版本:**1.0.4 2014** 年 **06** 月

音视频库总体使用说明

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	增加一个获取资源长度的回调	2012 年 06 月
V1.0.3	增加媒体合成库相关内容	2012 年 10 月
V1.0.4	增加芯片类型 AK3751C	2014 年 06 月

目录

1 模块介绍

1.1 功能概述

本模块提供的库适应于 RTOS 平台和 Android、WINCE 等智能平台，用于实现音视频编解码、播放和录像功能。

1.1.1 单线程播放功能

调用 media_player_lib.h 头文件中的接口实现，库内部会自动调用媒体解析、音视频解码接口将音视频码流解码成 PCM 数据和 YUV 数据，平台进行相应的输出。

1.1.2 多线程播放功能

需要平台根据需要创建多个线程，分别调用媒体解析、音视频解码三个接口实现。

媒体解析调用 media_demuxer_lib.h 头文件中的接口；音频解码调用 sdcodec.h 头文件中的接口；视频解码调用 video_stream_lib.h 头文件中的接口。

1.1.3 录像功能

调用 media_recorder_lib.h 头文件中的接口实现，如果音频数据需要压缩，还需要平台调用 sdcodec.h 头文件中的接口进行音频编码。

1.1.4 音视频解码功能

音频解码调用 sdcodec.h 头文件中的接口；视频解码调用 video_stream_lib.h 头文件中的接口。主要用于解码接收的音视频码流或其他解析器解析出的码流，如 CMMB 播放。

1.1.5 CMMB录像功能

调用 mobiletv_recorder_lib.h 头文件中的接口实现，在 CMMB 播放时需要调用音视频解码，CMMB 录像时的音频码流要求是来自于音频解码器处理后的无误码的码流，这样才能保证录制的文件在播放时是同步的。

1.1.6 媒体合成功能

调用 media_muxer_lib.h 头文件中的接口实现，需要平台提供已经编码的音视频码流，传入媒体合成库，同时给出正确的时间戳信息，这样才能保证合成的文件在播放时是同步的。

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 库列表、头文件与相关文档

2.1 头文件与相关文档

头文件	对应的接口文档
media_player_lib.h	《媒体播放库接口说明》
media_recorder_lib.h	《媒体录制库接口说明》
media_demuxer_lib.h	《媒体解析库接口说明》
mobiletv_recorder_lib.h	《CMMB 录制库接口说明》
media_muxer_lib.h	《媒体合成库接口说明》
video_stream_lib.h	《视频驱动库接口说明》
sdcodec.h	《音频库接口说明》

头文件	依赖的公共头文件
media_player_lib.h	medialib_global.h、medialib_struct.h
media_recorder_lib.h	medialib_global.h、medialib_struct.h
media_demuxer_lib.h	medialib_global.h、medialib_struct.h
mobiletv_recorder_lib.h	medialib_global.h、medialib_struct.h
media_muxer_lib.h	medialib_global.h、medialib_struct.h
video_stream_lib.h	medialib_global.h
sdcodec.h	medialib_global.h

2.2 各平台的库列表

平台	媒体库（**media_lib**）	视频驱动库（**video_lib**）	音频编解码库（**audio_lib**）
RTOS	media_lib_aspen.a	video_lib_aspen.a	sdcodec.a
Wince	medialib_ce.lib	video_lib_ce.lib	soundlibce.lib
Android	libakmedialib.so	libakvideocodec.so	libakaudiocodec.so
Linux	libakmedialib.so.0.1.0	libakvideocodec.so.0.1.0	libakaudiocodec.so.0.1.0

3 集成指南

3.1 层次结构

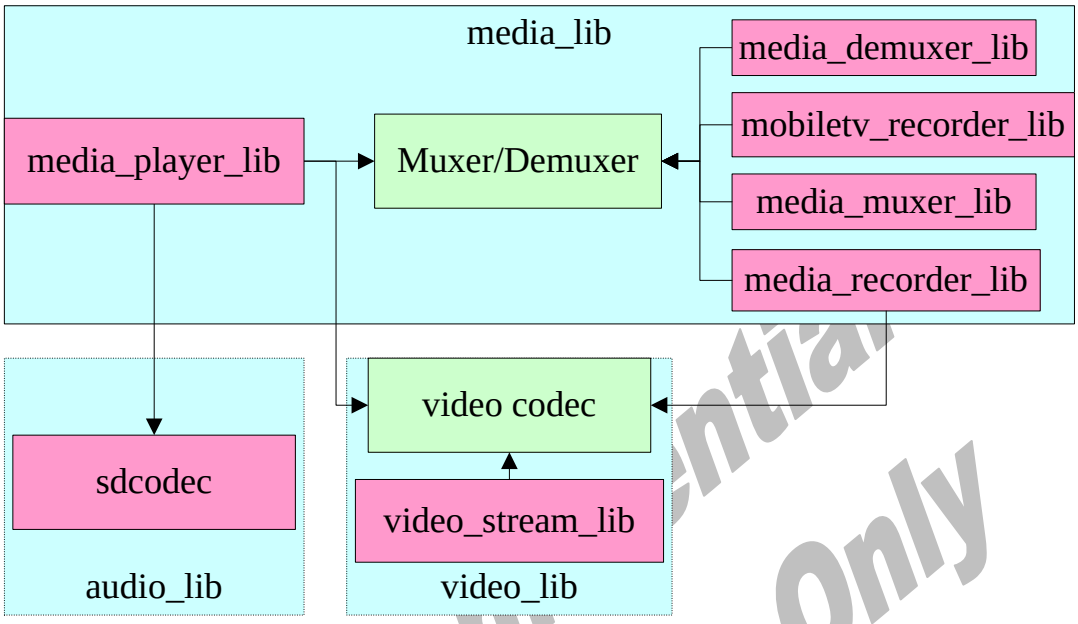

图 3-1 模块关系图

AudioVideoDriver 包含媒体库（media_lib）、视频驱动库（video_lib）、音频编解码库（audio_lib），各接口在库中的位置和调用关系如上图所示。

3.2 集成步骤

3.2.1 获得库相关文件

包括库二进制文件、头文件和相关文档，列表见第二章节。

3.2.2 实现库依赖的系统API

请仔细阅读本文的第五部分，准备好库所依赖的系统 API，为提供足够的灵活性，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

在目标系统根据不同的功能进行调试。

3.3 注意事项

集成过程中对于设置芯片类型的调用十分重要，因为如果没有设置芯片类型或设置了错误的芯片类型会造成无法正常编解码甚至死机。

有两个函数可设置芯片类型，分别是媒体库接口中的 MediaLib_Init 和视频驱动库中的 VideoStream_Init 函数。MediaLib_Init 的说明在本文第 4 章。VideoStream_Init 的说明在《视频驱动库接口说明》的第 4 章。

4 接口说明

此处列出一些在多个接口中共用的结构体定义，其他见各接口对应的说明文档（详见第 2 章的列表）。

4.1 媒体库初始化结构体

```
typedef struct
{
        T_eMEDIALIB_CHIP_TYPE       m_ChipType;
        T_AUDIO_I2S_APP             m_AudioI2S;  //AK78/AK88/AK98/AK37 系列不使用
}T_MEDIALIB_INIT_INPUT;

typedef enum
{
        MEDIALIB_CHIP_UNKNOWN,      //default AK3223
        MEDIALIB_CHIP_AK3223,       //AK3220/3221/3223/3224 first version
        MEDIALIB_CHIP_AK3224,       //AK3224 metal fix, AK3225
        MEDIALIB_CHIP_AK3610,       //AK3610 and AK3620
        MEDIALIB_CHIP_AK3810,       //AK3810, AK7801 and AK7802
        MEDIALIB_CHIP_AK3610_2,     //AK3631 and AK3631L, AK322L
        MEDIALIB_CHIP_AK8801,       //AK8801 and AK8802
        MEDIALIB_CHIP_AK3671,       //AK3671, AK3675, AK3650...
        MEDIALIB_CHIP_AK9802,       //AK9801, AK9802, AK9805...
        MEDIALIB_CHIP_AK3751,       //AK3751, AK3760, AK3771...
        MEDIALIB_CHIP_AK3751C       //AK3751C, AK3750C, AK3760C...
}T_eMEDIALIB_CHIP_TYPE;

typedef enum
{
        I2S_UNUSE,
        I2S_DA,
        I2S_AD,
```

I2S_DA_AD

}T_AUDIO_I2S_APP;

结构名	T_eMEDIALIB_CHIP_TYPE	
定义概述	芯片类型	
成员说明	MEDIALIB_CHIP_UNKNOWN	未知，默认为 AK3223M
	MEDIALIB_CHIP_AK3223	AK3220M, AK3221M, AK3223M
	MEDIALIB_CHIP_AK3224	AK3224M
	MEDIALIB_CHIP_AK3610	AK3610M and AK3620M
	MEDIALIB_CHIP_AK3810	AK3810M
	MEDIALIB_CHIP_AK3610_2	AK3631 and AK3631L, AK322L
	MEDIALIB_CHIP_AK8801	AK8801 and AK8802
	MEDIALIB_CHIP_AK3671	AK3671, AK3675, AK3650...
	MEDIALIB_CHIP_AK9802	AK9801, AK9802, AK9805...
	MEDIALIB_CHIP_AK3751	AK3751, AK3760, AK3771...
	MEDIALIB_CHIP_AK3751C	AK3751C, AK3750C, AK3760C...

typedef struct

{

MEDIALIB_CALLBACK_FUN_PRINTF m_FunPrintf;

MEDIALIB_CALLBACK_FUN_LOADRESOURCE m_FunLoadResource;

MEDIALIB_CALLBACK_FUN_RELEASERESOURCE m_FunReleaseResource;

}T_MEDIALIB_INIT_CB;

结构名	T_MEDIALIB_INIT_CB	
定义概述	初始化时需要的回调函数	
成员说明	m_FunPrintf	打印函数回调
	m_FunLoadResource	音频 load 资源的回调，AK78/88/98/37 系列不使用
	m_FunReleaseResource	音频 release 资源的回调，AK78/88/98/37 系列不使用

4.2 媒体输入结构体

4.2.1 媒体类型

```
typedef enum
{
        MEDIALIB_MEDIA_UNKNOWN,
        MEDIALIB_MEDIA_AKV,
        MEDIALIB_MEDIA_AVI,
        MEDIALIB_MEDIA_WAV,
        MEDIALIB_MEDIA_MP4,
        MEDIALIB_MEDIA_3GP,
        MEDIALIB_MEDIA_M4A,
        MEDIALIB_MEDIA_MOV,
        MEDIALIB_MEDIA_ASF,
        MEDIALIB_MEDIA_REAL,
        MEDIALIB_MEDIA_MP3,
        MEDIALIB_MEDIA_AKV2,    //add below
        MEDIALIB_MEDIA_AAC,
        MEDIALIB_MEDIA_AMR,
        MEDIALIB_MEDIA_MIDI,
        MEDIALIB_MEDIA_FLV,
        MEDIALIB_MEDIA_APE,
        MEDIALIB_MEDIA_FLAC,
        MEDIALIB_MEDIA_OGG_FLAC,
        MEDIALIB_MEDIA_RIFF_AV,
        MEDIALIB_MEDIA_OGG_VORBIS,
        MEDIALIB_MEDIA_MKV,
        MEDIALIB_MEDIA_F4V,
        MEDIALIB_MEDIA_AC3,
        MEDIALIB_MEDIA_SPEEX
}T_eMEDIALIB_MEDIA_TYPE;
```

结构名	T_eMEDIALIB_MEDIA_TYPE	
定义概述	媒体类型	
成员说明	MEDIALIB_MEDIA_UNKNOWN	未知类型
	MEDIALIB_MEDIA_AKV	AKV
	MEDIALIB_MEDIA_AVI	AVI
	MEDIALIB_MEDIA_WAV	WAV
	MEDIALIB_MEDIA_MP4	MP4
	MEDIALIB_MEDIA_3GP	3GP
	MEDIALIB_MEDIA_M4A	M4P
	MEDIALIB_MEDIA_MOV	MOV
	MEDIALIB_MEDIA_ASF	ASF
	MEDIALIB_MEDIA_REAL	REAL
	MEDIALIB_MEDIA_MP3	MP3
	MEDIALIB_MEDIA_AKV2	AKV2
	MEDIALIB_MEDIA_AAC	AAC
	MEDIALIB_MEDIA_AMR	AMR
	MEDIALIB_MEDIA_MIDI	MIDI
	MEDIALIB_MEDIA_FLV	FLV
	MEDIALIB_MEDIA_APE	APE
	MEDIALIB_MEDIA_FLAC	FLAC
	MEDIALIB_MEDIA_OGG_FLAC	OGG FLAC
	MEDIALIB_MEDIA_RIFF_AV	RIFF AV File
	MEDIALIB_MEDIA_OGG_VORBIS	OGG
	MEDIALIB_MEDIA_MKV	MKV
	MEDIALIB_MEDIA_F4V	F4V
	MEDIALIB_MEDIA_AC3	AC3
	MEDIALIB_MEDIA_SPEEX	SPEEX

4.2.2 媒体录制文件类型

typedef enum

{

MEDIALIB_REC_AVI_NORMAL,

MEDIALIB_REC_AVI_CYC,

MEDIALIB_REC_3GP,

MEDIALIB_REC_AVI_SEGMENT

}T_eMEDIALIB_REC_TYPE;

结构名	T_eMEDIALIB_AUDIO_CODE	
定义概述	媒体录制文件类型	
成员说明	MEDIALIB_REC_AVI_NORMAL	录像保存为 avi 文件格式
	MEDIALIB_REC_AVI_CYC	avi 循环录像模式，目前已不提供，由上层实现该功能
	MEDIALIB_REC_3GP	录像保存为 3gp 文件格式
	MEDIALIB_REC_AVI_SEGMENT	分段录像模式，目前已不提供，与循环录像类似由上层实现该功能

4.2.3 媒体音频类型

typedef enum

{

MEDIALIB_AUDIO_UNKNOWN,

MEDIALIB_AUDIO_AMR,

MEDIALIB_AUDIO_MP3,

MEDIALIB_AUDIO_AAC,

MEDIALIB_AUDIO_PCM,

MEDIALIB_AUDIO_WMA,

MEDIALIB_AUDIO_MIDI,

MEDIALIB_AUDIO_ADPCM,

MEDIALIB_AUDIO_APE,

MEDIALIB_AUDIO_FLAC,

MEDIALIB_AUDIO_RA,

MEDIALIB_AUDIO_VORBIS,

MEDIALIB_AUDIO_AC3,

MEDIALIB_AUDIO_G711,

MEDIALIB_AUDIO_SPEEX

}T_eMEDIALIB_AUDIO_CODE;

结构名	T_eMEDIALIB_AUDIO_CODE	
定义概述	媒体音频类型	
成员说明	MEDIALIB_AUDIO_UNKNOWN	未知类型
	MEDIALIB_AUDIO_AMR	AMR
	MEDIALIB_AUDIO_MP3	MP3
	MEDIALIB_AUDIO_AAC	AAC
	MEDIALIB_AUDIO_PCM	PCM
	MEDIALIB_AUDIO_WMA	WMA
	MEDIALIB_AUDIO_MIDI	MIDI
	MEDIALIB_AUDIO_ADPCM	ADPCM
	MEDIALIB_AUDIO_APE	APE
	MEDIALIB_AUDIO_FLAC	Flac
	MEDIALIB_AUDIO_RA	RealAudio
	MEDIALIB_AUDIO_VORBIS	Vorbis
	MEDIALIB_AUDIO_AC3	AC3
	MEDIALIB_AUDIO_G711	G.711
	MEDIALIB_AUDIO_SPEEX	Speex

4.2.4 媒体视频类型

typedef enum

{

MEDIALIB_VIDEO_UNKNOWN,

MEDIALIB_VIDEO_MPEG4,

MEDIALIB_VIDEO_H263,

MEDIALIB_VIDEO_WMV,

MEDIALIB_VIDEO_FLV263,

MEDIALIB_VIDEO_H264,

MEDIALIB_VIDEO_RV,

MEDIALIB_VIDEO_MJPEG,

MEDIALIB_VIDEO_MPEG2

}T_eMEDIALIB_VIDEO_CODE;

结构名	T_eMEDIALIB_VIDEO_CODE	
定义概述	媒体视频类型	
成员说明	MEDIALIB_VIDEO_UNKNOWN	未知类型
	MEDIALIB_VIDEO_MPEG4	MPEG4
	MEDIALIB_VIDEO_H263	H263
	MEDIALIB_VIDEO_WMV	WMV
	MEDIALIB_VIDEO_FLV263	FLV263
	MEDIALIB_VIDEO_H264	H264
	MEDIALIB_VIDEO_RV	RealVideo
	MEDIALIB_VIDEO_MJPEG	Motion JPEG
	MEDIALIB_VIDEO_MPEG2	MPEG2

4.2.5 媒体回调函数结构

结构体定义见 5.5，简要说明见下表，具体回调函数定义见第 5 章节。

结构名	T_MEDIALIB_CB	
定义概述	媒体回调函数结构	
成员说明	m_FunPrintf	打印回调
	m_FunRead	读文件回调
	m_FunWrite	写文件回调
	m_FunSeek	Seek 文件回调,set pos
	m_FunTell	Tell 文件回调,get pos
	m_FunMalloc	申请内存回调
	m_FunFree	释放内存回调
	m_FunLoadResource	载入资源
	m_FunReleaseResource	释放资源
	m_FunRtcDelay	延时

结构名	T_MEDIALIB_CB	
	m_FunDMAMemcpy	DMA 方式内存拷贝
	m_FunMMUInvalidateDCache	刷新 DCache
	m_FunCleanInvalidateDcache	刷新并清空 Dcache
	m_FunCheckDecBuf	检测 buffer 是否正在显示
	m_FunFileSysIsBusy	检验是否文件系统忙
	m_FunFileHandleExist	文件句柄存在否
	m_FunFileGetLength	获取文件长度
	m_FunDMAMalloc	DMA 方式 Malloc
	m_FunDMAFree	DMA 方式 free
	m_FunVaddrToPaddr	虚地址到物理地址
	m_FunMapAddr	映射地址
	m_FunUnmapAddr	不映射地址
	m_FunRegBitsWrite	写寄存器位
	m_FunVideoHWLock	锁定视频硬件模块
	m_FunVideoHWUnlock	解锁视频硬件模块

4.3 音视频编解码结构体

4.3.1 音频编解码类型

typedef enum

{

_SD_MEDIA_TYPE_UNKNOWN ,

_SD_MEDIA_TYPE_MIDI ,

_SD_MEDIA_TYPE_MP3 ,

_SD_MEDIA_TYPE_AMR ,

_SD_MEDIA_TYPE_AAC ,

_SD_MEDIA_TYPE_WMA ,

_SD_MEDIA_TYPE_PCM ,

_SD_MEDIA_TYPE_ADPCM_IMA ,

_SD_MEDIA_TYPE_ADPCM_MS ,

_SD_MEDIA_TYPE_ADPCM_FLASH ,

_SD_MEDIA_TYPE_APE ,

_SD_MEDIA_TYPE_FLAC ,

_SD_MEDIA_TYPE_OGG_FLAC ,

_SD_MEDIA_TYPE_RA8LBR ,

_SD_MEDIA_TYPE_DRA ,

_SD_MEDIA_TYPE_OGG_VORBIS,

_SD_MEDIA_TYPE_AC3,

_SD_MEDIA_TYPE_PCM_ALAW,

_SD_MEDIA_TYPE_PCM_ULAW,

_SD_MEDIA_TYPE_SPEEX,

}T_AUDIO_TYPE;

结构名	T_AUDIO_TYPE	
定义概述	音频编码类型	
成员说明	_SD_MEDIA_TYPE_UNKNOWNNOWN	未知类型
	_SD_MEDIA_TYPE_MIDI	Midi
	_SD_MEDIA_TYPE_MP3	MP3
	_SD_MEDIA_TYPE_AMR	AMR
	_SD_MEDIA_TYPE_AAC	AAC
	_SD_MEDIA_TYPE_WMA	WMA
	_SD_MEDIA_TYPE_PCM	PCM
	_SD_MEDIA_TYPE_ADPCM_IMA	ADPCM_IM
	_SD_MEDIA_TYPE_ADPCM_MS	ADPCM_MS
	_SD_MEDIA_TYPE_ADPCM_FLASH	ADPCM_FL
	_SD_MEDIA_TYPE_ APE	APE
	_SD_MEDIA_TYPE_NES	NES
	_SD_MEDIA_TYPE_FLAC	FLAC
	_SD_MEDIA_TYPE_OGG_FLAC	OGG FLAC
	_SD_MEDIA_TYPE_RA8LBR	RA8LBR
	_SD_MEDIA_TYPE_DRA	DRA

结构名	T_AUDIO_TYPE	
	_SD_MEDIA_TYPE_OGG_VORBIS	OGG VORBIS
	_SD_MEDIA_TYPE_AC3	AC3
	_SD_MEDIA_TYPE_PCM_ALAW	PCM_ALAW
	_SD_MEDIA_TYPE_PCM_ULAW	PCM_ULAW
	_SD_MEDIA_TYPE_SPEEX	SPEEX

4.3.2 视频编解码类型

typedef enum

{

VIDEO_DRV_UNKNOWN = 0,

VIDEO_DRV_H263,

VIDEO_DRV_MPEG,

VIDEO_DRV_FLV263,

VIDEO_DRV_H264,

VIDEO_DRV_RV,

VIDEO_DRV_AVC1,

VIDEO_DRV_MJPEG,

VIDEO_DRV_MPEG2,

VIDEO_DRV_H264DMX

}T_eVIDEO_DRV_TYPE;

结构名	T_eVIDEO_DRV_TYPE	
定义概述	视频编码类型	
成员说明	VIDEO_DRV_UNKNOWN	未知类型
	VIDEO_DRV_H263	H263
	VIDEO_DRV_MPEG	MPEG
	VIDEO_DRV_FLV263	FLV263
	VIDEO_DRV_H264	H264
	VIDEO_DRV_RV	RV
	VIDEO_DRV_AVC1	AVC1

结构名	T_eVIDEO_DRV_TYPE	
	VIDEO_DRV_MJPEG	MJPEG
	VIDEO_DRV_MPEG2	MPEG2
	VIDEO_DRV_H264DMX	H264 经过处理的码流

4.3.3 音频解码输出结构体

```
typedef struct
{
        T_U32   m_SampleRate;   //sample rate, sample per second
        T_U16   m_Channels;     //channel number
        T_U16   m_BitsPerSample; //bits per sample

        T_U32   m_ulSize;
        T_U32   m_ulDecDataSize;
        T_U8    *m_pBuffer;
}T_AUDIO_DECODE_OUT;
```

结构名	T_AUDIO_DECODE_OUT	
定义概述	音频缓冲区结构体	
成员说明	m_SampleRate	音频采样率
	m_Channels	声道数
	m_BitsPerSample	每个采样点的比特数
	m_ulSize	缓冲区实际大小
	m_ulDecDataSize	解码出的音频数据大小
	m_pBuffer	Buffer 指针

4.3.4 视频解码输出结构体

```
typedef struct
{
        T_U32   m_ulSize;
        T_U16   m_uDispWidth;
        T_U16   m_uDispHeight;
```

```
    T_U8    *m_pBuffer;
    T_U16   m_uOriWidth;            //原始宽，不一定是 16 的倍数
    T_U16   m_uOriHeight;           //原始高，不一定是 16 的倍数
    T_U8    *m_pBuffer_u;
    T_U8    *m_pBuffer_v;
}T_VIDEO_DECODE_OUT;
```

结构名	T_VIDEO_DECODE_OUT	
定义概述	视频缓冲区结构体	
成员说明	m_ulSize	缓冲区实际大小
	m_uDispWidth	用于显示的图像宽（16 的倍数）
	m_uDispHeight	用于显示的图像高（16 的倍数）
	m_pBuffer	亮度 Y 的 Buffer 指针
	m_uOriWidth	原始宽，不一定是 16 的倍数
	m_uOriHeight	原始高，不一定是 16 的倍数
	m_pBuffer_u	色度 U 的 Buffer 指针
	m_pBuffer_v	色度 V 的 Buffer 指针

4.4 解析相关结构体

4.4.1 Meta信息结构体

```
typedef struct _T_MEDIALIB_META_TYPE_INFO
{
    T_U8    VersionType;    //0: unicode; 1: non-unicode
    T_U8    TitleType;
    T_U8    ArtistType;
    T_U8    AlbumType;
    T_U8    YearType;
    T_U8    CommentType;
    T_U8    GenreType;
    T_U8    TrackType;
    T_U8    ComposerType;
```

```
}T_MEDIALIB_META_TYPE_INFO;      //meta 类型列表，标识每种 meta 信息是否为 unicode

typedef struct _T_MEDIALIB_META_SIZE_INFO
{
        T_U16   uVersionLen;
        T_U16   uTitleLen;
        T_U16   uArtistLen;
        T_U16   uAlbumLen;
        T_U16   uYearLen;
        T_U16   uCommentLen;
        T_U16   uGenreLen;
        T_U16   uTrackLen;
        T_U16   uComposerLen;
}T_MEDIALIB_META_SIZE_INFO;      //meta 长度列表，以字节为单位

typedef struct _MEDIALIB_META_CONTENT
{
        T_VOID          *pVersion;      //metainfo version
        T_VOID          *pTitle;        //Title
        T_VOID          *pArtist;       //Artist
        T_VOID          *pAlbum;        //Album
        T_VOID          *pYear;         //Year
        T_VOID          *pComment;      //Comment
        T_VOID          *pGenre;        //Genre
        T_VOID          *pTrack;        //Track
        T_VOID          *pComposer;     //Composer
}T_MEDIALIB_META_CONTENT;        //meta 内容列表

typedef struct _MEDIALIB_META_INFO
{
        T_MEDIALIB_META_TYPE_INFO       m_MetaTypeInfo;
        T_MEDIALIB_META_SIZE_INFO       m_MetaSizeInfo;
        T_MEDIALIB_META_CONTENT         m_MetaContent;
```

}T_MEDIALIB_META_INFO;

结构名	T_MEDIALIB_META_INFO	
定义概述	Meta 信息结构体	
成员说明	m_MetaTypeInfo	各 meta 信息的类型列表，标识每种 meta 信息是否为 unicode，0: unicode; 1: non-unicode
	m_MetaSizeInfo	各 meta 信息长度列表，以字节为单位
	m_MetaContent	各 meta 信息的内容列表

4.5 接口函数列表

4.5.1 MediaLib_GetVersion

原　　型	const T_CHR *MediaLib_GetVersion(T_VOID)	
功能概述	获取媒体库版本号	
参数说明		
返回值说明	返回版本号信息	
返回值说明	版本号字符串	
注意事项	无	
调用示例		

4.5.2 MediaLib_Init

原　　型	T_BOOL MediaLib_Init(const T_MEDIALIB_INIT_INPUT *init_input, const T_MEDIALIB_INIT_CB *init_cb_fun)	
功能概述	初始化媒体库，全局资源初始化	
参数说明	init_input	初始化输入信息：其中芯片类型必须正确赋值，否则可能导致死机
	init_cb_fun	见 T_MEDIALIB_INIT_CB
返回值说明	初始化的结果	
返回值说明	AK_TRUE　　初始化成功 AK_FALSE　　初始化失败	

原　　型	T_BOOL MediaLib_Init(const T_MEDIALIB_INIT_INPUT *init_input, const T_MEDIALIB_INIT_CB *init_cb_fun)
注意事项	1. 在系统启动时必须首先调用本函数，整个系统运行过程中必须并且只能调用一次，后续调用会返回初始化失败，但只要第一次调用返回成功，则可正常使用媒体库各项功能 2. 在执行该函数时必须提供相应的回调函数
调用示例	见典型调用示例

4.5.3 MediaLib_Destroy

原　　型	T_VOID MediaLib_Destroy(T_VOID)	
功能概述	全局资源释放	
参数说明	-	-
返回值说明	-	
注意事项	-	
调用示例	见典型调用示例	

5 依赖接口说明

各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功能接口，该类函数由目标平台实现。

5.1 资源管理相关接口

资源管理模块提供文件或缓冲区的读写、seek 等操作。库内部不对文件和缓冲区播放进行区分，而是通过提供不同的资源管理接口来屏蔽两者的区别。

■ 读数据

类　型	typedef T_S32 (*MEDIALIB_CALLBACK_FUN_READ)(T_S32 hFile, T_pVOID buf, T_S32 size);	
功能概述	从资源中读取数据到 buffer 中	
参数说明	hFile	需要读取数据的资源
	buf	读取的数据放入此缓冲中
	size	从资源中读取的字节数
返回值说明	返回 －1 说明不成功，其他值说明成功，为实际读取的数量。	
返回值说明	-	
注意事项	-	
调用示例	-	

■ 写数据

类　型	typedef T_S32 (*MEDIALIB_CALLBACK_FUN_WRITE)(T_S32 hFile, T_pVOID buf, T_S32 size);	
功能概述	把 buffer 中的数据写入资源中	
参数说明	hFile	需要写入数据的资源
	buf	需要写入的数据放在此 buffer 中
	size	需要写入的数据字节数
返回值说明	返回 －1 说明不成功，其他值说明成功，为实际写入的数量。	
注意事项	-	
调用示例	-	

■ 资源定位

类　型	typedef T_S32 (*MEDIALIB_CALLBACK_FUN_SEEK)(T_S32 hFile, T_S32 offset, T_S32 whence);	
功能概述	定位到资源的某一地方	
参数说明	hFile	需要定位的资源
	offset	相对基地址的偏移量，正数向后，负数向前
	whence	定位基位置,偏移量是相对此基位置（资源开始 0/当前位置 1/资源结束 2）
返回值说明	定位到资源的实际位置	
注意事项	注意该回调函数需要的返回值是定位后相对于资源起始的偏移； 因为有些系统或平台的 seek 返回值是返回成功或失败，特提醒注意。	
调用示例	-	

■ 取资源当前位置

类　型	typedef T_S32 (*MEDIALIB_CALLBACK_FUN_TELL)(T_S32 hFile);	
功能概述	取得资源的当前位置	
参数说明	hFile	资源句柄
返回值说明	得到资源的当前位置，为-1 说明失败，其他值则为资源的当前位置	
注意事项	-	
调用示例	-	

■ 资源是否处于忙的状态

类　型	typedef T_BOOL (*MEDIALIB_CALLBACK_FUN_FILESYS_ISBUSY)(T_VOID);	
功能概述	获取资源是否处于忙的状态	
参数说明	-	-
返回值说明	为 TRUE 表示当前资源忙，无法读取数据	
注意事项	-	
调用示例	-	

■ 检查资源是否存在

类　型	typedef T_S32 (*MEDIALIB_CALLBACK_FUN_FILE_HANDLE_EXIST) (T_S32 hFile);	
功能概述	检查资源是否存在	
参数说明	hFile	资源句柄
返回值说明	AK_TRUE 表示当前资源存在 AK_FALSE 表示当前资源不存在	
注意事项	如果该回调设置为空，则库内部不对资源有效性进行判断，需要上层对资源句柄进行有效性检查	
调用示例	-	

■ 获取资源的长度

类　型	typedef T_U32 (*MEDIALIB_CALLBACK_FUN_FILE_GET_LENGTH)(T_S32 hFile);	
功能概述	获取资源总长度，以字节为单位	
参数说明	hFile	资源句柄
返回值说明	资源长度，如果资源长度未知，可返回(T_U32)(-1)。	
注意事项	如果该回调设置为空，库内部会通过 seek 到资源尾部的方式来获取文件长度，这样可能造成一定的时间损耗	
调用示例	-	

5.2 内存管理相关的接口

■ 申请内存

类　型	typedef T_pVOID (*MEDIALIB_CALLBACK_FUN_MALLOC)(T_U32 size)	
功能概述	申请内存	
参数说明	size	需要申请的内存的字节数
返回值说明	申请成功，则返回内存的地址，如不成功，则返回 NULL	
注意事项	-	
调用示例	-	

■ 释放内存

类　型	typedef T_VOID (*MEDIALIB_CALLBACK_FUN_FREE)(T_pVOID mem);	
功能概述	释放申请的内存空间	
参数说明	mem	申请的内存空间的首地址
返回值说明	返回 NULL 指针，方便对指针赋值成 NULL.	
注意事项	-	
调用示例	-	

■ 用 DMA 方式复制内存内容，功能等同与标准 C 的 memcpy()，目前已不再使用

类　型	typedef T_pVOID (*MEDIALIB_CALLBACK_FUN_DMA_MEMCPY)(T_pVOID dest, T_pCVOID src, T_U32 size);	
功能概述	使用 DMA 机制进行复制	
参数说明	dst	指向目标内存的指针
	src	指向源内存的指针
	size	复制的字节数
返回值说明	不使用标准 C 的 memcpy()是因为 DMA 复制能显著提高视频播放的性能	
注意事项	目前基本不使用，一般赋为 NULL 即可	
调用示例	-	

■ 刷新 cache

类　型	typedef T_VOID (*MEDIALIB_CALLBACK_FUN_MMU_INVALIDATEDCACHE)(T_VOID);	
功能概述	刷新 cache，包括读写 cache	
参数说明	-	-
返回值说明	-	
注意事项	-	
调用示例	-	

■ 申请物理连续的内存

类　型	typedef T_pVOID (*MEDIALIB_CALLBACK_FUN_DMA_MALLOC)(T_U32 size)	
功能概述	申请物理连续的内存	
参数说明	size	需要申请的内存的字节数

类　　型	typedef T_pVOID (*MEDIALIB_CALLBACK_FUN_DMA_MALLOC)(T_U32 size)	
返回值说明	申请成功，则返回内存的地址，如不成功，则返回 NULL	
注意事项	-	
调用示例	-	

■ 释放物理连续的内存

类　　型	typedef T_VOID (*MEDIALIB_CALLBACK_FUN_DMA_FREE)(T_pVOID mem)	
功能概述	释放申请的物理连续的内存空间	
参数说明	mem	申请的内存空间的首地址
返回值说明	返回 NULL 指针，方便对指针赋值成 NULL.	
注意事项	-	
调用示例	-	

■ 虚实地址转换

类　　型	typedef T_pVOID (*MEDIALIB_CALLBACK_FUN_VADDR_TO_PADDR)(T_pVOID mem)	
功能概述	将虚地址转为实地址	
参数说明	mem	虚地址指针
返回值说明	实地址指针	
注意事项	-	
调用示例	-	

■ map 寄存器地址

类　　型	typedef T_U32 (*MEDIALIB_CALLBACK_FUN_MAP_ADDR)(T_U32 phyAddr, T_U32 size)	
功能概述	Map 寄存器地址	
参数说明	phyAddr	需要 map 的物理首地址
	size	需要 map 的字节数
返回值说明	Map 后的地址	
注意事项	-	
调用示例	-	

■ unmap 地址

类　型	typedef T_VOID (*MEDIALIB_CALLBACK_FUN_UNMAP_ADDR)(T_U32 addr, T_U32 size)	
功能概述	Unmap 地址	
参数说明	addr	Map 函数返回的地址
	size	Map 时对应的字节数
返回值说明	-	
注意事项	-	
调用示例	-	

■ 按位写寄存器

类　型	typedef T_VOID (*MEDIALIB_CALLBACK_FUN_REG_BITS_WRITE)(T_U32 phyAddr, T_U32 val, T_U32 mask)	
功能概述	Map 物理地址成虚地址	
参数说明	phyAddr	寄存器实地址
	val	需写入的值
	mask	Mask
返回值说明	-	
注意事项	实现参考： *(volatile T_U32 *)(phyAddr) &= ~mask; *(volatile T_U32 *)(phyAddr) |= (val & mask);	
调用示例	如需对寄存器 0x20000000 比特位 bit20；bit22 进行操作，bit20 写入 0，bit22 写入 1；那么传入参数(reg, ((1<<22) |(0<<20)), ((1<<22)|(1<<20)))	

5.3 互斥量相关的接口

在多个线程中调用 video_stream_lib.h 中的接口分别进行输入码流数据和视频解码时使用，其他情况下，本小节中的回调函数可赋值为 NULL。

■ 创建互斥量

类　型	typedef T_pVOID (*MEDIALIB_CALLBACK_FUN_MUTEX_CREATE)(T_VOID);
功能概述	创建互斥量

类　　型	typedef T_pVOID (*MEDIALIB_CALLBACK_FUN_MUTEX_CREATE)(T_VOID);	
参数说明	-	-
返回值说明	NULL：创建失败；其他：创建的互斥量句柄	
注意事项	-	
调用示例	-	

■ 互斥锁定

类　　型	typedef T_S32 (*MEDIALIB_CALLBACK_FUN_MUTEX_LOCK)(T_pVOID pMutex, T_S32 nTimeOut);	
功能概述	互斥锁定	
参数说明	pMutex	创建互斥量函数的返回值
	nTimeOut	超时模式，输入-1 表示不进行超时判断，阻塞式锁定；其他值：以毫秒为单位的超时锁定
返回值说明	1：锁定成功；0：超时	
注意事项	目前该回调在 android 平台上使用，其他平台可设置为 NULL	
调用示例	-	

■ 互斥解锁

类　　型	typedef T_S32 (*MEDIALIB_CALLBACK_FUN_MUTEX_UNLOCK)(T_pVOID pMutex);	
功能概述	互斥解锁	
参数说明	pMutex	创建互斥量函数的返回值
返回值说明	1：解锁成功；0：解锁失败	
注意事项	-	
调用示例	-	

■ 释放互斥量

类　　型	typedef T_VOID (*MEDIALIB_CALLBACK_FUN_MUTEX_RELEASE)(T_pVOID pMutex);	
功能概述	释放互斥量	
参数说明	pMutex	创建互斥量函数的返回值
返回值说明	-	

类　型	typedef T_VOID (*MEDIALIB_CALLBACK_FUN_MUTEX_RELEASE)(T_pVOID pMutex);
注意事项	-
调用示例	-

5.4 其他

■ 打印函数

类　型	typedef T_S32 (*MEDIALIB_CALLBACK_FUN_PRINTF)(T_pCSTR format, ...);	
功能概述	释放申请的内存空间	
参数说明	mem	申请的内存空间的首地址
返回值说明	-	
注意事项	-	
调用示例	-	

■ 载入资源

```
typedef struct
{
    T_U16   ResourceID;     //资源 ID 号
    T_U8    *Buff;          //资源首地址
    T_U32   Resource_len;   //资源长度
}T_AUDIO_LOADRESOURCE_CB_PARA;
```

类　型	typedef T_VOID (*MEDIALIB_CALLBACK_FUN_LOADRESOURCE) (T_AUDIO_LOADRESOURCE_CB_PARA *pPara);	
功能概述	载入资源	
参数说明	pPara	要载入的资源地址
返回值说明	-	
注意事项	目前基本不使用，一般赋为 NULL 即可	
调用示例	-	

■ 释放载入的资源

类　型	typedef T_VOID (*MEDIALIB_CALLBACK_FUN_RELEASERESOURCE)(T_U8 *Buff);	
功能概述	释放载入的资源	
参数说明	Buff	要释放的资源地址
返回值说明	-	
注意事项	目前基本不使用，一般赋为 NULL 即可	
调用示例	-	

■ 延时函数

类　型	typedef T_BOOL (*MEDIALIB_CALLBACK_FUN_RTC_DELAY) (T_U32 ulTicks);	
功能概述	延时或 sleep	
参数说明	ulTicks	一个 tick 是 5ms
返回值说明	-	
注意事项	-	
调用示例	-	

■ 检测某地址是否可以用于解码

类　型	typedef T_BOOL (*MEDIALIB_CALLBACK_FUN_CHECK_DEC_BUF)(T_pDATA pBuf);	
功能概述	检测 buffer 是否可以用于解码	
参数说明	pBuf	待检测的地址
返回值说明	返回 FALSE 表示该 buffer 正在显示，暂时不能用于解码	
注意事项	-	

■ 硬件模块锁定

类　型	typedef T_pVOID (*MEDIALIB_CALLBACK_FUN_VIDEO_HW_LOCK)(T_S32 hw_id);	
功能概述	锁定硬件资源	
参数说明	hw_id	表示哪个硬件(用于扩展)
	-	-
返回值说明	返回被锁定的硬件资源	

类　型	typedef T_pVOID (*MEDIALIB_CALLBACK_FUN_VIDEO_HW_LOCK)(T_S32 hw_id);
注意事项	目前该回调在 android/linux 平台上使用，其他平台可设置为 NULL
调用示例	

■ 硬件模块解锁

类　型	typedef T_S32 (*MEDIALIB_CALLBACK_FUN_VIDEO_HW_UNLOCK)(T_pVOID hLock);	
功能概述	解锁硬件资源	
参数说明	hLock	硬件模块锁定函数的返回值
返回值说明	-	
注意事项	目前该回调在 android/linux 平台上使用，其他平台可设置为 NULL	
调用示例	-	

5.5 回调函数定义

```
typedef struct
{
    MEDIALIB_CALLBACK_FUN_PRINTF                m_FunPrintf;

    MEDIALIB_CALLBACK_FUN_READ                  m_FunRead;
    MEDIALIB_CALLBACK_FUN_WRITE                 m_FunWrite;
    MEDIALIB_CALLBACK_FUN_SEEK                  m_FunSeek;
    MEDIALIB_CALLBACK_FUN_TELL                  m_FunTell;
    MEDIALIB_CALLBACK_FUN_MALLOC                m_FunMalloc;
    MEDIALIB_CALLBACK_FUN_FREE                  m_FunFree;

    MEDIALIB_CALLBACK_FUN_LOADRESOURCE          m_FunLoadResource;
    MEDIALIB_CALLBACK_FUN_RELEASERESOURCE       m_FunReleaseResource;

    MEDIALIB_CALLBACK_FUN_RTC_DELAY             m_FunRtcDelay;
    MEDIALIB_CALLBACK_FUN_DMA_MEMCPY            m_FunDMAMemcpy;
    MEDIALIB_CALLBACK_FUN_MMU_INVALIDATEDCACHE  m_FunMMUInvalidateDCache;
    MEDIALIB_CALLBACK_FUN_MMU_INVALIDATEDCACHE  m_FunCleanInvalidateDcache;
    MEDIALIB_CALLBACK_FUN_CHECK_DEC_BUF         m_FunCheckDecBuf;
```

```
        MEDIALIB_CALLBACK_FUN_FILESYS_ISBUSY          m_FunFileSysIsBusy;

        //add for file handle
        MEDIALIB_CALLBACK_FUN_FILE_HANDLE_EXIST       m_FunFileHandleExist;
        MEDIALIB_CALLBACK_FUN_FILE_GET_LENGTH         m_FunFileGetLength;

        //add for Linux and CE
        MEDIALIB_CALLBACK_FUN_DMA_MALLOC              m_FunDMAMalloc;
        MEDIALIB_CALLBACK_FUN_DMA_FREE                m_FunDMAFree;
        MEDIALIB_CALLBACK_FUN_VADDR_TO_PADDR          m_FunVaddrToPaddr;
        MEDIALIB_CALLBACK_FUN_MAP_ADDR                m_FunMapAddr;
        MEDIALIB_CALLBACK_FUN_UNMAP_ADDR              m_FunUnmapAddr;
        MEDIALIB_CALLBACK_FUN_REG_BITS_WRITE          m_FunRegBitsWrite;

        //add for hardware mutex
        MEDIALIB_CALLBACK_FUN_VIDEO_HW_LOCK           m_FunVideoHWLock;
        MEDIALIB_CALLBACK_FUN_VIDEO_HW_UNLOCK         m_FunVideoHWUnlock;
}T_MEDIALIB_CB;
```

对于与硬件无关的接口调用，如 media_demuxer_lib、mobiletv_recorder_lib，只需要对上述红色部分的回调函数赋值即可。

6 常见问题

1、看不到打印信息

请首先确认串口连接正常，串口调试工具正常接收打印信息；然后请查调用传递给信息打印函数指针所指的函数能正常运作；打印级别是否太低导致无法输出。调试过程中建议输出所有打印信息方便调试，正常使用后再把函数指针赋为 AK_NULL 关闭打印信息。

2、Open 过程中死机

有可能的一个原因是某些回调没有初始化，或者必须设置的回调为 NULL。首先请检查输入结构体在定义后是否进行了初始化的操作，使用 memset 对整个结构体清零即可，这样即使结构体中增加了变量，上层应用也可以在不修改的情况下正常运行，库内部对于新增的结构体变量在使用时会进行判断；其次检查是否所有必需的回调函数都进行了正确的赋值。