	void ResetVideoDecoder(VideoDecoder* pDecoder)
功能	重置视频编码器
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针
返回值	成功：返回 0； 失败：返回-1；
调用说明	无

3.1.6. ReopenVideoEngine

函数原型	int ReopenVideoEngine (VideoDecoder* pDecoder)
功能	重新打开解码器内 Video Engine 模块
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针；
返回值	0： 表示成功；-1：失败；
调用说明	如果视频分辨率发生改变，DecodeVideoStream 函数会返回对应的值 VDECODE_RESULT_RESOLUTION_CHANGE，并将码流归还到码流 Buffer；

	此时应用应该调用本函数重新打开 Video Engine 模块； 重新打开 Video Engine 模块会导致图像 Buffer 被释放，图像 Buffer 管理模块重新初始化。这一点显示控制需要注意。

3.1.7. DecodeVideoStream

函数原型	int DecodeVideoStream(VideoDecoder* pDecoder, int bEndOfStream, int bDecodeKeyFrameOnly, int bDropBFrameIfDelay, int64_t nCurrentTimeUs)
功能	视频码流解码函数
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； bEndOfStream：最后一笔码流数据， 取值为 0/1 bDropBFrameIfDelay：在解码过时的情况下是否允许丢 B 帧 nCurrentTimeUs：当前系统时间，单位 us
返回值	(1) VDECODE_RESULT_UNSUPPORTED = -1, 视频格式或规格不支持 (2) VDECODE_RESULT_OK = 0, 解码器中间状态 (3)VDECODE_RESULT_FRAME_DECODED = 1, 解码器解完一个 P 帧或 B 帧 (4)VDECODE_RESULT_CONTINUE = 2, 解码一帧未完成 (5)VDECODE_RESULT_KEYFRAME_DECODED = 3, 解完一个关键帧（I 帧） (6)VDECODE_RESULT_NO_FRAME_BUFFER = 4, 解码器申请不到 frame buffer (7)VDECODE_RESULT_NO_BITSTREAM = 5, 解码器中没有可以解码的码流 (8)VDECODE_RESULT_RESOLUTION_CHANGE = 6, 码流的分辨率发生改变
调用说明	

3.1.8. RequestVideoStreamBuffer

函数原型	int RequestVideoStreamBuffer(VideoDecoder* pDecoder, int nRequireSize, char** ppBuf, int* pBufSize,

	char** ppRingBuf, int* pRingBufSize, int nStreamBufIndex)
功能	向解码器申请传输视频码流的 buffer
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针 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； nStreamBufIndex：对于蓝光 MVC 等 3D 视频，解码器需要处理两路码流，nStreamBufIndex 指定从第几路视频码流 Buffer 获取 Buffer，0 表示第 0 路（MVC 主码流）、1 表示第 1 路（MVC 从码流）。
返回值	成功：返回 0； 失败：返回-1；
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。

3.1.9. SubmitVideoStreamData

函数原型	int SubmitVideoStreamData(VideoDecoder* pDecoder, VideoStreamDataInfo* pDataInfo, int nStreamBufIndex)
功能	向解码器提交码流数据
参数	pDecoder： 通过 CreateVideoDecoder 函数创建的视频解码器指针； pDataInfo：码流数据信息，包括地址、长度、时间戳、边界信息等；
返回值	成功：返回 0； 失败：返回-1；
调用说明	提交数据时，数据可以是一笔包含整数个数据单元的完整码流帧，也可以只包含一个数据单元的部分数据，只需将 VideoStreamDataInfo 结构体中的两个表示数据边界信息的变量正确填写即可。两个边界信息变量为 bIsFirstPart：表示该笔数据第一个字节是否是一个数据单元的开始； bIsLastPart：表示该笔数据最后一个有效字节是否是一个数据单元的结束；

3.1.10. NextPictureInfo

函数原型	VideoPicture* NextPictureInfo(VideoDecoder* pDecoder, int nStreamIndex)
功能	获取下一帧输出图像的信息
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； nStreamIndex：对于蓝光 MVC 等 3D 视频，解码器有两路视频可供显示，nStreamIndex 指定从获取第几路视频的图像。
返回值	成功：返回下一帧待显示图像 Buffer 的指针； 失败：返回 NULL；
调用说明	与 RequestPicture 函数相比，本函数只是获取视频图像的信息，不会使该图像从解码器的显示队列中删除。

3.1.11. RequestPicture

函数原型	VideoPicture* RequestPicture(VideoDecoder* pDecoder, int nStreamIndex)
功能	获取一帧图像
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； nStreamIndex：对于蓝光 MVC 等 3D 视频，解码器有两路视频可供显示，nStreamIndex 指定从获取第几路视频的图像。
返回值	成功：返回图像 Buffer 指针； 失败：返回 NULL；
调用说明	

3.1.12. ReturnPicture

函数原型	int ReturnPicture(VideoDecoder* pDecoder, VideoPicture* pPicture)
功能	归还通过 RequestPicture 获取的视频图像给解码器
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； pPicture：通过 RequestPicture 函数获取的图像 Buffer；
返回值	0：表示成功；-1：失败；
调用说明	

3.1.13. RotatePicture

函数原型	int RotatePicture (VideoDecoder* pDecoder, VideoPicture* pPictureIn, VideoPicture* pPictureOut, int nRotateDegree)
功能	把图像 pPictureIn 旋转输出到图像 Buffer pPictureOut
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针；

	pPictureIn：输入图像； pPictureOut：输出图像； nRotateDegree：顺时针旋转角度，0 表示不旋转、1 表示旋转 90 度、2 表示 180 度、3 表示 270 度；
返回值	0：表示成功； -1：表示失败；
调用说明	除旋转图像外，本函数还支持像素格式的转换，输出像素格式由 pPictureOut 图像的 ePixelFormat 参数指定； 目前已经支持的像素格式转换有： (1) PIXEL_FORMAT_YUV_MB32_420 转为 PIXEL_FORMAT_YV12 (2) PIXEL_FORMAT_YUV_MB32_422 转为 PIXEL_FORMAT_YV12 (3) PIXEL_FORMAT_YUV_MB32_420 转为 PIXEL_FORMAT_NV21 (4) PIXEL_FORMAT_YUV_MB32_422 转为 PIXEL_FORMAT_NV21 (5) PIXEL_FORMAT_YV12 转为 PIXEL_FORMAT_NV21 (6) PIXEL_FORMAT_NV21 转为 PIXEL_FORMAT_YV12

3.1.14. RotatePictureHw

函数原型	int RotatePictureHw(VideoDecoder* pDecoder, VideoPicture* pPictureIn, VideoPicture* pPictureOut, int nRotateDegree)
功能	采用硬件旋转模， 把图像 pPictureIn 旋转输出到图像 Buffer pPictureOut
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； pPictureIn：输入图像； pPictureOut：输出图像； nRotateDegree：顺时针旋转角度，0 表示不旋转、1 表示旋转 90 度、2 表示 180 度、3 表示 270 度；
返回值	0：表示成功； -1：表示失败；
调用说明	

3.1.15. GetVideoStreamInfo

函数原型	int GetVideoStreamInfo(VideoDecoder* pDecoder, VideoStreamInfo* pVideoInfo);
功能	获取 VideoStreamInfo 结构体的信息
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； pVideoInfo：VideoStreamInfo 结构体指针
返回值	成功：0 失败：-1

3.1.16. VideoStreamBufferSize

函数原型	int VideoStreamBufferSize(VideoDecoder* pDecoder, int nStreamBufIndex)
功能	查询码流 Buffer 的总大小，单位为字节
参数	pDecoder: 通过 CreateVideoDecoder 函数创建的视频解码器指针； nStreamBufIndex: 对于蓝光 MVC 等 3D 视频，解码器需要处理两路码流，nStreamBufIndex 指定从第几路视频码流 Buffer 获取 Buffer，0 表示第 0 路（MVC 主码流）、1 表示第 1 路（MVC 从码流）。
返回值	成功：返回 0； 失败：返回-1；
调用说明	在解码器初始化后才能正确返回码流 Buffer 的大小，否则返回 0.

3.1.17. VideoStreamDataSize

函数原型	int VideoStreamDataSize(VideoDecoder* pDecoder, int nStreamBufIndex)
功能	查询码流 Buffer 中有效数据（未解码的数据）大小，单位为字节
参数	pDecoder: 通过 CreateVideoDecoder 函数创建的视频解码器指针； nStreamBufIndex: 对于蓝光 MVC 等 3D 视频，解码器需要处理两路码流，nStreamBufIndex 指定从第几路视频码流 Buffer 获取 Buffer。
返回值	码流 Buffer 中未解码的数据量，单位为字节。
调用说明	
函数原型	int ReleaseAllocInputBuffer(VideoEncoder* pEncoder)

3.1.18. VideoStreamFrameNum

函数原型	int VideoStreamFrameNum (VideoDecoder* pDecoder, int nStreamBufIndex)
功能	查询码流 Buffer 中有效数据（未解码的数据）有多少帧；
参数	pDecoder: 通过 CreateVideoDecoder 函数创建的视频解码器指针； nStreamBufIndex: 对于蓝光 MVC 等 3D 视频，解码器需要处理两路码流，nStreamBufIndex 指定从第几路视频码流 Buffer 获取 Buffer，0 表示第 0 路（MVC 主码流）、1 表示第 1 路（MVC 从码流）。
返回值	码流 Buffer 中未解码的数据有多少帧。
调用说明	

3.1.19. VideoStreamDataInfoPointer

函数原型	void* VideoStreamDataInfoPointer(VideoDecoder* pDecoder, int nStreamBufIndex)
功能	获取码流 Buffer 中将要解码的数据所携带的 videoinfo 信息的指针；

参数	pDecoder: 通过 CreateVideoDecoder 函数创建的视频解码器指针； nStreamBufIndex: 码流所在 buffer 的 index 标签
返回值	
调用说明	

3.1.20. TotalPictureBufferNum

函数原型	int TotalPictureBufferNum(VideoDecoder* pDecoder, int nStreamIndex)
功能	询问解码器内总共有多少个图像 Buffer
参数	pDecoder: 通过 CreateVideoDecoder 函数创建的视频解码器指针； nStreamIndex: 对于蓝光 MVC 等 3D 视频，解码器有两路视频可供显示，nStreamIndex 指定从获取第几路视频的图像。
返回值	图像 Buffer 个数
调用说明	某些视频格式(H264 和 MPEG2)需要解码部分码流(SPS/PPS、Sequence Header)信息后才申请图像 Buffer，在此之前，本函数返回 0。

3.1.21. EmptyPictureBufferNum

函数原型	int EmptyPictureBufferNum(VideoDecoder* pDecoder, int nStreamIndex)
功能	询问解码器内有多少个空闲的图像 Buffer，即未被解码器和显示占用的图像 Buffer 个数。 nStreamIndex: 对于蓝光 MVC 等 3D 视频，解码器有两路视频可供显示，nStreamIndex 指定从获取第几路视频的图像。
参数	pDecoder: 通过 CreateVideoDecoder 函数创建的视频解码器指针；
返回值	空闲图像 Buffer 个数
调用说明	

3.1.22. ValidPictureNum

函数原型	int ValidPictureNum(VideoDecoder* pDecoder, int nStreamIndex)
功能	询问解码器内显示队列中有多少个待显示的图像。
参数	pDecoder: 通过 CreateVideoDecoder 函数创建的视频解码器指针； nStreamIndex: 对于蓝光 MVC 等 3D 视频，解码器有两路视频可供显示，nStreamIndex 指定从获取第几路视频的图像。
返回值	待显示的图像个数
调用说明	

3.1.23. AllocatePictureBuffer

函数原型	VideoPicture* AllocatePictureBuffer (struct ScMemOpsS* memOps, int nWidth, int nHeight,int nLineStride, int ePixelFormat)
功能	申请一个图像 Buffer。
参数	memOps: memory 管理器接口 nWidth: 图像像素宽度; nHeight: 图像像素高度; nLineStride: 图像在内存中存放的行宽，以像素为单位; ePixelFormat: 图像像素格式;
返回值	0: 表示成功; -1: 失败;
调用说明	本函数独立于 VideoDecoder 模块，是全局函数。

3.1.24. FreePictureBuffer

函数原型	int FreePictureBuffer (struct ScMemOpsS* memOps, VideoPicture* pPicture)
功能	释放一个由 AllocatePictureBuffer 函数申请的图像 Buffer。
参数	memOps: memory 管理器接口 pPicture: 通过 AllocatePictureBuffer 函数申请的图像 Buffer;
返回值	成功: 返回图像 Buffer 指针; 失败: 返回 NULL
调用说明	本函数独立于 VideoDecoder 模块，是全局函数。

3.1.25. VideoDecoderPallocIonBuf

函数原型	void *VideoDecoderPallocIonBuf(VideoDecoder* pDecoder, int nSize);
功能	获取 ion buffer（物理连续的 buffer）
参数	pDecoder: 通过 CreateVideoDecoder 函数创建的视频解码器指针; nSize：buffer 大小
返回值	成功： buffer 地址 失败：NULL
调用说明	

3.1.26. VideoDecoderFreeIonBuf

函数原型	void VideoDecoderFreeIonBuf(VideoDecoder* pDecoder, void *pIonBuf);
功能	释放 ion buffer（物理连续的 buffer）
参数	pDecoder: 通过 CreateVideoDecoder 函数创建的视频解码器指针; pIonBuf： buffer 地址

返回值	
调用说明	

3.1.27. GetVideoFbmBufInfo

函数原型	FbmBufInfo* GetVideoFbmBufInfo(VideoDecoder* pDecoder)
功能	获取解码器开辟的 fbm buffer 的结构体信息
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针；
返回值	成功：返回 FbmBufInfo 结构体 失败：返回 NULL
调用说明	

3.1.28. SetVideoFbmBufAddress

函数原型	VideoPicture* SetVideoFbmBufAddress(VideoDecoder* pDecoder, \ VideoPicture* pVideoPicture, int bForbidUseFlag)
功能	在新显架构下，将显示模块开辟的视频帧 buffer 的地址信息传到解码器
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； pVideoPicture：显示模块开辟的视频帧 buffer bForbidUseFlag：此 buffer 此时刻是否允许使用
返回值	成功：返回 VideoPicture 结构体 失败：返回 NULL
调用说明	此函数将新显模式下显示模块开辟的视频帧 buffer 地址传到解码器，然后从解码器获取相应视频帧 buffer 的全部信息

3.1.29. SetVideoFbmBufRelease

函数原型	VideoPicture* SetVideoFbmBufRelease(VideoDecoder* pDecoder)
功能	在新显架构下，在遇到 nativeWindow 发生改变时，需要通知解码器将设置到解码器的显示 buffer 设置成 release 状态
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针；
返回值	成功：返回 VideoPicture 结构体 失败：返回 NULL
调用说明	解码器会将设成 release 状态且空闲的 buffer 添加到 release buffer 队列，供 RequestReleasePicture 函数调用

3.1.30. RequestReleasePicture

函数原型	VideoPicture* RequestReleasePicture (VideoDecoder* pDecoder)
功能	在新显架构下，获取解码器设置为 release 状态的视频帧 buffer
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针；
返回值	成功：返回 VideoPicture 结构体 失败：返回 NULL
调用说明	

3.1.31. ReturnRelasePicture

函数原型	VideoPicture* ReturnReleasePicture (VideoDecoder* pDecoder, VideoPicture* pVpicture, int bForbidUseFlag)
功能	在新显架构下，将获取到的标记为 release 的 picture 中的 buffer 相关信息修改后重新将 picture 传到解码器
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； pVpicture：包含新 buffer 信息的 pVpicture 指针 bForbidUseFlag：pVpicture 包含的 buffer 是否可用的标记
返回值	成功：返回 VideoPicture 结构体 失败：返回 NULL
调用说明	中间件获取到 release 状态的 buffer 后，会将原先申请的 buffer 内存空间释放，在新的 nativeWIndow 下重新申请 buffer， 然后将包含新 buffer 的内存信息的 pVpicture 设置到解码器中

3.1.32. ConfigExtraScaleInfo

函数原型	int ConfigExtraScaleInfo(VideoDecoder* pDecoder, int nWidthTh, int nHeightTh, int nHorizonScaleRatio, int nVerticalScaleRatio)
功能	设置 scaleDown 相关信息到解码器
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； nWidthTh：要进行缩放的图像的阈值（对>=nWidthTh 的图像进行缩放）； nHeightTh：要进行缩放的图像的阈值（对>= nHeightTh 的图像进行缩放）； nHorizonScaleRatio：水平缩放比例 nVerticalScaleRatio：竖直缩放比例
返回值	成功：返回 0； 失败：返回-1；
调用说明	通常设置解码器的 scaledown 信息是通过调用 InitializeVideoDecoder 函数进行设置， 但当在调用 InitializeVideoDecoder 阶段无法确定 scaleDown 参数时，可以通过调用 ConfigExtraScaleInfo 函数进行设置， 但要注意此函数一定在 DecodeVideoStream 函数调用之前调用

3.1.33. DecoderSetSpecialData

函数原型	int DecoderSetSpecialData(VideoDecoder* pDecoder, void *pArg);
功能	设置特殊的参数，目前用于设置 jpeg 解码器里关于 skipConfig 的参数
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； pArg：参数结构体的指针
返回值	成功：0 失败：-1
调用说明	

3.1.34. SetDecodePerformCmd

函数原型	Int SetDecodePerformCmd(VideoDecoder* pDecoder, enum EVDECODERSETPERFORMCMD performCmd)
功能	在视频播放过程中，开启或关闭统计丢帧信息的功能
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； performCmd： 默认取值 VDECODE_SETCMD_DEFAULT 取值为：VDECODE_SETCMD_START_CALDROPFRAME 表示开始统计丢帧信息 取值为：VDECODE_SETCMD_STOP_CALDROPFRAME 表示结束统计丢帧信息
返回值	成功：返回 0 失败：返回 -1
调用说明	

3.1.35. GetDecodePerformInfo

函数原型	int GetDecodePerformInfo(VideoDecoder* pDecoder, enum EVDECODERGETPERFORMCMD performCmd, VDecodePerformaceInfo** performInfo)
功能	在视频播放过程中，获取统计的丢帧信息
参数	pDecoder：通过 CreateVideoDecoder 函数创建的视频解码器指针； performCmd： 默认取值 VDECODE_SETCMD_DEFAULT 取值为：VDECODE_GETCMD_DROPFRAME_INFO 表示获取统计丢帧信息
返回值	成功：返回 0 失败：返回 -1
调用说明	

4. 数据结构说明

4.1. VideoStreamInfo

名称	VideoStreamInfo	
功能描述	初始化解码器时码流相关的基本信息	
属性	类型	描述

eCodecFormat	int	视频源编码格式，取值如下
		enum EVIDEOCODECFORMAT
		{
		VIDEO_CODEC_FORMAT_UNKNOWN
		VIDEO_CODEC_FORMAT_MJPEG
		VIDEO_CODEC_FORMAT_MPEG1
		VIDEO_CODEC_FORMAT_MPEG2
		VIDEO_CODEC_FORMAT_MPEG4
		VIDEO_CODEC_FORMAT_MSMPEG4V1
		VIDEO_CODEC_FORMAT_MSMPEG4V2
		VIDEO_CODEC_FORMAT_DIVX3
		VIDEO_CODEC_FORMAT_DIVX4
		VIDEO_CODEC_FORMAT_DIVX5
		VIDEO_CODEC_FORMAT_XVID
		VIDEO_CODEC_FORMAT_H263
		VIDEO_CODEC_FORMAT_SORENSSON_H263
		VIDEO_CODEC_FORMAT_RXG2
		VIDEO_CODEC_FORMAT_WMV1
		VIDEO_CODEC_FORMAT_WMV2
		VIDEO_CODEC_FORMAT_WMV3
		VIDEO_CODEC_FORMAT_VP6
		VIDEO_CODEC_FORMAT_VP8
		VIDEO_CODEC_FORMAT_VP9
		VIDEO_CODEC_FORMAT_RX
		VIDEO_CODEC_FORMAT_H264
		VIDEO_CODEC_FORMAT_H265
		VIDEO_CODEC_FORMAT_AVS
		};

nWidth	int	视频源宽度
nHeight	int	视频源高度
nFrameRate	int	视频源帧率
nFrameDuration	int	视频帧持续时间
nAspectRatio	int	视频源像素宽高比
bIs3DStream	int	视频源是否为 3D 双流片源
nCodecSpecificDataLen	int	视频源对应的 specialData 的长度
pCodecSpecificData	Char*	视频源对应的 specialData 的 buffer

bSecureStreamFlag	int	视频源是否为加密视频
bSecureStreamFlagLevel1	int	视频源加密等级
bIsFramePackage	int	视频源是否为帧封装格式
h265ReferencePictureNum	int	H265 视频参考帧个数
bReOpenEngine	int	视频数据是否为分辨率切换后的视频
bIsFrameCtsTestFlag	int	视频源是否为 cts 测试片源

4.2. VConfig

名称	VConfig	
功能描述	初始化解码器时的基本配置信息	
属性	类型	描述
bScaleDownEn	int	解码器缩放功能使能标记
bRotationEn	int	解码器旋转功能使能标记
bSecOutputEn	int	MJPEG 解码器第二路输出标记
nHorizonScaleDownRatio	int	视频帧水平方向缩放比例，取值 0,1,2,3
nVerticalScaleDownRatio	int	视频帧竖直方向多方比例，取值 0,1,2,3
nSecHorizonScaleDownRatio	int	第二路输出的视频帧水平方向缩放比例，取值 0,1,2,3
nSecVerticalScaleDownRatio	int	第二路输出的视频帧竖直方向缩放比例，取值 0,1,2,3
nRotateDegree	int	输出视频帧旋转角度，取值 0,1,2,3
bThumbnailMode	int	缩略图模式标记
eOutputPixelFormat	int	输出视频帧 yuv 排列方式，取值如下 PIXEL_FORMAT_DEFAULT PIXEL_FORMAT_YUV_PLANER_420 PIXEL_FORMAT_YUV_PLANER_422 PIXEL_FORMAT_YUV_PLANER_444 PIXEL_FORMAT_YV12 PIXEL_FORMAT_NV21 PIXEL_FORMAT_NV12 PIXEL_FORMAT_YUV_MB32_420 PIXEL_FORMAT_YUV_MB32_422 PIXEL_FORMAT_YUV_MB32_444 PIXEL_FORMAT_RGBA PIXEL_FORMAT_ARGB PIXEL_FORMAT_ABGR PIXEL_FORMAT_BGRA PIXEL_FORMAT_YUYV PIXEL_FORMAT_YVYU PIXEL_FORMAT_UYVY PIXEL_FORMAT_VYUY PIXEL_FORMAT_PLANARUV_422

		PIXEL_FORMAT_PLANARVU_422 PIXEL_FORMAT_PLANARUV_444 PIXEL_FORMAT_PLANARVU_444
eSecOutputPixelFormat	int	第二路输出视频帧 yuv 排列方式，取值同 eOutputPixelFormat
bNoBFrames	int	视频源没有 B 帧标记
bDisable3D	int	解码器不支持 3D 双流标记
bDispErrorFrame	int	不显示错误帧标记
nVbvBufferSize	int	设置 vbv buffer 的取值
nFrameBufferNum	int	设置 frame buffer 个数
bSecureosEn	int	加密视频使能标记
bGpuBufValid	int	Frame buffer 由 GPU 统一申请的标记
nAlignStride	int	Frame buffer 的对齐值
bIsSoftDecoderFlag	int	是否选择软解的标记
bVirMallocSbm	int	Vbv buffer 是否采用开辟虚拟内存的标记
bSupportPallocBufBeforeDecode	int	支持在解码前开辟 frame buffer 的标记
nDeInterlaceHoldingFrameBufferNum	int	离线 interlace 模块占用 frame buffer 个数
nDisplayHoldingFrameBufferNum	int	显示模块占用 frame buffer 个数
nRotateHoldingFrameBufferNum	int	离线旋转模块专用 frame buffer 个数
nDecodeSmoothFrameBufferNum	int	解码器保持平滑度占用 frame buffer 个数
bIsTvStream	int	视频片源是电视信号标记
memops	struct ScMemOpsS *	memory 管理器接口
eCtlAfbcMode	eControlAfbcMode	设置 afbc 功能的模式，取值如下： DISABLE_AFBC_ALL_SIZE ENABLE_AFBC_JUST_BIG_SIZE ENABLE_AFBC_ALL_SIZE
eCtlIptvMode	eControlIptvMode	设置“iptv 场景下获取特定信息”功能的模式，取值如下： DISABLE_IPTV_ALL_SIZE ENABLE_IPTV_JUST_SMALL_SIZE ENABLE_IPTV_ALL_SIZE
veOpsS	VeOpsS*	Ve 模块的操作句柄
pVeOpsSelf	void*	Ve 模块的内部结构体数据
bConvertVp910bitTo8bit	int	在 vp9 10bit 的码流场景下，是否将 10bit 的图像数据转换成 8bi 的标志位
nVeFreq	unsigned int	设置 Ve 频率值
bCalledByOmxFlag	int	上层调用者是否为 omx 的标志位
bSetProcInfoEnable	int	是否设置 proc 信息的标志位
nSetProcInfoFreq	int	设置 proc 信息的频率，如每隔 1s 设置

		一次
nChannelNum	int	设置 proc 信息的通道数

4.3. VideoStreamDataInfo

名称	VideoStreamDataInfo	
功能描述	中间件传到解码器的每笔数据结构信息	
属性	类型	描述
pData	Char*	传输的一笔视频数据所用的 buffer 指针
nLength	int	传输的一笔视频数据的长度
nPts	int64_t	传输的一笔视频数据所携带的时间戳
nPcr	int64_t	传输的一笔视频数据所携带的 PCR
bIsFirstPart	int	传输的一笔视频数据的第一部分的标记
bIsLastPart	int	传输的一笔视频数据的最后一部分的标记
nID	int	传输的视频数据所携带的 ID 号（目前没用）
nStreamIndex	int	传输的视频数据属于第几路视频的标记，（主流为 0，从流为 1）
bValid	int	视频数据有效的标记
bVideoInfoFlag	Unsigned int	传输的视频帧数据是否携带额外信息的标记
pVideoInfo	int	传输的视频帧数据携带额外信息所在的 buffer 指针

4.4. VideoPicture

名称	VideoPicture	
功能描述	解码器解码输出的视频帧的结构体信息	
属性	类型	描述
nID	int	视频帧所用 buffer 的 ID 号
nStreamIndex	int	视频帧对应码流的标记号，（主流为 0，从流为 1）
ePixelFormat	int	输出视频帧 yuv 排列方式，取值如下 PIXEL_FORMAT_DEFAULT PIXEL_FORMAT_YUV_PLANER_420 PIXEL_FORMAT_YUV_PLANER_422 PIXEL_FORMAT_YUV_PLANER_444 PIXEL_FORMAT_YV12 PIXEL_FORMAT_NV21 PIXEL_FORMAT_NV12 PIXEL_FORMAT_YUV_MB32_420 PIXEL_FORMAT_YUV_MB32_422 PIXEL_FORMAT_YUV_MB32_444

		PIXEL_FORMAT_RGBA PIXEL_FORMAT_ARGB PIXEL_FORMAT_ABGR PIXEL_FORMAT_BGRA PIXEL_FORMAT_YUYV PIXEL_FORMAT_YVYU PIXEL_FORMAT_UYVY PIXEL_FORMAT_VYUY PIXEL_FORMAT_PLANARUV_422 PIXEL_FORMAT_PLANARVU_422 PIXEL_FORMAT_PLANARUV_444 PIXEL_FORMAT_PLANARVU_444
nWidth	int	视频帧的宽度（做对齐后的宽度）
nHeight	int	视频帧的高度（做对齐后的高度）
nLineStride	int	视频帧宽度对齐要求值
nTopOffset	int	视频帧有效显示区域顶端开始值
nLeftOffset	int	视频帧有效显示区域左侧开始值
nBottomOffset	int	视频帧有效显示区域底端结束值
nRightOffset	int	视频帧有效显示区域右侧结束值
nFrameRate	int	视频帧率
nAspectRatio	int	视频帧宽高像素比
bIsProgressive	int	视频帧 interlace 格式标记
bTopFieldFirst	int	视频帧顶场优先标记
bRepeatTopField	int	视频帧顶场重复显示标记
nPts	int64_t	视频帧所携带的显示时间戳
nPcr	int64_t	视频帧多携带的 pcr
pData0	Char*	视频帧 y buffer 的地址指针
pData1	Char*	视频帧 u buffer 的地址指针
pData2	Char*	视频帧 v buffer 的地址指针
pData3	Char*	视频帧其它 buffer 的地址指针
bMafValid	int	Maf 有效标记
pMafData	Char*	Maf 信息所在的 buffer 地址
bPreFrmValid	int	视频帧前面帧有效标记
nBufId	int	视频帧 Buffer 所携带的 buf id
phyYBufAddr	size_addr	视频帧亮度 buffer 对应的物理地址
phyCBufAddr	size_addr	视频帧色度 buffer 对应的物理地址
pPrivate	void*	视频帧私有信息 buffer 指针
nBufStatus	int	视频帧 buffer 的状态
bTopFieldError	int	视频帧顶场有错的标记
bBottomFieldError	int	视频帧底场有错的标记
nColorPrimary	int	解码器给出的 ColorPrimary 值
bFrameErrorFlag	int	图像是否包含错误数据的标志位
pMetaData	Void*	用于存放 afbc info 和 hdr info 的

		buffer
video_full_range_flag	VIDEO_FULL_RANGE_FLAG	与 hdr 相关的参数
transfer_characteristics	VIDEO_TRANSFER	与 hdr 相关的参数
matrix_coeffs	VIDEO_MATRIX_COEFFS	与 hdr 相关的参数
colour_primaries	u8	与 hdr 相关的参数
nLower2BitBufSize	int	低 2bit 的 buffer 大小
nLower2BitBufOffset	int	低 2bit 的 buffer 的偏移值
nLower2BitBufStride	int	低 2bit 的 buffer 线宽值
b10BitPicFlag	int	10bit 图像标志位
bEnableAfbcFlag	int	是否开启 afbc 功能的标志位
nBufSize	int	Buffer size
nAfbcSize	int	Afbc buffer size
nDebugCount	int	用于 debug
nCurFrameInfo	VIDEO_FRM_STATUS_INFO	与码流特性相关的信息

4.5. FbmBufInfo

名称	FbmBufInfo	
功能描述	Frame buffer 相关信息	
属性	类型	描述
nBufNum	int	开辟 frame buffer 的个数
nBufWidth	int	开辟 frame buffer 的宽度
nBufHeight	int	开辟 frame buffer 的高度
ePixelFormat	int	视频帧的 yuv 排列方式
nAlignValue	int	视频帧 buffer 的宽度对齐方式
bProgressiveFlag	int	视频码流是 progressive 片源的标记
bIsSoftDecoderFlag	int	视频片源格式对应的解码器是否为软解格式
bHdrVideoFlag	int	是否为 hdr 视频的标志位
b10bitVideoFlag	int	是否为 10bit 码流的标志位
bAfbcModeFlag	int	开启 afbc 功能的模式参数
nTopOffset	int	视频帧有效显示区域顶端开始值
nBottomOffset	int	视频帧有效显示区域左侧开始值
nLeftOffset	int	视频帧有效显示区域底端结束值
nRightOffset	int	视频帧有效显示区域右侧结束值