ANYKA
芯 随 你 动

版本: 1.0.3 2014 年 12 月

媒体解析库接口说明

声　　明

联　系　方　式

安凯（广州）微电子技术有限公司

地址：广州科学城科学大道 182 号创新大厦 C1 区 3 楼
电话: (86)-20-3221 9000

传真: (86)-20-3221 9258

邮编: 510663

销售热线:

(86)-20-3221 9499

电子邮箱:

sales@anyka.com

主页:

http: //www. anyka.com

版 本 变 更 说 明

以下表格对于本文档的版本变更做一个简要的说明。版本变更仅限于技术内容的变更，不包括版式、格式、句法等的变更。

版本	说明	完成日期
V1.0.0	正式发布	2011 年 12 月
V1.0.1	更新描述	2011 年 12 月
V1.0.2	Add MediaLib_Dmx_FF_FR	2012 年 05 月
V1.0.3	Add MediaLib_Dmx_GetAudioPts	2014 年 12 月

媒体解析库接口说明与媒体解析库对应的关系

Version	Corresponding demuxer lib
V1.0.3	V1.16.09
V1.0.2	V1.13.00
V1.0.1	V1.11.00
V0.2.0	V0.10.0
V0.1.0	V0.9.0

目录

1 模块介绍

1.1 功能概述

本模块实现了各种文件格式的解析以及音视频的分离，支持的格式包括 AVI、MP4/3GP、FLV、RM/RMVB、MKV、ASF/WMV/WMA、WAV、MP3、AMR、AAC、APE、FLAC 和 MIDI。

表 1-1 支持文件格式列表

文件格式	常用音视频组合	
AVI	XVID（MPEG4 SP/ASP）	MP3
	H263	MP3
	MJPEG	PCM
MP4	H264/MPEG4 SP/ASP	AAC
3GP	H263	AMR
FLV	FLV263	MP3
RM/RMVB	RealVideo	COOK
MKV	H264	AAC
ASF/WMV/WMA	WMV	WMA
音频格式		
WAV	-	PCM/ADPCM
MP3	-	MP3
AMR	-	AMR
AAC/ADTS/ADIF	-	AAC
APE	-	APE
FLAC	-	FLAC
MIDI	-	MIDI
OGG	-	VORBIS
AC3	-	AC3

1.2 与其它模块关系

本模块依赖于以下模块：

- 资源管理模块
- 内存管理模块
- 系统功能模块

本模块需要调用以上模块相关接口进行资源读取与写入、内存分配释放等，具体见第5章“依赖接口说明”。

2 相关文档

《音视频库总体使用说明》。

3 集成指南

如果是首次使用媒体解析库，请首先阅读《音视频库总体使用说明》。

3.1 层次结构

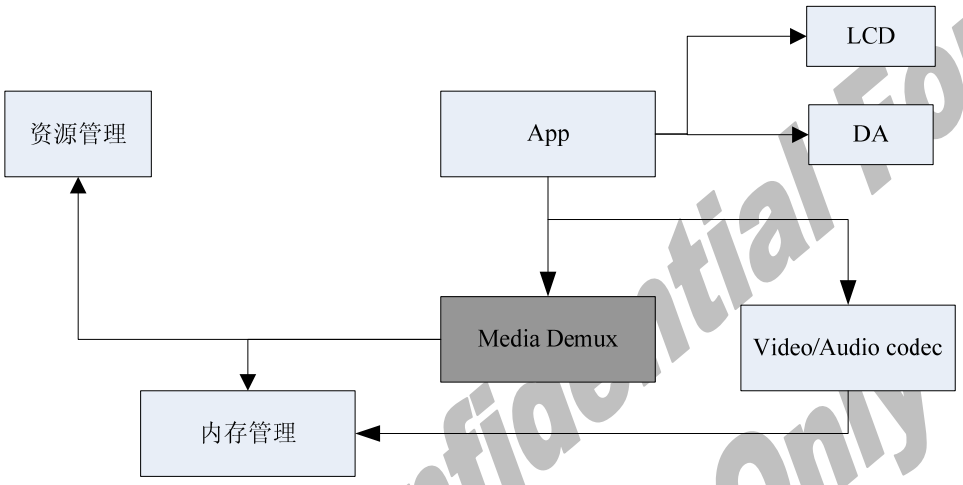

图 3-1 模块关系图

Media Demux 模块完成媒体文件的解复用过程，集成后在系统中的位置如上图所示。媒体解析库依赖的模块都应该在使用前由系统初始化。App 应用模块实现播放，调用系统层模块、控制 LCD 及 DA 实现播放；内存管理模块用于申请和释放内存；资源管理模块提供基本的资源读写，定位等功能。箭头标明在运行播放过程中模块的调用情况。

3.2 集成步骤

3.2.1 获得媒体解析库相关文件

媒体解析库包括库二进制文件、头文件和相关文档。

3.2.2 实现媒体解析库依赖的系统API

请仔细阅读本文的第五部分，准备好媒体解析库所依赖的系统 API，为系统控制媒体解析库提供足够的灵活性，需要在集成时根据实际情况由系统集成人员实现。

3.2.3 集成链接测试

把媒体解析库、依赖的系统 API 实现和目标系统进行链接。

3.2.4 联合调试

在目标系统上联合调试一般分成以下步骤，调试和使用媒体解析库常见问题请参考本文的相关部分。

1、调试播放无声媒体，确定解析层和视频解码内部播放基本正常运作。

2、调试播放音频媒体，确定解析层和音频解码内部播放基本正常运作。

3、调试播放有声媒体，确定解析层、视频解码和音频解码相关接口正常运作。

4、调试两路音频媒体播放，确定解析层和音频解码相关接口及混音正常运作。

5、调试各种类型媒体的播放，确定媒体解析库正常运作。

4 接口说明

4.1 模块功能概述

本模块起媒体解析的作用，请参考功能概述。

4.2 数据结构/格式

4.2.1 回调函数定义

见第 5 章“依赖接口说明”。

4.2.2 媒体解析句柄

typedef T_pVOID T_MEDIALIB_STRUCT;

用于媒体解析，存放解析库的所有信息，外部调用不需要知道具体定义，传入库后做强制转换。

4.2.3 媒体解析打开输入结构体

```
typedef struct
{
    T_eMEDIALIB_MEDIA_TYPE      m_MediaType;
    T_S32                       m_hMediaSource;
    T_MEDIALIB_CB               m_CBFunc;
}T_MEDIALIB_DMX_OPEN_INPUT;
```

结构名	T_MEDIALIB_DMX_OPEN_INPUT	
定义概述	打开时输入参数	
成员说明	m_MediaType	媒体文件类型
	m_hMediaSource	媒体文件句柄
	m_CBFunc	回调函数结构

4.2.4 媒体解析打开输出结构体

```
typedef struct
{
    T_eMEDIALIB_STATE          m_State;
}T_MEDIALIB_DMX_OPEN_OUTPUT;
```

结构名	T_MEDIALIB_DMX_OPEN_OUTPUT	
定义概述	打开时输出状态信息	
成员说明	m_State	状态信息

4.2.5 媒体信息结构

```
typedef struct
{
    T_eMEDIALIB_MEDIA_TYPE  m_MediaType;
    T_BOOL          m_bHasVideo;
    T_BOOL          m_bHasAudio;
    T_BOOL          m_bAllowSeek;
    T_BOOL          m_bSelfRecord;
    T_U32           m_ulTotalTime_ms;

    //video
    T_eVIDEO_DRV_TYPEm_VideoDrvType;
    T_U16           m_uWidth;
    T_U16           m_uHeight;
    T_U16           m_uFPS;
    T_U32           m_ulVideoBitrate;

    //audio
    T_AUDIO_TYPEm_AudioType;
    T_U32           m_ulAudioBitRate;
    T_U16           m_wFormatTag;
```

```
    T_U16       m_nChannels;
    T_U32       m_nSamplesPerSec;
    T_U32       m_nAvgBytesPerSec;
    T_U16       m_nBlockAlign;
    T_U16       m_wBitsPerSample;
    T_U16       m_cbSize;
    T_U8        m_szData[MEDIALIB_DMX_INFO_EX_SIZE];

    T_MEDIALIB_META_INFO    *m_pMetaInfo;
}T_MEDIALIB_DMX_INFO;
```

结构名	T_MEDIALIB_DMX_INFO	
定义概述	媒体信息结构	
成员说明	m_MediaType	媒体类型
	m_bHasVideo	有视频
	m_bHasAudio	有音频
	m_bAllowSeek	允许 seek
	m_bSelfRecord	自录文件
	m_ulTotalTime_ms	总时间（ms）
	m_VideoDrvType	视频驱动类型（T_eVIDEO_DRV_TYPE）
	m_uWidth	宽
	m_uHeight	高
	m_uFPS	帧率
	m_ulVideoBitrate	视频比特率
	m_AudioType	音频类型
	m_ulAudioBitRate	音频比特率
	m_wFormatTag	格式标记
	m_nChannels	声道数
	m_nSamplesPerSec	采样率
	m_nAvgBytesPerSec	平均采样率

结构名	T_MEDIALIB_DMX_INFO	
	m_nBlockAlign	块对其字节数
	m_wBitsPerSample	采样位宽
	m_cbSize	音频数据大小
	m_szData	音频数据
	m_pMetaInfo	T_MEDIALIB_META_INFO 类型，参见《音视频库总体使用说明》

4.2.6 媒体解析状态

```
typedef enum
{
    MEDIALIB_DMX_END,
    MEDIALIB_DMX_PLAY,
    MEDIALIB_DMX_PAUSE,
    MEDIALIB_DMX_STOP,
    MEDIALIB_DMX_ERR,
    MEDIALIB_DMX_SEEK,
    MEDIALIB_DMX_FF,
    MEDIALIB_DMX_FR
}T_eMEDIALIB_DMX_STATUS;
```

结构名	T_eMEDIALIB_DMX_STATUS	
定义概述	媒体解析状态	
成员说明	MEDIALIB_DMX_END	播放结束
	MEDIALIB_DMX_PLAY	正常播放
	MEDIALIB_DMX_PAUSE	暂停
	MEDIALIB_DMX_STOP	停止
	MEDIALIB_DMX_ERR	出错
	MEDIALIB_DMX_SEEK	Seek

结构名	T_eMEDIALIB_DMX_STATUS	
	MEDIALIB_DMX_FF	快进
	MEDIALIB_DMX_FR	快退

4.2.7 媒体类型

T_eMEDIALIB_MEDIA_TYPE，参见《音视频库总体使用说明》。

4.2.8 视频编码类型

T_eVIDEO_DRV_TYPE，参见《音视频库总体使用说明》。

4.2.9 音频编码类型

T_AUDIO_TYPE，参见《音视频库总体使用说明》。

4.2.10 媒体回调函数结构

T_MEDIALIB_CB，参见《音视频库总体使用说明》。

4.3 接口函数列表

MediaLib_GetVersion 参见《音视频库总体使用说明》。

4.3.1 MediaLib_Dmx_Open

原　　型	T_MEDIALIB_STRUCTMediaLib_Dmx_Open(T_MEDIALIB_DMX_OPEN_INPUT*dmx_open_input,T_MEDIALIB_DMX_OPEN_OUTPUT *dmx_open_output);	
功能概述	打开 demuxer,分析媒体头并检验参数	
参数说明	dmx_open_input	打开传入参数结构
	dmx_open_output	打开输出参数结构,打开状态
返回值说明	T_MEDIALIB_STRUCT 结构,为 demuxer 句柄	
注意事项	首先调用该函数才能调用其他函数	

原　　型	T_MEDIALIB_STRUCTMediaLib_Dmx_Open(T_MEDIALIB_DMX_OPEN_INPUT*dmx_open_input,T_MEDIALIB_DMX_OPEN_OUTPUT *dmx_open_output);
调用示例	见典型调用示例

4.3.2 MediaLib_Dmx_Close

原　　型	T_BOOL MediaLib_Dmx_Close(T_MEDIALIB_STRUCT hMedia);	
功能概述	关闭 demuxer	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针,即 Dmx 句柄
返回值说明	AK_TRUE　关闭成功 AK_FALSE　关闭失败	
注意事项		
调用示例	见典型调用示例	

4.3.3 MediaLib_Dmx_Start

原　　型	T_S32 MediaLib_Dmx_Start(T_MEDIALIB_STRUCT hMedia, T_U32 start_pos);	
功能概述	准备读取视频和音频数据	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针,即 Dmx 句柄
	start_pos	开始时间，0 或者是由 setpos 返回的 pts
返回值说明	< 0　失败 Other　如果有音频返回音频 pts,否则返回视频 pts;	
注意事项	必须已经成功执行 MediaLib_Dmx_Open 函数。	
调用示例	见典型调用示例	

4.3.4 MediaLib_Dmx_Resume

原　　型	T_BOOL MediaLib_Dmx_Resume(T_MEDIALIB_STRUCT hMedia);	
功能概述	重新读取音频和视频数据	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针,即 Dmx 句柄
返回值说明	AK_TRUE　Resume 成功 AK_FALSE　Resume 失败	
注意事项		
调用示例	见典型调用示例	

4.3.5 MediaLib_Dmx_Stop

原　　型	T_BOOL MediaLib_Dmx_Stop(T_MEDIALIB_STRUCT hMedia);	
功能概述	停止读取音频和视频数据	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针,即 Dmx 句柄
返回值说明	AK_TRUE　停止成功 AK_FALSE　停止失败	
注意事项	必须已经成功执行 MediaLib_Dmx_Open 函数。	
调用示例	见典型调用示例	

4.3.6 MediaLib_Dmx_Pause

原　　型	T_BOOL MediaLib_Dmx_Pause(T_MEDIALIB_STRUCT hMedia);	
功能概述	暂停读取音频和视频数据	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针,即 Dmx 句柄
返回值说明	AK_TRUE　暂停成功 AK_FALSE　暂停失败	
注意事项	必须已经成功执行 MediaLib_Dmx_Open 函数。	

原　　型	T_BOOL MediaLib_Dmx_Pause(T_MEDIALIB_STRUCT hMedia);	
调用示例	见典型调用示例	

4.3.7 MediaLib_Dmx_GetInfo

原　　型	T_BOOL MediaLib_Dmx_GetInfo(T_MEDIALIB_STRUCT hMedia, T_MEDIALIB_DMX_INFO *pInfo);	
功能概述	获取当前的媒体信息	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针,即 Dmx 句柄
	pInfo	输出参数，T_MEDIALIB_DMX_INFO 结构指针
返回值说明	AK_TRUE　获取成功 AK_FALSE　获取失败	
注意事项	1.　必须已经成功执行 MediaLib_Dmx_Open 函数。 2.　必须保证 pInfo 不为空指针。	
调用示例	见典型调用示例	

4.3.8 MediaLib_Dmx_ReleaseInfoMem

原　　型	T_BOOL MediaLib_Dmx_ReleaseInfoMem(T_MEDIALIB_STRUCT hMedia);	
功能概述	释放存贮信息的内存空间	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针,即 Dmx 句柄
返回值说明	AK_TRUE　释放成功 AK_FALSE　释放失败	
注意事项	1. 必须已经成功执行 MediaLib_Dmx_Open 函数 2. MediaLib_Dmx_GetInfo 调用之后，上层已获取媒体相关信息并处理，通过调用该函数释放部分与播放无关的资源，减少播放过程中对内存的占用 3. 该函数不调用不会影响播放，也不会造成内存泄漏，MediaLib_Dmx_Close 时会释放所有资源	
调用示例	见典型调用示例	

4.3.9 MediaLib_Dmx_GetAudioSeekInfo

原　　型	T_AUDIO_SEEK_INFO *MediaLib_Dmx_GetAudioSeekInfo(T_MEDIALIB_STRUCT hMedia);	
功能概述	获取音频 seek 的信息	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针,即 Dmx 句柄
返回值说明	T_AUDIO_SEEK_INFO 结构指针	
注意事项	-	
调用示例	见典型调用示例	

4.3.10 MediaLib_Dmx_GetStatus

原　　型	T_eMEDIALIB_DMX_STATUS MediaLib_Dmx_GetStatus(T_MEDIALIB_STRUCT hMedia);	
功能概述	获取 demuxer 的状态	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针，即 Dmx 句柄
返回值说明	demuxer 状态，详见 T_eMEDIALIB_DMX_STATUS 说明	
注意事项	-	
调用示例	见典型调用示例	

4.3.11 MediaLib_Dmx_SetPosition

原　　型	T_S32 MediaLib_Dmx_SetPosition(T_MEDIALIB_STRUCT hMedia, T_U32 uTimeMs, T_BOOL bSeekNext);	
功能概述	设置当前位置	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针,即 Dmx 句柄
	uTimeMs	媒体文件的时间（ms）
	bSeekNext	是否定位到下一关键帧

原　　型	T_S32 MediaLib_Dmx_SetPosition(T_MEDIALIB_STRUCT hMedia, T_U32 uTimeMs, T_BOOL bSeekNext);
返回值说明	< 0　失败； Other　如果有音频返回音频 pts,否则返回视频 pts;
注意事项	定位到最近的关键帧
调用示例	见典型调用示例

4.3.12 MediaLib_Dmx_ResetAudioPos

原　　型	T_S32 MediaLib_Dmx_ResetAudioPos(T_MEDIALIB_STRUCT hMedia);	
功能概述	重新设置音频位置，用于获取第一个音频包	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针,即 Dmx 句柄
返回值说明	< 0　Reset 失败 Other　音频 pts	
注意事项	-	
调用示例	见典型调用示例	

4.3.13 MediaLib_Dmx_GetAudioData

原　　型	T_U32 MediaLib_Dmx_GetAudioData(T_MEDIALIB_STRUCT hMedia, T_pDATA pData, T_U32 uDataLen);	
功能概述	获取音频数据	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针，即 Dmx 句柄
	pData	输出参数，音频数据
	uDataLen	获取数据长度，MediaLib_Dmx_GetAudioDataSize 函数返回的大小
返回值说明	< 0　没有音频数据 Other　实际获取的数据长度	
注意事项	-	
调用示例	见典型调用示例	

4.3.14 MediaLib_Dmx_GetAudioDataSize

原　　型	T_U32 MediaLib_Dmx_GetAudioDataSize(T_MEDIALIB_STRUCT hMedia);	
功能概述	获取音频大小	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针，即 Dmx 句柄
返回值说明	< 0　没有音频数据 Other 音频数据的大小	
注意事项	-	
调用示例	见典型调用示例	

4.3.15 MediaLib_Dmx_CheckAudioEnd

原　　型	T_BOOL MediaLib_Dmx_CheckAudioEnd(T_MEDIALIB_STRUCT hMedia);	
功能概述	检验是否音频已经结束	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针，即 Dmx 句柄
返回值说明	AK_TRUE　已结束 AK_FALSE 没结束	
注意事项	-	
调用示例	见典型调用示例	

4.3.16 MediaLib_Dmx_CheckVideoEnd

原　　型	T_BOOL MediaLib_Dmx_CheckVideoEnd(T_MEDIALIB_STRUCT hMedia);	
功能概述	检验是否视频已经结束	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针，即 Dmx 句柄
返回值说明	AK_TRUE　已结束 AK_FALSE 没结束	
注意事项	-	

原　　型	T_BOOL MediaLib_Dmx_CheckVideoEnd(T_MEDIALIB_STRUCT hMedia);
调用示例	见典型调用示例

4.3.17 MediaLib_Dmx_GetFirstVideoSize

原　　型	T_U32 MediaLib_Dmx_GetFirstVideoSize(T_MEDIALIB_STRUCT hMedia);	
功能概述	获取第一个关键桢的大小，包含码流头和第一个关键帧的数据	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针，即 Dmx 句柄
返回值说明	< 0　没有视频数据 Other　第一个关键桢的大小	
注意事项	-	
调用示例	见典型调用示例	

4.3.18 MediaLib_Dmx_GetFirstVideo

原　　型	T_BOOL MediaLib_Dmx_GetFirstVideo(T_MEDIALIB_STRUCT hMedia, T_pDATA pData, T_U32 *pDataLen);	
功能概述	获取第一个关键桢的数据，含码流头和第一个关键桢	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针，即 Dmx 句柄
	pData	输出参数，视频数据
	pDataLen	输入 / 输出参数，MediaLib_Dmx_GetVideoDataSize 函数返回的数据长度
返回值说明	AK_TRUE　成功 AK_FALSE　失败	
注意事项	-	
调用示例	见典型调用示例	

4.3.19 MediaLib_Dmx_GetVideoFrameSize

原　　型	T_U32 MediaLib_Dmx_GetVideoFrameSize(T_MEDIALIB_STRUCT hMedia);	
功能概述	获取视频帧大小	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针，即 Dmx 句柄
返回值说明	< 0　没有视频数据 Other 帧大小	
注意事项	-	
调用示例	见典型调用示例	

4.3.20 MediaLib_Dmx_GetVideoFrame

原　　型	T_BOOL MediaLib_Dmx_GetVideoFrame(T_MEDIALIB_STRUCT hMedia, T_pDATA pData, T_U32 *pDataLen);	
功能概述	获取视频帧数据	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针，即 Dmx 句柄
	pData	输出参数，视频帧数据
	pDataLen	输入输出参数，数据长度，MediaLib_Dmx_GetVideoDataSize 函数返回值
返回值说明	AK_TRUE　成功 AK_FALSE　失败	
注意事项	-	
调用示例	见典型调用示例	

4.3.21 MediaLib_Dmx_DisableVideo

原　　型	T_BOOL MediaLib_Dmx_ DisableVideo(T_MEDIALIB_STRUCT hMedia);	
功能概述	使视频无效	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针，即 Dmx 句柄

原　　型	T_BOOL MediaLib_Dmx_ DisableVideo(T_MEDIALIB_STRUCT hMedia);
返回值说明	AK_TRUE　成功 AK_FALSE　失败
注意事项	调用者发现视频解码方式不支持时调用，库内部停止获取视频数据
调用示例	-

4.3.22 MediaLib_Dmx_DisableAudio

原　　型	T_BOOL MediaLib_Dmx_DisableAudio (T_MEDIALIB_STRUCT hMedia);	
功能概述	使音频无效	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针，即 Dmx 句柄
返回值说明	AK_TRUE　成功 AK_FALSE　失败	
注意事项	调用者发现音频解码方式不支持时调用，库内部停止获取音频数据	
调用示例	-	

4.3.23 MediaLib_Dmx_GetNextBlockInfo

```
typedef enum
{
    T_eMEDIALIB_BLKTYPE_UNKNOWN,
    T_eMEDIALIB_BLKTYPE_VIDEO,
    T_eMEDIALIB_BLKTYPE_AUDIO
}T_MEDIALIB_BLKTYPE;


typedef struct
{
    T_MEDIALIB_BLKTYPE      m_eBlkType;    //数据类型：音频、视频
    T_U32                   m_ulBlkLen;    //数据长度
}T_MEDIALIB_DMX_BLKINFO;
```

原　　型	T_BOOL MediaLib_Dmx_GetNextBlockInfo(T_MEDIALIB_STRUCT hMedia, T_MEDIALIB_DMX_BLKINFO *dmxBlockInfo);	
功能概述	获取数据类型和长度	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针，即 Dmx 句柄
	dmxBlockInfo	输出参数，见 T_MEDIALIB_DMX_BLKINFO 结构体
返回值说明	AK_TRUE　成功 AK_FALSE　失败	
注意事项	该函数用于按文件中音视频包的存放顺序来读取音视频数据，类似于推模式	
调用示例	见典型调用示例	

4.3.24 MediaLib_Dmx_FF_FR

原　　型	T_BOOL MediaLib_Dmx_FF_FR(T_MEDIALIB_STRUCT hMedia, T_U32 video_pts, T_eMEDIALIB_DMX_STATUS dmx_status)	
功能概述	设置获取视频数据的状态为快进或快退	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针，即 Dmx 句柄
	video_pts	快进/快退的起始时间点，以毫秒为单位
	dmx_status	MEDIALIB_DMX_FF：快进； MEDIALIB_DMX_FR：快退 其它：无效
返回值说明	AK_TRUE　成功 AK_FALSE　失败	
注意事项	1. 设置为快进/快退模式后只能获取视频帧，且获取到的帧全是关键帧； 2. 当该函数返回失败时，可能的原因是文件中没有视频或只有一个关键帧； 3. MEDIALIB_DMX_STOP 和 MEDIALIB_DMX_END 状态下调用该函数无效；	

原　　型	T_BOOL MediaLib_Dmx_FF_FR(T_MEDIALIB_STRUCT hMedia, T_U32 video_pts, T_eMEDIALIB_DMX_STATUS dmx_status)
调用示例	... MediaLib_Dmx_FF_FR(hMedia, 10000, MEDIALIB_DMX_FF); streamLen = MediaLib_Dmx_GetVideoFrameSize(hMedia); ...

4.3.25 MediaLib_Dmx_GetAudioPts

原　　型	T_BOOL MediaLib_Dmx_GetAudioPts(T_MEDIALIB_STRUCT hMedia, T_U32 *pAudioPts)	
功能概述	获取每个音频包的时间戳	
参数说明	hMedia	MediaLib_Dmx_Open 函数返回的 T_MEDIALIB_STRUCT 结构指针，即 Dmx 句柄
	pAudioPts	用于获取每个音频包的时间戳
返回值说明	AK_TRUE　成功 AK_FALSE　失败	
注意事项	1. 该函数必须在 MediaLib_Dmx_GetAudioDataSize 和 MediaLib_Dmx_GetAudioData 之间调用 2. 只有 avi、3gp、mp4 提供该功能 3. 需要获取音频包时间才调用，普通文件播放时一般不需要调用	

调用示例	...
	streamLen = MediaLib_Dmx_GetAudioDataSize(hMedia);
	if (streamLen != 0)
	{
	　　bRet = MediaLib_Dmx_GetAudioPts(hMedia, &audioPts);
	　　MediaLib_Dmx_GetAudioData(hMedia, streamBuf, streamLen);
	}	

4.4 典型调用范例

```
/*************************************************************
* The following is an example to use demuxing APIs
*************************************************************/
T_S32 Demux_Media(char* filename)
{
    T_MEDIALIB_DMX_OPEN_INPUT open_input;
    T_MEDIALIB_DMX_OPEN_OUTPUT open_output;
    T_MEDIALIB_DMX_INFO media_info;
    T_U8 streamBuf[960*480*2] = {0};
    T_eMEDIALIB_DMX_STATUS demux_status;
    T_VOID *hMedia;
    T_S32 fid;
    T_U32 streamLen = 0;
    T_S32 pts = 0;

    FILE *fp;
    FILE *fp_a;

    fid = _open(filename, _O_RDONLY | _O_BINARY);
    if(fid <= 0)
    {
            printf("open file failed\r\n");
            return 0;
    }

    memset(&open_input, 0, sizeof(T_MEDIALIB_DMX_OPEN_INPUT));
    open_input.m_hMediaSource = fid;
   open_input.m_CBFunc.m_FunPrintf = (MEDIALIB_CALLBACK_FUN_PRINTF)printf;
    open_input.m_CBFunc.m_FunRead = (MEDIALIB_CALLBACK_FUN_READ)_read;
   open_input.m_CBFunc.m_FunWrite = (MEDIALIB_CALLBACK_FUN_WRITE)_write;
```

```
open_input.m_CBFunc.m_FunSeek = (MEDIALIB_CALLBACK_FUN_SEEK)_lseek;
open_input.m_CBFunc.m_FunTell = (MEDIALIB_CALLBACK_FUN_TELL)_tell;
    open_input.m_CBFunc.m_FunMalloc=
(MEDIALIB_CALLBACK_FUN_MALLOC)my_malloc;
open_input.m_CBFunc.m_FunFree = (MEDIALIB_CALLBACK_FUN_FREE)free;
open_input.m_CBFunc.m_FunFileHandleExist = file_handle_exist;

hMedia = MediaLib_Dmx_Open(&open_input, &open_output);
if (AK_NULL == hMedia)
{
        _close(fid);
        return 0;
}

MediaLib_Dmx_GetInfo(hMedia, &media_info);
MediaLib_Dmx_ReleaseInfoMem(hMedia);

bSeekFlag = 0;

streamLen = MediaLib_Dmx_GetFirstVideoSize(hMedia);
if (MediaLib_Dmx_GetFirstVideo(hMedia, streamBuf, &streamLen) == AK_FALSE)
{
        MediaLib_Dmx_Close(hMedia);
        _close(fid);
        return 0;
}

//here decode first video

pts = MediaLib_Dmx_Start(hMedia, 0);

while (1)
```

```
    {
        if (!MediaLib_Dmx_CheckAudioEnd(hMedia))
        {
            streamLen = MediaLib_Dmx_GetAudioDataSize(hMedia);
            if (streamLen != 0)
            {
                MediaLib_Dmx_GetAudioData(hMedia, streamBuf,
streamLen);
            }

            //here decode audio
        }

        if (!MediaLib_Dmx_CheckVideoEnd(hMedia))
        {
            streamLen = MediaLib_Dmx_GetVideoFrameSize(hMedia);
            if (streamLen != 0)
            {
                MediaLib_Dmx_GetVideoFrame(hMedia, streamBuf,
&streamLen);
            }

            //here decode video
        }

        demux_status = MediaLib_Dmx_GetStatus(hMedia);
        if (demux_status == MEDIALIB_DMX_END ||
    demux_status == MEDIALIB_DMX_ERR)
        {
            break;
        }
    }
```

```
    if (fp != 0)
    {
            fclose(fp);
    }
    if (fp_a != 0)
    {
            fclose(fp_a);
    }

    MediaLib_Dmx_Close(hMedia);

    _close(fid);

    return 1;
}
```

**********instead of the code above in while(1)**********

以下两种模式的代码分别替换上述代码中 while(1)包含的内容，实现相应的功能：

模式一、forward read mode:顺序读取模式，按照文件中音视频包的数据获取音视频

```
    while (1)
    {
    if (MediaLib_Dmx_GetNextBlockInfo(hMedia, &dmxBlockInfo) == AK_FALSE)
    {
            //error
            break;
    }

    streamLen = dmxBlockInfo.m_ulBlkLen;
    switch (dmxBlockInfo.m_eBlkType)
    {
    case T_eMEDIALIB_BLKTYPE_VIDEO:
```

```
                if (streamLen != 0)
                {
                        MediaLib_Dmx_GetVideoFrame(hMedia, streamBuf, &streamLen);
                }
                        break;
        case T_eMEDIALIB_BLKTYPE_AUDIO:
                if (streamLen != 0)
                {
                        MediaLib_Dmx_GetAudioData(hMedia, streamBuf, streamLen);
                }
                break;
                default:
                        break;
                }
                demux_status = MediaLib_Dmx_GetStatus(hMedia);
                if (demux_status == MEDIALIB_DMX_END || demux_status ==
MEDIALIB_DMX_ERR)
                {
                        break;
                }
        }
```

模式二、mix read mode:混序模式，顺序读取与强制读取结合，可根据实际需要进行切换

```
        while (1)
        {
        if (MediaLib_Dmx_GetNextBlockInfo(hMedia, &dmxBlockInfo) == AK_FALSE)
                {
                        //error
                        break;
                }
                streamLen = dmxBlockInfo.m_ulBlkLen;
```

```
switch (dmxBlockInfo.m_eBlkType)
{
case T_eMEDIALIB_BLKTYPE_VIDEO:
        if (streamLen != 0)
        {
        MediaLib_Dmx_GetVideoFrame(hMedia, streamBuf, &streamLen);
        }
        break;
case T_eMEDIALIB_BLKTYPE_AUDIO:
        if (streamLen != 0)
        {
        MediaLib_Dmx_GetAudioData(hMedia, streamBuf, streamLen);
        }
        break;
default:
        break;
}
if (!MediaLib_Dmx_CheckAudioEnd(hMedia))
{
        streamLen = MediaLib_Dmx_GetAudioDataSize(hMedia);
        if (streamLen != 0)
        {
        MediaLib_Dmx_GetAudioData(hMedia, streamBuf, streamLen);
        }
}
if (!MediaLib_Dmx_CheckVideoEnd(hMedia))
{
        streamLen = MediaLib_Dmx_GetVideoFrameSize(hMedia);
        if (streamLen != 0)
        {
        MediaLib_Dmx_GetVideoFrame(hMedia, streamBuf, &streamLen);
        }
```

```
                }
                demux_status = MediaLib_Dmx_GetStatus(hMedia);
                if (demux_status == MEDIALIB_DMX_END || demux_status =
MEDIALIB_DMX_ERR)
                {
                        break;
                }
        }
```

4.5 注意事项

MediaLib_Dmx_GetAudioSeekInfo 仅对 APE 格式的文件有效，其他格式返回 NULL。

5 依赖接口说明

从 3.1 层次关系可以看出媒体解析库所依赖的外部接口主要有资源管理、内存管理及其他一些媒体解析库需要的功能接口，该类函数由目标平台实现。参见《音视频库总体使用说明》。

6 常见问题

6.1 播放常见问题

1、调用 MediaLib_Dmx_Open 失败

检查传入参数是否正确；该文件不是媒体文件或暂时不支持此类文件的播放。

2、定位误差偏大，例如定位到第 10 秒播放，实际从第 9 秒开始播放

这属于正常现象。如果定位指定的位置不是一个视频关键帧，根据视频压缩的特性必须从之前最接近的一个关键帧开始依次每帧解码才能得到指定位置的视频帧。不同媒体资源关键帧的分布差别很大，从关键帧依次解码耗费的时间可能很长造成响应时间过长，为此只能从最接近的关键帧初开始播放。

3、某些媒体资源不能定位

这属于正常现象。某些媒体资源只有一个关键帧或没有索引列表。可以通过媒体信息得知文件是否可以定位，见 T_MEDIALIB_DMX_INFO。

4、媒体解析时有大量的对资源进行 seek 的操作

根据各个媒体格式的特点，有的文件格式中信息和数据分隔很远，因此在获取音视频时会有较大的跳转动作，这是正常现象。

6.2 其他问题

参见《音视频库总体使用说明》。