全志科技
Allwinner Technology

音视频解码库项目

Recoder API 说明

文档履历

版本号	日期	制/修订人	内容描述
V0.1	2015-11-24		初稿
V0.2	2016-1-20		把编码器部分独立出来，调整接口
V0.3	2016-2-16		把编码数据放到应用去做格式封装、调整一些接口
V1.0	2016-11-10		更新文档,添加 CdxWriter 接口说明，把文档纳入 CedarX 2.7

目 录

1. 概述

1.1. 编写目的

设计 Cedarx 2.0 Recoder 部分的应用访问接口，指导具体应用的开发、使用和后续维护。

1.2. 适用范围

使用 Cedarx 2.0 有硬件编码 VE 模块的芯片。

1.3. 相关人员

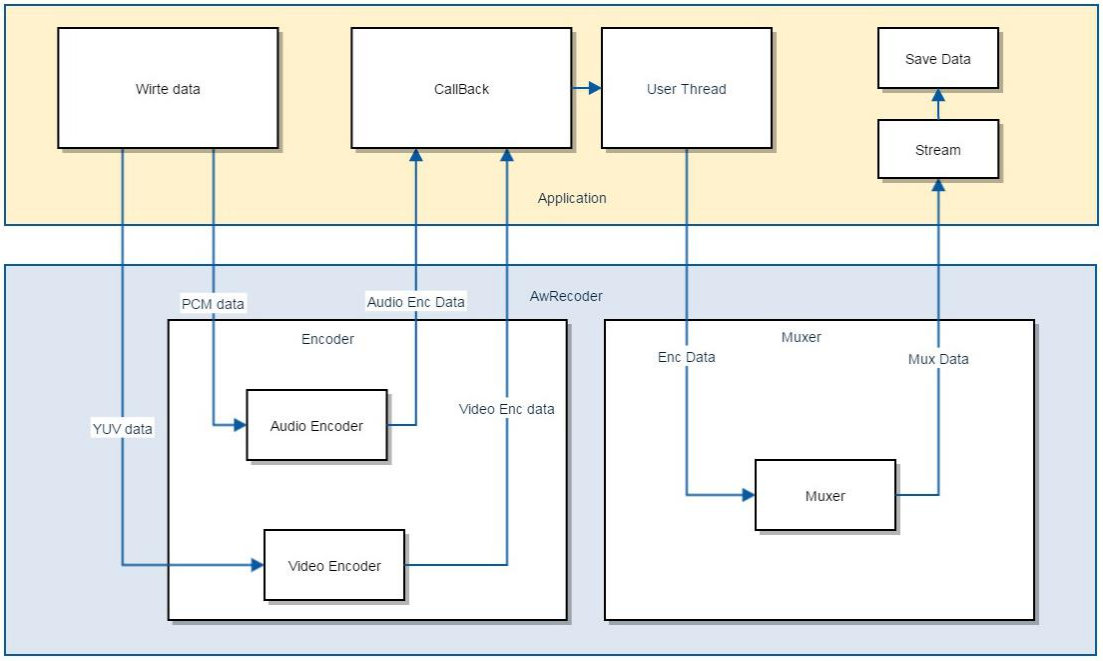
相应的开发者和维护者。

2. 模块介绍

2.1. 功能介绍

针对要求编码并且保存到需求的场景，提供访问硬件编解码的接口和 Muxer 的接口。

编码处理流程如下图：



2.2. 相关术语介绍

2

3. 编码器 awrecorder 接口

3.1. AwEncoderCreate

函数原型	AwEncoder*　　AwEncoderCreate(void * app)
功能	创建编码器；
参数	app 应用句柄，用于回调函数做数据传输的句柄
返回值	成功：返回编码器句柄 失败：NULL
调用说明	NA

3.2. AwEncoderDestory

函数原型	void AwRecorderDestory(AwRecorder* p)
功能	销毁编码器；
参数	p：　编码器句柄
返回值	
调用说明	NA

3.3. AwEncoderInit

函数原型	int　AwEncoderInit(AwEncoder* p, VideoEncodeConfig *videoConfig, AudioEncodeConfig *audioConfig,EncDataCallBackOps *ops)
功能	初始化音视频编码信息、设置数据回调接口；
参数	p: 编码器句柄 videoConfig: 视频编码器配置 audioConfig：音频编码器配置 ops: 编码数据回调注册句柄
返回值	成功：0 失败：-1
调用说明	在 AwEncoderCreate 之后调用，输入输出宽高以及编码类型，比特率、帧率；其他参数如果不设置，编码器会使用默认值，音频编后数据的 pts 会根据输入数据采样率和通道数计算，当音频设置为 NULL 的时候不编码音频。 ops 里面的两个函数根据需要在应用注册，编码好的数据编码器会通过回调出来，应用层再根据需要去做数据的格式封装。

3.4. AwEncoderGetExtradata

函数原型	int　AwEncoderGetExtradata(AwEncoder* v, unsigned char** buf, unsigned int* length)
功能	获取编码器的 Extradata
参数	p：　编码器句柄
返回值	成功：0 失败：-1
调用说明	需要在调用 AwEncoderInit 后才能调用

3.5. AwEncoderStart

函数原型	Int AwRecorderStart(AwRecorder* p)
功能	开始编码;
参数	p： 编码器句柄
返回值	成功：0 失败：-1
调用说明	NA

3.6. AwEncoderSetParamete

函数原型	int AwRecorderSetParamete(AwRecorder* p,AwRecorderParameteIndex nIndex, void* para);
功能	设置参数;
参数	P:编码器句柄 nIndex：设置的参数参数值 Para：参数
返回值	成功：0 失败：-1
调用说明	该函数可以在编码过程中动态设置帧率、码率等信息，后续会扩展

3.7. AwEncoderStop

函数原型	Int AwRecorderStop(AwRecorder* p)
功能	停止编码;
参数	p： 编码器句柄
返回值	成功：0 失败：-1
调用说明	NA

3.8. AwEncoderReset

函数原型	Int AwRecorderReset(AwRecorder* p)
功能	重置编码器;
参数	p： 编码器句柄
返回值	成功：0 失败：-1
调用说明	NA

3.9. AwEncoderWriteYUVdata

函数原型	Int AwRecorderWriteYUVdata(AwRecorder* p, VideoInputBuffer* buf)
功能	写入视频 yuv 数据;
参数	p： 编码器句柄 buf: 写入数据指针
返回值	成功：返回写入数据长度 失败：0

调用说明	当写入数据超出底层 buffer 大小，将无法继续写入数据，是否丢弃该数据由应用决定；

3.10. AwEncoderWritePCMdata

函数原型	int AwEncoderWritePCMdata(AwEncoder* p, AudioInputBuffer* buf)
功能	写入音频数据；
参数	p： 编码器句柄 buf: 写入数据指针
返回值	成功：返回写入数据长度 失败：0
调用说明	当写入数据超出底层 buffer 大小，将无法继续写入数据，是否丢弃该数据由应用决定

3.11. AwEncoderSetNotifyCallback

函数原型	int AwRecorderSetNotifyCallback(AwRecorder* p, NotifyCallback notifier, void* pUserData);
功能	设置消息回调函数；
参数	p： 编码器句柄 Notifier：消息回调函数（由应用实现） pUserData：回调消息处理对象
返回值	成功：0 失败：-1
调用说明	NA

4. Muxer 接口

4.1. CdxMuxerCreate

函数原型	CdxMuxerT *CdxMuxerCreate(CdxMuxerTypeT type, CdxWriterT *stream);
功能	创建 Muxer；
参数	type：封装类型 stream：IO 流句柄
返回值	成功：返回 Muxer 句柄 失败：NULL
调用说明	NA

4.2. CdxMuxerSetMediaInfo

函数原型	int CdxMuxerSetMediaInfo(CdxMuxerT *mux, CdxMuxerMediaInfoT *mediaInfo);
功能	设置 muxer 的媒体信息
参数	mux：muxer 句柄 mediaInfo：媒体信息
返回值	成功：0 失败：-1
调用说明	NA

4.3. CdxMuxerWriteExtraData

函数原型	int CdxMuxerWriteExtraData(CdxMuxerT *mux, unsigned char* pdata, int data_len, int index);
功能	设置某些编码格式的 ExtraData
参数	mux：muxer 句柄 pdata：数据指针 data_len：数据长度 Index：数据对应的媒体索引
返回值	成功：0 失败：-1
调用说明	NA

4.4. CdxMuxerWriteHeader

函数原型	int CdxMuxerWriteHeader(CdxMuxerT *mux);
功能	写头信息；
参数	mux：muxer 句柄

返回值	成功：0 失败：-1
调用说明	NA

4.5. CdxMuxerWritePacket

函数原型	int CdxMuxerWritePacket(CdxMuxerT *mux, CdxMuxerPacketT *pkt);
功能	写入音视频数据信息
参数	mux: muxer 句柄 pkt: 数据指针
返回值	成功：0 失败：-1
调用说明	NA

4.6. CdxMuxerWriteTrailer

函数原型	int CdxMuxerWriteTrailer(CdxMuxerT *mux);
功能	写尾操作
参数	mux: muxer 句柄
返回值	成功：0 失败：-1
调用说明	NA

4.7. CdxMuxerControl

函数原型	int CdxMuxerControl(CdxMuxerT *mux, int uCmd, void * pParam);
功能	Muxer 的控制操作接口；
参数	mux: muxer 句柄 uCmd: 操作索引 pParam：操作参数
返回值	成功：0 失败：-1
调用说明	NA

4.8. CdxMuxerClose

函数原型	int CdxMuxerClose(CdxMuxerT *mux);
功能	创建编码器；
参数	mux: muxer 句柄
返回值	成功：0

	失败：-1
调用说明	NA

8

5. CdxWriter 接口

CdxWriter 是应用程序根据自己的应用需求去实现对应的数据写入器，CdxMuxer 会通过下列接口去做数据的操作。

5.1. CdxWriterRead

函数原型	int CdxWriterRead(CdxWriterT *w, void *buf, int size)
功能	从 w 里面读取大小为 size 的数据；
参数	w：CdxWriterT 句柄 buf：数据存放地址 size：数据大小
返回值	成功：0 失败：-1

5.2. CdxWriterWrite

函数原型	int CdxWriterRead(CdxWriterT *w, void *buf, int size)
功能	从 w 里面写入大小为 size 的数据；
参数	w：CdxWriterT 句柄 buf：数据句柄 size：数据大小
返回值	成功：0 失败：-1

5.3. CdxWriterSeek

函数原型	int CdxWriterSeek(CdxWriterT *w, long moffset, int mwhere)
功能	从跳到指定位置；
参数	w：CdxWriterT 句柄 moffset：偏移大小 mwhere：offset 起始位置
返回值	成功：0 失败：-1

5.4. CdxWriterTell

函数原型	int CdxWriterRead(CdxWriterT *w)
功能	获取当前的位置；
参数	w：CdxWriterT 句柄
返回值	成功：返回当前位置 失败：-1

9

5.5. CdxWriterClose

函数原型	int CdxWriterClose(CdxWriterT *w)
功能	关闭 writer；
参数	w：CdxWriterT 句柄
返回值	成功：0 失败：-1

5.6. CdxWriterDestroy

函数原型	void CdxWriterRead(CdxWriterT *writer)
功能	释放 writer 资源；
参数	writer：CdxWriterT 句柄
返回值	

6. 数据结构设计

6.1. 视频信息 VideoEncodeConfig

```
typedef struct VideoEncodeConfig
{
    VIDEO_ENCODE_TYPE nType;     //视频编码格式，1663 只支持 JPEG
    Int               nFrameRate;          //视频帧率
    Int               nBitRate;            //视频码率
    Int               nOutWidth;           //编码输出宽度
    int               nOutHeight;          //编码输出高度
    int               nSrcFrameRate;
    int               nSrcWidth;           //输入 YUV 数据宽度
    int               nSrcHeight;          //输入 YUV 数据高度
    int               bUsePhyBuf;          //是否使用物理内存
}VideoEncodeConfig;
```

注意事项：

当使用物理内存传递 YUV 数据时，需要申请使用物理内存，并且要注意物理内存的使用状况，当编码器使用完一帧物理数据是会通过 NotifyCallbackForAwEncorder 回调 AWENCODER_VIDEO_ENCODER_NOTIFY_RETURN_BUFFER 信息给到应用。

6.2. 音频信息 AudioEncodeConfig

```
typedef struct AudioEncodeConfig
{
    AUDIO_ENCODE_TYPE  nType;   //音频编码格式，目前只支持 PCM
    int                nInSamplerate;    //输入采样率
    int                nInChan;          //输入通道数
    int                nBitrate;         //比特率
    int                nSamplerBits;     //位深
    int                nOutSamplerate;   //输出采样率
    int                nOutChan;         //编码输出通道数
    int                nFrameStyle;      //1663 上置 0
}AudioEncodeConfig;
```

6.3. 编码数据回调注册接口

```
typedef struct EncDataCallBackOps
{
    int (*onVideoDataEnc)(void *app,CdxMuxerPacketT *buff);      //编码后的视频数据
    int (*onAudioDataEnc)(void *app,CdxMuxerPacketT *buff);      //编码后的音频数据
} EncDataCallBackOps;
```

这两个函数可以根据需要注册相应的实现即可，如不需要可以把某个函数置为 null

6.4. VideoInputBuffer 写入的视频数据

```
typedef struct VideoInputBuffer
{
    unsigned char*  pData;          //虚拟内存数据地址
    int             nLen;           //虚拟内存数据长度
    long long       nPts;           //pts

    unsigned long   nID;            //输入 buffer 编号
    unsigned char*  pAddrPhyY;      //物理内存 Y 分量数据地址
    unsigned char*  pAddrPhyC;      //物理内存 UV 分量数据地址
}VideoInputBuffer;
```

6.5. AudioInputBuffer 写入的音频数据

```
typedef struct AudioInputBuffer
{
    char        *pData; //数据指针
    int         nLen;   //数据长度
    long long   nPts;    //pts
}AudioInputBuffer;
```

6.6. 封装文件类型 CdxMuxerTypeE

目前只支持 mov（mp4）

```
enum CdxMuxerTypeE
{
    CDX_MUXER_UNKNOW = -1,
    CDX_MUXER_MOV,
    CDX_MUXER_TS,
    CDX_MUXER_AVI,
    CDX_MUXER_AAC,
    CDX_MUXER_MP3,
};
```

6.7. 编码数据 CdxMuxerPacketS

```
struct CdxMuxerPacketS
{
    cdx_void *buf;      //数据指针
    cdx_int32 buflen;   //数据长度
    int64_t   pts;      //当前数据的 pts
    cdx_int64 duration; //当前数据的时长
    cdx_int32 type;     //数据的类型
    cdx_int32 length;   //数据长度
    cdx_int32 streamIndex; //数据的索引
};
```

7. 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.