目录

一、 bm_video Decode 数据结构 & API说明

bm_video Decode 数据结构

- BmVpuDecStreamFormat
- BmVpuDecSkipMode
- BmVpuDecDMABuffer
- BmVpuDecOutputMapType
- BmVpuDecBitStreamMode
- BmVpuDecPixFormat
- BMVidDecParam
- BMDecStatus
- BMDecOutputMapType
- BMVidStream
- BMVidFrame
- BMVidStreamInfo
- BmVpuDecPicType

1. BmVpuDecStreamFormat

设置输入码流的格式。

- BMDEC_AVC 表示输入码流满足 AVC 编码标准；
- BMDEC_HEVC 表示输入码流满足 HEVC 编码标准。

2. BmVpuDecSkipMode

设置跳帧模式

成员变量	描述
BMDEC_FRAME_SKIP_MODE	不开启跳帧模式
BMDEC_SKIP_NON_REF_NON_I	开启跳帧模式，跳过除参考帧和I帧外的视频帧
BMDEC_SKIP_NON_I	开启跳帧模式，跳过除I帧外的视频帧

3. BmVpuDecDMABuffer

保存 VPU 缓冲区的信息

成员变量	描述
size	缓冲区的大小
phys_addr	缓冲区的物理地址
virt_addr	缓冲区的虚拟地址

4. BmVpuDecOutputMapType

设置输出数据的类型。

- BMDEC_OUTPUT_UNMAP 输出 yuv 数据；
- BMDEC_OUTPUT_COMPRESSED 输出压缩模式数据。

5. BmVpuDecBitStreamMode

设置 VPU 解码方式。

- BMDEC_BS_MODE_INTERRUPT 采用流模式解码，当输入缓冲区填满后送入解码器；
- BMDEC_BS_MODE_PIC_END 采用帧模式解码，获取到一帧数据就送入解码器。

6. BmVpuDecPixFormat

设置输出数据的格式

成员变量	描述
BM_VPU_DEC_PIX_FORMAT_YUV420P	输出 YUV420P 数据
BM_VPU_DEC_PIX_FORMAT_YUV422P	输出 YUV422P 数据，BM1684 不支持
BM_VPU_DEC_PIX_FORMAT_YUV444P	输出 YUV444P 数据，BM1684 不支持
BM_VPU_DEC_PIX_FORMAT_YUV400	输出 YUV400 数据，BM1684 不支持
BM_VPU_DEC_PIX_FORMAT_NV12	输出 NV12 数据
BM_VPU_DEC_PIX_FORMAT_NV21	输出 NV21 数据
BM_VPU_DEC_PIX_FORMAT_NV16	输出 NV16 数据，BM1684 不支持
BM_VPU_DEC_PIX_FORMAT_NV24	输出 NV24 数据，BM1684 不支持
BM_VPU_DEC_PIX_FORMAT_COMPRESSED	输出压缩格式数据
BM_VPU_DEC_PIX_FORMAT_COMPRESSED_10BITS	输出10bits压缩格式数据，BM1684 不支持

7. BMVidDecRetStatus

解码器接口返回的错误码类型

成员变量	描述
BM_ERR_VDEC_INVALID_CHNID	无效的解码channel id
BM_ERR_VDEC_ILLEGAL_PARAM	非法参数
BM_ERR_VDEC_EXIST	解码channel已存在
BM_ERR_VDEC_UNEXIST	解码channel不存在
BM_ERR_VDEC_NULL_PTR	空指针
BM_ERR_VDEC_NOT_CONFIG	解码器未配置
BM_ERR_VDEC_NOT_SUPPORT	不支持的解码业务
BM_ERR_VDEC_NOT_PERM	参数异常
BM_ERR_VDEC_INVALID_PIPEID	非法的PIPEID
BM_ERR_VDEC_INVALID_GRPID	非法的GRPID
BM_ERR_VDEC_NOMEM	存储空间异常
BM_ERR_VDEC_NOBUF	缓冲区异常
BM_ERR_VDEC_BUF_EMPTY	缓冲区空
BM_ERR_VDEC_BUF_FULL	缓冲区满
BM_ERR_VDEC_SYS_NOTREADY	解码器未准备就绪
BM_ERR_VDEC_BADADDR	错误地址
BM_ERR_VDEC_BUSY	解码器忙
BM_ERR_VDEC_SIZE_NOT_ENOUGH	空间不足
BM_ERR_VDEC_INVALID_VB	无效的VB
BM_ERR_VDEC_ERR_INIT	解码器初始化错误
BM_ERR_VDEC_ERR_INVALID_RET	无效返回值

成员变量	描述
BM_ERR_VDEC_ERR_SEQ_OPER	队列处理异常
BM_ERR_VDEC_ERR_VDEC_MUTEX	信号量异常
BM_ERR_VDEC_ERR_SEND_FAILED	发送失败
BM_ERR_VDEC_ERR_GET_FAILED	获取失败
BM_ERR_VDEC_ERR_HUNG	解码器挂起

8. BMVidDecParam

BMVidDecParam 用于设置解码器的初始化参数，在调用接口 bmvpu_dec_create 前需要创建 BMVidDecParam 对象，并对其进行初始化。

成员变量	类型	描述
streamFormat	BmVpuDecStreamFormat	设置输入码流类型，BMDEC_AVC 为 H.264(AVC)，BMDEC_HEVC 为 H.265(HEVC)
wtlFormat	BmVpuDecOutputMapType	设置输出数据格式
skip_mode	BmVpuDecSkipMode	设置跳帧模式
bsMode	BmVpuDecBitStreamMode	设置解码器工作方式。 0 以 INTERRPUT 模式工作； 2 以 PIC_END 模式工作。
enableCrop	int	是否启用裁剪选项，此参数无效
pixel_format	BmVpuDecPixFormat	输出图像格式
secondaryAXI	int	是否开启 secondary AXI。SDK 中会根据码流类型，自动选择，不需要手动开启
mp4class	int	MPEG_4，此参数无效
frameDelay	int	帧延迟输出，大于0时，在解码frameDelay帧后输出显示帧，此参数无效
pcie_board_id	int	PCIE板卡的设备id
pcie_no_copyback	int	pcie模式，解码输出数据不拷贝回host端
enable_cache	int	启用缓存，提高内存拷贝速度，但会增加算力、带宽等的消耗
perf	int	性能监测功能
core_idx	int	解码核选择。1684x core_idx 可以配置为 0，1，-1；配置为-1时，会根据解码器负载自动选择解码核
timeout	int	解码超时时间，默认为3000ms（即VPU_WAIT_TIME_OUT）
timeout_count	int	解码超时重试次数，默认为5
extraFrameBufferNum	int	除去vpu所必要的Frame Buffer 外，用户额外需要的 Frame Buffer 的数量。
min_framebuf_cnt	int	输入码流所需要的最小的 Frame Buffer 的数量。
framebuf_delay	int	解码延迟出帧所需要的 Frame Buffer 的数量。
streamBufferSize	int	设置输入码流的缓冲区大小。 若设置为 0，则默认缓冲区大小为 0x700000。
bitstream_buffer	BmVpuDecDMABuffer	输入码流缓冲区信息
frame_buffer	BmVpuDecDMABuffer	Frame Buffer 信息
Ytable_buffer	BmVpuDecDMABuffer	压缩模式 Y table 缓冲区信息
Ctable_buffer	BmVpuDecDMABuffer	压缩模式 C table 缓冲区信息

备注：

- 解码器支持用户自行分配 Bitstream Buffer 和 Frame Buffer。当外部分配内存时，extraFrameBufferNum、min_framebuf_cnt、framebuf_delay、streamBufferSize、bitstream_buffer、frame_buffer、Ytable_buffer、Ctable_buffer 必须配置。
- Frame Buffer的计算参考（Frame Buffer 计算方法）

9. BMDecStatus

枚举类型，用于指示解码器的状态。

状态	含义
BMDEC_UNCREATE	解码器未创建(用户无需处理)
BMDEC_UNLOADING	解码器未加载(用户无需处理)
BMDEC_UNINIT	解码器未初始化
BMDEC_WRONG_RESOLUTION	设置的分辨率不匹配
BMDEC_FRAMEBUFFER_NOTENOUGH	分配的 Frame Buffer 不足
BMDEC_DECODING	解码器正在解码
BMDEC_ENDOF	解码器送帧结束
BMDEC_STOP	解码器停止解码
BMDEC_HUNG	解码器无响应
BMDEC_CLOSE	关闭解码器，表示可以开始关闭解码器
BMDEC_CLOSED	解码器关闭状态

10. BMDecOutputMapType

枚举类型，用于定义编码器输出的YUV格式

类型	含义
BMDEC_OUTPUT_UNMAP	原始YUV数据
BMDEC_OUTPUT_TILED	TILED YUV数据(已弃用)
BMDEC_OUTPUT_COMPRESSED	压缩格式YUV数据

11. BMVidStream

保用于表示视频流的实际数据，包括像素值、时间戳等。

成员变量	类型	描述
buf	unsigned char*	码流信息存储地址
length	unsigned int	码流信息大小
header_buf	unsigned char*	码流头信息存储地址
header_size	unsigned int	码流头信息大小
extradata	unsigned char*	已弃用：不再接受 extradata 数据。
extradata_size	unsigned int	已弃用：不再接受 extradata 数据。
pts	unsigned long	显示时间戳。
dts	unsigned long	解码时间戳。

12.BMVidFrame

保存解码器输出的视频帧信息。

成员变量	类型	描述
picType	BmVpuDecPicType	图片类型
buf[8]	unsigned char*	存放输出数据的地址。前四个通道存储YUV的虚拟地址，后四通道存储YUV的物理地址。0：Y虚拟地址，1：Cb虚拟地址，2：Cr虚拟地址。4：Y物理地址，5：Cb物理地址，6：Cr物理地址。3和7为特殊格式数据的存储通道（如存放透明度数据）
stride[8]	int	和 buf 对应，存放对应通道的步长。 对于 FBC 数据，stride 存放的数据稍有不同。channel 0 和 4，存放 Y 分量的宽度； channel 1 和 5，存放 Cb 分量的宽度； channel 2 和 6，存放 Y table 的长度； channel 3 和 7，存放 Cb table 的长度
width	unsigned int	存放 Frame 的宽度
height	unsigned int	存放 Frame 的高度
frameFormat	int	输出 yuv的格式 frameFormat为0 ： 输出为yuv，需要和cbcrinterleve 结合使用。 frameFormat为0，cbcrInterleave为 0 输出:yuv420p frameFormat为0 cbcrInterleave为1 输出:nv12 frameFormat为116：压缩数据需要调用vpp解压缩
interlacedFrame	BmVpuDecLaceFrame	图像扫描方式。0 为逐行扫描模式，1为隔行扫描模式
lumaBitDepth	int	亮度数据的深度
chromaBitDepth	int	色度数据的深度
pixel_format	BmVpuDecPixFormat	图像格式
endian	int	表示帧缓冲区的段序。 endian=0，以小端模式存储； endian=1，以大端模式存储； endian=2，以 32 位小端模式存储； endian=3，以 32 位大端模式存储
sequenceNo	int	表示码流序列的状态。当码流序列改变时，sequenceNo 的值会进行累加
frameIdx	int	图像帧缓冲区的索引。用于表示该帧缓冲区在解码器中位置
pts	unsigned long	显示时间戳戳
dts	unsigned long	解码时间戳戳
size	int	帧缓冲区的大小
colorPrimaries	int	指定视频使用的色彩原色标准，影响颜色再现方式
colorTransferCharacteristic	int	定义了颜色从原色到可显示色彩的转换特性或曲线
colorSpace	int	表明视频色彩的编码空间，如RGB、YCbCr等
colorRange	int	指明色彩的动态范围，如广色域或有限色域（全范围或限定范围）
chromaLocation	int	定义色度样本相对于亮度样本的位置。
coded_width	unsigned int	用于编码的图片宽度
coded_height	unsigned int	用于编码的图片高度

13.BMVidStreamInfo

用于描述视频流的基本信息，例如图像大小、帧率、编码标准等。

成员变量	类型	描述
picWidth	int	图片的水平像素大小
picHeight	int	图片的垂直像素大小
fRateNumerator	int	帧率分数的分子
fRateDenominator	int	帧率分数的分母
picCropRect	CropRect	图片裁剪矩形信息（仅适用于H.264/HEVC解码器）
mp4DataPartitionEnable	int	MPEG4 VOL头中的 data_partitioned 标志位
mp4ReversibleVlcEnable	int	MPEG4 VOL头中的 reversible_vlc 标志位
mp4ShortVideoHeader	int	0：非H.263流 1：H.263流（mpeg4 short video header）
h263AnnexJEnable	int	0：禁用Annex J， 1：启用Annex J（可选的解块滤波器模式）
minFrameBufferCount	int	解码所需的最小帧缓冲区数量
frameBufDelay	int	最大显示帧缓冲区延迟
normalSliceSize	int	正常情况下保存切片的推荐缓冲区大小（仅适用于H.264）
worstSliceSize	int	最坏情况下保存切片的推荐缓冲区大小（仅适用于H.264）
maxSubLayers	int	H.265/HEVC的子层数量。
profile	int	不同视频编码标准的配置文件信息
level	int	不同视频编码标准的级别信息。
tier	int	层次指示器（0：主层，1：高层）
interlace	int	Indication of interlaced or progressive frame
constraint_set_flag	int[4]	H.264/AVC SPS中的 constraint_set0_flag 至 constraint_set3_flag。指定了视频编码的一些约束集，每个元素对应一个约束集的标志
direct8x8Flag	int	标志指定了解码器在进行运动估计时是否使用直接模式进行 8x8 推导。 1表示启用了直接模式; 0表示禁用
vc1Psf	int	VC1序列层中的渐进分段帧（PSF）标志
isExtSAR	int	H.264中的SAR（Sample Aspect Ratio,样本宽高比）扩展标志
maxNumRefFrmFlag	int	H.264 中的 max_num_ref_frames 的标志位。 0表示 max_num_ref_frames 为 0； 1表示 max_num_ref_frames 不为 0
maxNumRefFrm	int	H.264 中的 max_num_ref_frames 的具体数值，仅在maxNumRefFrmFlag==1时有效
aspectRateInfo	int	图像的宽高比信息
bitRate	int	码流写入时的比特率
mp2LowDelay	int	MPEG2规范中sequence extension的low_delay语法
mp2DispVerSize	int	MPEG2规范中sequence display extension的display_vertical_size语法
mp2DispHorSize	int	MPEG2规范中sequence display extension的display_horizontal_size语法
userDataHeader	Uint32	用户数据头
userDataNum	Uint32	用户数据的数量
userDataSize	Uint32	用户数据的大小

成员变量	类型	描述
userDataBufFull	Uint32	当 userDataEnable 启用时，解码器将帧缓冲区状态报告到 userDataBufAddr 和以字节为单位的 userDataSize 中。当用户数据报告模式为1且用户数据大小大于用户数据缓冲区大小时，VPU将报告与缓冲区大小一样多的用户数据，跳过剩余部分并设置 serDataBufFull
chromaFormatIDC	int	色度格式指示器
lumaBitdepth	int	亮度样本的位深度
chromaBitdepth	int	色度样本的位深度
seqInitErrReason	int	序列头解码错误原因
warnInfo	int	警告信息
sequenceNo	unsigned int	序列信息的序号，增加1表示检测到序列变化

14.BmVpuDecPicType

- 枚举类型，表示图片类型。
- 0 表示I帧；1 表示P帧；2 表示B帧；5表示IDR帧。

bm_video Decode API说明

- bmvpu_dec_create
- bmvpu_dec_get_status
- bmvpu_dec_decode
- bmvpu_dec_get_caps
- bmvpu_dec_get_output
- bmvpu_dec_clear_output
- bmvpu_dec_flush
- bmvpu_dec_delete
- bmvpu_dec_get_stream_buffer_empty_size
- bmvpu_dec_get_all_frame_in_buffer
- bmvpu_dec_get_empty_input_buf_cnt
- bmvpu_dec_get_pkt_in_buf_count
- bmvpu_dec_vpu_reset
- bmvpu_dec_get_core_idx
- bmvpu_dec_dump_stream
- bmvpu_get_inst_idx
- bmvpu_dec_get_stream_info
- bmvpu_dec_set_logging_threshold

1. bmvpu_dec_create

[功能和说明]

- 用于创建视频解码器的实例，返回创建的视频解码器实例句柄。
- 初始化一些变量和数据结构，申请解码器实例所需要的内存。将创建的解码器实例的句柄返回给用户，用户通过句柄来操作解码器。

[函数名]

```
BMVidDecRetStatus bmvpu_dec_create(BMVidCodHandle* pVidCodHandle, BMVidDecParam decParam)
```

[参数说明]

- `BMVidCodHandle *pVidCodHandle` 存储创建的视频解码器实例的句柄
- `BMVidDecParam decParam` 视频解码器的配置参数

[返回值]

返回值为0表示成功，其他值表示失败。

2. bmvpu_dec_get_status

[功能和说明]

- 获取当前视频解码器的状态。
- 该函数接受一个视频编码器句柄 BMVidCodHandle，并返回该句柄对应视频解码器的状态 BMDecStatus。
- 如果句柄有效，函数返回解码器的当前状态；否则，返回 BMDEC_CLOSED 表示句柄错误。

[函数名]

```
BMDecStatus bmvpu_dec_get_status(BMVidCodHandle vidCodHandle)
```

[参数说明]

- `BMVidCodHandle vidCodHandle`：视频编码器句柄。

[返回值]

解码器的状态

3. bmvpu_dec_decode

[功能和说明]

- 解码视频流并将解码后的帧放入输出队列。
- 该函数接受一个视频编码器句柄 BMVidCodHandle 和一个包含视频流信息的结构体 BMVidStream。
- 函数首先检查输入参数的有效性，包括输入队列是否已满、解码器是否处于正确状态等。
- 若满足调用条件，函数将视频流数据填充到环形缓冲区，并触发解码器开始解码。

[函数名]

```
BMVidDecRetStatus bmvpu_dec_decode(BMVidCodHandle vidCodHandle, BMVidStream vidStream)
```

[参数说明]

- `BMVidCodHandle vidCodHandle` 视频编码器句柄。
- `BMVidStream vidStream` 包含视频流信息的结构体，包括帧数据、长度、时间戳等。

[返回值]

返回值为0表示成功，其他值表示失败。

4. bmvpu_dec_get_caps

[功能和说明]

- 获取视频编码器和码流的相关信息。
- 该函数接受一个视频编码器句柄 BMVidCodHandle 和一个用于存储信息的结构体指针 BMVidStreamInfo。
- 若满足调用条件，函数从视频编码器实例中提取初始信息，填充到给定的 BMVidStreamInfo 结构体中。
- 返回 0 表示成功，其他值表示错误。

[函数名]

```
BMVidDecRetStatus bmvpu_dec_get_caps(BMVidCodHandle vidCodHandle, BMVidStreamInfo* streamInfo)
```

[参数说明]

- `BMVidCodHandle vidCodHandle` 视频编码器句柄。
- `BMVidStreamInfo *streamInfo` 用于存储视频解码器信息的结构体指针。

[返回值]

返回值为0表示成功，其他值表示失败。

5. bmvpu_dec_get_output

[功能和说明]

- 从视频解码器获取输出帧信息。
- 该函数接受一个视频编码器句柄 BMVidCodHandle，并返回一个指向 BMVidFrame 结构的指针，该结构包含解码器输出的帧信息。

[函数名]

```
BMVidDecRetStatus bmvpu_dec_get_output(BMVidCodHandle vidCodHandle, BMVidFrame* frame)
```

[参数说明]

- `BMVidCodHandle vidCodHandle` 视频编码器句柄。

[返回值]

- 如果成功获取输出帧信息，则返回指向 BMVidFrame 结构的指针，其中包含帧的详细信息。
- 如果未成功获取输出帧信息，返回 NULL。
- 此函数的返回值可能为 NULL，因此在使用返回值前建议进行有效性检查。

6. bmvpu_dec_clear_output

[功能和说明]

释放指定的视频帧所占用的缓冲区资源，供后续输出视频帧使用。

[函数名]

```
BMVidDecRetStatus bmvpu_dec_clear_output(BMVidCodHandle vidCodHandle, BMVidFrame* frame)
```

[参数说明]

- `BMVidCodHandle vidCodHandle` 视频解码器的句柄，用于标识特定的视频解码器实例。
- `BMVidFrame *frame` 要清除的输出帧

[返回值] 返回值为0表示成功，其他值表示失败。

7. bmvpu_dec_flush

[功能和说明]

- 刷新（Flush）视频解码器输出缓冲区
- 该函数用于刷新视频解码器，应在关闭解码器之前调用，确保获取到所有的解码输出数据。
- 解码器解码通常会存在延迟出帧的情况，当输入数据全部送入解码器后，并不能得到所有输出数据。需要调用 bmvpu_dec_flush 告诉解码器当前输入文件已经结束，刷新解码器中的数据，并更新解码器状态。
- 如果不调用 bmvpu_dec_flush，可能会存在丢帧问题。

[函数名]

```
BMVidDecRetStatus bmvpu_dec_flush(BMVidCodHandle vidCodHandle)
```

[参数说明]

- `BMVidCodHandle vidCodHandle` 视频解码器的句柄，用于标识特定的视频解码器实例

[返回值] 返回值为0表示成功，其他值表示失败。

8. bmvpu_dec_delete

[功能和说明]

调用 bmvpu_dec_delete 用于关闭解码器实例，并释放资源。

[函数名]

```
BMVidDecRetStatus bmvpu_dec_delete(BMVidCodHandle vidCodHandle)
```

[参数说明]

- `BMVidCodHandle vidCodHandle` 视频解码器的句柄，用于标识特定的视频解码器实例

[返回值]

返回值为0表示成功，其他值表示失败。

9. bmvpu_dec_get_stream_buffer_empty_size

[功能和说明]

- 获取视频解码器的比特流缓冲的可用空间大小。
- INTERRUPT MODE: 返回比特流缓冲区的剩余空间大小。
- PIC_END MODE：返回比特流缓冲区当前能够存储的压缩帧数据的最大容量。该容量并不一定等于比特流缓冲区的剩余空间大小。
- 返回比特流缓冲的可用空间大小或错误码。

[函数名]

```
int bmvpu_dec_get_stream_buffer_empty_size(BMVidCodHandle vidCodHandle)
```

[参数说明]

`BMVidCodHandle vidCodHandle` 视频解码器实例的句柄

[返回值]

返回值为比特流缓冲的剩余空间大小，如果出现错误，返回对应错误码。

10. bmvpu_dec_get_all_frame_in_buffer

[功能和说明]

- 刷新视频解码器输出缓冲区。
- 通过句柄获取解码器实例相关信息。
- 作用和 bmvpu_dec_flush 一样，用于刷新解码器中剩余的帧数据。

[函数名]

```
int bmvpu_dec_get_all_frame_in_buffer(BMVidCodHandle vidCodHandle)
```

[参数说明]

- `BMVidCodHandle vidCodHandle` 视频解码器实例的句柄

[返回值]

返回值为0表示成功通知视频解码器获取所有剩余帧并放入缓冲区。

11. bmvpu_dec_get_all_empty_input_buf_cnt

[功能和说明]

- 该函数用于获取视频解码器当前空闲输入缓冲区的数量。
- 通过句柄获取解码器实例相关信息。
- 检查vidHandle是否为空或者解码器状态是否已关闭。如果是，返回0表示没有空闲输入缓冲区。
- 返回计算得到的空闲输入缓冲区的数量。
- 函数返回表示空闲输入缓冲区数量的整数值。

[函数名]

```
int bmvpu_dec_get_all_empty_input_buf_cnt(BMVidCodHandle vidCodHandle)
```

[参数说明]

- `BMVidCodHandle vidCodHandle` 视频解码器实例的句柄

[返回值]

返回值表示空闲输入缓冲区的数量。

12. bmvpu_dec_get_pkt_in_buf_count

[功能和说明]

- 获取视频解码器输入缓冲区中数据包数量。
- 检查vidHandle是否为空或者解码器状态是否已关闭。如果是，返回0表示没有已有数据包。
- 获取队列中的元素个数，该数量表示还未进行解码的数据包的个数。
- 函数返回获取到的数据包的数量。

[函数名]

```
int bmvpu_dec_get_pkt_in_buf_count(BMVidCodHandle vidCodHandle)
```

[参数说明]

- `BMVidCodHandle vidCodHandle` 视频解码器实例的句柄

13. bmvpu_dec_vpu_reset

[功能和说明]

- 对Sophon设备的VPU进行硬件复位（reset）。
- 该函数可用于复位指定设备上的所有VPU核心，或者只复位设备上的特定VPU核心。

[函数名]

```
BMVidDecRetStatus bmvpu_dec_reset(int devIdx, int coreIdx);
```

[参数说明]

- `int devIdx` Sophon设备的索引，范围为[0, MAX_PCIE_BOARD_NUM-1]。
- `int coreIdx` VPU核心的索引，范围为[-1, MAX_NUM_VPU_CORE_CHIP-1]。若为-1，表示复位设备上的所有VPU核心。

[返回值]

返回值为0表示复位成功，其他值表示复位失败。

14. bmvpu_dec_get_core_idx

[功能和说明]

- 获取视频编码器实例的 VPU 核心索引
- 该函数接受一个视频编码器句柄 BMVidCodHandle 作为参数。
- 通过将输入句柄强制类型转换为 BMVidHandle，然后获取其 codecInst，最终取得 coreIdx。
- 返回 VPU 核心索引。

[函数名]

`int getcoreidx(BMVidCodHandle handle)` [返回值]

返回VPU核心索引

15. bmvpu_dec_dump_stream

[功能和说明]

- 用于将解码器的输入缓冲区进行映射，用于转储位本地文件。
- 若出现错误，将会返回0.

[函数名]

```
int bmvpu_dec_dump_stream(BMVidCodHandle vidCodHandle, unsigned char *p_stream, int size)
```

[参数说明]

- `BMVidCodHandle vidCodHandle` 视频编解码器句柄，表示与Sophon设备中的VPU关联的视频编解码器。
- `unsigned char *p_stream` 用于存储解码器输入比特流的本地内存缓冲区的指针。
- `int size` 缓冲区的大小，表示用户提供的本地内存缓冲区的长度。

[返回值]

返回值表示成功转储到本地内存的比特流的长度。如果发生错误，返回值为0。

16. bmvpu_get_inst_idx

[功能和说明]

用于获取与Sophon设备中的VPU关联的视频编解码器的实例索引。

[函数名]

`int bmvpu_get_inst_idx(BMVidCodHandle vidCodHandle)` [参数说明]

- `BMVidCodHandle vidCodHandle` 视频编解码器句柄，表示与Sophon设备中的VPU关联的视频编解码器。

[返回值]

返回值表示与Sophon设备中的VPU关联的视频编解码器的实例索引。

17. bmvpu_dec_get_stream_info

[功能和说明]

- 该接口用于外部分配内存时查询码流信息。
- 调用该接口可以获取码流的宽高和缓冲区数量信息。
- 当用户申请的内存和解码器所需要的内存不匹配时，会返回错误。可以通过该接口获取正确的信息。

[函数名]

`BMVidDecRetStatus bmvpu_dec_get_stream_info(BMVidCodHandle vidCodHandle, int* width, int* height, int* mini_fb, int* frame_delay)` [参数说明]

- `BMVidCodHandle vidCodHandle` 视频编解码器句柄。
- `int* width` 存储码流宽度
- `int* height` 存储码流高度
- `int* mini_fb` 存储解码所需的最小缓冲区数量
- `int* frame_delay` 存储延迟出帧所需的缓冲区数量

[返回值]

返回值表示接口执行的状态

18. bmvpu_dec_set_logging_threshold

[功能和说明]

该接口用于设置 SDK 调试信息的打印等级。

[函数名]

`void bmvpu_dec_set_logging_threshold(BmVpuDecLogLevel log_level)` [参数说明]

- `BmVpuDecLogLevel log_level` 设置打印等级。
- `BmVpuDecLogLevel` 为枚举类型，定义如下：

类型	含义
BMVPU_DEC_LOG_LEVEL_NONE	不打印任何调试信息
BMVPU_DEC_LOG_LEVEL_ERR	打印 ERR 类调试信息
BMVPU_DEC_LOG_LEVEL_WARN	打印 ERR 和 WARN 类调试信息
BMVPU_DEC_LOG_LEVEL_INFO	打印 ERR、WARN 和 INFO 类调试信息
BMVPU_DEC_LOG_LEVEL_TRACE	打印 ERR、WARN、INFO 和 TRACE 类调试信息
BMVPU_DEC_LOG_LEVEL_MAX_LOG_LEVEL	打印所有调试信息

[返回值]

返回值表示接口执行的状态

Frame Buffer 计算方法

1. frame buffer count

解码器的 frame buffer 分为两种，compress frame 和 linear frame。所需要的 frame buffer 的数量和种类由输入码流和解码器输出数据的类型（wtlFormat）决定。 解码器的输出数据类型通过 BMVidDecParam 的 wtlFormat 参数来设置。

（1）压缩模式（BMDEC_OUTPUT_COMPRESSED）

压缩模式下，只需要申请 compress frame data 和 compress frame table。

（2）YUV模式（BMDEC_OUTPUT_UNMAP）

YUV模式下，既需要申请 compress frame data 和 compress frame table 也需要申请 linear frame buffer。

2. compress frame data

(1) buffer count

compress frame 的 count 由 minFrameBufferCount 和 extraFrameBufferCount 来决定

压缩模式

```
compressedFbCount = minFrameBufferCount + extraFrameBufferCount
```

YUV模式，若压缩数据不需要用于后续处理，compressedFbCount 可以缩减

```
compressedFbCount = minFrameBufferCount
```

其中 extraFrameBufferCount 由用户指定，数量必须大于0。

- Frame Buffer 通过 sdk 内部分配，此参数有默认值为 5，若用户不指定该参数，则按照默认参数来分配
- Frame Buffer 由用户在外部分配，此参数必须设置，不设置则会报错。

minFrameBufferCount 因码流而异，是 VPU 解码需要的 Frame Buffer 最小值。此参数可以通过 VPU 分析码流得到，但如需要外部分配内存，则需要用户自己计算。

- 对于 HEVC 码流， minFrameBufferCount 由 VPS 参数中的 vps_max_dec_pic_buffering_minus1 参数决定，minFrameBufferCount = vps_max_dec_pic_buffering_minus1 + 2；
- 对于 AVC 码流，minFrameBufferCount 由 max_dec_frame_buffering 参数决定，minFrameBufferCount = max_dec_frame_buffering + 2；

(2) buffer size

定义 ALIGN_xx（）表示进行 xx 字节对齐。如 ALIGN_16（）表示进行 16字节对齐。 compress frame 的 size 由 图像的宽高决定，具体计算方法如下

```
stride = ALIGN_32(width)

height = ALIGN_32(height)

LumaSize = stride * height

ChromaSize = ALIGN_16(stride / 2) * height

FramebufSize = LumaSize + ChromaSize
```

3. compress frame table

(1) buffer count

table 的数量和 compress frame 匹配，计算方法参考 compress frame data。

(2) buffer size

```
YtableBufferSize = ALIGN_16(height) * ALIGN_256(width) / 32

YtableBufferSize = ALIGN_4096(YtableBufferSize) + 4096

CtableBufferSize = ALIGN_16(height) * ALIGN_256(width / 2) / 32

CtableBufferSize = ALIGN_4096(CtableBufferSize) + 4096
```

4. linear frame buffer

(1) buffer count

linear frame 的 count 由 frameBufDelay 和 extraFrameBufferCount 决定。

```
linearFbCount = frameBufDelay + extraFrameBufferCount + 1
```

- 对于 HEVC 码流， frameBufDelay 由 VPS 参数中的 num_reorder_pics参数决定， frameBufDelay = vps_max_num_reorder_pics + 1
- 对于 AVC 码流，frameBufDelay 由 num_reorder_frames 参数决定， frameBufDelay = num_reorder_frames + 1

(2) buffer size

stride 的对齐方式和 compress frame 一致, height 不进行对齐

```
LumaSize = stride * height

ChromaSize = (stride / 2) * (height / 2)

FramebufSize = LumaSize + ChromaSize * 2
```

二、 bm_video Encode 数据结构 & API说明

bm_video Encode 枚举类型

- BmVpuEncReturnCodes
- BmVpuEncOutputCodes
- BmVpuEncHeaderDataTypes
- BmVpuCodecFormat
- BmVpuEncPixFormat
- BMVpuEncGopPreset
- BMVpuEncMode
- BmVpuMappingFlags

1. BmVpuEncReturnCodes

- 编码器返回值代码。除了BM_VPU_ENC_RETURN_CODE_OK，其他的返回值应该被视为发生错误，此时编码器应该被关闭。

枚举值	返回值	描述
BM_VPU_ENC_RETURN_CODE_OK	0	操作成功完成。
BM_VPU_ENC_RETURN_CODE_ERROR	1	通用错误代码，用作其他错误返回代码不匹配时的通用错误。
BM_VPU_ENC_RETURN_CODE_INVALID_PARAMS	2	输入参数无效。
BM_VPU_ENC_RETURN_CODE_INVALID_HANDLE	3	VPU 编码器句柄无效，内部错误，可能是库中的错误，请报告此类错误。
BM_VPU_ENC_RETURN_CODE_INVALID_FRAMEBUFFER	4	帧缓冲区信息无效，通常发生在将包含无效值的 BmVpuFramebuffer 结构传递给 bmvpu_enc_register_framebuffers() 函数时。
BM_VPU_ENC_RETURN_CODE_INSUFFICIENT_FRAMEBUFFERS	5	注册用于编码的帧缓冲区失败，因为未提供足够的帧缓冲区给 bmvpu_enc_register_framebuffers() 函数。
BM_VPU_ENC_RETURN_CODE_INVALID_STRIDE	6	步幅值无效，例如帧缓冲区的一个步幅值无效。
BM_VPU_ENC_RETURN_CODE_WRONG_CALL_SEQUENCE	7	在不适当的时间调用函数。
BM_VPU_ENC_RETURN_CODE_TIMEOUT	8	操作超时。
BM_VPU_ENC_RETURN_CODE_RESEND_FRAME	9	重复送帧。
BM_VPU_ENC_RETURN_CODE_ENC_END	10	编码结束。
BM_VPU_ENC_RETURN_CODE_END	11	编码结束。

2. BmVpuEncOutputCodes

- 编码器内部输出代码。这些代码可以通过按位 OR 进行组合，通过使用按位 AND 检查 `bmvpu_enc_encode()` 返回的 `output_codes` 位掩码，来确认编码器状态。

枚举值	返回值	描述
BM_VPU_ENC_OUTPUT_CODE_INPUT_USED	1 << 0	表示输入数据已被使用。如果未设置该标志位，则编码器尚未使用输入数据，因此请将其再次输入给编码器，直到此标志位被设置或返回错误。

枚举值	返回值	描述
BM_VPU_ENC_OUTPUT_CODE_ENCODED_FRAME_AVAILABLE	1 << 1	表示现在有一个完全编码的帧可用。传递给 `bmvpu_enc_encode()` 的 `encoded_frame` 参数包含有关此帧的信息。
BM_VPU_ENC_OUTPUT_CODE_CONTAINS_HEADER	1 << 2	表示编码帧中的数据还包含头信息，如 h.264 的 SPS/PSS。头信息始终放置在编码数据的开头，如果未设置 `BM_VPU_ENC_OUTPUT_CODE_ENCODED_FRAME_AVAILABLE`，则该标志位不会被设置。

3. BmVpuEncHeaderDataTypes

- 定义了编码器头部数据的不同类型

枚举值	返回值	描述
BM_VPU_ENC_HEADER_DATA_TYPE_VPS_RBSP	0	视频参数集 (VPS) 的 RBSP（Raw Byte Sequence Payload）数据类型。
BM_VPU_ENC_HEADER_DATA_TYPE_SPS_RBSP	1	序列参数集 (SPS) 的 RBSP 数据类型。
BM_VPU_ENC_HEADER_DATA_TYPE_PPS_RBSP	2	图像参数集 (PPS) 的 RBSP 数据类型。

4. BmVpuCodecFormat

- 编码器支持的编码格式

枚举值	返回值	描述
BM_VPU_CODEC_FORMAT_H264	0	编码类型 h264
BM_VPU_CODEC_FORMAT_H265	1	编码类型 h265

5. BmVpuEncPixFormat

- 编码器输入 yuv 格式
- 目前仅支持 nv12，nv21，yuv420p

枚举值	返回值	描述
BM_VPU_ENC_PIX_FORMAT_YUV420P	0	planar 4:2:0
BM_VPU_ENC_PIX_FORMAT_YUV422P	1	planar 4:2:2
BM_VPU_ENC_PIX_FORMAT_YUV444P	3	planar 4:4:4
BM_VPU_ENC_PIX_FORMAT_YUV400	4	8位灰度图像
BM_VPU_ENC_PIX_FORMAT_NV12	5	semi-planar 4:2:0
BM_VPU_ENC_PIX_FORMAT_NV16	6	semi-planar 4:2:2
BM_VPU_ENC_PIX_FORMAT_NV24	7	semi-planar 4:4:4

6. BMVpuEncGopPreset

- 编码器输入 gop_preset设置

枚举值	返回值	描述
BM_VPU_ENC_GOP_PRESET_ALL_I	1	全I帧模式 gopsize=1
BM_VPU_ENC_GOP_PRESET_IPP	2	全IP帧模式 gopsize=1
BM_VPU_ENC_GOP_PRESET_IBBB	3	全IB帧模式 gopsize=1

枚举值	返回值	描述
BM_VPU_ENC_GOP_PRESET_IBPBP	4	全IBP帧模式 gopsize=2
BM_VPU_ENC_GOP_PRESET_IBBBP	5	全IBP帧模式 gopsize=4
BM_VPU_ENC_GOP_PRESET_IPPPP	6	全IP帧模式 gopsize=4
BM_VPU_ENC_GOP_PRESET_IBBBB	7	全IB帧模式 gopsize=4
BM_VPU_ENC_GOP_PRESET_RA_IB	8	Random IB帧模式 gopsize=8

7. BMVpuEncMode

- 编码器输入 编码模式

枚举值	返回值	描述
BM_VPU_ENC_CUSTOM_MODE	0	自定义模式
BM_VPU_ENC_RECOMMENDED_MODE	1	推荐模式（慢编码速度，最高画质）
BM_VPU_ENC_BOOST_MODE	2	提升模式（正常编码速度，正常画质）
BM_VPU_ENC_FAST_MODE	3	快速模式（高编码速度，低画质）

8. BmVpuMappingFlags

- 用于使用vpu_EncMmap()函数时指定映射类型。

枚举值	返回值	描述
BM_VPU_MAPPING_FLAG_WRITE	1 << 0	可写权限标志
BM_VPU_MAPPING_FLAG_READ	1 << 1	可读权限标志

bm_video Encode 数据结构

- BmVpuEncH264Params
- BmVpuEncH265Params
- BmVpuEncOpenParams
- BmVpuEncInitialInfo
- BmCustomMapOpt
- BmVpuEncParams
- BmVpuEncoder
- BmVpuFbInfo
- BmVpuEncodedFrame
- BmVpuEncDMABuffer
- BmVpuRawFrame
- BmVpuFramebuffer

1. BmVpuEncH264Params

- 定义了新的 H.264 编码器实例的参数

成员变量	类型	描述
enable_transform8x8	int	启用 8x8 帧内预测和 8x8 变换。默认值为 1。

2. BmVpuEncH265Params

- 定义了新的 H.265 编码器实例的参数

成员变量	类型	描述
enable_tmvp	int	启用时域运动矢量预测。默认值为 1。

成员变量	类型	描述
enable_wpp	int	启用线性缓冲区模式的波前并行处理。默认值为 0。
enable_sao	int	如果设置为 1，则启用 SAO；如果设置为 0，则禁用。默认值为 1。
enable_strong_intra_smoothing	int	启用对带有少量 AC 系数的区域进行强烈的帧内平滑，以防止伪影。默认值为 1。
enable_intra_trans_skip	int	启用帧内 CU 的变换跳过。默认值为 0。
enable_intraNxN	int	启用帧内 NxN PUs。默认值为 1。

3. BmVpuEncOpenParams

- 用于初始化编码器的全局参数，设置编码器的基本属性和编码器类型等。

成员变量	类型	描述
codec_format	BmVpuCodecFormat	指定要生成的编码数据的编码格式。
pix_format	BmVpuEncPixFormat	指定传入帧使用的图像格式。
frame_width	uint32_t	传入帧的宽度（以像素为单位），无需对齐
frame_height	uint32_t	传入帧的高度（以像素为单位），无需对齐
timebase_num	uint32_t	时间基数，以分数形式给出。
timebase_den	uint32_t	时间分母，以分数形式给出。
fps_num	uint32_t	帧率，以分数形式给出。
fps_den	uint32_t	帧率分母，以分数形式给出。
bitrate	int64_t	比特率（以 bps 为单位）。如果设置为 0，则禁用码率控制，而使用常量质量模式。默认值为 100000。
vbv_buffer_size	uint64_t	vbv 缓冲区的大小，以 bit 为单位。仅在启用码率控制时（*BmVpuEncOpenParams* 中的 *bitrate* 非零）使用。0 表示不检查缓冲区大小约束。默认值为 0。
cqp	int	常量质量模式的量化参数。
enc_mode	BMVpuEncMode	编码模式：0 自定义模式，1 推荐的编码器参数（慢编码速度，最高画质），2 提升模式（正常编码速度，正常画质），3 快速模式（高编码速度，低画质）。默认值为 2。
max_num_merge	int	RDO 中的合并候选数（1 或 2）。1：提高编码性能，2：提高编码图像的质量。默认值为 2。
enable_constrained_intra_prediction	int	启用受限帧内预测。如果设置为 1，则启用；如果设置为 0，则禁用。默认值为 0。
enable_wp	int	启用加权预测。默认值为 1。
disable_deblocking	int	如果设置为 1，则禁用去块滤波器。如果设置为 0，则保持启用。默认值为 0。
offset_tc	int	deblocking 滤波器的 Alpha/Tc 偏移。默认值为 0。
offset_beta	int	deblocking 滤波器的 Beta 偏移。默认值为 0。
enable_cross_slice_boundary	int	启用帧内循环滤波中的跨切片边界滤波。默认值为 0。
enable_nr	int	启用降噪。默认值为 1。
h264_params	BmVpuEncH264Params	H.264 编码器参数。(union，从 *h264_params* 和 *h265_params* 中选择一个)
h265_params	BmVpuEncH265Params	H.265 编码器参数。(union，从 *h264_params* 和 *h265_params* 中选择一个)

成员变量	类型	描述
soc_idx	int	仅用于 PCIe 模式。对于 SOC 模式，此值必须为 0。默认值为 0。
gop_preset	BMVpuEncGopPreset	GOP 结构预设选项。 1：全部为 I帧，gopsize = 1； 2：I-P-P，连续 P，循环 gopsize = 1； 3：I-B-B-B，连续 B，循环 gopsize = 1； 4：I-B-P-B-P，gopsize = 2； 5：I-B-B-B-P，gopsize = 4； 6：I-P-P-P-P，连续 P，循环 gopsize = 4； 7：I-B-B-B-B，连续 B，循环 gopsize = 4； 8：随机访问，I-B-B-B-B-B-B-B-B，循环 gopsize = 8。低延迟情况为 1、2、3、6、7。默认值为 5。
intra_period	int	GOP 大小内的帧内图片周期。默认值为 28。
bg_detection	int	启用背景检测。默认值为 0。
mb_rc	int	启用 MB 级/CU 级码率控制。默认值为 1。
delta_qp	int	码率控制的最大 delta QP。默认值为 5。
min_qp	int	码率控制的最小 QP。默认值为 8。
max_qp	int	码率控制的最大 QP。默认值为 51。
roi_enable	int	ROI 编码标志。默认值为 0。
cmd_queue_depth	int	设置命令队列深度，默认值为4，取值范围[1, 4]
timeout	int	编码超时时间，默认为1000ms（即VPU_WAIT_TIMEOUT）
timeout_count	int	编码超时重试次数，默认为40（即VPU_MAX_TIMEOUT_COUNTS）
buffer_alloc_func	BmVpuEncBufferAllocFunc	缓冲区内存分配函数接口
buffer_free_func	BmVpuEncBufferFreeFunc	缓冲区内存释放函数接口
buffer_context	void*	缓冲区上下文信息

4. BmVpuEncInitialInfo

- 初始编码信息，由编码器生成。这个结构体对于实际开始编码至关重要，因为它包含了创建和注册足够的帧缓冲区所需的所有信息。

成员变量	类型	描述
min_num_rec_fb	uint32_t	最小推荐帧缓冲区数量，分配少于此数量可能会影响编码质量。
min_num_src_fb	uint32_t	输入 YUV 数据帧的最小数量，分配少于此数量可能会影响编码。
framebuffer_alignment	uint32_t	物理帧缓冲区地址的对齐要求。
rec_fb	BmVpuFbInfo	用于重建的帧缓冲区大小信息。包括宽度、高度等信息。
src_fb	BmVpuFbInfo	输入 YUV 数据的宽高信息。

5. BmCustomMapOpt

- 自定义映射选项（H.265编码器）

成员变量	类型	描述
roiAvgQp	int	ROI 映射的平均 QP
customRoiMapEnable	int	是否开启 ROI 映射
customLambdaMapEnable	int	是否开启 Lambda 映射

成员变量	类型	描述
customModeMapEnable	int	是否指定 CTU 使用帧间编码，否则跳过
customCoefDropEnable	int	对于每个 CTU，是否设置 TQ 系数为全0，系数全0的 CTU 将被丢弃
addrCustomMap	bmvpu_phys_addr_t	自定义映射缓冲区的起始地址

6. BmVpuEncParams

- 用于配置编码器的运行时参数，**影响每次编码操作的具体行为。**

成员变量	类型	描述
skip_frame	int	默认值为 0,禁用跳帧生成。如果设置为 1，则 VPU 忽略给定的原始帧，而生成一个“跳帧”，它是前一帧的复制。这个跳帧被编码为 P 帧。
forcePicTypeEnable	int	是否强制指定编码帧类型
forcePicType	int	强制指定的编码帧类型（I帧、P帧、B帧、IDR帧、CRA帧），只有当 *forcePicTypeEnable* 为1时有效
acquire_output_buffer	BmVpuEncAcquireOutputBuffer	用于获取输出缓冲区的函数。
finish_output_buffer	BmVpuEncFinishOutputBuffer	用于释放输出缓冲区的函数。
output_buffer_context	void*	传递给上述函数的用户提供的值。
customMapOpt	BmCustomMapOpt*	指向自定义映射选项的指针。

7. BmVpuEncoder

- 具体的编码器实例，会接收BmVpuEncOpenParams的参数信息

成员变量	类型	描述
handle	void*	编码器句柄。
soc_idx	int	Sophon SoC 的索引。对于 PCIE 模式，请参考 /dev/bm-sophonxx 中的编号。对于 SOC 模式，请将其设置为零。
core_idx	int	所有 Sophon SoC 中 VPU 编码器core的统一索引。
codec_format	BmVpuCodecFormat	编码器使用的视频编解码格式。
color_format	BmVpuEncPixFormat	传入帧使用的图像格式。
frame_width	uint32_t	传入帧的宽度（以像素为单位）。
frame_height	uint32_t	传入帧的高度（以像素为单位）。
fps_n	uint32_t	帧率的分子。
fps_d	uint32_t	帧率的分母。
first_frame	int	是否为第一帧。
rc_enable	int	是否启用码率控制。
cqp	int	在禁用码率控制时，使用恒定的量化参数 QP。
work_dmabuffer	BmVpuEncDMABuffer*	用于编码器工作的 DMA 缓冲区。
bs_dmabuffer	BmVpuEncDMABuffer*	用于存储码流的 DMA 缓冲区。
bs_virt_addr	unsigned long long	码流的虚拟地址。
bs_phys_addr	bmvpu_phys_addr_t	码流的物理地址。
num_framebuffers	uint32_t	帧缓冲区的数量。
internal_framebuffers	void*	编码器内部的帧缓冲区。

成员变量	类型	描述
framebuffers	BmVpuFramebuffer*	帧缓冲区。
buffer_mv	BmVpuEncDMABuffer*	用于存储运动矢量的 DMA 缓冲区。
buffer_fbc_y_tbl	BmVpuEncDMABuffer*	用于存储 FBC 亮度表的 DMA 缓冲区。
buffer_fbc_c_tbl	BmVpuEncDMABuffer*	用于存储 FBC 色度表的 DMA 缓冲区。
buffer_sub_sam	BmVpuEncDMABuffer*	用于 ME 的子采样 DMA 缓冲区。
headers_rbsp	uint8_t*	帧头 RBSP 数据。
headers_rbsp_size	size_t	帧头 RBSP 数据的大小。
initial_info	BmVpuEncInitialInfo	编码器的初始信息。
timeout	int	编码超时时间，默认为1000ms（即VPU_WAIT_TIMEOUT）
timeout_count	int	编码超时重试次数，默认为40（即VPU_MAX_TIMEOUT_COUNTS）
video_enc_ctx	void*	编码上下文信息，内部使用

8. BmVpuFbInfo

- 与 `bmvpu_calc_framebuffer_sizes()` 一起使用，用于计算帧缓冲区的大小

成员变量	类型	描述
width	int	帧的宽度，按照 VPU 要求的 16 像素边界对齐。
height	int	帧的高度，按照 VPU 要求的 16 像素边界对齐。
y_stride	int	对齐后的 Y 分量的跨距大小，以字节为单位。
c_stride	int	对齐后的 Cb 和 Cr 分量的跨距大小，以字节为单位（可选）。
y_size	int	Y 分量的 DMA 缓冲区大小，以字节为单位。
c_size	int	Cb 和 Cr 分量的 DMA 缓冲区大小，以字节为单位。
size	int	帧缓冲区 DMA 缓冲区的总大小，以字节为单位。这个值包括所有通道的大小。

9. BmVpuEncodedFrame

- 编码帧的详细信息

成员变量	类型	描述
data	uint8_t*	在解码时，data 必须指向包含码流数据的内存块，编码器不使用。
data_size	size_t	编码数据的大小。在编码时，由编码器设置，表示获取的输出块的大小，以字节为单位。
frame_type	BmVpuEncFrameType	编码帧的帧类型（I、P、B 等）。由编码器填充。仅由编码器使用。
acquired_handle	void*	在编码时由用户定义的 **acquire_output_buffer** 函数生成的句柄。仅由编码器使用。
context	void*	用户定义的指针。编码器不会更改此值。这个指针和相应原始帧的指针将具有相同的值，在编码器中传递。
pts	uint64_t	用户定义的显示时间戳（Presentation Timestamp）。与 *context* 指针一样，编码器只是将其传递到关联的原始帧，并不实际更改其值。
dts	uint64_t	用户定义的解码时间戳（Decoding Timestamp）。与 *pts* 指针一样，编码器只是将其传递到关联的原始帧，并不实际更改其值。
src_idx	int	原始帧的索引。
u64CustomMapPhyAddr	bmvpu_phys_addr_t	用户自定义映射选项的起始地址

成员变量	类型	描述
avg_ctu_qp	int	平均 CTU QP（Quantization Parameter）。

10. BmVpuEncDMABuffer

- 保存 YUV 数据的物理内存.

成员变量	类型	描述
size	unsigned int	物理内存的大小
phys_addr	uint64_t	物理内存的地址
virt_addr	uint64_t	物理内存mmap后的虚拟地址
enable_cache	int	是否开启cache

11. BmVpuRawFrame

- 结构体包含了关于原始、未压缩帧的详细信息

成员变量	类型	描述
framebuffer	BmVpuFramebuffer*	原始帧的帧缓冲区。
context	void*	用户定义的指针。编码器不会更改此值。这个指针和相应编码帧的指针将具有相同的值，在编码器中传递。
pts	uint64_t	用户定义的显示时间戳（Presentation Timestamp）。与 *context* 指针一样，编码器只是将其传递到关联的编码帧，并不实际更改其值。
dts	uint64_t	用户定义的解码时间戳（Decoding Timestamp）。与 *pts* 指针一样，编码器只是将其传递到关联的编码帧，并不实际更改其值。

12. BmVpuFramebuffer

- 帧缓冲区的相关信息，用于容纳视频帧的像素数据，同时用于编码和解码。

成员变量	类型	描述
dma_buffer	BmVpuEncDMABuffer*	保存 YUV 数据的物理内存
myIndex	int	YUV 索引，用户设置，用于释放 YUV 数据
y_stride	unsigned int	Y 通道对齐后的大小
cbcr_stride	unsigned int	UV 通道对齐后的大小
width	unsigned int	编码 YUV 图像的宽
height	unsigned int	编码 YUV 图像的高
y_offset	size_t	Y 通道 offset。相对于缓冲区起始位置，指定每个分量的起始偏移量。以字节为单位指定。
cb_offset	size_t	U 通道 offset。
cr_offset	size_t	V 通道 offset。
already_marked	int	如果帧缓冲区已在编码器中标记为已使用，则设置为 1。仅供内部使用。不要从外部读取或写入。
internal	void*	内部实现定义的数据。不要修改。
context	void*	用户定义的指针，编码器不会修改此值。用法由用户决定，例如，可以用于标识在编码器中包含该帧的帧缓冲区的序号。

bm_video Encode API

1. bmvpu_enc_error_string

[功能和说明]

- 返回编码错误码的具体描述

[函数名]

```
char const * bmvpu_enc_error_string(BmVpuEncReturnCodes code)
```

[参数说明]

- `BmVpuEncReturnCodes code` 编码错误码

2. bmvpu_enc_get_core_idx

[功能和说明]

- 在指定的Sophon SoC上，获取VPU编码器core的唯一索引。

[函数名]

```
int bmvpu_enc_get_core_idx(int soc_idx)
```

[参数说明]

- `int soc_idx` 设备索引号

3. bmvpu_enc_load

[功能和说明] **加载Sophon设备上的视频处理单元（VPU）的编码模块。**

- unload()和load()的调用次数要一致。
- 在对编码器执行任何其他操作之前，必须先加载(load)编码器。
- 同样，在完成所有编码器活动之前，不得卸载(unload)编码器，包括打开编码器实例。

[函数名]

```
int bmvpu_enc_load(int soc_idx)
```

[参数说明]

- `int soc_idx` 设备索引号

4. bmvpu_enc_unload

[功能和说明]

- 卸载(unload)编码器

[函数名]

```
int bmvpu_enc_unload(int soc_idx)
```

[参数说明]

- `int soc_idx` 设备索引号

5. bmvpu_enc_get_bitstream_buffer_info

[功能和说明]

- 该函数得到编码器所需的 bitstream buffer 的大小 (size) 和所需要的 对齐（alignment）值。
- 返回编码器的码流缓冲区所需的物理内存块的对齐方式和大小。
- 用户必须分配至少此大小的 DMA 缓冲区，并且必须根据对齐值对其物理地址进行对齐。
- **需要在bmvpu_enc_open()之前调用**

[函数名]

```
void bmvpu_enc_get_bitstream_buffer_info(size_t *size, uint32_t *alignment)
```

[参数说明]

- `size_t *size` 码流缓冲区所需的物理内存块的大小
- `uint32_t *alignment` 码流缓冲区所需的物理内存块的对齐方式

6. bmvpu_enc_set_default_open_params

[功能和说明]

- 设置编码器的默认变量，用于编码器初始化时传递参数
- 如果调用方只想修改几个成员变量（或者不做修改），可以调用此函数
- **需要在 bmvpu_enc_open 之前调用**

[函数名]

```
void bmvpu_enc_set_default_open_params(BmVpuEncOpenParams *open_params, BmVpuCodecFormat
codec_format)
```

[参数说明]

- `BmVpuEncOpenParams *open_params` 用于返回编码器的参数
- `BmVpuCodecFormat codec_format` 编码器选择，h264 或 h265

7. bmvpu_fill_framebuffer_params

[功能和说明]

- 根据传入的 *fb_info* 填充 *BmVpuFramebuffer* 结构中的参数
- 可以在此指定帧缓冲区及上下文信息

[函数名]

```
int bmvpu_fill_framebuffer_params(BmVpuFramebuffer *framebuffer, BmVpuFbInfo *fb_info,
BmVpuEncDMABuffer *fb_dma_buffer, int fb_id, void *context)
```

[参数说明]

- `BmVpuFramebuffer *framebuffer` 需要填充的帧缓冲区信息
- `BmVpuFbInfo *fb_info` 从编码器获取的初始化信息，包含所需的帧缓冲区的最小个数及大小
- `BmVpuEncDMABuffer *fb_dma_buffer` 分配给帧缓冲区的保存 YUV 数据的物理内存
- `int fb_id` 用户设置的 YUV 数据索引
- `void *context` 用户设置的上下文信息

8. bmvpu_enc_open

[功能和说明]

- 打开一个新的编码器实例, 设置编码器参数并开始接收视频帧
- **`BmVpuEncOpenParams *open_params` 和 `BmVpuDMABuffer *bs_dmabuffer` 必须不为空**

[函数名]

```
int bmvpu_enc_open(BmVpuEncoder **encoder, BmVpuEncOpenParams *open_params, BmVpuDMABuffer
*bs_dmabuffer, BmVpuEncInitialInfo *initial_info)
```

[参数说明]

- `BmVpuEncoder **encoder` 指向编码器实例的二级指针，接收编码器的属性和视频帧的部分设置, 例如设备 id、缓冲区设置和帧率、宽高等
- `BmVpuEncOpenParams *open_params` 编码器各项参数
- `BmVpuDMABuffer *bs_dmabuffer` 指向码流缓冲区的指针，使用之前已经分配的码流缓冲区
- `BmVpuEncInitialInfo *initial_info` 编码器的初始化信息，返回给用户编码器需要的帧缓冲区最小个数和大小

9. bmvpu_enc_close

[功能和说明]

- 关闭编码器实例
- 多次尝试关闭同一实例会导致未定义的行为

[函数名]

```
int bmvpu_enc_close(BmVpuEncoder *encoder)
```

[参数说明]

- `BmVpuEncoder *encoder` 视频编码器实例

10. bmvpu_enc_encode

[功能和说明]

- 使用给定的编码参数对给定的原始输入帧进行编码。*encoded_frame* 填充有关于所得到的编码输出帧的信息。
- 编码的帧数据本身被存储在由用户提供的函数（在 *encoding_params* 中被设置为 *acquire_output_buffer* 和 *finish_output_buffer* 函数指针）分配的缓冲区中

[函数名]

```
int bmvpu_enc_encode(BmVpuEncoder *encoder, BmVpuRawFrame const *raw_frame, BmVpuEncodedFrame
*encoded_frame, BmVpuEncParams *encoding_params, uint32_t *output_code)
```

[参数说明]

- `BmVpuEncoder *encoder` 视频编码器实例
- `BmVpuRawFrame const *raw_frame` 原始视频帧，包括帧数据、时间戳等。
- `BmVpuEncodedFrame *encoded_frame` 编码后的视频帧，包括帧数据、帧类型、时间戳等。
- `BmVpuEncParams *encoding_params` 用于编码的参数
- `uint32_t *output_code` 返回输出状态代码

11. bmvpu_enc_dma_buffer_allocate

[功能和说明]

- 根据用户指定的size分配设备内存。

[函数名]

```
int bmvpu_enc_dma_buffer_allocate(int vpu_core_idx, BmVpuEncDMABuffer *buf, unsigned int size)
```

[参数说明]

- `int vpu_core_idx` 输入参数，指定编码器所在core的索引
- `BmVpuEncDMABuffer *buf` 输出参数，函数执行后，将会填充该结构体的 *phys_addr*、*size*、*enable_cache* 成员变量
- `unsigned int size` 输入参数，以字节为单位，指定需要的缓冲区大小

[返回值]

- 返回BM_SUCESS(=0)表示分配成功 ，否则分配失败

12. bmvpu_enc_dma_buffer_deallocate

[功能和说明]

- 释放由 **bmvpu_enc_dma_buffer_allocate** 函数分配的设备内存。

[函数名]

```
int bmvpu_enc_dma_buffer_deallocate(int vpu_core_idx, BmVpuEncDMABuffer *buf)
```

[参数说明]

- `int vpu_core_idx` 输入参数，指定编码器所在core的索引
- `BmVpuDMABuffer *buf` 输入参数，调用前用户必须要填充该结构体的 *phys_addr*、*size*、*virt_addr* 成员变量

[返回值]

- 返回BM_SUCESS(=0)表示执行成功 ，否则执行失败

13. bmvpu_enc_dma_buffer_attach

[功能和说明]

- 将用户通过 **bmvpu_enc_dma_buffer_allocate** 函数以外的其它方式申请的设备内存地址绑定至编码器。

[函数名]

```
int bmvpu_enc_dma_buffer_attach(int vpu_core_idx, uint64_t paddr, unsigned int size)
```

[参数说明]

- `int vpu_core_idx` 输入参数，指定编码器所在core的索引
- `uint64_t paddr` 输入参数，由用户通过 **bmvpu_enc_dma_buffer_allocate** 函数以外的其它方式申请的设备内存地址
- `unsigned int size` 输入参数，该块设备内存大小(byte)

[返回值]

- 返回BM_SUCESS(=0)表示执行成功 ，否则执行失败

14. bmvpu_enc_dma_buffer_deattach

[功能和说明]

- 将用户通过 **bmvpu_enc_dma_buffer_attach** 函数绑定的设备内存解绑。

[函数名]

```
int bmvpu_enc_dma_buffer_deattach(int vpu_core_idx, uint64_t paddr, unsigned int size)
```

[参数说明]

- `int vpu_core_idx` 输入参数，指定编码器所在core的索引
- `uint64_t paddr` 输入参数，由用户通过 **bmvpu_enc_dma_buffer_attach** 函数绑定的设备内存物理地址
- `unsigned int size` 输入参数，该块设备内存大小(byte)

[返回值]

- 返回BM_SUCESS(=0)表示执行成功 ，否则执行失败

15. bmvpu_dma_buffer_map

[功能和说明]

- 将对应core上申请的设备内存映射到系统内存。

[函数名]

```
int bmvpu_dma_buffer_map(int vpu_core_idx, BmVpuEncDMABuffer *buf, int port_flag)
```

[参数说明]

- `int vpu_core_idx` 输入参数，指定编码器所在core的索引
- `BmVpuEncDMABuffer *buf` 输入参数，指定设备内存的地址、大小等信息
- `int port_flag` 输入参数，配置 **mmap** 内存可读(*BM_VPU_MAPPING_FLAG_READ*)或可写(*BM_VPU_MAPPING_FLAG_WRITE*)

[返回值]

- 返回BM_SUCESS(=0)表示执行成功 ，否则执行失败

16. bmvpu_dma_buffer_unmap

[功能和说明]

- 对某个core上映射过的设备内存解除映射

[函数名]

```
int bmvpu_dma_buffer_unmap(int vpu_core_idx, BmVpuEncDMABuffer *buf)
```

[参数说明]

- `int vpu_core_idx` 输入参数，指定编码器所在core的索引
- `BmVpuEncDMABuffer *buf` 输入参数，指定设备内存的地址、大小等信息

[返回值]

- 返回BM_SUCESS(=0)表示执行成功 ，否则执行失败

17. bmvpu_enc_dma_buffer_flush

[功能和说明]

- 对已分配的设备内存进行flush操作。

[函数名]

```
int bmvpu_enc_dma_buffer_flush(int vpu_core_idx, BmVpuEncDMABuffer *buf)
```

[参数说明]

- `int vpu_core_idx` 输入参数，指定编码器所在core的索引
- `BmVpuEncDMABuffer *buf` 输入参数，调用前用户至少要填充该结构体的 *phys_addr*、*size* 成员变量

[返回值]

- 返回BM_SUCESS(=0)表示执行成功，否则执行失败

18. bmvpu_enc_dma_buffer_invalidate

[功能和说明]

- 对已分配的设备内存进行invalid操作。

[函数名]

```
int bmvpu_enc_dma_buffer_invalidate(int vpu_core_idx, BmVpuEncDMABuffer *buf)
```

[参数说明]

- `int vpu_core_idx` 输入参数，指定编码器所在core的索引
- `BmVpuEncDMABuffer *buf` 输入参数，调用前用户至少要填充该结构体的 *phys_addr*、*size* 成员变量

[返回值]

- 返回BM_SUCESS(=0)表示执行成功，否则执行失败

19. bmvpu_enc_dma_buffer_get_physical_address

[功能和说明]

- 返回已分配的设备内存的地址。

[函数名]

```
uint64_t bmvpu_enc_dma_buffer_get_physical_address(BmVpuEncDMABuffer *buf)
```

[参数说明]

- `BmVpuEncDMABuffer *buf` 输入参数，已分配的设备内存

[返回值]

- 已分配的设备内存的物理地址

20. bmvpu_enc_dma_buffer_get_size

[功能和说明]

- 返回已分配的设备内存的大小。

[函数名]

```
unsigned int bmvpu_enc_dma_buffer_get_size(BmVpuEncDMABuffer *buf)
```

[参数说明]

- `BmVpuDMABuffer *buf` 输入参数

[返回值]

- 已分配的设备内存的大小

21. bmvpu_enc_upload_data

[功能和说明]

- 向使用 **bmvpu_enc_dma_buffer_allocate()** 分配的设备内存地址传输数据。

[函数名]

```
int bmvpu_enc_upload_data(int vpu_core_idx, const uint8_t* host_va, int host_stride, uint64_t vpu_pa,
int vpu_stride, int width, int height)
```

[参数说明]

- `int vpu_core_idx` 输入参数，指定编码器所在core的索引
- `const uint8_t* host_va` 输入参数, 待传输数据的host端虚拟地址
- `int host_stride` 输入参数，host端的数据跨距
- `uint64_t vpu_pa` 输入参数，传输数据的目标物理地址
- `int vpu_stride` 输入参数，device端的数据跨距
- `int width` 输入参数，数据宽度
- `int height` 输入参数，数据高度

[返回值]

- 返回BM_SUCESS(=0)表示执行成功 ，否则执行失败

22. bmvpu_enc_download_data

[功能和说明]

- 从 **bmvpu_enc_dma_buffer_allocate()** 分配的设备内存地址向host端传输数据。

[函数名]

```
int bmvpu_enc_download_data(int vpu_core_idx, uint8_t* host_va, int host_stride, uint64_t vpu_pa, int vpu_stride, int width, int height)
```

[参数说明]

- `int vpu_core_idx` 输入参数，指定编码器所在core的索引
- `const uint8_t* host_va` 输入参数, 传输数据的目标地址
- `int host_stride` 输入参数，host端的数据跨距
- `uint64_t vpu_pa` 输入参数，传输数据的物理地址
- `int vpu_stride` 输入参数，device端的数据跨距
- `int width` 输入参数，数据宽度
- `int height` 输入参数，数据高度

[返回值]

- 返回BM_SUCESS(=0)表示执行成功 ，否则执行失败

三、 JPU通用结构体

JPU枚举类型

- BmJpuLogLevel
- BmJpuImageFormat
- BmJpuColorFormat
- BmJpuChromaFormat
- BmJpuRotateAngle
- BmJpuMirrorDirection

1. BmJpuLogLevel

枚举变量	描述
BM_JPU_LOG_LEVEL_ERROR	日志等级ERROR
BM_JPU_LOG_LEVEL_WARNING	日志等级WARNING
BM_JPU_LOG_LEVEL_INFO	日志等级INFO
BM_JPU_LOG_LEVEL_DEBUG	日志等级DEBUG
BM_JPU_LOG_LEVEL_LOG	日志等级LOG
BM_JPU_LOG_LEVEL_TRACE	日志等级TRACE

2. BmJpuImageFormat

枚举变量	描述
BM_JPU_IMAGE_FORMAT_YUV420P	YUV 4:2:0 planar
BM_JPU_IMAGE_FORMAT_YUV422P	YUV 4:2:2 planar
BM_JPU_IMAGE_FORMAT_YUV444P	YUV 4:4:4 planar
BM_JPU_IMAGE_FORMAT_NV12	YUV 4:2:0 NV12
BM_JPU_IMAGE_FORMAT_NV21	YUV 4:2:0 NV21
BM_JPU_IMAGE_FORMAT_NV16	YUV 4:2:2 NV16
BM_JPU_IMAGE_FORMAT_NV61	YUV 4:2:2 NV61
BM_JPU_IMAGE_FORMAT_GRAY	YUV 4:0:0
BM_JPU_IMAGE_FORMAT_RGB	RGB 8:8:8 packed, for opencv

3. BmJpuColorFormat

枚举变量	描述
BM_JPU_COLOR_FORMAT_YUV420	YUV 4:2:0 format
BM_JPU_COLOR_FORMAT_YUV422_HORIZONTAL	YUV 4:2:2 format
BM_JPU_COLOR_FORMAT_YUV422_VERTICAL	YUV 2:2:4 format (JPU中定义的一种格式，很少使用)
BM_JPU_COLOR_FORMAT_YUV444	YUV 4:4:4 format
BM_JPU_COLOR_FORMAT_YUV400	YUV 4:0:0 format

4. BmJpuChromaFormat

枚举变量	描述
BM_JPU_CHROMA_FORMAT_CBCR_SEPARATED	Cb、Cr分量非交织，分别存放在不同的通道
BM_JPU_CHROMA_FORMAT_CBCR_INTERLEAVE	Cb、Cr分量交织，存放在一个通道中交错排列，Cb在前，Cr在后
BM_JPU_CHROMA_FORMAT_CRCB_INTERLEAVE	Cb、Cr分量交织，存放在一个通道中交错排列，Cr在前，Cb在后

5. BmJpuRotateAngle

枚举变量	描述
BM_JPU_ROTATE_NONE	不做旋转
BM_JPU_ROTATE_90	逆时针旋转90度
BM_JPU_ROTATE_180	逆时针旋转180度
BM_JPU_ROTATE_270	逆时针旋转270度

6. BmJpuMirrorDirection

枚举变量	描述
BM_JPU_MIRROR_NONE	不做镜像
BM_JPU_MIRROR_VER	竖直方向镜像
BM_JPU_MIRROR_HOR	水平方向镜像
BM_JPU_MIRROR_HOR_VER	水平和竖直方向镜像

JPU通用结构体

- BmJpuFramebuffer
- BmJpuFramebufferSizes
- BmJpuRawFrame

1. BmJpuFramebuffer

成员变量	类型	描述
y_stride	unsigned int	Y 分量的步幅（Stride），单位: byte。
cbcr_stride	unsigned int	Cb 和 Cr 分量的步幅（Stride），单位: byte。
dma_buffer	bm_device_mem_t*	用于存放 YUV 数据的一块设备内存，由 bmlib 分配。
y_offset	size_t	Y 分量起始地址相对于 dma_buffer 中物理地址的偏移量，单位: byte。
cb_offset	size_t	Cb 分量起始地址相对于 dma_buffer 中物理地址的偏移量，单位: byte。
cr_offset	size_t	Cr 分量起始地址相对于 dma_buffer 中物理地址的偏移量，单位: byte。
context	void*	用户定义的指针，库不会更改此值。使用方法由用户决定，例如：用于保存解码上下文信息。
already_marked	int	如果帧缓冲区已标记为已显示，则设置为 1 。仅供内部使用，请不要在外部修改。
internal	void*	内部定义的数据。请不要修改。

2. BmJpuFramebufferSizes

成员变量	类型	描述
aligned_frame_width	unsigned int	帧的宽度（单位: pixel），按照 64 像素边界对齐。
aligned_frame_height	unsigned int	帧的高度（单位: pixel），按照 16 像素边界对齐。
y_stride	unsigned int	Y 分量的跨度大小（单位: byte），包含了对齐要求。
cbcr_stride	unsigned int	Cb 和 Cr 分量的跨度大小（单位: byte），包含了对齐要求。Cb和Cr分量始终使用相同的跨度，因此它们共享相同的值。
y_size	unsigned int	Y 分量的 DMA 缓冲区大小（单位: byte）。
cbcr_size	unsigned int	Cb 和 Cr 分量的 DMA 缓冲区大小（单位: byte）。Cb和Cr分量始终使用相同的大小，因此它们共享相同的值。
total_size	unsigned int	帧缓冲区 DMA 缓冲区的总大小，包括所有分量的大小、对齐和填充字节。
image_format	BmJpuImageFormat	帧的图像格式，可选项请参考 **BmJpuImageFormat** 定义。

3. BmJpuRawFrame

成员变量	类型	描述
framebuffer	BmJpuFramebuffer*	在解码时，指向包含解码后的原始帧的帧缓冲区。在编码时，指向包含要编码的原始帧的帧缓冲区。
context	void*	用户定义的指针。库不会修改此值。该指针和相应的编码帧中的context指向相同的地址，库会将它们传递给用户。使用方法由用户决定，例如：可以用于标识与此编码帧关联的原始帧。
pts	uint64_t	用户定义的时间戳（PTS - Presentation Time Stamp）。 通常用于关联原始/编码帧的时间戳信息。库只是将其传递给关联的编码帧，不会实际更改其值。
dts	uint64_t	用户定义的时间戳（DTS - Decoding Time Stamp）。 通常用于关联原始/编码帧的时间戳信息。库只是将其传递给关联的编码帧，不会实际更改其值。

四、 jpeg Decode 数据结构 & API说明

jpeg Decode 结构体

- BmJpuJPEGDecInfo
- BmJpuJPEGDecoder
- BmJpuDecOpenParams
- BmJpuDecInitialInfo
- BmJpuDecReturnCodes

1. BmJpuJPEGDecInfo

- JPU帧缓冲区的宽度和高度与内部边界对齐。
- 帧aligned_frame由实际图像像素actual_frame和额外的填充像素组成。

成员变量	类型	描述
aligned_frame_width	unsigned int	对齐后的宽度，包括额外的填充像素，已对齐到内部边界
aligned_frame_height	unsigned int	对齐后的高度，包括额外的填充像素，已对齐到内部边界
actual_frame_width	unsigned int	实际帧宽度，不包括额外的填充像素
actual_frame_height	unsigned int	实际帧高度，不包括额外的填充像素
y_stride	unsigned int	Y分量的跨度（每行占用的字节数）
cbcr_stride	unsigned int	Cr和Cb分量的跨度，通常与Y分量相同。 Cb和Cr的跨度经常是相同的
y_size	unsigned int	Y分量在帧缓冲区中的大小（单位: byte）
cbcr_size	unsigned int	Cr和Cb分量在帧缓冲区中的大小（单位: byte）
y_offset	unsigned int	Y分量在帧缓冲区中的偏移量（单位: byte）
cb_offset	unsigned int	Cr分量在帧缓冲区中的偏移量（单位: byte）
cr_offset	unsigned int	Cb分量在帧缓冲区中的偏移量（单位: byte）
framebuffer	BmJpuFramebuffer*	包含解码后的帧像素数据的指针和相关信息
image_format	BmJpuImageFormat	解码后的帧的图像格式
framebuffer_recycle	bool	表示是否开启framebuffer_recycle模式
framebuffer_size	size_t	帧缓冲区的总大小（单位: byte）

2. BmJpuJPEGDecoder

成员变量	类型	描述
decoder	BmJpuDecoder*	指向内部JPEG解码器的指针
bitstream_buffer	bm_device_mem_t*	码流缓冲区的指针，用于存储JPEG码流数据。 **bm_device_mem_t: bmlib内存描述符**
bitstream_buffer_size	size_t	码流缓冲区的大小
bitstream_buffer_alignment	unsigned int	码流缓冲区的内存对齐要求（单位: byte）
initial_info	BmJpuDecInitialInfo	包含JPEG解码器的初始化信息
framebuffers	BmJpuFramebuffer*	记录解码器中帧缓冲区中各分量的地址偏移及大小
framebuffer_addrs	bm_jpu_phys_addr_t*	存储解码器中帧缓冲区的设备内存对应的物理地址，由 bmlib 分配
framebuffer_size	size_t	记录解码器中帧缓冲区的总内存大小（各缓冲区大小相同）

成员变量	类型	描述
num_framebuffers	unsigned int	解码器申请的帧缓冲区总帧数。
num_extra_framebuffers	unsigned int	解码需要的帧缓冲区额外帧数，通常为 0，暂未使用。
calculated_sizes	BmJpuFramebufferSizes	记录对齐后的帧缓冲区大小信息。
raw_frame	BmJpuRawFrame	表示原始帧数据，用于存储图像的原始数据和时间戳。
device_index	int	设备索引，标识使用的解码设备的索引
opaque	void*	用户自定义数据
rotationEnable	int	是否启用图像旋转。0 表示不旋转，1 表示旋转。
rotationAngle	BmJpuRotateAngle	旋转角度。可选项请参考 **BmJpuRotateAngle** 定义
mirrorEnable	int	是否启用图像镜像。0 表示不镜像，1 表示镜像。
mirrorDirection	BmJpuMirrorDirection	镜像方向。可选项请参考 **BmJpuMirrorDirection** 定义
framebuffer_recycle	bool	表示是否开启framebuffer_recycle模式
bitstream_from_user	bool	表示码流缓冲区的设备内存是否由外部分配
framebuffer_from_user	bool	表示帧缓冲区的设备内存是否由外部分配

3. BmJpuDecOpenParams

成员变量	类型	描述
min_frame_width	unsigned int	解码器能够处理的最小宽度（单位: pixel），设为0表示无限制。
min_frame_height	unsigned int	解码器能够处理的最小高度（单位: pixel），设为0表示无限制。
max_frame_width	unsigned int	解码器能够处理的最大宽度（单位: pixel），设为0表示无限制。
max_frame_height	unsigned int	解码器能够处理的最大高度（单位: pixel），设为0表示无限制。
color_format	BmJpuColorFormat	解码器输出的YUV格式，**注意：BM1684/BM1684X上不支持该设置**。
chroma_interleave	BmJpuChromaFormat	色度分量存储方式，可选项请参考 **BmJpuChromaFormat** 定义。
scale_ratio	unsigned int	缩放比例，用于指定解码输出的缩放级别。0 表示不进行缩放，n (取值1-3) 表示等比缩放为 2^n 倍。
bs_buffer_size	size_t	用于码流的 DMA 缓冲区大小，这里记录了存储输入图片需要的字节大小。
buffer	uint8_t*	码流缓冲区的指针。仅在 Windows 环境下使用，在其他环境下已弃用，不建议使用。
device_index	int	设备索引。
rotationEnable	int	是否启用图像旋转。0 表示不旋转，1 表示旋转。
rotationAngle	BmJpuRotateAngle	旋转角度。可选项请参考 **BmJpuRotateAngle** 定义。
mirrorEnable	int	是否启用图像镜像。0 表示不镜像，1 表示镜像。
mirrorDirection	BmJpuMirrorDirection	镜像方向。可选项请参考 **BmJpuMirrorDirection** 定义。
roiEnable	int	是否启用感兴趣区域（ROI）。
roiWidth	int	ROI 的宽度。
roiHeight	int	ROI 的高度。
roiOffsetX	int	ROI 相对图像左上角的水平偏移量。
roiOffsetY	int	ROI 相对图像左上角的垂直偏移量。

成员变量	类型	描述
framebuffer_recycle	bool	是否启用framebuffer_recycle模式。如果开启recycle模式，则解码器会使用固定大小的帧缓冲区，当输入码流的分辨率或格式切换时，不会重新申请设备内存。此时，帧缓冲区的大小由用户指定。
framebuffer_size	size_t	用户指定的帧缓冲区大小，*framebuffer_recycle = 1* 或 *framebuffer_from_user = 1* 时生效，生效时要求该值大于0。
bitstream_from_user	bool	是否由外部分配码流缓冲区设备内存
bs_buffer_phys_addr	bm_jpu_phys_addr_t	用户指定的码流缓冲区的设备内存的物理地址，*bitstream_from_user = 1* 时生效。
framebuffer_from_user	bool	是否由外部分配帧缓冲区设备内存
framebuffer_num	int	用户指定的帧缓冲区个数，*framebuffer_from_user = 1* 时生效，默认为1
framebuffer_phys_addrs	bm_jpu_phys_addr_t*	用户指定的帧缓冲区的设备内存的物理地址，*framebuffer_from_user = 1* 时生效，以数组的形式传入，个数由 *framebuffer_num* 指定。
timeout	int	解码超时时间，默认为2s
timeout_count	int	解码超时重试次数，默认为5

4. BmJpuDecInitialInfo

成员变量	类型	描述
frame_height	unsigned int	帧的高度（单位: pixel）。
		请注意，这些值不一定对齐到 16 像素边界（JPU 帧缓冲区对齐要求）。这些是实际像素内容的宽度和高度。如果需要对齐，可能会在帧的右侧有填充列和在帧的下方有填充行。
frame_width	unsigned int	帧的宽度（单位: pixel）。
		请注意，这些值不一定对齐到 16 像素边界（JPU 帧缓冲区对齐要求）。这些是实际像素内容的宽度和高度。如果需要对齐，可能会在帧的右侧有填充列和在帧的下方有填充行。
min_num_required_framebuffers	unsigned int	解码需要的缓冲区最小个数，调用者必须向解码器注册至少该数量的帧缓冲区。
image_format	BmJpuImageFormat	解码后帧的图像格式。可选项请参考 **BmJpuImageFormat** 定义。
framebuffer_alignment	unsigned int	帧缓冲区的内存对齐要求（单位: byte）。
roiFrameHeight	int	感兴趣区域 (ROI) 帧的高度（单位: pixel）。
roiFrameWidth	int	感兴趣区域 (ROI) 帧的宽度（单位: pixel）。

5. BmJpuDecReturnCodes

- 解码器返回代码。除BM_JPU_DEC_RETURN_CODE_OK外，这些都应被视为错误，返回时应关闭解码器。

错误码	返回值	描述
BM_JPU_DEC_RETURN_CODE_OK	0	操作成功完成。
BM_JPU_DEC_RETURN_CODE_ERROR	1	通用错误码，用于当其他错误码无法匹配错误时的情况。
BM_JPU_DEC_RETURN_CODE_INVALID_PARAMS	2	输入参数无效。
BM_JPU_DEC_RETURN_CODE_INVALID_HANDLE	3	JPU 解码器句柄无效。这是一个内部错误，很可能是JPU库中的BUG，请报告此类错误。

错误码	返回值	描述
BM_JPU_DEC_RETURN_CODE_INVALID_FRAMEBUFFER	4	帧缓冲区信息无效。通常发生在 *bm_jpu_jpeg_dec_get_info()* 函数中获取的 **BmJpuFramebuffer** 结构包含无效值的情况。
BM_JPU_DEC_RETURN_CODE_INSUFFICIENT_FRAMEBUFFERS	5	使用已注册给解码器的帧缓冲区解码失败，因为未提供足够大小的帧缓冲区。
BM_JPU_DEC_RETURN_CODE_INVALID_STRIDE	6	某个跨度值（例如帧缓冲区的Y跨度值）无效。
BM_JPU_DEC_RETURN_CODE_WRONG_CALL_SEQUENCE	7	在不适当的时间调用函数。这是一个内部错误，很可能是JPU库中的BUG，请报告此类错误。
BM_JPU_DEC_RETURN_CODE_TIMEOUT	8	操作超时。
BM_JPU_DEC_RETURN_CODE_ALREADY_CALLED	9	调用了一个只应在解码会话的持续时间内调用一次的函数。这是一个内部错误，很可能是JPU库中的BUG，请报告此类错误。
BM_JPU_DEC_RETURN_ALLOC_MEM_ERROR	10	分配内存失败。
BM_JPU_DEC_RETURN_OVER_LIMITED	11	超过解码分辨率限制

jpeg Decode API

- bm_jpu_dec_load
- bm_jpu_jpeg_dec_open
- bm_jpu_jpeg_dec_decode
- bm_jpu_jpeg_dec_get_info
- bm_jpu_jpeg_dec_frame_finished
- bm_jpu_jpeg_dec_close
- bm_jpu_dec_unload
- bm_jpu_calc_framebuffer_sizes
- bm_jpu_dec_error_string
- bm_jpu_dec_get_bm_handle
- bm_jpu_jpeg_dec_flush

1. bm_jpu_dec_load

[功能和说明]

- 根据传入的 设备ID 打开指定的解码设备节点，可以通过 bmlib 管理内存分配。

[函数名]

```
BmJpuDecReturnCodes bm_jpu_dec_load(int device_index)
```

[参数说明]

- `int device_index` 解码设备ID。

2. bm_jpu_jpeg_dec_open

[功能和说明]

- 初始化一个JPEG解码器实例。
- 分配额外的帧缓冲区，如果需要的话（例如，用于快速解码多个JPEG图像或者保留解码后的DMA缓冲区）。
- 分配一个码流缓冲区，用于存储JPEG数据以供解码器使用。
- 调用 `bm_jpu_dec_open` 来实际打开解码器并传入必要的回调和参数。
- 如果在任何步骤中出现错误，函数将清理已分配的资源并返回相应的错误码。

[函数名]

```
BmJpuDecReturnCodes bm_jpu_jpeg_dec_open(BmJpuJPEGDecoder **jpeg_decoder,BmJpuDecOpenParams
*open_params, unsigned int num_extra_framebuffers)
```

[参数说明]

- `BmJpuJPEGDecoder **jpeg_decoder` 指向解码器的二级指针，在接口内部完成初始化，返回给用户一个解码器实例。
- `BmJpuDecOpenParams *open_params` 指向解码参数的指针，该结构体包含打开解码器时所需的参数，如设备索引、解码帧配置参数等。
- `unsigned int num_extra_framebuffers` 指示函数分配额外帧缓冲区的数量。这些额外的帧缓冲区用于解码多个JPEG图像或在其他地方保留解码帧的DMA缓冲区。

3. bm_jpu_jpeg_dec_decode

[功能和说明]

- 解码JPEG数据。
- 设置输入JPEG数据块及其大小。
- 注意，这个解码器只支持基线（Baseline）JPEG数据，不支持渐进式（Progressive）编码。

[函数名]

```
BmJpuDecReturnCodes bm_jpu_jpeg_dec_decode(BmJpuJPEGDecoder *jpeg_decoder, uint8_t const *jpeg_data,
size_t const jpeg_data_size, int timeout, int timeout_count)
```

[参数说明]

- `BmJpuJPEGDecoder *jpeg_decoder` 指向JPEG解码器实例的指针。
- `uint8_t const *jpeg_data` 待解码的图像数据。
- `size_t const jpeg_data_size` 待解码的图像数据大小，单位: byte。
- `int timeout` 解码超时时间。
- `int timeout_count` 解码超时重试次数

4. bm_jpu_jpeg_dec_get_info

[功能和说明]

- 从解码器获取解码信息

[函数名]

```
void bm_jpu_jpeg_dec_get_info(BmJpuJPEGDecoder *jpeg_decoder, BmJpuJPEGDecInfo *info)
```

[参数说明]

- `BmJpuJPEGDecoder *jpeg_decoder` 指向JPEG解码器实例的指针。
- `BmJpuJPEGDecInfo *info` 用来存储解码图像的详细信息的结构体。

5. bm_jpu_jpeg_dec_frame_finished

[功能和说明]

- 通知解码器一个解码帧已经被处理完毕，并且相关的资源可以被回收。
- 一旦用户处理完一帧，就必须始终调用此函数，否则JPU无法回收此帧缓冲区，如果解码器内部的帧缓冲区均被占用，将无法继续解码。

[函数名]

```
BmJpuDecReturnCodes bm_jpu_jpeg_dec_frame_finished(BmJpuJPEGDecoder *jpeg_decoder, BmJpuFramebuffer *framebuffer)
```

[参数说明]

- `BmJpuJPEGDecoder *jpeg_decoder` 指向JPEG解码器实例的指针。
- `BmJpuFramebuffer *framebuffer` 包含了已解码的图像数据的帧缓冲区。

6. bm_jpu_jpeg_dec_close

[功能和说明]

- 用于关闭JPEG解码器实例，并释放资源。

[函数名]

```
BmJpuDecReturnCodes bm_jpu_jpeg_dec_close(BmJpuJPEGDecoder *jpeg_decoder)
```

[参数说明]

- `BmJpuJPEGDecoder *jpeg_decoder` 指向JPEG解码器实例的指针。

7. bm_jpu_dec_unload

[功能和说明]

- 释放指定的解码设备节点

[函数名]

```
BmJpuDecReturnCodes bm_jpu_dec_unload(int device_index)
```

[参数说明]

- `int device_index` 解码设备ID

8. bm_jpu_calc_framebuffer_sizes

[功能和说明]

- 用于计算存放解码数据的帧缓冲区的各分量对齐后的跨度、大小等信息

[函数名]

```
BmJpuDecReturnCodes bm_jpu_calc_framebuffer_sizes(unsigned int frame_width, unsigned int frame_height, unsigned int framebuffer_alignment, BmJpuImageFormat image_format, BmJpuFramebufferSizes *calculated_sizes)
```

[参数说明]

- `unsigned int frame_width` 图像实际宽度
- `unsigned int frame_height` 图像实际高度
- `unsigned int framebuffer_alignment` 解码帧缓冲区内存对齐要求，设为0或1表示不对齐，单位: byte
- `BmJpuImageFormat image_format` 图像格式
- `BmJpuFramebufferSizes *calculated_sizes` 计算后的帧缓冲区大小信息

9. bm_jpu_dec_error_string

[功能和说明]

- 返回解码错误码的具体描述

[函数名]

```
char const * bm_jpu_dec_error_string(BmJpuDecReturnCodes code)
```

[参数说明]

- `BmJpuDecReturnCodes code` 解码错误码

10. bm_jpu_dec_get_bm_handle

[功能和说明]

- 获取该解码设备上bmlib的句柄

[函数名]

```
bm_handle_t bm_jpu_dec_get_bm_handle(int device_index)
```

[参数说明]

- `int device_index` 解码设备ID

11. bm_jpu_jpeg_dec_flush

[功能和说明]

- 刷新解码器帧缓冲区状态，解码器内部所有帧缓冲区将解除占用，可用于后续解码

[函数名]

```
BmJpuDecReturnCodes bm_jpu_jpeg_dec_flush(BmJpuJPEGDecoder *jpeg_decoder)
```

[参数说明]

- `BmJpuJPEGDecoder *jpeg_decoder` 指向JPEG解码器实例的指针。

五、 jpeg Encode 数据结构 & API说明

jpeg Encode 结构体

- BmJpuJPEGEncParams
- BmJpuJPEGEncoder
- BmJpuEncInitialInfo
- BmJpuEncReturnCodes

1. BmJpuJPEGEncParams

成员变量	类型	描述
frame_width	unsigned int	输入帧的宽度，这是实际大小，这些大小不能为零。如果需要，它们将在内部对齐。
frame_height	unsigned int	输入帧的高度，这是实际大小，这些大小不能为零。如果需要，它们将在内部对齐。
quality_factor	unsigned int	JPEG编码的图像质量因子。1表示最佳压缩，100表示最佳质量。
image_format	BmJpuImageFormat	输入帧的图像格式
acquire_output_buffer	BmJpuEncAcquireOutputBuffer	获取编码码流输出 buffer 的回调函数, 具体定义请参考 **BmJpuEncAcquireOutputBuffer** 。
finish_output_buffer	BmJpuEncFinishOutputBuffer	释放上述 buffer 的回调函数，具体定义请参考 **BmJpuEncFinishOutputBuffer**。
write_output_data	BmJpuWriteOutputData	指定编码码流输出方式的回调函数，如：写入文件或写入指定的内存地址。具体定义请参考 **BmJpuWriteOutputData**
		使用此函数将不会调用 `acquire_output_buffer` 和 `finish_output_buffer` 函数。
output_buffer_context	void*	保存输出数据的上下文，将传递给 `acquire_output_buffer`、`finish_output_buffer` 和 `write_output_data` 函数。
rotationEnable	int	是否启用图像旋转。0 表示不旋转，1 表示旋转。
rotationAngle	BmJpuRotateAngle	旋转角度。可选项请参考 **BmJpuRotateAngle** 定义。
mirrorEnable	int	是否启用图像镜像。0 表示不镜像，1 表示镜像。
mirrorDirection	BmJpuMirrorDirection	镜像方向。可选项请参考 **BmJpuMirrorDirection** 定义。
bs_in_device	int	表示编码的码流数据输出到设备内存还是系统内存，0表示输出到系统内存，1表示输出到设备内存。
timeout	int	编码超时时间，默认为2s。
timeout_count	int	编码超时重试次数，默认为5。
bs_buffer_phys_addr	bm_jpu_phys_addr_t	（可选）用户外部分配的码流缓冲区的设备内存的物理地址。
bs_buffer_size	int	（可选）用户外部分配的码流缓冲区的小大。

2. BmJpuJPEGEncoder

成员变量	类型	描述
encoder	BmJpuEncoder *	指向内部JPEG编码器的指针。
bitstream_buffer_addr	bm_jpu_phys_addr_t	码流缓冲区的设备内存的物理地址。
bitstream_buffer_size	size_t	码流缓冲区的大小，单位: byte。

成员变量	类型	描述
bitstream_buffer_alignment	unsigned int	码流缓冲区的对齐要求，单位: byte。
initial_info	BmJpuEncInitialInfo	编码器的初始化信息。
frame_width	unsigned int	输入帧的宽度，单位: pixel。
frame_height	unsigned int	输入帧的高度，单位: pixel。
calculated_sizes	BmJpuFramebufferSizes	由输入帧计算得到的帧缓冲区大小信息。
quality_factor	unsigned int	JPEG 编码的质量因子。1 表示最佳压缩质量，100 表示最佳图像质量。
image_format	BmJpuImageFormat	输入帧的图像格式。
device_index	int	设备索引。
rotationEnable	int	是否启用图像旋转。0 表示不旋转，1 表示旋转。
rotationAngle	BmJpuRotateAngle	旋转角度。可选项请参考 **BmJpuRotateAngle** 定义。
mirrorEnable	int	是否启用图像镜像。0 表示不镜像，1 表示镜像。
mirrorDirection	BmJpuMirrorDirection	镜像方向。可选项请参考 **BmJpuMirrorDirection** 定义。
bitstream_from_user	bool	表示码流缓冲区的设备内存是否由外部分配

3. BmJpuEncInitialInfo

成员变量	类型	描述
min_num_required_framebuffers	unsigned int	编码需要的缓冲区最小个数，调用者必须向编码器注册至少该数量的帧缓冲区。
framebuffer_alignment	unsigned int	帧缓冲区的内存对齐要求（单位: byte）。

4. BmJpuEncReturnCodes

- 编码器返回代码。除BM_JPU_ENC_RETURN_CODE_OK外，这些都应被视为错误，返回时应关闭编码器。

错误码	返回值	描述
BM_JPU_ENC_RETURN_CODE_OK	0	操作成功完成。
BM_JPU_ENC_RETURN_CODE_ERROR	1	通用错误码，用于当其他错误码无法匹配错误时的情况。
BM_JPU_ENC_RETURN_CODE_INVALID_PARAMS	2	输入参数无效。
BM_JPU_ENC_RETURN_CODE_INVALID_HANDLE	3	JPU 编码器句柄无效。这是一个内部错误，很可能是JPU库中的BUG，请报告此类错误。
BM_JPU_ENC_RETURN_CODE_INVALID_FRAMEBUFFER	4	帧缓冲区信息无效。通常发生在传递给 *bm_jpu_jpeg_enc_encode* 函数的 **BmJpuFramebuffer** 结构包含无效值的情况。
BM_JPU_ENC_RETURN_CODE_INSUFFICIENT_FRAMEBUFFERS	5	编码时使用的帧缓冲区无效。这是一个内部错误，很可能是JPU库中的BUG，请报告此类错误。
BM_JPU_ENC_RETURN_CODE_INVALID_STRIDE	6	某个跨度值（例如帧缓冲区的Y跨度值）无效。
BM_JPU_ENC_RETURN_CODE_WRONG_CALL_SEQUENCE	7	在不适当的时间调用函数。这是一个内部错误，很可能是JPU库中的BUG，请报告此类错误。
BM_JPU_ENC_RETURN_CODE_TIMEOUT	8	操作超时。
BM_JPU_ENC_RETURN_CODE_ALREADY_CALLED	9	调用了一个只应在编码会话的持续时间内调用一次的函数。这是一个内部错误，很可能是JPU库中的BUG，请报告此类错误。

错误码	返回值	描述
BM_JPU_ENC_RETURN_CODE_WRITE_CALLBACK_FAILED	10	编码输出回调函数 **BmJpuWriteOutputData** 调用失败
BM_JPU_ENC_RETURN_ALLOC_MEM_ERROR	11	分配内存失败
BM_JPU_ENC_BYTE_ERROR	12	编码数据异常
BM_JPU_ENC_RETURN_BS_BUFFER_FULL	13	码流缓冲区已满

jpeg Encode API

- bm_jpu_enc_load
- bm_jpu_jpeg_enc_open
- bm_jpu_jpeg_enc_encode
- bm_jpu_jpeg_enc_close
- bm_jpu_enc_unload
- bm_jpu_enc_error_string
- bm_jpu_enc_get_bm_handle

1. bm_jpu_enc_load

[功能和说明]

- 根据传入的 设备ID 打开指定的编码设备节点，可以通过 bmlib 管理内存分配。

[函数名]

```
BmJpuEncReturnCodes bm_jpu_enc_load(int device_index)
```

[参数说明]

- `int device_index` 编码设备ID。

2. bm_jpu_jpeg_enc_open

[功能和说明]

- 创建一个JPEG编码器实例，并申请指定大小的码流缓冲区。

[函数名]

```
BmJpuEncReturnCodes bm_jpu_jpeg_enc_open(BmJpuJPEGEncoder **jpeg_encoder, bm_jpu_phys_addr_t
bs_buffer_phys_addr, int bs_buffer_size, int device_index)
```

[参数说明]

- `BmJpuJPEGEncoder **jpeg_encoder` 指向编码器的二级指针，在接口内部完成初始化，返回给用户一个编码器实例。
- `bm_jpu_phys_addr_t bs_buffer_phys_addr` 用户指定的码流缓冲区设备内存物理地址，0表示由编码器内部分配。
- `int bs_buffer_size` 码流缓冲区的大小，0表示默认大小（5MB）。
- `int device_index` 编码设备ID

3. bm_jpu_jpeg_enc_encode

[功能和说明]

- 编码原始输入帧
- JPU编码器仅生成baseline JPEG数据，不支持渐进式编码。

[函数名]

```
BmJpuEncReturnCodes bm_jpu_jpeg_enc_encode(BmJpuJPEGEncoder *jpeg_encoder, BmJpuFramebuffer const
*framebuffer, BmJpuJPEGEncParams const *params, void **acquired_handle, size_t *output_buffer_size)
```

[参数说明]

- `BmJpuJPEGEncoder *jpeg_encoder` 指向 `BmJpuJPEGEncoder` 结构体的指针，它包含了与JPEG编码相关的设置和状态信息
- `mJpuFramebuffer const *framebuffer` 包含要编码的原始输入像素。它的跨度（stride）和偏移（offset）值必须有效，并且其 dma_buffer 指针必须指向包含像素数据的 DMA 缓冲区。
- `BmJpuJPEGEncParams const *params` 包含了JPEG编码的参数，如图像尺寸、质量因子等, params必须填充有效的数值；帧的宽度和高度不能为零。
- `void **acquired_handle` 用于返回获取编码后的图像数据的句柄。
- `size_t *output_buffer_size` 用于返回编码后的图像数据的大小。

4. bm_jpu_jpeg_enc_close

[功能和说明]

- 关闭编码器，释放资源。
- 获取与 jpeg_encoder 相关的设备句柄 handle，通常用于与硬件设备交互。
- 如果 jpeg_encoder 的 bitstream_buffer 不为空（即已分配了码流缓冲区），且设备内存由编码器内部申请，它会释放该码流缓冲区的设备内存，并释放 jpeg_encoder->bitstream_buffer 占用的内存。
- 最后释放 jpeg_encoder 结构体本身的内存，并将 jpeg_encoder 指针设置为 NULL。

[函数名]

```
BmJpuEncReturnCodes bm_jpu_jpeg_enc_close(BmJpuJPEGEncoder *jpeg_encoder)
```

[参数说明]

- `BmJpuJPEGEncoder *jpeg_encoder` 指向JPEG编码器实例的指针。

5. bm_jpu_enc_unload

[功能和说明]

- 释放指定的编码设备节点

[函数名]

```
BmJpuDecReturnCodes bm_jpu_enc_unload(int device_index)
```

[参数说明]

- `int device_index` 编码设备ID

6. bm_jpu_enc_error_string

[功能和说明]

- 返回编码错误码的具体描述

[函数名]

```
char const * bm_jpu_enc_error_string(BmJpuEncReturnCodes code)
```

[参数说明]

- `BmJpuEncReturnCodes code` 编码错误码

7. bm_jpu_enc_get_bm_handle

[功能和说明]

- 获取该编码设备上bmlib的句柄

[函数名]

```
bm_handle_t bm_jpu_enc_get_bm_handle(int device_index)
```

[参数说明]

- `int device_index` 编码设备ID

jpeg Encode Callback

- BmJpuEncAcquireOutputBuffer
- BmJpuEncFinishOutputBuffer
- BmJpuWriteOutputData

1. BmJpuEncAcquireOutputBuffer

[功能和说明]

- 用于申请编码数据的输出buffer，可根据编码器的 *bs_in_device* 配置，选择输出到设备内存还是系统内存。
- 编码完成后，数据会从编码器中的码流缓冲区拷贝到上述申请的buffer中，用户可以从 *acquired_handle* 获取输出数据。
- **BmJpuWriteOutputData** 回调函数为空时执行该操作

[接口实现]

```
typedef void* (*BmJpuEncAcquireOutputBuffer)(void *context, size_t size, void **acquired_handle)
```

[参数说明]

- `void *context` 输出上下文信息，由用户在编码参数中的 **output_buffer_context** 指定
- `size_t size` 申请buffer的大小
- `void **acquired_handle` 用户获取输出数据的句柄，由编码接口中的 **acquired_handle** 参数指定

2. BmJpuEncFinishOutputBuffer

[功能和说明]

- 用于释放 **BmJpuEncAcquireOutputBuffer** 接口申请的buffer，与上述接口配套使用。
- **BmJpuWriteOutputData** 回调函数为空时执行该操作

[接口实现]

```
typedef void (*BmJpuEncFinishOutputBuffer)(void *context, void *acquired_handle)
```

[参数说明]

- `void *context` 输出上下文信息，与 **BmJpuEncAcquireOutputBuffer** 接口中的 **context** 参数相同
- `void *acquired_handle` 需要释放的buffer，等价于 **BmJpuEncAcquireOutputBuffer** 接口中的 ***acquired_handle** 参数值

3. BmJpuWriteOutputData

[功能和说明]

- 用于用户指定编码数据输出方式的回调函数

[接口实现]

```
typedef int (*BmJpuWriteOutputData)(void *context, uint8_t const *data, uint32_t size,
BmJpuEncodedFrame *encoded_frame)
```

[参数说明]

- `void *context` 输出上下文信息，由用户在编码参数中的 **output_buffer_context** 指定
- `uint8_t const *data` 编码器码流缓冲区映射后的虚拟地址，由编码器内部mmap得到
- `uint32_t size` 编码输出码流大小
- `BmJpuEncodedFrame *encoded_frame` 编码器内部生成的编码帧信息，用于保存PTS、DTS等信息（暂未使用）