Linux Encoder
开发指南

版本号: 1.0
发布日期: 2023.07.25

版本历史

版本号	日期	制/修订人	内容描述
1.0	2023.07.25	AWA1908	初始版本。

目　录

1 概述

1.1 编写目的

介绍视频编码库对外 API 接口及相关的数据结构，指导基于视频编码库的开发、使用。

1.2 适用范围

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

1.3 目标读者

基于视频编码库开发和使用的相关人员。

2 模块介绍

2.1 功能介绍

视频编码库是一个提供视频编码功能的库，编译输出的库文件为 libvencoder.so。基于视频编码库，应用程序可以在 Allwinner 的各个 IC 平台上实现高效的、多种压缩格式的视频编码功能，所支持的压缩格式为：JPEG、H264、H265。

2.2 相关术语介绍

- QP：量化参数；
- Exif：可交换图像文件格式，用于记录照片的属性信息和拍摄数据，通常作为 jpeg 图片的附件信息。

3 接口和流程设计

3.1 接口函数

视频编码库 APIs	
VideoEncCreate	创建一个视频编码器
VideoEncDestroy	销毁视频编码器
VideoEncInit	初始化视频编码器
VideoEncUnInit	去初始化视频编码器
AllocInputBuffer	通过 vencoder 申请输入图像帧 buffer
GetOneAllocInputBuffer	获取一块由 vencoder 分配的图像帧
FlushCacheAllocInputBuffer	刷 cache 保持数据的一致性
ReturnOneAllocInputBuffer	还回由 vencoder 申请的图像帧
ReleaseAllocInputBuffer	释放由 vencoder 申请的图像帧
AddOneInputBuffer	添加一块输入的图像帧到编码器
VideoEncodeOneFrame	编码一帧图像
AlreadyUsedInputBuffer	获取编码器已经使用过的图像帧
ValidBitstreamFrameNum	获取有效的输出码流 buffer 的个数
GetOneBitstreamFrame	获取一个码流 buffer
FreeOneBitStreamFrame	还回码流 buffer
VideoEncGetParameter	获取编码器参数
VideoEncSetParameter	设置编码器参数
VideoEncoderReset	重启编码器
VideoEncoderGetUnencodedBufferNum	获取编码器未完成编码的输入 buffer 个数

3.1.1 VideoEncCreate

函数原型	VideoEncoder* VideoEncCreate(VENC_CODEC_TYPE eCodecType)
功能	创建一个视频编码器
参数	eCodecType：创建的编码器 codec 类型
返回值	成功：视频编码器指针；失败：返回 NULL；
调用说明	视频编码器支持创建多个编码器，支持多路编码

3.1.2 VideoEncDestroy

函数原型	void VideoEncDestroy(VideoEncoder* pEncoder)
功能	销毁视频编码器
参数	pEncoder: 通过 VideoEncCreate 函数创建的视频编码器指针
返回值	无
调用说明	无

3.1.3 VideoEncInit

函数原型	int VideoEncInit(VideoEncoder* pEncoder, VencBaseConfig* pConfig)
功能	初始化视频编码器
参数	pEncoder: 通过 VideoEncCreate 函数创建的视频编码器指针 pConfig：编码器基本初始化信息，包括是否做 scaler，颜色格式等
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；nInputWidth：输入图像帧的宽度，以像素为单位；nInputHeight：输入图像帧的高度，以像素为单位；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；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；eInputFormat：输入的颜色格式；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；

3.1.4 VideoEncUnInit

函数原型	int VideoEncUnInit(VideoEncoder* pEncoder)
功能	去初始化视频编码器
参数	pEncoder: 通过 VideoEncCreate 函数创建的视频编码器指针
返回值	成功：返回 0；失败：返回-1；
调用说明	无

3.1.5 AllocInputBuffer

函数原型	int AllocInputBuffer(VideoEncoder* pEncoder, VencAllocateBufferParam *pBufferParam)
功能	通过 vencoder 申请输入图像帧 buffer
参数	pEncoder: 通过 VideoEncCreate 函数创建的视频编码器指针；pBufferParam：指定申请 buffer 的格式和 size；
返回值	成功：返回 0；失败：返回-1；
调用说明	当需要由编码器来提供输入图像帧的 buffer 时，由此接口来申请图像帧 buffer；当外部模块有自己的 buffer 管理模块，并且所使用的 buffer 为物理连续的 buffer 的时候，从效率上考虑可以不使用此接口来申请输入图像帧 buffer，可以直接把相应的 buffer 的物理地址配给 VE，从而可以少一次数据 copy；

3.1.6 GetOneAllocInputBuffer

函数原型	int GetOneAllocInputBuffer(VideoEncoder* pEncoder, VencInputBuffer *pInputbuffer)
功能	获取到的由 AllocInputBuffer 申请的输入图像帧
参数	pEncoder: 通过 VideoEncCreate 函数创建的视频编码器指针；pInputbuffer（输出）：获取到的由 AllocInputBuffer 申请的输入图像帧 buffer；
返回值	成功：返回 0；失败：返回-1；
调用说明	pInputbuffer 的相应变量的说明：nID: 用来区分不同的 buffer；nPts：当前图像帧的时间戳，以 us 为单位；pAddrPhyY：当前图像帧 Y 分量的物理地址，配给硬件使用；pAddrPhyC：当前图像帧的 C 分量的物理地址，配给硬件使用；pAddrVirY：当前图像帧 Y 分量的虚拟地址，可由 CPU 来搬移图像数据到此 buffer；pAddrVirC：当前图像帧 C 分量的虚拟地址，可由 CPU 来搬移图像数据到此 buffer；

3.1.7 FlushCacheAllocInputBuffer

函数原型	Int FlushCacheAllocInputBuffer(VideoEncoder* pEncoder, VencInputBuffer * pInputbuffer)
功能	刷 cache 保存数据的一致性
参数	pEncoder: 通过 VideoEncCreate 函数创建的视频编码器指针；pInputbuffer（输入）：由 AllocInputBuffer 申请的输入图像帧 buffer；
返回值	成功：返回 0；失败：返回-1；
调用说明	当调用 GetOneAllocInputBuffer 获取到由 AllocInputBuffer 申请的输入图像帧 buffer 的时，如果通过 CPU 来搬移输入的图像帧数据到此 buffer，在把此 buffer 送给编码器之前，需要调用此接口来保证 dram 和 cache 中的数据一致性；

3.1.8 ReturnOneAllocInputBuffer

函数原型	Int ReturnOneAllocInputBuffer(VideoEncoder* pEncoder, VencInputBuffer *pInputbuffer)
功能	还回由 AllocInputBuffer 申请的输入图像帧 buffer
参数	pEncoder: 通过 VideoEncCreate 函数创建的视频编码器指针 pInputbuffer（输入）：由 AllocInputBuffer 申请的输入图像帧 buffer
返回值	成功：返回 0；失败：返回-1；
调用说明	无

3.1.9 ReleaseAllocInputBuffer

函数原型	int ReleaseAllocInputBuffer(VideoEncoder* pEncoder)
功能	释放由 AllocInputBuffer 申请的输入图像帧 buffer
参数	pEncoder: 通过 VideoEncCreate 函数创建的视频编码器指针
返回值	成功：返回 0；失败：返回-1；
调用说明	无

3.1.10 AddOneInputBuffer

函数原型	int AddOneInputBuffer(VideoEncoder* pEncoder, VencInputBuffer* pInputbuffer)
功能	添加输入图像帧到编码器
参数	pEncoder: 通过 VideoEncCreate 函数创建的视频编码器指针；pInputbuffer（输入）：输入图像帧 buffer；
返回值	成功：返回 0；失败：返回-1；
调用说明	pInputbuffer 的来源可以是由 AllocInputBuffer 申请的输入图像帧 buffer，也可以由外部模块来提供；

3.1.11 VideoEncodeOneFrame

函数原型	int VideoEncodeOneFrame(VideoEncoder* pEncoder);
功能	编码一帧数据
参数	pEncoder: 通过 VideoEncCreate 函数创建的视频编码器指针

函数原型	int VideoEncodeOneFrame(VideoEncoder* pEncoder);
返回值	VENC_RESULT_ERROR(-1): 编码出错；VENC_RESULT_OK（0）：编码成功； VENC_RESULT_NO_FRAME_BUFFER （1）：无法获取到输入帧； VENC_RESULT_BITSTREAM_IS_FULL （2）：输出码流 buffer 已经溢出；
调用说明	无

3.1.12 AlreadyUsedInputBuffer

函数原型	int AlreadyUsedInputBuffer(VideoEncoder* pEncoder, VencInputBuffer* pBuffer)
功能	获取 VideoEncodeOneFrame 已经使用过的输入图像帧；
参数	pEncoder: 通过 VideoEncCreate 函数创建的视频编码器指针； pInputbuffer（输出）：图像帧 buffer；
返回值	成功：返回 0；失败：返回-1；
调用说明	无

3.1.13 ValidBitstreamFrameNum

函数原型	ValidBitstreamFrameNum(VideoEncoder* pEncoder)
功能	获取有效的的输出码流 buffer 的格式；
参数	pEncoder: 通过 VideoEncCreate 函数创建的视频编码器指针；
返回值	有效的输出码流的个数（value>=0）；
调用说明	无

3.1.14 GetOneBitstreamFrame

函数原型	int GetOneBitstreamFrame(VideoEncoder* pEncoder, VencOutputBuffer* pBuffer);
功能	获取有效的的输出码流 buffer 的格式
参数	pEncoder: 通过 VideoEncCreate 函数创建的视频编码器指针；pBuffer（输出）：输出码流 buffer；
返回值	成功：返回 0；失败：返回-1；

函数原型	int GetOneBitstreamFrame(VideoEncoder* pEncoder, VencOutputBuffer* pBuffer);
调用说明	pBuffer 中结构体变量说明：nID：用来识别不同的 buffer：nPts：编码器不对时间戳信息做处理，输出 buffer 中的 pts 对应相应输入 buffer 中的 pts；nSize0：输出码流的第一部分的大小；nSize1：输出码流的第二部分的大小；pData0：输出码流的第一部分的地址 pData1：输出码流的第二部分的地址；　　输出的一笔码流可能由两部分组成：nSize0 表示第一部分的大小，nSize1 表示第二部分的大小；nSize0 一定大于 0，当 nSize1 = 0 的时候，输出码流只在地址 pData0 中；当 nSize1 > 0 时，输出码流由两部分组成，第一部分在 pData0 中，第二部分在 pData1 中，此时需要外部应用程序把这两部分数据组合成一帧；

3.1.15 FreeOneBitStreamFrame

函数原型	int FreeOneBitstreamFrame(VideoEncoder* pEncoder, VencOutputBuffer* pBuffer);
功能	还回输出码流 buffer
参数	pEncoder: 通过 VideoEncCreate 函数创建的视频编码器指针；pBuffer（输入）：由 GetOneBitstreamFrame 获取到的输出码流 buffer；
返回值	成功：返回 0；失败：返回-1；
调用说明	pBuffer 表示由 GetOneBitstreamFrame 获取到的输出码流 buffer

3.1.16 VideoEncGetParameter

函数原型	int VideoEncGetParameter(VideoEncoder* pEncoder, VENC_INDEXTYPE indexType, void* paramData);
功能	获取编码器参数；
参数	pEncoder: 通过 VideoEncCreate 函数创建的视频编码器指针；indexType：参数类型索引号；paramData（输出）：参数数据指针；
返回值	成功：返回 0；失败：返回-1；
调用说明	调用成功后将会返回参数到 paramData 指针所指的地址中；

3.1.17 VideoEncSetParameter

函数原型	int VideoEncSetParameter(VideoEncoder* pEncoder, VENC_INDEXTYPE indexType, void* paramData);
功能	设置编码器参数；
参数	pEncoder: 通过 VideoEncCreate 函数创建的视频编码器指针；indexType：参数类型索引号；paramData（输出）：参数数据指针；
返回值	成功：返回 0；失败：返回-1；
调用说明	编码器将从 paramData 指针所指的地址中获取参数信息；

3.1.18 VideoEncoderReset

函数原型	int VideoEncoderReset(VideoEncoder*pEncoder);
功能	重启编码器；
参数	pEncoder: 通过 VideoEncCreate 函数创建的视频编码器指针；
返回值	成功：返回 0；失败：返回-1；
调用说明	编码器配置参数不变，仅把输入帧 buffer 队列和输出比特流 buffer 队列清零；

3.1.19 VideoEncoderGetUnencodedBufferNum

函数原型	int VideoEncoderGetUnencodedBufferNum(VideoEncoder*pEncoder);
功能	获取编码器未完成编码的输入 buffer 个数；
参数	pEncoder: 通过 VideoEncCreate 函数创建的视频编码器指针；
返回值	成功：返回 0；失败：返回-1；
调用说明	无；

3.2 内部模块接口设计

3.2.1 Frame Buffer Manager 模块接口设计

Frame Buffer Manager 模块接口如下所示：

1	FrameBufferManagerCreate	创建帧缓冲管理模块
2	FrameBufferManagerDestroy	销毁帧缓冲管理模块，释放内存
3	AddInputBuffer	添加一帧图像帧到输入帧队列
4	GetInputBuffer	从输入帧队列获取一个图像帧
5	AddUsedInputBuffer	把编码用过的图像帧添加到编码 used 图像帧队列

6	GetUsedInputBuffer	从编码 used 图像帧队列获取一个图像帧，外部模块可以通过此接口，来获知图像帧是否已经被编码器用过
7	AllocateInputBuffer	申请输入图像帧内存，当不使用外部应用程序的 buffer 管理队列的时候，需要调用此接口来申请物理连续的图像帧内存
8	GetOneAllocateInputBuffer	获取一帧由 AllocateInputBuffer 申请的图像帧
9	FlushCacheAllocateInputBuffer	刷 cache 保证数据的一致性
10	ReturnOneAllocateInputBuffer	返回由 AllocateInputBuffer 申请的图像帧
11	FreeAllocateInputBuffer	释放由 AllocateInputBuffer 申请的图像帧
12	ResetFrameBuffer	帧 buffer 队列清零
13	GetUnencodedBufferNum	获取未完成编码的帧 buffer 个数

3.2.1.1 FrameBufferManagerCreate

函数原型	FrameBufferManager* FrameBufferManagerCreate(int num, struct ScMemOpsS *memops)
功能	创建图像帧缓冲管理模块
参数	num: 缓冲帧 buffer 的个数；memops: memory 管理器接口
返回值	成功：返回图像缓冲帧管理模块的指针；失败：返回 NULL；
调用说明	Num 的个数一般设置为 2~6 之间

3.2.1.2 FrameBufferManagerDestroy

函数原型	void FrameBufferManagerDestroy(FrameBufferManager* fbm)
功能	创建图像帧缓冲管理模块
参数	fbm: 通过 FrameBufferManagerCreate 创建的图像帧管理模块的指针
返回值	无
调用说明	无

3.2.1.3 AddInputBuffer

函数原型	int AddInputBuffer(FrameBufferManager* fbm, VencInputBuffer *inputbuffer)
功能	添加一帧图像帧到输入帧队列
参数	fbm: 通过 FrameBufferManagerCreate 创建的图像帧管理模块的指针； Inputbuffer（输入）：输入图像帧的信息；

函数原型	int AddInputBuffer(FrameBufferManager* fbm, VencInputBuffer *inputbuffer)
返回值	成功：返回 0；失败：返回-1；
调用说明	Inputbuffer 参数中包含输入图像帧的信息，包括 pts，地址等信息；nID：用来识别不同的图像帧；nPts：输入图像帧的时间戳，以 us 为单位 nFlag: 标记此 buffer 中的数据是否属于关键帧；pAddrPhyY：输入图像帧的 Y 分量的物理地址，当有 C 分量的时候，此地址必须有效，硬件编码需要物理地址，当输入的数据为 RGB 格式的时候，pAddrPhyY 用来存储 RGB 数据的地址；pAddrPhyC：输入图像帧的 C 分量的物理地址，此地址必须有效，硬件编码需要物理地址。当输入的数据为 RGB 格式的时候，此地址不使用，可以为空；pAddrVirY：输入图像帧的 Y 分量的虚拟地址；pAddrVirC：输入图像帧的 C 分量的虚拟地址；BEnableCorp: 标记是否使用 crop；sCropInfo：如果使用 crop，给出 crop 区域信息；ispPicVar: isp 对 YUV 噪声强度的评价，默认不使用；roi_param[4]: 图像处理程序对 roi 区域标定，增减编码时的 QP，仅个别芯片会用到；

3.2.1.4 GetInputBuffer

函数原型	int GetInputBuffer(FrameBufferManager* fbm, VencInputBuffer *inputbuffer)
功能	从输入图像帧队列获取一个图像帧，供编码器使用
参数	fbm: 通过 FrameBufferManagerCreate 创建的图像帧管理模块的指针； inputbuffer：输入图像帧的信息；
返回值	成功：返回 0；失败：返回-1；
调用说明	编码器编码之前会先调用此函数来获取一个图像帧；

3.2.1.5 AddUsedInputBuffer

函数原型	int AddUsedInputBuffer(FrameBufferManager* fbm, VencInputBuffer *inputbuffer)
功能	编码器用过图像帧后，把图像帧加入到 used 图像帧队列
参数	fbm: 通过 FrameBufferManagerCreate 创建的图像帧管理模块的指针； inputbuffer：图像帧的信息；
返回值	成功：返回 0；失败：返回-1；
调用说明	当编码器编码一帧图像后，会调用此函数把当前编码使用的图像帧加入到 used 图像帧队列

3.2.1.6 GetUsedInputBuffer

函数原型	int GetUsedInputBuffer(FrameBufferManager* fbm, VencInputBuffer *inputbuffer);
功能	编码器用过图像帧后，把图像帧加入到 used 图像帧队列
参数	fbm: 通过 FrameBufferManagerCreate 创建的图像帧管理模块的指针； inputbuffer：图像帧的信息；
返回值	成功：返回 0；失败：返回-1；
调用说明	从编码 used 图像帧队列获取一个图像帧，外部模块可以通过此接口，来获知图像帧是否已经被编码器用过

3.2.1.7 AllocateInputBuffer

函数原型	int AllocateInputBuffer(FrameBufferManager* fbm, VencAllocateBufferParam *buffer_param)
功能	编码器用过图像帧后，把图像帧加入到 used 图像帧队列
参数	fbm: 通过 FrameBufferManagerCreate 创建的图像帧管理模块的指针； buffer_param：申请图像帧参数；buffer_param->nBufferNum：申请的图像帧个数；buffer_param->nSizeY：申请图像帧 Y 分量的大小； buffer_param->nSizeC：申请图像帧的 C 分量的大小；
返回值	成功：返回 0；失败：返回-1；
调用说明	申请输入图像帧内存，当不使用外部应用程序的 buffer 管理队列的时候，需要调用此接口来申请物理连续的图像帧内存

3.2.1.8 GetOneAllocateInputBuffer

函数原型	int GetOneAllocateInputBuffer(FrameBufferManager* fbm, VencInputBuffer* inputbuffer)
功能	获取一帧由 AllocateInputBuffer 申请的图像帧
参数	fbm: 通过 FrameBufferManagerCreate 创建的图像帧管理模块的指针； inputbuffer：图像帧的信息；
返回值	成功：返回 0；失败：返回-1；
调用说明	当使用由 AllocateInputBuffer 申请的图像帧的时候，需要调用此函数来获取一帧图像帧；

3.2.1.9 FlushCacheAllocateInputBuffer

函数原型	int FlushCacheAllocateInputBuffer(FrameBufferManager* fbm, VencInputBuffer *inputbuffer)
功能	对于由 GetOneAllocateInputBuffer 获取到的图像帧，可以使用此函数保证 cache 和 dram 中的数据的一致性
参数	fbm: 通过 FrameBufferManagerCreate 创建的图像帧管理模块的指针； inputbuffer：图像帧的信息；
返回值	成功：返回 0；失败：返回-1；
调用说明	当用 CPU 向由 GetOneAllocateInputBuffer 获取的图像帧中搬移数据后，在把相应的图像帧送给硬件编码之前，需要调用此接口来刷擦车，以此来保证 cache 和 dram 中的数据一致性；

3.2.1.10 ReturnOneAllocateInputBuffer

函数原型	int ReturnOneAllocateInputBuffer(FrameBufferManager* fbm, VencInputBuffer *inputbuffer)
功能	还回由 GetOneAllocateInputBuffer 获取到的图像帧
参数	fbm: 通过 FrameBufferManagerCreate 创建的图像帧管理模块的指针； inputbuffer：图像帧的信息；
返回值	成功：返回 0；失败：返回-1；
调用说明	无

3.2.1.11 FreeAllocateInputBuffer

函数原型	void FreeAllocateInputBuffer(FrameBufferManager* fbm)
功能	释放由 AllocateInputBuffer 申请的图像帧
参数	fbm: 通过 FrameBufferManagerCreate 创建的图像帧管理模块的指针；
返回值	无
调用说明	无

3.2.1.12 ResetFrameBuffer

函数原型	int ResetFrameBuffer(FrameBufferManager* fbm)
功能	帧 buffer 队列清零
参数	fbm: 通过 FrameBufferManagerCreate 创建的图像帧管理模块的指针；

函数原型	int ResetFrameBuffer(FrameBufferManager* fbm)
返回值	成功：返回 0；失败：返回-1；
调用说明	无

3.2.1.13 GetUnencodedBufferNum

函数原型	unsigned int GetUnencodedBufferNum(FrameBufferManager* fbm)
功能	释放由 AllocateInputBuffer 申请的图像帧
参数	fbm: 通过 FrameBufferManagerCreate 创建的图像帧管理模块的指针；
返回值	fbm->added_buff_num
调用说明	无

3.2.2 BitStream Manager 模块接口设计

BitStream Manager 模块接口如下所示：

1	BitStreamCreate	创建码流管理模块，申请一块物理连续的内存
2	BitStreamDestroy	销毁码流管理模块，释放内存
3	BitStreamBaseAddress	获取码流内存的虚拟基地址
4	BitStreamBasePhyAddress	获取码流内存的物理基地址
5	BitStreamEndPhyAddress	获取码流内存的 end 地址
6	BitStreamBufferSize	获取码流 buffer 大小
7	BitStreamFreeBufferSize	获取空余的，未被编码器使用的码流内存大小
8	BitStreamFrameNum	获取码流输出队列中的 buffer 数量
9	BitStreamWriteOffset	获取当前可写的码流内存的 offset
10	BitStreamAddOneBitstream	添加一笔码流到码流输出队列
11	BitStreamGetOneBitstream	从码流输出队列中获取一个码流 buffer
12	BitStreamReturnOneBitstream	还回码流 buffer，更新码流内存有效的数据长度
13	BitStreamReset	码流 buffer 队列清零

3.2.2.1 BitStreamCreate

函数原型	BitStreamManager* BitStreamCreate(int nBufferSize, struct ScMemOpsS *memops)
功能	创建码流管理模块，申请一块物理连续的内存
参数	nBufferSize：申请的码流内存的大小；memops: memory 管理器接口
返回值	成功：返回码流管理模块的指针；失败：返回 NULL；

函数原型	BitStreamManager* BitStreamCreate(int nBufferSize, struct ScMemOpsS *memops)
调用说明	一般申请码流内存为 4~8MB

3.2.2.2 BitStreamDestroy

函数原型	void BitStreamDestroy(BitStreamManager* handle)
功能	销毁码流管理模块，释放内存
参数	Handle：由 BitStreamCreate 申请的码流管理模块的指针
返回值	无
调用说明	无

3.2.2.3 BitStreamBaseAddress

函数原型	void* BitStreamBaseAddress(BitStreamManager* handle)
功能	获取码流内存的虚拟基地址
参数	Handle：由 BitStreamCreate 申请的码流管理模块的指针
返回值	成功：返回码流的基地址（虚拟地址）；失败：返回 NULL；
调用说明	码流输出 buffer 的地址计算需要此函数，通过基地址和 offset 可以计算出对应码率的地址；

3.2.2.4 BitStreamBasePhyAddress

函数原型	void* BitStreamBasePhyAddress(BitStreamManager* handle)
功能	获取码流内存的物理基地址
参数	Handle：由 BitStreamCreate 申请的码流管理模块的指针
返回值	成功：返回码流的基地址（物理地址）；失败：返回 NULL；
调用说明	编码器需要调用此函数获取到码流内存的基地址（物理地址），此地址会配给硬件

3.2.2.5 BitStreamEndPhyAddress

函数原型	void* BitStreamEndPhyAddress(BitStreamManager* handle)
功能	获取码流内存的物理 end 地址
参数	Handle：由 BitStreamCreate 申请的码流管理模块的指针
返回值	成功：返回码流的 end 地址（物理地址）；失败：返回 NULL；

函数原型	void* BitStreamEndPhyAddress(BitStreamManager* handle)
调用说明	编码器需要调用此函数获取到码流内存的 end 地址（物理地址）

3.2.2.6 BitStreamBufferSize

函数原型	int BitStreamBufferSize(BitStreamManager* handle)
功能	获取码流 buffer 大小
参数	Handle：由 BitStreamCreate 申请的码流管理模块的指针
返回值	码流内存的总大小；
调用说明	无

3.2.2.7 BitStreamFreeBufferSize

函数原型	int BitStreamFreeBufferSize(BitStreamManager* handle);
功能	获取码流内存的物理基地址
参数	Handle：由 BitStreamCreate 申请的码流管理模块的指针
返回值	码流内存空余空余大小；
调用说明	无

3.2.2.8 BitStreamFrameNum

函数原型	int BitStreamFrameNum(BitStreamManager* handle)
功能	获取输出码流队列中有效 buffer 的个数
参数	Handle：由 BitStreamCreate 申请的码流管理模块的指针；
返回值	输出码流队列中有效 buffer 的个数；
调用说明	无

3.2.2.9 BitStreamWriteOffset

函数原型	int BitStreamWriteOffset(BitStreamManager* handle)
功能	获取输出码流队列中有效 buffer 的个数
参数	Handle：由 BitStreamCreate 申请的码流管理模块的指针；
返回值	输出码流队列中有效 buffer 的个数；
调用说明	无

3.2.2.10 BitStreamAddOneBitstream

函数原型	int BitStreamAddOneBitstream(BitStreamManager* handle, StreamInfo* pStreamInfo)
功能	添加一笔码流到码流输出队列
参数	Handle：由 BitStreamCreate 申请的码流管理模块的指针；pStreamInfo：码流 buffer 信息
返回值	成功：返回 0；失败：返回-1；
调用说明	pStreamInfo 中变量的说明：nStreamOffset：当前码流 buffer，相应码流内存起始地址的 offset；nStreamLength：当前码流 buffer 的长度；nPts：当前码流 buffer 的时间戳；nFlags：当前码流 buffer 的标记，可以用此来标记当前码率是否是关键帧；nID：当前码流 buffer 的 ID, 用来区分不用的码流 buffer；VE 编码结束后，会调用此函数把一笔码流添加到码流 buffer 队列中;

3.2.2.11 BitStreamGetOneBitstream

函数原型	StreamInfo* BitStreamGetOneBitstream(BitStreamManager* handle)
功能	从码流输出队列中获取一个码流 buffer
参数	Handle：由 BitStreamCreate 申请的码流管理模块的指针；
返回值	成功：返回码流 buffer 信息的指针；失败：返回 NULL；
调用说明	外部模块会调用此函数从码流输出队列中获取一个码流 buffer

3.2.2.12 BitStreamReturnOneBitstream

函数原型	int BitStreamReturnOneBitstream(BitStreamManager* handle, StreamInfo* pStreamInfo)
功能	还回码流 buffer，更新码流内存有效的数据长度
参数	Handle：由 BitStreamCreate 申请的码流管理模块的指针； pStreamInfo：码流 buffer 信息；
返回值	成功：返回 0；失败：返回-1；
调用说明	无

3.2.2.13 BitStreamReset

函数原型	int BitStreamReset(BitStreamManager* handle, struct ScMemOpsS *memops)
功能	码流 buffer 队列清零
参数	Handle：由 BitStreamCreate 申请的码流管理模块的指针；memops：memory 管理器接口；
返回值	成功：返回 0；失败：返回-1；
调用说明	无

3.2.3 Video Encoder Device 模块接口设计

Video Encoder Device 模块接口如下所示：

1	VencoderDeviceCreate	创建编码设备
2	VencoderDeviceDestroy	销毁编码设备

3.2.3.1 VencoderDeviceCreate

函数原型	VENC_DEVICE *VencoderDeviceCreate(VENC_CODEC_TYPE type)
功能	创建编码设备
参数	type：Codec 类型
返回值	成功：返回编码设备的指针；失败：返回 NULL；
调用说明	编码设备的返回值是类型为 VENC_DEVICE 的指针，此结构体中封装了对编码器操作的相关函数；

3.2.3.2 VencoderDeviceDestroy

函数原型	void VencoderDeviceDestroy(void *handle)
功能	销毁编码设备
参数	handle：由 VencoderDeviceCreate 创建的编码设备的指针
返回值	无
调用说明	无

4 数据结构设计

4.1 VencBaseConfig

名称		VencBaseConfig
功能描述		初始化编码器时的基本配置信息
属性	类型	描述
nInputWidth	unsigned int	输入图像帧的宽度；
nInputHeight	unsigned int	输入图像帧的高度；
nDstWidth	unsigned int	编码输出的图像宽度；
nDstHeight	unsigned int	编码输出的图像高度；
nStride	unsigned int	输入图像在内存中的宽度；
eInputFormat	VENC_PIXEL_FMT	输入图像的颜色格式：typedef enum VENC_PIXEL_FMT{ VENC_PIXEL_YUV420SP, VENC_PIXEL_YVU420SP, VENC_PIXEL_YUV420P, VENC_PIXEL_YVU420P, VENC_PIXEL_YUV422SP, VENC_PIXEL_YVU422SP, VENC_PIXEL_YUV422P, VENC_PIXEL_YVU422P, VENC_PIXEL_YUYV422, VENC_PIXEL_UYVY422, VENC_PIXEL_YVYU422, VENC_PIXEL_VYUY422, VENC_PIXEL_ARGB, VENC_PIXEL_RGBA, VENC_PIXEL_ABGR, VENC_PIXEL_BGRA, VENC_PIXEL_TILE_32X32, VENC_PIXEL_TILE_128X32,} VENC_PIXEL_FMT
memops	struct ScMemOpsS*	memory 管理器接口

4.2 VencH264ProfileLevel

名称	VencH264ProfileLevel	
功能描述	H264 编码的 profile 和 level	
属性	类型	描述
nProfile	VENC_H264PROFILETYPE	

名称	VencH264ProfileLevel
nLevel	VENC_H264LEVELTYPE

4.3 encQPRange

名称		VencQPRange
功能描述		H264 编码的 QP 区间
属性	类型	描述
nMaxqp	int	取值范围（0~51）
nMinqp	int	取值范围（0~51）

4.4 MotionParam

名称		MotionParam
功能描述		移动侦测的参数
属性	类型	描述
nMotionDetectEnable	int	0：关闭移动侦测；1：打开移动侦测；
nMotionDetectRatio	int	取值范围（0~12）；0 为最高灵敏度，值越小灵敏度越高，值越大灵敏度越低；

4.5 VencHeaderData

名称		VencHeaderData
功能描述		存储头信息的结构体
属性	类型	描述
pBuffer	unsigned char*	在 H264 编码的时候，会用此来存储 SPS、PPS 信息；JPEG 编码不需要此结构体
nLength	unsigned int	头信息的长度

4.6 VencInputBuffer

名称		VencInputBuffer
功能描述		输入图像帧的信息
属性	类型	描述
nID	int	用来区分不同的 buffer
nPts	long long	当前图像帧的时间戳
nFlag	unsigned int	标记此 buffer 的数据是否属于关键帧
pAddrPhyY	unsigned char*	当前图像帧 Y 分量的物理地址，配给硬件使用
pAddrPhyC	unsigned char*	当前图像帧的 C 分量的物理地址，配给硬件使用
pAddrVirY	unsigned char*	当前图像帧 Y 分量的虚拟地址
pAddrVirC	unsigned char*	当前图像帧 C 分量的虚拟地址
bEnableCorp	int	0: 关闭 corp；1：打开 corp；
sCropInfo	VencRect	当 corp 打开的时候的 corp 矩形区域
ispPicVar	int	isp 对 YUV 数据的噪声评价，默认不使用
roi_param[4]	VencROIConfig	图像处理识别的 ROI 区域，编码器会对这些区域进行 QP 特殊调整，仅个别芯片会用到

4.7 VencOutputBuffer

名称		VencOutputBuffer
功能描述		输出图像帧的信息
属性	类型	描述
nID	int	用来区分不同的 buffer
nPts	long long	当前输出 buffer 的时间戳
nFlag	int	用来标记是否为关键帧
nSize0	unsigned int	输出码流的第一部分长度，存储的数据在地址 pData0 中
nSize1	unsigned int	输出码流的第二部分长度，存储的数据在地址 pData1 中，当 nSize1 = 0 的时候，码流只有一部分，不存在第二部分；
pData0	unsigned char*	输出码流的第一部分地址
pData1	unsigned char*	输出码流的第二部分地址
frame_info	FrameInfo	buffer 中的数据所属帧的 QP 和 GOP 信息，用于码率控制

4.8 VencAllocateBufferParam

名称		VencAllocateBufferParam
功能描述		申请图像帧内存的参数
属性	类型	描述
nBufferNum	unsigned int	申请的图像帧个数；
nSizeY	unsigned int	申请图像帧 Y 分量的大小；
nSizeC	unsigned int	申请图像帧的 C 分量的大小；

4.9 VencH264FixQP

名称		VencH264FixQP
功能描述		固定 QP 参数
属性	类型	描述
bEnable	int	0: 码率控制固定 QP 模式关闭；1：码率控制估计 QP 模式打开；
nIQp	int	I 帧的 QP(0~51)；
nPQp	int	P 帧的 QP(0~51)；

4.10 VencCyclicIntraRefresh

名称		VencCyclicIntraRefresh
功能描述		Cyclic Intra Refresh 信息
属性	类型	描述
bEnable	int	0: 关闭；1：打开；
nBlockNumber	int	一个图像帧划分的区域个数

4.11 VencH264Param

名称		VencH264Param
功能描述		H264 参数
属性	类型	描述
sProfileLevel	VencH264ProfileLevel	Profile 和 level 信息；
bEntropyCodingCABAC	int	0：熵编码使用 CAVLC；1：熵编码使用 CABAC；
sQPRange	VencQPRange	设置 QP 区间；
nFramerate	int	单位为：fps

名称		VencH264Param
nBitrate	int	单位为：bps
nMaxKeyInterval	int	关键帧间隔
nCodingMode	VENC_CODING_MODE	可以选择帧编码，还是场编码：VENC_FRAME_CODINGVENC_FIELD_CODING

4.12 VencROIConfig

名称		VencROIConfig
功能描述		ROI 感兴趣区域设置
属性	类型	描述
bEnable	int	0：关闭；1：打开；
index	int	可使用 4 个 ROI，区域，index 的值可设为（0~3），来选择这四个 ROI 区域；
nQPoffset	int	通过 nQPoffset 可以设置 QP：ROI 区域的 QP 为码率控制产生的 QP 与用户设定的 nQPoffset 的差；例如：一帧图像使用固定 QP=30；nQPoffset = 10；那么 ROI 区域的 QP 为：30 - 10 = 20;
sRect	VencRect	感兴趣区域所表示的矩形区域

4.13 VencH264AspectRatio

名称		VencH264AspectRatio
功能描述		VUI 扩展选项，对播放视频时的显示比例限制
属性	类型	描述
aspect_ratio_idc	unsigned char	一般取值 255，表示启用自定义显示比例；
sar_width	unsigned short	显示比例宽度；
sar_height	unsigned short	显示比例高度;

4.14 VencH264VideoSignal

功能描述		VUI 扩展选项，对颜色空间控制
属性	类型	描述
video_format	VENC_VIDEO_FORMAT	视频制式，一般取值 5；

功能描述		VUI 扩展选项，对颜色空间控制
full_range_flag	unsigned char	全范围色彩空间标识；
src_colour_primaries	VENC_COLOR_SPACE	输入源色彩空间；typedef enum {RESERVED0 = 0, VENC_BT709 = 1, RESERVED1 = 2, RESERVED2 = 3, RESERVED3 = 4, NC_BT601 = 5, 601_525 = 6, SERVED4 = 7, NC_YCC = 8,} VENC_COLOR_SPACE;
st_colour_primaries	VENC_COLOR_SPACE	输出图色彩空间；

4.15 VencH264SVCSkip

名称		VencH264SVCSkip
功能描述		时域可伸缩编码及跳帧，不能与插帧混用
属性	类型	描述
nTemporalSVC	T_LAYER	时域分层数：typedef enum { NO_T_SVC = 0, T_LAYER_2 = 2, T_LAYER_3 = 3, T_LAYER_4 = 4} T_LAYER;
nSkipFrame	SKIP_FRAME	跳帧倍数，若 nTemporalSVC 为 0，则可独立使用；否则没意义，实际跳帧受 nTemporalSVC 控制；typedef enum { NO_SKIP = 0, SKIP_2 = 2, SKIP_4 = 4, SKIP_8 = 8}SKIP_FRAME;

4.16 VENC_INDEXTYPE

对函数 VideoEncGetParameter 与 VideoEncSetParameter 中用到的枚举变量 VENC_INDEXTYPE 进行说明：

VENC_INDEXTYPE	引用的数据类型	描述
VENC_IndexParamBitrate	int	单位为：bps
VENC_IndexParamFramerate	int	单位为：fps
VENC_IndexParamMaxKeyInterval	int	设置关键帧最大间隔
VENC_IndexParamIfilter	int	I 帧滤波开关

VENC_INDEXTYPE	引用的数据类型	描述
VENC_IndexParam Rotation	int	设置旋转方向（支持 4 个方向）：0：不旋转；90：旋转 90 度；180：旋转 180 度；270：旋转 270 度；
VENC_IndexParam SliceHeight	int	设置一个 slice 的高度，一帧图像可以支持多个 slice，单位为像素，16 对齐；
VENC_IndexParam ForceKeyFrame	int	在编码过程中，可以强制设置下一帧为关键帧
VENC_IndexParam MotionDetectEnable	MotionParam	移动侦测开关
VENC_IndexParam MotionDetectStatus	int	编码一帧结束后，可以使用此接口获取当前图像帧是否有物体运动：0：静止；1：移动；
VENC_IndexParam Rgb2Yuv	VENC_COLOR_SPACE	颜色空间转换
VENC_IndexParam Yuv2Yuv	VENC_YUV2YUV	颜色空间标准转换
VENC_IndexParam ROIConfig	VencROIConfig	人眼感兴趣区域增强
VENC_IndexParam Stride	int	图片在内存中的行宽值
VENC_IndexParam ColorFormat	int	（VENC_PIXEL_FMT）输入编码器数据的颜色格式
VENC_IndexParam Size	VencSize	只读。获取图片输入的宽高
VENC_IndexParam SetVbvSize	unsigned int	预设申请 VBV（编码输出）buffer 大小
VENC_IndexParam VbvInfo	VbvInfo	只读。获取 VBV（编码输出）buffer 信息
VENC_IndexParam SuperFrameConfig	VencSuperFrameConfig	超大帧重编码处理设置
VENC_IndexParam SetPSkip	unsigned int	插帧开关
VENC_IndexParam ResetEnc	int	复位编码器输入输出 buffer，I 帧 QP 更改
VENC_IndexParam H264QPRange	VencQPRange	设置 QP 波动范围
VENC_IndexParam H264ProfileLevel	VencProfileLevel	nProfile 和 nLevel 取值参考 vencoder.h
VENC_IndexParam H264EntropyCodingCABAC	int	设置熵编码格式。0：CAVLC；1:CABAC
VENC_IndexParam H264CyclicIntraRefresh	VencCyclicIntraRefresh	循环帧内刷新，网络码流使用。

VENC_INDEXTYPE	引用的数据类型	描述
VENC_IndexParamH264FixQP	VencH264FixQP	不使用码率控制，固定 QP
VENC_IndexParamH264SVCSkip	VencH264SVCSkip	此选项不能与插帧选项混用。时域 SVC 和跳帧，时域分层取值 0/2/3/4。跳帧倍数取值 0/2/4/8。若时域分层不为 0，则跳帧倍数受时域分层控制，对其取值无意义；否则跳帧倍数可独立使用。
VENC_IndexParamH264AspectRatio	VencH264AspectRatio	VUI 扩展选项。限制视频播放时的显示比例。一般把 aspect_ratio_idc 设为 255，显示比例取 sar_width 和 sar_height 的比值。
VENC_IndexParamH264FastEnc	unsigned int	快速编码开关，简化编码操作，编码速度加快，但图像质量和压缩率下降。
VENC_IndexParamH264VideoSignal	VencH264VideoSignal	VUI 扩展选项。选择编码颜色空间。video_format 一般为 5。src_colour_primaries 取 5，dst_colour_primaries 取 8，颜色最明亮；两者取值相反，效果次之；其它取值颜色最灰暗。
VENC_IndexParamH264IQpOffset	int	I 帧 QP 偏移值
VENC_IndexParamJpegEncMode	int	JPEG 编码方式，若编单幅图片，设为 0；若编 MJPEG 序列，设为 1。
VENC_IndexParamJpegVideoSignal	VencJpegVideoSignal	颜色空间选择，同 H264 类似选项。
VENC_IndexParamJpegQuality	int	（0~100）值越大，编码质量越高
VENC_IndexParamJpegExifInfo	EXIFInfo	JPEG 图片的描述信息，包括快门速度，曝光时间，GPS 信息，缩略图信息等

4.17 VENC_DEVICE

名称	VENC_DEVICE
功能描述	编码器内部功能调用接口，用于调用 JPEG 编码、H264 编码等;
方法	描述
open	打开编码器设备
init	初始化编码设备上下文信息；
uninit	去初始化编码设备
close	关闭编码设备

名称	VENC_DEVICE
encode	编码一个图像帧
GetParameter	获取编码器参数
SetParameter	获取编码器参数

著作权声明

商标声明

免责声明