MI VENC API

Version 2.14

REVISION HISTORY

Revision No.	Description	Date
2.03	● Initial release	04/12/2018
2.04	● 增加 Mjpeg cbr 模式，增加 InterxxPredEn 参数，增加 VUI 参数	01/18/2019
2.05	● 增加 E_MI_VENC_BASE_P_REFTOIDR	02/25/2019
2.06	● 增加获取 FrameRate, Bitrate 信息	03/14/2019
2.07	● 增加 u32RestartMakerPerRowCnt	05/05/2019
2.08	● 增加 Smart Encoding 相关的 API	09/01/2019
2.09	● SSC33X 增加 MI_VENC_SetInputSourceConfig 设置 input ring info	10/29/2019
2.10	● SSC33X 修改 MI_VENC_AllocCustomMap, MI_VENC_RoiCfg_t, MI_VENC_AdvCustRcAttr_t 定义。	11/01/2019
2.11	● 增加 AVBR 相关的数据结构	12/30/2019
2.12	● SSC33X 为 Smart Encoding 之智能侦测新增接口 MI_VENC_SetSmartDetInfo 以及相关数据结构，修改 MI_VENC_AllocCustomMap	01/02/2020
2.13	● SSC33X 为 h264 和 h265 encoder 新增 intra refresh set 和 get api，新增 MI_VENC_IntraRefresh_t 定义 MI_VENC_PackInfo_t 新增 u32SliceId 成员变量。	04/08/2020
2.14	● 增加 MI_VENC_InitDev 和 MI_VENC_DeInitDev	04/19/2020

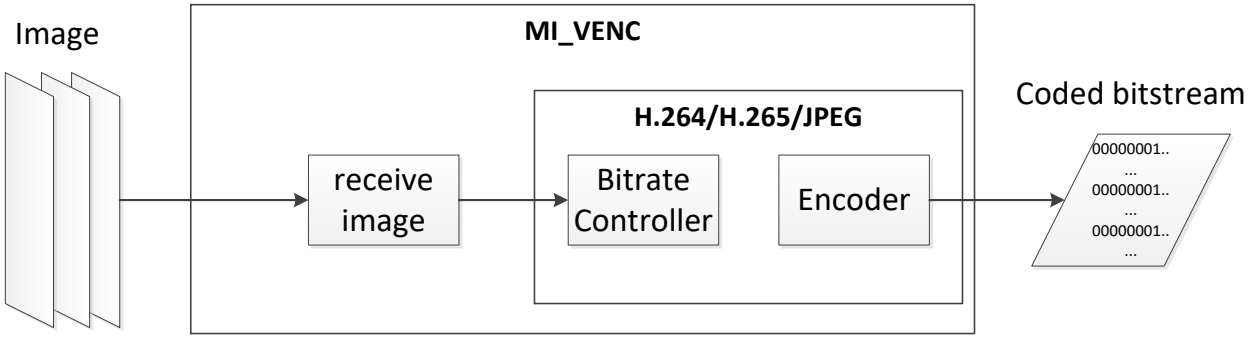
TABLE OF CONTENTS

1. API 参考

1.1. 概述

视频编码模块主要提供视频编码通道的创建和销毁、开启和停止接收图像、设置和获取编码通道属性、获取和释放码流等功能。本模块支持多路实时编码，每路独立，且可以设置不同编码协议和 profile。

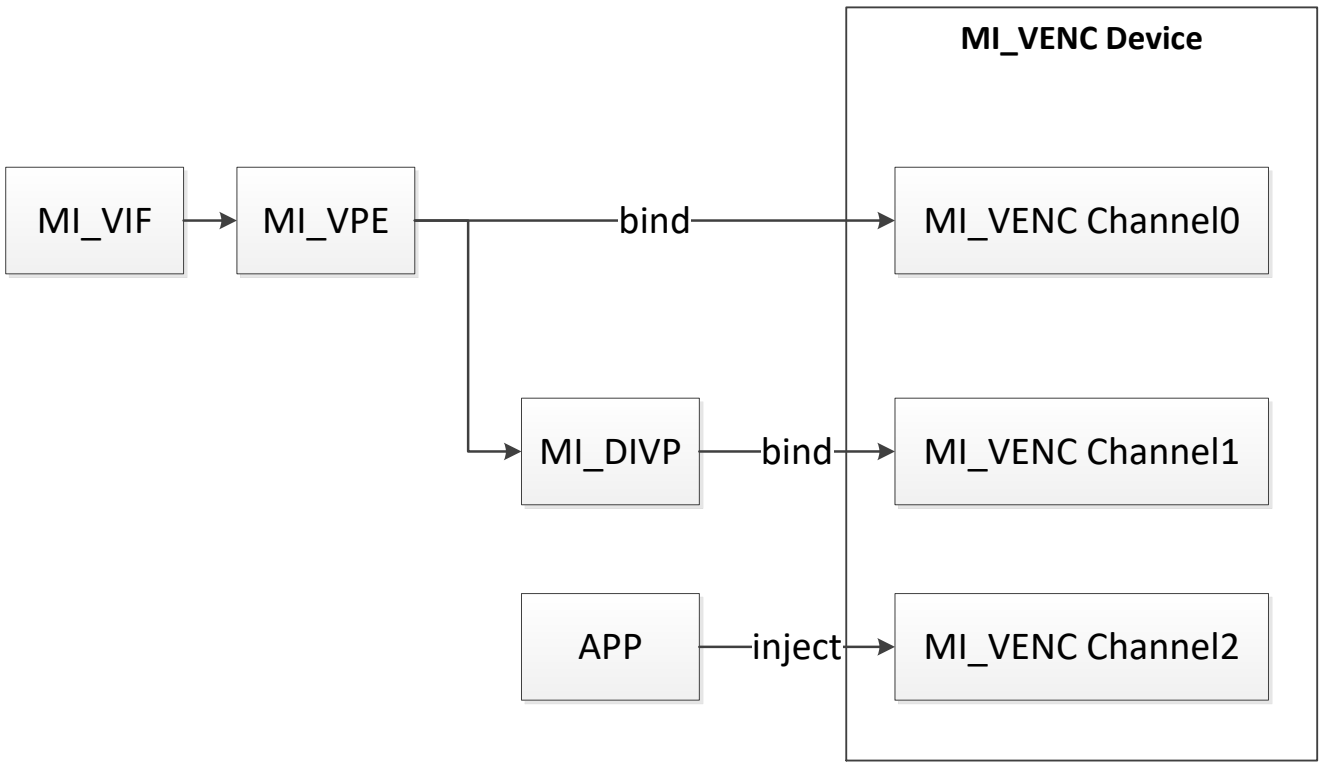
1.1.1 编码流程



编码流程包含了输入图像的接收，对输入图像的编码，和编码后码流的输出等过程。
输入图像 YUV 格式可以为 NV12 或 YUYV422。本模块的输入源包括以下两类：

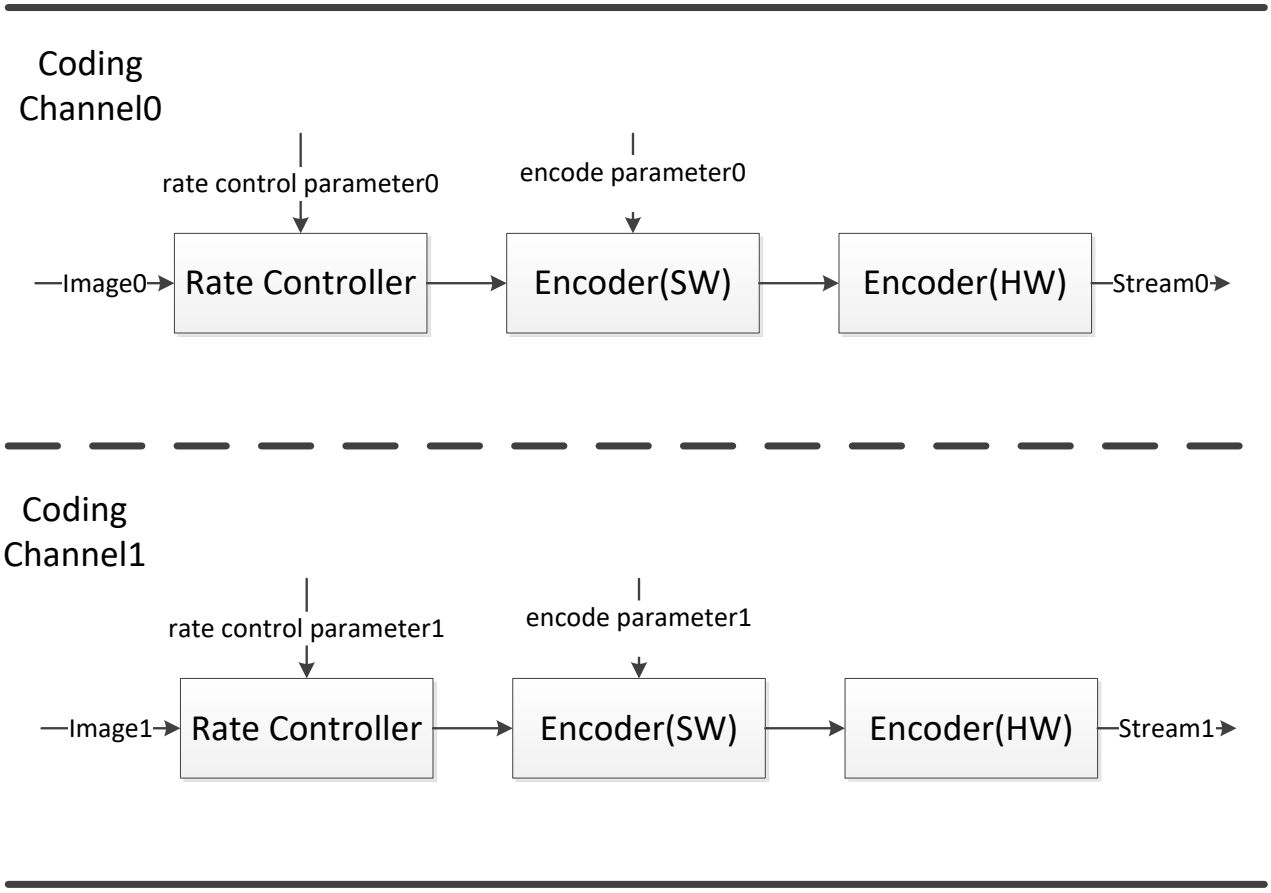
1. APP 直接注入图像数据到编码模块
2. 绑定的前级模块比如 VPE 或 DIVP 发送图像数据到编码模块

如下图所示：



1.1.2 编码通道

编码通道作为基本的控制单元，相互之间是完全独立的。Encoder(HW)完成原始图像转化为码流的功能，具体由 Rate Controller 和 Encoder(SW)协同控制完成。每个通道的 Rate Controller 和 Encoder(SW)会保存当前编码的所有用户设定和管理编码通道内部的所有资源，诸如码率控制，Buffer 的需求与分配等。码率控制保证输出码率与图像质量的平衡稳定，编码器则专注于各个 Codec 的语法元素实现。软件与硬件同时控制，同一个 Codec 硬件分时复用。以两个编码通道为例，基本框图如下：



1.2. 关键字说明

1.2.1 QP

Quantization Parameter：QP 值对应量化步长的序号，值越小，量化步长越小，量化的精度就越高，画质也就越好，编码出来的 size 也越大。

1.2.2 GOP

Group Of Pictures：指的是两个 I 帧的间隔。视频图像序列由一个或多个图像组 GOP 组成，GOP 之间是独立的。

1.2.3 MB

Macroblock：编码宏块，H.264 编码的基本单元。

1.2.4 CU

Coding Unit：编码单元，H.265 编码的基本单元。

1.2.5 SPS

Sequence Parameter Set：序列参数集，包含了一个 GOP 中所有图像的公有信息。

1.2.6 PPS

Picture Parameter Set：图像参数集，包含了一张图像编码所用的参数

1.2.7 SEI

Supplemental Enhancement information:辅助增强信息。

1.2.8 ECS

Entropy-coded segment:JPEG 熵编码后的压缩图像条带。

1.2.9 MCU

Minimum code unit:JPEG 编码的基本单元。

1.2.10 码率控制

从信息学角度来看，图像压缩比与质量成反比：压缩比越高，质量越低；压缩比越低，质量越高。以 H.264 为例，一般图像 Qp 越低，图像质量越高，码率越高；图像 Qp 越高，图像质量越低，码率越低。
不同编码协议对码率控制算法支持如下表：

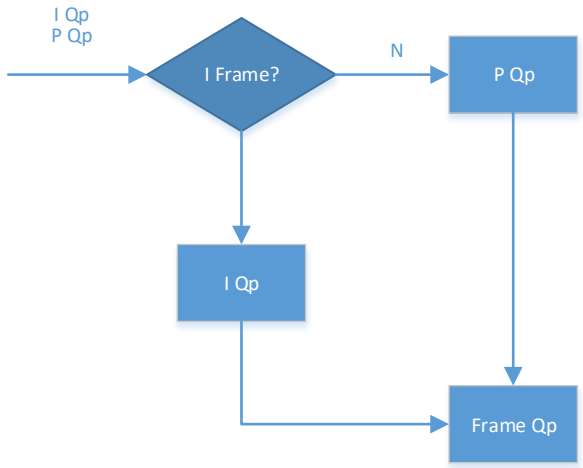
Chip	Rate Control Algorithm			
	FIXQP	CBR	VBR	AVBR
328Q/329D/326D	H.264/H.265/JPEG	H.264/H.265/JPEG	H.264/H.265	NONE
325/325DE/327DE	H.264/H.265/JPEG	H.264/H.265/JPEG	H.264/H.265	NONE
336D/336Q/339G	H.264/H.265/JPEG	H.264/H.265/JPEG	H.264/H.265	H.264/H.265
335/337DE	H.264/H.265/JPEG	H.264/H.265/JPEG	H.264/H.265	H.264/H.265

1.2.10.1. FIXQP

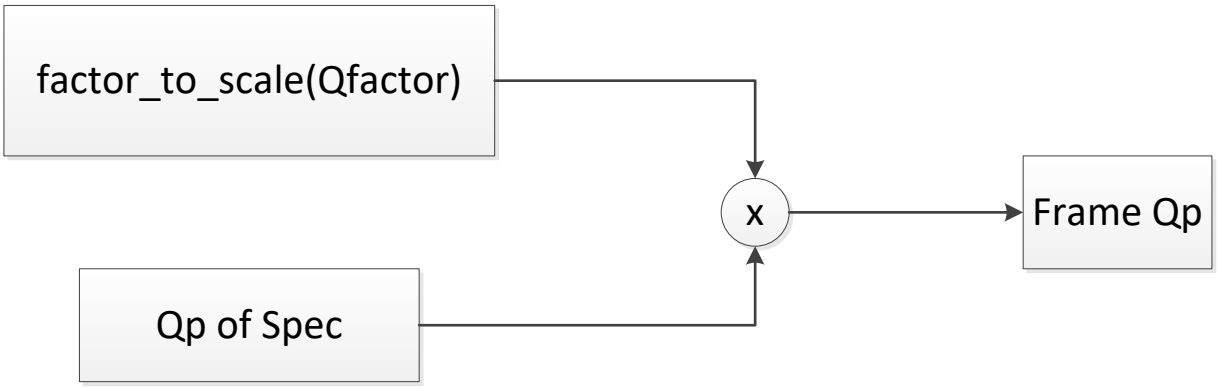
FIXQP(Fix Quantization Parameter)固定 Qp，在任意时间点，编码图像的所有基本单元 Qp 都直接采用用户设定值，H.264/H.265 I 帧和 P 帧的 Qp 可以分别设置，但有些 Chip 无法设置 P 帧 Qp，不同 chip 的行为如下表：

Chip	H.264/H.265	
	Can set I Qp?	Can set P Qp?
328Q/329D/326D	Y	Y
325/325DE/327DE	Y	Y
336D/336Q/339G	Y	N
335/337DE	Y	N

H.264 和 H.265 的控制流程图如下：



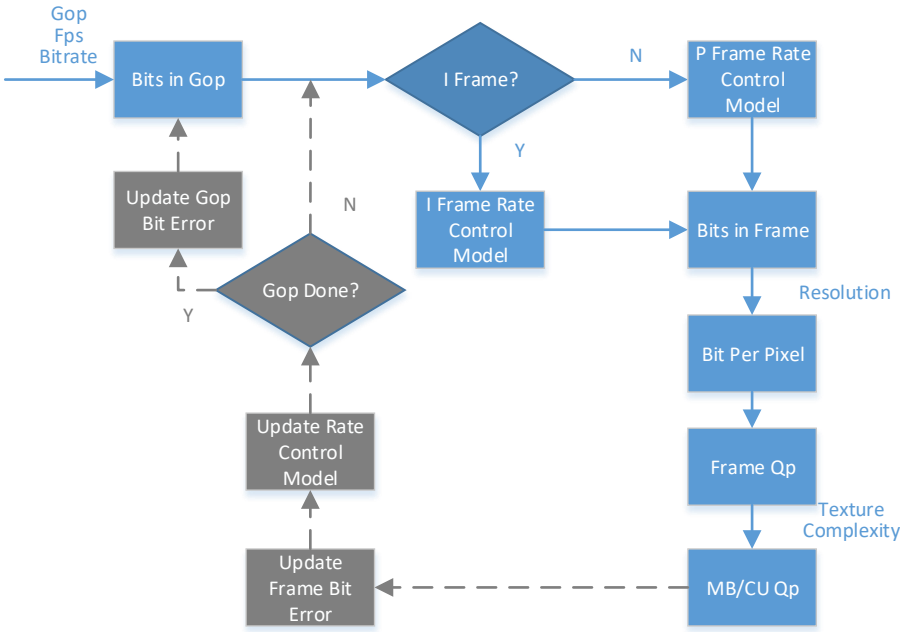
JPEG 的控制流程图如下：



采用 ITU-t81 K.1 节推荐的 Qp 表，将用户设置的 Qfactor 按照一定公式转化成比例因子，两者相乘就是最后的 Qp。

1.2.10.2. CBR

CBR(Constant Bit Rate)为固定比特率，在码率统计时间内保证编码码率稳定。
编码中 I 帧与 P 帧因为预测方式的不同会造成编码后 Size 的明显差异，底层的统计时间以 Gop 为基本单位，在 Gop 之间会实现比特的累积与补偿。
主要步骤如下：
1.将用户设定的 Fps/Gop/Bitrate 转换成每个 Gop 的 Bits；
2.I/P 区分处理，按照 Gop Bits 以及分辨率计算 Bpp(BitPerPixel)；
3.通过码率控制模型，将 Bpp 映射到 Frame Qp；
4.HW 在 Frame Qp 基础上通过画面纹理复杂度等信息进一步调整 MB/CU Qp；
5.编码结束后更新码率控制模型，以达到整个序列的不断稳定，同时累积 Bit 误差，用于后续帧 Bit 分配的微调；
6.整个 Gop 结束后累积整个 Gop 的 Bit 误差，用于下一个 Gop Bit 分配的微调。
控制流程图如下：



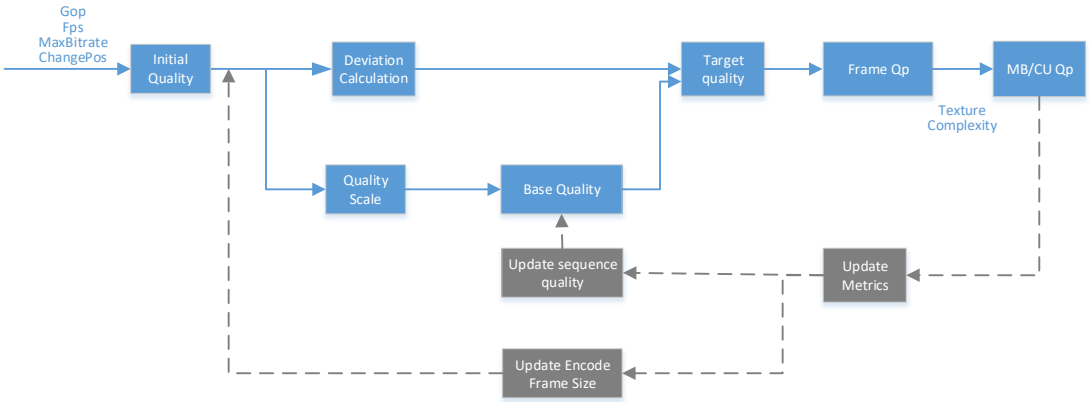
1.2.10.3. VBR

VBR(Variable Bit Rate)为可变比特率，允许在码率统计时间内编码码率波动，从而保证编码图像质量平稳。

以 H.264 为例，用户可设置 MaxQp,MinQp,MaxBitRate 和 ChangePos，具体可以参考 MI_VENC_ParamH264Vbr_t。MaxQp 和 MinQP 用于控制图像的质量范围;MaxBitRate 用于钳位码率统计时间内的最大编码码率;ChangePos 用于控制开始调整 Qp 的码率基准线，是一个相对于最大码率的百分比，即当编码码率大于 MaxBitRate*ChangePos 时，图像 Qp 会逐步向 MaxQp 调整，若图像 Qp 已到达 MaxQp，图像 Qp 会被钳位到最大值，MaxBitRate 的钳位效果失去，编码码率可能会超过 MaxBitRate;若编码码率小于 MaxBitRate*ChangePos,图像 Qp 会逐步向 MinQp 调整，若图像 Qp 已到达 MinQp,编码码率已达到最大值，图像质量最好。具体流程如下：

1. 将用户设定的 Fps/Gop/MaxBitrate/ChangePos 以及 Resolution 转换成序列的初始质量基准设定序列起始 Qp
2. HW 在 Frame Qp 基础上通过画面纹理复杂度等信息进一步调整 MB/CU Qp
3. 编码完后更新码率控制模型
4. 更新整体序列的质量
5. 根据当前期望的 Bit 与实际编码的 Bit 差计算偏差系数 Deviation 以及决定是否可以提高或者降低质量
6. 通过偏差系数以及两个质量系数计算当前帧的目标质量
7. 将目标质量映射为 Frame Qp，从 2 循环。

控制流程图如下：



1.2.10.4. AVBR

AVBR(Adaptive Variable Bit Rate)自适应可变比特率，允许在码率统计时间内编码码率波动，从而保证编码图像质量平稳。码率控制内部会检测当前场景的运动静止状态，在运动时采用较高码率编码，静止时主动降低码率。以 H.264 为例，用户可设置 MaxBitRate,ChangePos 和 MinStillPercent，具体可参考 MI_VENC_ParamH264Avbr_t。MaxBitRate 表示运动场景下的最大码率;MinStillPercent 表示静止状态下最小码率相对于调节阈值码率的百分比，MaxBitRate*ChangePos*MinStillPercent 表示静止场景下的最小码率，目标码率根据运动程度的不同会在最小码率和最大码率之间调整;MaxQp 和 MinQp 用于控制图像的质量范围，码率控制以 Qp 钳位为最高优先级，超出 MinQp 和 MaxQp 范围码率控制将失效。

1.2.11 QPMAP

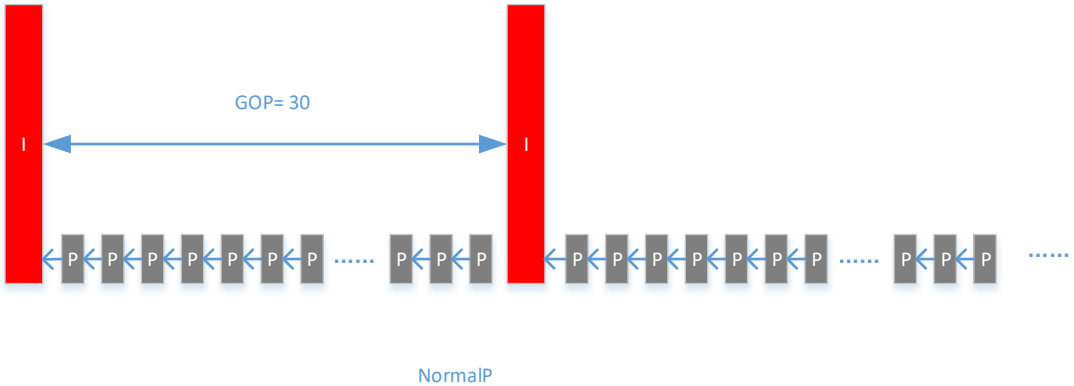
QPMAP 模式下允许用户自由决定码率控制的策略，支持绝对 QpMap 和相对 QpMap。请参考 MI_VENC_AllocCustomMap。

1.2.12 参考帧结构

H.264/H.265 单帧只支持参考 1 个参考帧，但是整个码流支持多个参考帧缓存。例如：LTR/TSVC3 模式，每张 P 都只可以参考一个，但是最多会保存 2 个参考帧供不同 p 帧参考。
参考帧共支持 5 种模式：NormalP，LTR(VI Ref IDR)，LTR(VI Ref VI)，TSVC-2 和 TSVC-3。所有参考帧结构由三个参数控制:u32Base，u32Enhance 和 bEnablePred，具体含义请参考 MI_VENC_ParamRef_t。u32Enhance 设置为 0 将转换为 NormalP 参考帧结构，其余结构开启请参考对应结构图，系统默认为 NormalP 参考帧结构。以下详细介绍每种参考关系结构图和参数设定。

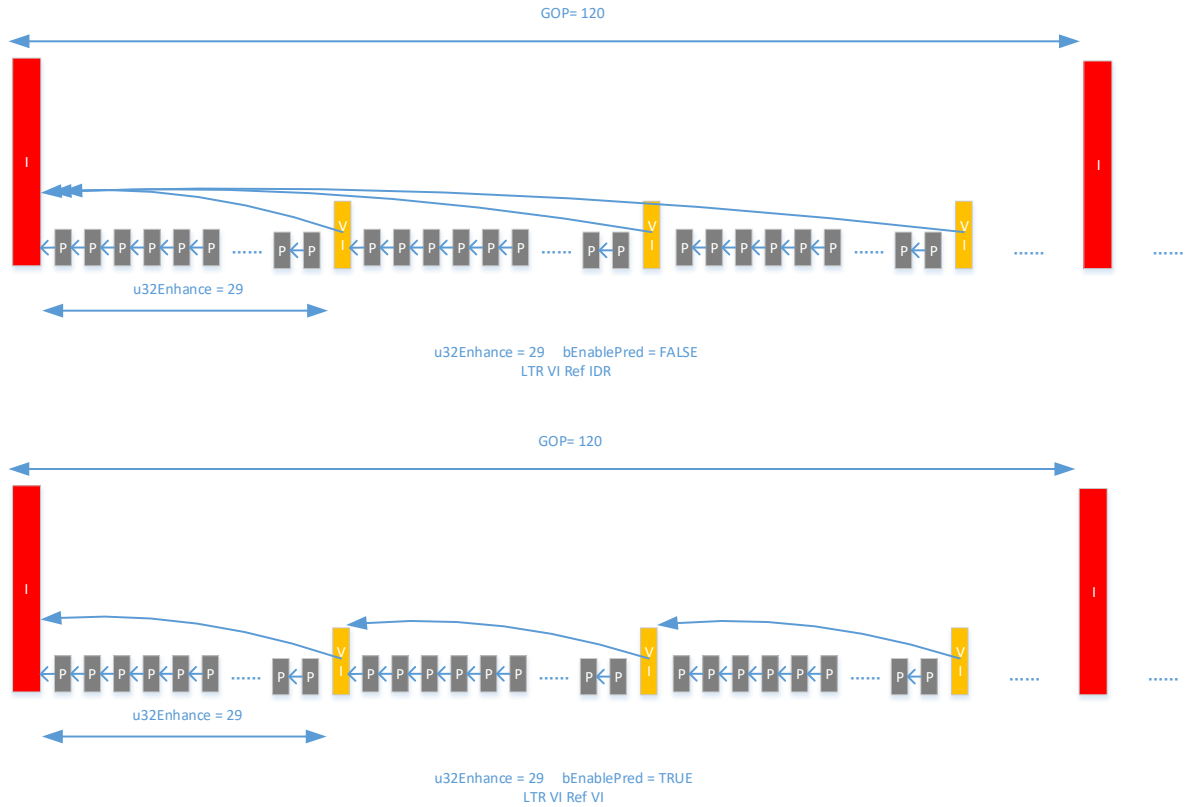
1.2.12.1. NormalP

NormalP 为最基本和常见的参考关系，P 帧直接参考前一张，其结构图如下图所示：



1.2.12.2. LTR 模式

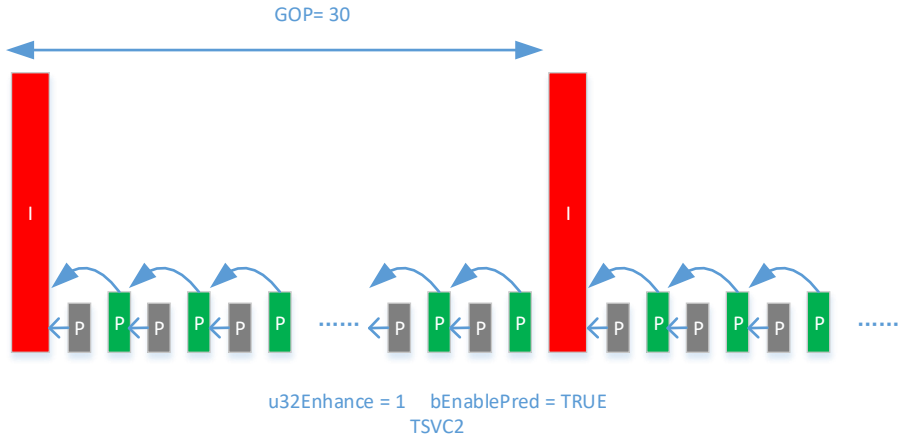
LTR(Long Term Reference)采用长期参考帧+短期参考帧实现同一帧可以被多张参考，该模式定义一种特殊的 P 为虚拟 I 帧（VI），与全部使用 IDR 相比，采用 VI 可以降低码率，同时又保持一定的错误恢复能力。目前 LTR 支持两种模式：VI 全部参考最近的 IDR 和 VI 参考上一个 VI 或者 IDR(第一个 VI 的情况下)。两种参考模式结构示意图如下：



注意：开启 LTR 底层需要额外占用一张 Buffer。

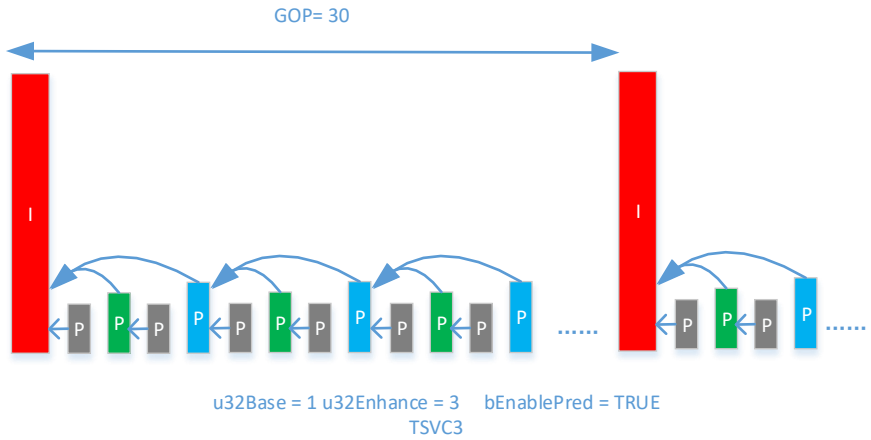
1.2.12.3. TSVC-2

TSVC-2 提供两层编码，帧率可以在 1/2 和全帧率之间变化，其结构图如下：



1.2.12.4. TSVC-3

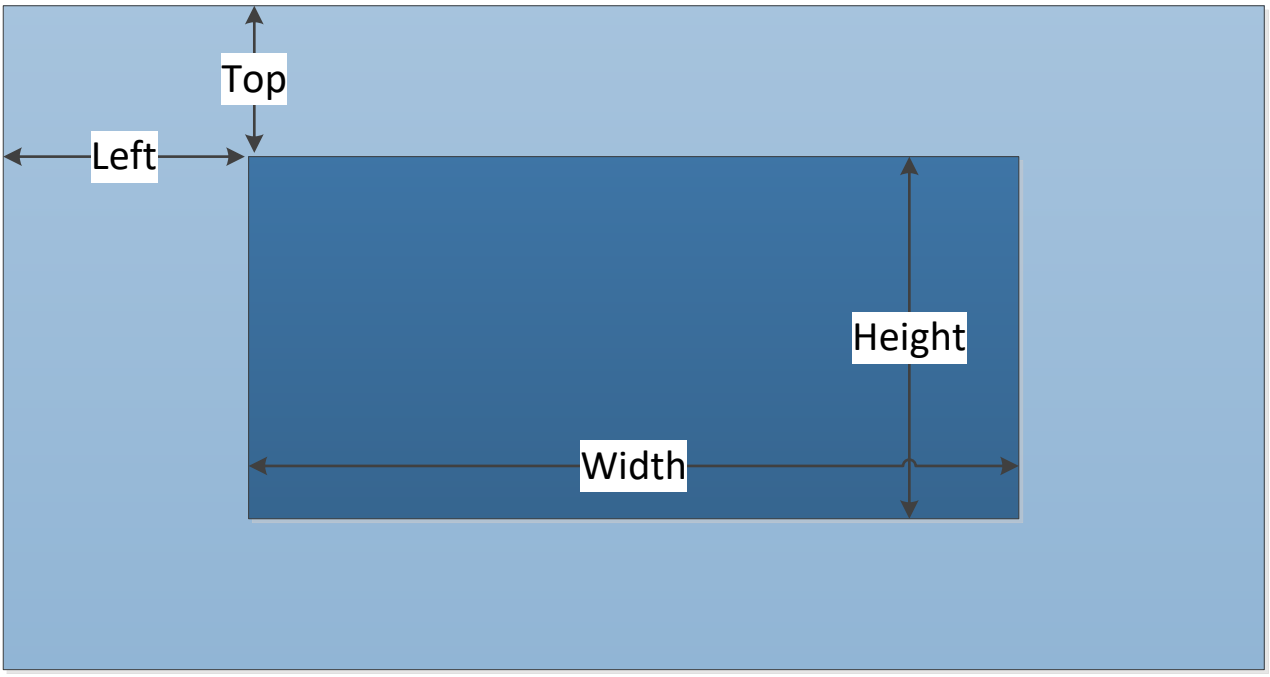
TSVC-3 提供三层编码，帧率可以在 1/4，1/2，3/4 和全帧率之间变化。其结构图如下：



注意：开启 TSVC-3 底层需要额外占用一张 Buffer。

1.2.13 裁剪编码

裁剪编码，即从图像中裁剪出一部分进行编码，用户可以设置裁剪的起始点 left/top、width 和 height，具体功能请参考相关 API:MI_VENC_SetCrop。示意图如下：



1.2.14 ROI

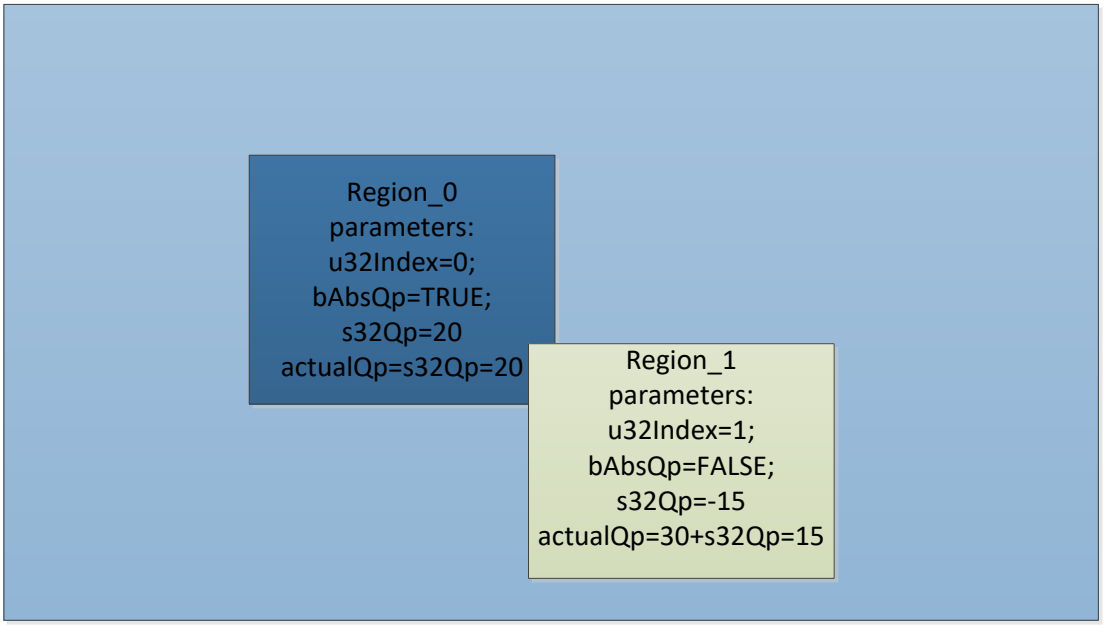
ROI(Region Of Interest)编码，感兴趣区域编码，用户可以通过配置 ROI 区域，对该区域的图像 Qp 进行限制，从而实现图像中该区域的 Qp 与其他图像区域的差异化。系统支持 H264 和 H265 编码设置 ROI，且提供 8 个 ROI 区域供用户同时使用。

8 个 ROI 区域可以互相叠加，且叠加时的优先级按照 0～7 的索引号依次提高，也即叠加区域的 Qp 最终判定只按最高优先级的区域处理。ROI 区域可以配置绝对 Qp 与相对 Qp 两种模式：

绝对 Qp:ROI 区域的 Qp 为用户设定的 Qp 值

相对 Qp:ROI 区域的 Qp 为码率控制产生的 Qp 与用户设定的 Qp 偏移值的和

以下示例为编码图像采用 FixQp 模式，设置图像 Qp 为 30，即图像所有宏块 Qp 值为 30。ROI 区域 0 设置为绝对 Qp 模式，Qp 值为 20，索引为 0；ROI 区域 1 设置为相对 Qp 模式，Qp 为-15，索引为 1。因为 ROI 区域 0 的 index 小于 ROI 区域 1 的 index，所以在发生重叠的图像区域按高优先级的 ROI 区域 1 的 Qp 设置。除了重叠部分的 ROI 区域 0 的 Qp 值为 20，区域 1 的 Qp 值为 30-15=15。



1.2.15 非 ROI 区域的低帧率编码

为了减低传输码率，对于非 ROI 区域，可采用低帧率编码，即开启 ROI 后 ROI 区域正常编码，而非 ROI 区域则可以通过设定源和目标的比例关系降低帧率，根据该比例关系，一个 GOP 里面一些帧的非 ROI 区域会直接用上一帧对应区域的数据，用户可以根据实际情况设置非 ROI 区域相对帧率，注意只有 ROI 开启才生效。
请参考 API：MI_VENC_SetRoiBgFrameRate，其中 s32SrcFrmRate 与 s32DstFrmRate 只表示比例关系，与实际帧率无关

1.2.16 Bind Type

一般来说前级与 venc 的 bind type 是 E_MI_SYS_BIND_TYPE_FRAME_BASE，即前级完整做完一张 Frame buffer，再输出给 venc，这种模式至少需要分配 3 张 Frame buffer。不过若前级是 vpe，还有另外两种省内存的 bind type: E_MI_SYS_BIND_TYPE_HW_RING 和 E_MI_SYS_BIND_TYPE_REALTIME。E_MI_SYS_BIND_TYPE_HW_RING 是指 vpe 与 venc 只在同一张 Frame buffer 上边读边写，另外 335/337DE 还支持在半张 Frame buffer 上读写，可参考 MI_VENC_SetInputSourceConfig；而 E_MI_SYS_BIND_TYPE_REALTIME 是指 vpe 与 venc 之间无需额外分配 Dram，硬件直连。

不同 Chip 和编码协议对 bind type 的支持如下：

Chip	Codec	Bind Type		
		FRAME_BASE	HW_RING	REALTIME
328Q/329D/326D	H.264/H.265	Y	N	N
	JPEG	Y	N	N
325/325DE/327DE	H.264/H.265	Y	Y	N
	JPEG	Y	N	Y
336D/336Q/339G	H.264/H.265	Y	N	N
	JPEG	Y	N	Y
335/337DE	H.264/H.265	Y	Y	N
	JPEG	Y	N	Y

1.2.17 编码码流输出 Buffer 配置

不同 Chip 由于 SW 架构不同，支持不同的方式，底层的默认分配方式如下表：

Chip	H.264	H.265	JPEG
328Q/329D/326D	WidthxHeight/2	WidthxHeight/2	WidthxHeight
325/325DE/327DE	Ring pool:WidthxHeight/2	Ring pool:WidthxHeight/2	Ring pool:WidthxHeight/2
336D/336Q/339G	Ring pool:WidthxHeight/2	Ring pool:WidthxHeight/2	Ring pool:WidthxHeight/2
335/337DE	Ring pool:WidthxHeight/2	Ring pool:WidthxHeight/2	Ring pool:WidthxHeight/2

以上为各 Chip 的默认输出 Buffer 配置，用户可以通过 MI_VENC_CreateChn 去指定输出 Buffer 配置。以 H.264 为例，可以通过设置 stVeAttr.stAttrH264e.u32BufSize 去指定输出 Buffer 配置，若为 0，则采用上表默认配置。原先 328Q/329D/326D 采用的是帧模式，即每编码一张，都要重新申请一块特定大小的 Buffer，且通常实际编码使用的 size 会小于申请的 size，会造成内存碎片；而采取 Ring pool 模式，每个 Channle 会单独分配一个 output pool，每次编码都从该 pool 里获取空闲的最大 Buffer，有效提高了 Buffer 的使用率，减少了内存碎片。

1.3. 功能模块 API

API 名	**功能**
MI_VENC_GetChnDevid	获取编码通道的设备 id
MI_VENC_SetModParam	设置编码相关的模块参数
MI_VENC_GetModParam	获取编码相关的模块参数
MI_VENC_CreateChn	创建编码通道
MI_VENC_DestroyChn	销毁编码通道
MI_VENC_ResetChn	复位编码通道
MI_VENC_StartRecvPic	开启编码通道接收输入图像
MI_VENC_StartRecvPicEx	开启编码通道接收输入图像，超出指定的帧数后自动停止接收图像

API 名	**功能**
MI_VENC_StopRecvPic	停止编码通道接收输入图像
MI_VENC_Query	查询编码通道状态
MI_VENC_SetChnAttr	设置编码通道的编码属性
MI_VENC_GetChnAttr	获取编码通道的编码属性
MI_VENC_GetStream	获取编码码流。
MI_VENC_ReleaseStream	释放码流缓存
MI_VENC_GetStreamBufInfo	获取码流 buffer 的物理地址和大小
MI_VENC_InsertUserData	插入用户数据
MI_VENC_SetMaxStreamCnt	设置最大码流缓存帧数
MI_VENC_GetMaxStreamCnt	获取最大码流缓存帧数
MI_VENC_RequestIdr	请求 IDR 帧
MI_VENC_EnableIdr	使能 IDR 帧
MI_VENC_SetH264IdrPicId	设置 IDR 帧的 idr_pic_id
MI_VENC_GetH264IdrPicId	获取 IDR 帧的 idr_pic_id
MI_VENC_GetFd	获取编码通道对应的设备文件句柄
MI_VENC_CloseFd	关闭编码通道对应的设备文件句柄
MI_VENC_SetRoiCfg	设置编码通道的感兴趣区域编码配置
MI_VENC_GetRoiCfg	获取编码通道的感兴趣区域编码配置
MI_VENC_SetRoiBgFrameRate	设置编码通道非感兴趣区域的帧率配置
MI_VENC_GetRoiBgFrameRate	获取编码通道非感兴趣区域的帧率配置
MI_VENC_SetH264SliceSplit	设置 H.264 编码的 slice 分割配置
MI_VENC_GetH264SliceSplit	获取 H.264 编码的 slice 分割配置
MI_VENC_SetH264InterPred	设置 H.264 编码的帧间预测配置
MI_VENC_GetH264InterPred	获取 H.264 编码的帧间预测配置
MI_VENC_SetH264IntraPred	设置 H.264 编码的帧内预测配置
MI_VENC_GetH264IntraPred	获取 H.264 编码的帧内预测配置
MI_VENC_SetH264Trans	设置 H.264 编码的变换、量化配置
MI_VENC_GetH264Trans	获取 H.264 编码的变换、量化配置
MI_VENC_SetH264Entropy	设置 H.264 编码的熵编码配置
MI_VENC_GetH264Entropy	获取 H.264 编码的熵编码配置
MI_VENC_SetH265InterPred	设置 H.265 编码的帧间预测配置
MI_VENC_GetH265InterPred	获取 H.265 编码的帧间预测配置
MI_VENC_SetH265IntraPred	设置 H.265 编码的帧内预测配置
MI_VENC_GetH265IntraPred	获取 H.265 编码的帧内预测配置

API 名	功能
MI_VENC_SetH265Trans	设置 H.265 编码的变换、量化配置
MI_VENC_GetH265Trans	获取 H.265 编码的变换、量化配置
MI_VENC_SetH264Dblk	设置 H.264 编码的 deblocking 配置
MI_VENC_GetH264Dblk	获取 H.264 编码的 deblocking 配置
MI_VENC_SetH265Dblk	设置 H.265 编码的 deblocking 配置
MI_VENC_GetH265Dblk	获取 H.265 编码的 deblocking 配置
MI_VENC_SetH264Vui	设置 H.264 编码的 VUI 配置
MI_VENC_GetH264Vui	获取 H.264 编码的 VUI 配置
MI_VENC_SetH265Vui	设置 H.265 编码的 VUI 配置
MI_VENC_GetH265Vui	获取 H.265 编码的 VUI 配置
MI_VENC_SetH265SliceSplit	设置 H.265 编码的 slice 分割配置
MI_VENC_GetH265SliceSplit	获取 H.265 编码的 slice 分割配置
MI_VENC_SetJpegParam	设置 JPEG 编码的参数集合
MI_VENC_GetJpegParam	获取 JPEG 编码的参数集合
MI_VENC_SetRcParam	设置通道码率控制高级参数
MI_VENC_GetRcParam	获取通道码率控制高级参数
MI_VENC_SetRefParam	设置 H.264/H.265 编码通道高级跳帧参考参数
MI_VENC_GetRefParam	获取 H.264/H.265 编码通道高级跳帧参考参数
MI_VENC_SetCrop	设置 VENC 的裁剪属性
MI_VENC_GetCrop	获取 VENC 的裁剪属性
MI_VENC_SetFrameLostStrategy	设置瞬时码率超出阈值时丢帧策略的配置
MI_VENC_GetFrameLostStrategy	获取瞬时码率超出阈值时丢帧策略的配置
MI_VENC_SetSuperFrameCfg	设置超大帧处理配置
MI_VENC_GetSuperFrameCfg	获取超大帧处理配置
MI_VENC_SetRcPriority	设置码率控制的优先级类型
MI_VENC_AllocCustomMap	分配智能编码所用的 Custom Map 的内存
MI_VENC_ApplyCustomMap	应用已经配置的 Custom Map
MI_VENC_GetLastHistoStaticInfo	获取最新的图像帧编码后的相关输出信息
MI_VENC_ReleaseHistoStaticInfo	释放保存的最新的图像帧编码后的相关输出信息占用的内存
MI_VENC_SetAdvCustRcAttr	设置自定义的高级码控相关属性配置
MI_VENC_SetInputSourceConfig	设置 H.264/H.265 的输入源配置信息
MI_VENC_SetSmartDetInfo	设置智能侦测算法相关的统计信息
MI_VENC_SetIntraRefresh	设置 H.264/H.265 的 P 帧刷新 Islice

API 名	功能
MI_VENC_GetIntraRefresh	获取 H.264/H.265 的 Intra Refresh 参数
MI_VENC_InitDev	初始化 venc 设备
MI_VENC_DeInitDev	反初始化 venc 设备

1.3.1 MI_VENC_GetChnDevid

➢ 功能

获取编码通道的设备 id。

➢ 语法

MI_S32 MI_VENC_GetChnDevid(MI_VENC_CHN VeChn, MI_U32 *pu32Devid);

➢ 形参

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pu32Devid	编码设备 id 指针。	输出

➢ 返回值

返回值
- 0 成功。
- 非 0 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

➢ 芯片差异

不同 Chip 下不同编码协议对应的设备 id 是不同的，如下表所示：

Chip	H.264	H.265	JPEG
328Q/329D/326D	0	0	1
325/325DE/327DE	0	0	1
336D/336Q/339G	0	0	1
335/337DE	0	0	1

※ 注意

- **此接口在通道创建后调用，如果通道未创建，则返回失败**

由于不同编码协议对应的设备 id 有可能不同，因此如果接下来调用的接口参数包含设备 id，需要先调用该接口获取当前 channel 的对应设备 id。

➢ 举例

```
MI_S32 BindVpeToJpeg()
{
    MI_S32 s32Ret;
    MI_SYS_ChnPort_t stVencInputPort;
    MI_SYS_ChnPort_t stVpeOutputPort;
    MI_U32 u32DevId = 0;
    MI_U32 u32ChnId = 0;

    /*call MI_VPE_CreateChannel to create vpe channel*/
    /*call MI_VENC_CreateChn to create jpeg channel*/

    memset(&stVencInputPort, 0, sizeof(MI_SYS_ChnPort_t));
    s32Ret = MI_VENC_GetChnDevid(u32ChnId, &u32DevId);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_GetChnDevid err 0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }
    stVencInputPort.eModId = E_MI_MODULE_ID_VENC;
    stVencInputPort.u32DevId = u32DevId;
    stVencInputPort.u32ChnId = u32ChnId;
    stVencInputPort.u32PortId = 0;

    stVpeOutputPort.eModId = E_MI_MODULE_ID_VPE;
    stVpeOutputPort.u32DevId = 0;
    stVpeOutputPort.u32ChnId = 0;
    stVpeOutputPort.u32PortId = 0;

    s32Ret = MI_SYS_BindChnPort2(&stVpeOutputPort, &stVencInputPort, 30, 30,
E_MI_SYS_BIND_TYPE_FRAME_BASE, 0);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_SYS_BindChnPort2 err 0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }

    return MI_SUCCESS;
}
```

- **相关主题**

  无。

1.3.2 MI_VENC_SetModParam

➢ 功能

设置编码相关的模块参数。

➢ 语法

MI_S32 MI_VENC_SetModParam(MI_VENC_ModParam_t *pstModParam)

➢ 形参

参数名称	描述	输入/输出
pstModParam	编码模块参数指针。	输入

➢ 返回值

返回值
- 0 成功。
- 非 0 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 此接口在通道创建前调用，如果通道已经创建，则返回失败
- 可以设置 mi_venc.ko、mi_h264e.ko、mi_h265e.ko、mi_jpge.ko 模块的参数。
- 暂不支持

➢ 举例

无。

➢ 相关主题

无。

1.3.3 MI_VENC_GetModParam

➢ 功能

获取编码相关的模块参数。

➢ 语法

MI_S32 MI_VENC_GetModParam(MI_VENC_ModParam_t *pstModParam)

➢ 参数

参数名称	描述	输入/输出
pstModParam	编码模块参数指针。	输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 可以获取 mi_venc.ko、mi_h264e.ko、mi_h265e.ko、mi_jpge.ko 模块的参数。
- 暂不支持

➢ 举例

无。

➢ 相关主题

无。

1.3.4 MI_VENC_CreateChn

➢ 功能

创建编码通道。

➢ 语法

MI_S32 MI_VENC_CreateChn(MI_VENC_CHN Vehn, MI_VENC_ChnAttr_t *pstAttr);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstAttr	编码通道属性指针。	输入

➢ 返回值

- MI_OK 成功。

返回值

非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_comm_rc.h、mi_venc.h
- 库文件：libmi.a

➢ 芯片差异

不同 Chip 支持的编码规格如下表所示：

Chip	H.264			H.265	JPEG
	Baseline Profile	Main Profile	High Profile	Main Profile	Baseline Profile
328Q/329D/326D	Y	Y	N	Y	Y
325/325DE/327DE	Y	Y	N	Y	Y
336D/336Q/339G	Y	Y	Y	Y	Y
335/337DE	Y	Y	Y	Y	Y

不同 Chip 支持的编码通道宽高差异如下表所示：

Chip	Resolution Spec	H.264	H.265	JPEG
328Q/329D/326D	MIN_WIDTHxMIN_HEIGHT	192x128	192x128	16x16
	MAX_WIDTHxMAX_HEIGHT	3840x2176	3840x2176	3840x2176
	MIN_ALIGN	8x2	8x2	8x2
325/325DE/327DE	MIN_WIDTHxMIN_HEIGHT	192x128	192x128	16x16
	MAX_WIDTHxMAX_HEIGHT	2688x1920	2688x1920	2688x2688
	MIN_ALIGN	8x2	8x2	8x2
336D/336Q/339G	MIN_WIDTHxMIN_HEIGHT	256x128	256x128	16x16
	MAX_WIDTHxMAX_HEIGHT	3840x2176	3840x2176	3840x2176
	MIN_ALIGN	8x2	8x2	8x2
335/337DE	MIN_WIDTHxMIN_HEIGHT	256x128	256x128	16x16
	MAX_WIDTHxMAX_HEIGHT	2592x1952	2592x1952	2688x2688
	MIN_ALIGN	8x2	8x2	8x2

※ 注意

- 编码通道属性由两部分组成，编码器属性和码率控制器属性。
- 编码器属性首先需要选择编码协议，然后分别对各种协议对应的属性进行赋值。
- 编码器属性最大宽高，通道宽高必须满足如下约束：

  MaxPicWidth∈[MIN_WIDTH,MAX_WIDTH]
  MaxPicHeight∈[MIN_HEIGHT,MAX_HEIGHT]

PicWidth∈[MIN_WIDTH,MaxPicwidth]
PicHeight∈[MIN_HEIGHT,MaxPicHeight]

- 最大宽高，通道宽高必须是 MIN_ALIGN 的整数倍。
- 其中 MIN_WIDTH，MAX_WIDTH，MIN_HEIGHT，MAX_HEIGHT，MIN_ALIGN 分别表示编码通道支持的最小宽度，最大宽度，最小高度，最大高度，最小对齐单元（像素）。
- 当输入图像大小不大于通道编码图像最大宽高时，才能启动编码通道进行编码。
- 推荐的编码宽高为：3840x2160（4k*2k）、1920x1080(1080P)、1280x720（720P）、960x540、640x360、704x576、704x480、352x288、352x240。
- 编码器属性 stVeAttr 中除通道宽高（u32PicWidth/u32PicHeight）和 profile 以外都是静态属性，一旦创建编码通道成功，静态属性不支持被修改，除非该通道被销毁，重新创建。设置时需要注意的事项请参考 MI_VENC_SetChnAttr 接口说明。
- 码率控制器属性首先需要配置 RC 模式，JPEG 抓拍通道不需要配置码率控制器属性，其他协议类型通道（H.264/H.265/MJPEG）都必须配置。码率控制器属性 RC 模式必须与编码器属性协议类型匹配。
- 三种编码协议（H.264/H.265/MJPEG），码率控制均支持三种模式：CBR、VBR 和 FIXQP。并且对于不同协议，相同 RC 模式的属性变量基本一致。
- u32SrcFrmRateNum 指编码模块帧率的分子部分，u32SrcFrmRateDen 指编码模块帧率的分母部分。u32SrcFrmRateNum/u32SrcFrmRateDen 应该设置为产生 TimeRef 的实际帧率，RC 需要根据 u32SrcFrmRateNum/u32SrcFrmRateDen 统计实际的帧率以及进行码率控制。如果编码图像的源是 VI 时，则 u32SrcFrmRateNum/u32SrcFrmRateDen 设置为 VI 的实际输出帧率，因为 TimeRef 是在 VI 输出的时候产生。假如 VI 的实际输出帧率为 30，则 u32SrcFrmRateNum 设为 30，u32SrcFrmRateDen 设为 1。

CBR 除了上述的属性之外，还需要设置平均比特率和波动等级。平均比特率的单位是 kbps。平均比特率的设置与编码通道宽高以及图像帧率都有关系。典型的平均比特率的设置如下表所示。注意，下表中的平均比特率的设置是在通道编码帧率为满帧率（30fps）时的设置。当用户设置编码输出帧率不为满帧率时，可以对下表中的码率按用户设置帧率与满帧率（30fps）的比例进行换算。

图像宽高/码率等级	D1(720x576)	720p(1280x720)	1080p(1920x1080)
低码率	＜400kbps	＜800kbps	＜2000kbps
中码率	400～1000Kbps	800kbps～4000Kpbs	2000～8000Kbps
高码率	1000Kbps	4000Kbps	8000Kbps

- 波动等级设置分为 5 档，波动等级越大，系统允许码率的波动范围更大。如果波动等级设置高，对于一些图像复杂，变化剧烈的场景，图像质量可能会更平稳，适用于网络带宽富裕的场景；如果波动等级设置低，编码的码率会比较平稳，对于一些图像复杂，变化剧烈的场景，图像质量可能不如高波动等级，适用于带宽不富裕的场景，保留，暂时没有使用。
- VBR 除了上述属性之外，还需要设置 MaxBitRate，MaxQp，MinQp。MaxBitRate：编码通道在码率统计时间内允许的最大码率。MaxQp：图像允许的最大 QP。MinQp：图像允许的最小 QP。
- FIXQP 除了上述属性之外，还需要设置 IQp，PQp。IQp：I 帧时，图像固定使用的 QP 值。PQp：P 帧时，图像固定使用的 QP 值。在设置 I 帧 QP，P 帧 QP 时，可以根据当前的带宽限制对 I 帧 QP

和 P 帧 QP 同时进行向上或是向下调整。为了减少呼吸效应 ,推荐 I 帧 QP 始终比 P 帧的 QP 小 2~3。

- 举例

```
MI_S32 StartVenc()
{
    MI_S32 s32Ret;
    MI_VENC_CHN VeChn = 0;
    MI_VENC_ChnAttr_t stAttr;

    /*set h264 chnnel video encode attribute*/
    stAttr.stVeAttr.eType = E_MI_VENC_MODTYPE_H264E;
    stAttr.stVeAttr.stAttrH264e.u32PicWidth = u32PicWidth;
    stAttr.stVeAttr.stAttrH264e.u32PicHeight = u32PicHeigh;
    stAttr.stVeAttr.stAttrH264e.u32MaxPicWidth = u32MaxPicWidth;
    stAttr.stVeAttr.stAttrH264e.u32MaxPicHeight = u32MaxPicHeigh;
    stAttr.stVeAttr.stAttrH264e.u32Profile = 2;

    /*set h264 chnnel rate control attribute*/
    stAttr.stRcAttr.enRcMode = E_MI_VENC_RC_MODE_H264CBR;
    stAttr.stRcAttr.stAttrH264Cbr.u32BitRate = 10*1024;
    stAttr.stRcAttr.stAttrH264Cbr.fr32DstFrmRate = 30;
    stAttr.stRcAttr.stAttrH264Cbr.u32SrcFrmRate  = 30;
    stAttr.stRcAttr.stAttrH264Cbr.u32Gop = 30;
    stAttr.stRcAttr.stAttrH264Cbr.u32FluctuateLevel = 1;
    stAttr.stRcAttr.stAttrH264Cbr.u32StatTime = 1;

    s32Ret = MI_VENC_CreateChn(VeChn, &stAttr);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_CreateChn err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }
    s32Ret = MI_VENC_StartRecvPic(VeChn);
    if (s32Ret != MI_SUCCESS)
    {
        printf("MI_VENC_StartRecvPic err0x%x\n",s32Ret);
        return E_MI_ERR_FAILED;
    }

    return MI_SUCCESS;
}
```

- **相关主题**

  无。

1.3.5 MI_VENC_DestroyChn

➢ 描述

销毁编码通道。

➢ 语法

MI_S32 MI_VENC_DestroyChn(MI_VENC_CHN VeChn);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 销毁并不存在的通道，返回失败。
- 销毁前必须停止接收图像，否则返回失败。
- 如果开启了 OSD，则编码通道销毁前要调用接口 MI_RGN_DetachFrmChn 将 OSD 区域从当前通道撤出。

➢ 举例

```
MI_S32 StopVenc()
{
    MI_S32 s32Ret;
    MI_VENC_CHN VeChn = 0;

    s32Ret = MI_VENC_StopRecvPic(VeChn);
    if (s32Ret != MI_SUCCESS)
    {
        printf("MI_VENC_StopRecvPic err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }
    s32Ret = MI_VENC_DestroyChn(VeChn);
    if (s32Ret != MI_SUCCESS)
    {
        printf("MI_VENC_DestroyChn err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }

    return MI_SUCCESS;
}
```

- **相关主题**

  无。

1.3.6 MI_VENC_ResetChn

- **描述**

  复位通道，清除调用该接口之前缓存的图像和码流。

- **语法**

  MI_S32 MI_VENC_ResetChn(MI_VENC_CHN VeChn);

- **参数**

参数名称	描述	输入/输出
VeChn	通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入

- **返回值**

  返回值：
  - MI_OK 成功。
  - 非 MI_OK 失败，参照错误码。

➢ 依赖

- 依赖头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- Reset 并不存在的通道，返回失败 MI_ERR_VENC_UNEXIST。
- 如果一个通道没有停止接收图像而 reset 通道，则返回失败。

➢ 举例

请参见 MI_VENC_StartRecvPicEx 的举例。

➢ 相关主题

无。

1.3.7 MI_VENC_StartRecvPic

➢ 描述

开启编码通道接收输入图像。

➢ 语法

MI_S32 MI_VENC_StartRecvPic(MI_VENC_CHN VeChn);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 如果通道未创建，则返回失败 MI_ERR_VENC_UNEXIST。
- 此接口不判断当前是否已经开启接收，即允许重复开启不返回错误。

- 只有开启接收之后编码器才开始接收图像编码。

➢ **举例**

请参见 MI_VENC_CreateChn 的举例。

➢ 相关主题

无。

1.3.8 MI_VENC_StartRecvPicEx

➢ 描述

开启编码通道接收输入图像，超出指定的帧数后自动停止接收图像。

➢ 语法

MI_S32 MI_VENC_StartRecvPicEx(MI_VENC_CHN VeChn,MI_VENC_RecvPicParam_t *pstRecvParam);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstRecvParam	接收图像参数结构体指针，用于指定需要接收的图像帧数。	输入

➢ 返回值

返回值：
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 如果通道未创建，则返回失败 MI_ERR_VENC_UNEXIST。
- 如果通道已经调用了 MI_VENC_StartRecvPic 开始接收图像而没有停止接收图像，或者上次调用 MI_VENC_StartRecvPicEx 后还没有接收到足够的图像，再一次调用此接口返回操作不允许。
- 该接口用于连续接收 N 帧并编码的场景，当 N=0 时，该接口等同于 MI_VENC_StartRecvPic。
- 如果通道已经调用了 MI_VENC_StartRecvPic 开始接收图像，停止接收图像，再次调用 MI_VENC_StartRecvPicEx 启动编码时，建议用户调用 MI_VENC_ResetChn 清除编码模块在调用该接口之前缓存的图像和码流。
- 如果创建 jpeg 通道抓拍，建议用户调用 MI_VENC_StartRecvPicEx，可以设定抓拍通道接收整数张图像后自动停止接收。

- 举例

```
MI_S32 JpegSnapProcess()
{
    MI_S32 s32Ret;
    MI_VENC_CHN VeChn=0;
    MI_VENC_ChnAttr_t stAttr;
    MI_VENC_RecvPicParam_t stRecvParam;

    /*set jpeg channel video encode attribute*/
    stAttr.stVeAttr.eType = E_MI_VENC_MODTYPE_JPEGE;
    stAttr.stVeAttr.stAttrJpeg.u32PicWidth = u32PicWidth;
    stAttr.stVeAttr.stAttrJpeg.u32PicHeight = u32PicHeigh;
    stAttr.stVeAttr.stAttrJpeg.u32MaxPicWidth = u32MaxPicWidth;
    stAttr.stVeAttr.stAttrJpeg.u32MaxPicHeight = u32MaxPicHeigh;

    //…omit other video encode assignments here.

    //create jpeg channel
    s32Ret = MI_VENC_CreateChn(VeChn, &stAttr);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_CreateChn err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }
    //start snapping
    stRecvParam.s32RecvPicNum = 2;
    s32Ret = MI_VENC_StartRecvPicEx(VeChn, &stRecvParam);
    if (s32Ret != MI_SUCCESS)
    {
        printf("MI_VENC_StartRecvPicEx err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }

    //…wait until all pictures have been encoded.

    s32Ret = MI_VENC_StopRecvPic(VeChn);
    if (s32Ret != MI_SUCCESS)
    {
        printf("MI_VENC_StopRecvPic err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }

s32Ret = MI_VENC_ResetChn(VeChn);
    if (s32Ret != MI_SUCCESS)
    {
        printf("MI_VENC_ResetChn err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }
    //destroy jpeg channel
    s32Ret = MI_VENC_DestroyChn(VeChn);
    if (s32Ret != MI_SUCCESS)
    {
        printf("MI_VENC_DestroyChn err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }

    return MI_SUCCESS;
}
```

➢ 相关主题

无。

1.3.9 MI_VENC_StopRecvPic

➢ 描述

停止编码通道接收输入图像。

➢ 语法

MI_S32 MI_VENC_StopRecvPic(MI_VENC_CHN VeChn);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 如果通道未创建，则返回失败。
- 此接口并不判断当前是否停止接收，即允许重复停止接收不返回错误。
- 此接口用于编码通道停止接收图像来编码，在编码通道销毁或复位前必须停止接收图像。
- 调用此接口仅停止接收原始数据编码，码流 buffer 并不会被清除。
- 调用 MI_VENC_StartRecvPic 和 MI_VENC_StartRecvPicEx 接口开始接收图像，都可以调用该接口来停止接收。

➢ 举例

请参见 MI_VENC_DestroyChn 的举例。

➢ 相关主题

无。

1.3.10 MI_VENC_Query

➢ 描述

查询编码通道状态。

➢ 语法

MI_S32 MI_VENC_Query(MI_VENC_CHN VeChn, MI_VENC_ChnStat_t *pstStat);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstStat	编码通道的状态指针。	输入/输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 如果通道未创建，则返回失败。
- 此接口用于查询此函数调用时刻的编码器状态，pstStat 包含四个主要的信息：
  1) u32LeftPics 表示待编码的帧个数。在复位通道前，可以通过查询是否还有图像待编码来决定复位时机，防止复位时将可能需要编码的帧清理出去。
  2) u32LeftStreamBytes 表示码流 buffer 中剩余的 byte 数目。在复位通道前，可以通过查询是否还有码流没有被处理来决定复位时机，防止复位时将可能需要的码流清理出去。
  3) u32LeftStreamFrames 表示码流 buffer 中剩余的帧数目。在复位通道前，可以通过查询是否还有图像的码流没有被取走来决定复位时机，防止复位时将可能需要的码流清理出去。
  4) u32CurPacks 表示当前帧的码流包个数。在调用 MI_VENC_GetStream 之前应确保 u32CurPacks 大于 0，否则会返回无 buffer 可取的 error。
- 在编码通道状态结构体中，u32LeftRecvPics 表示调用 MI_VENC_StartRecvPicEx 接口后剩余等待接收的帧数目。
- 在编码通道状态结构体中，u32LeftEncPics 表示调用 MI_VENC_StartRecvPicEx 接口后剩余等待编码的帧数目。
- 如果没有调用 MI_VENC_StartRecvPicEx，u32LeftRecvPics 和 u32LeftEncPics 数目始终为 0。

- **举例**

  请参见 MI_VENC_GetStream 的举例。

- **相关主题**

  无。

1.3.11 MI_VENC_SetChnAttr

➢ 描述

设置编码通道动态属性，包含编码宽高，profile，以及码控参数。

➢ 语法

MI_S32 MI_VENC_SetChnAttr(MI_VENC_CHN VeChn, MI_VENC_ChnAttr_t *pstAttr);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstAttr	编码通道属性指针。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 不能动态设置编码通道最大宽、最大高等属性。
- 编码通道属性包括了编码器属性和码率控制器属性两部分。
- 设置未创建的通道的属性，则返回失败。
- 如果 pstAttr 为空，则返回失败。
- 设置编码图像大小时的限制和创建通道时的限制一样。
- 编码通道属性分为动态属性和静态属性两种。其中，动态属性的属性值在通道创建时配置，在通道销毁之前可以被修改；静态属性的属性值在通道创建时配置，在通道创建之后不能被修改。
- 此接口只能设置编码通道属性中的动态属性，如果设置静态属性，则返回失败。编码通道的编码协议、获取码流的方式（按帧还是按包获取码流）、编码图像最大宽高属性属于静态属性。另外，各个编码协议的静态属性由各个协议模块指定，具体请参见 MI_VENC_ChnAttr_t。
- 设置编码通道属性中码率控制器属性，码率控制模式为 VBR，当码率控制高级参数中 u32MinIQp 小于 u32MinQp，其中 u32MinQp 是码率控制器属性，此接口将 u32MinIQp 的值修改为 u32MinQp。
- 设置编码器属性中通道宽高属性时，其中通道的优先级并不会恢复默认，编码通道的其它所有参数配置恢复默认值，并且清空码流 buffer 和缓存图像队列。

➢ 举例

无。

➢ 相关主题

无。

1.3.12 MI_VENC_GetChnAttr

➢ 描述

获取编码通道属性。

➢ 语法

MI_S32 MI_VENC_GetChnAttr(MI_VENC_CHN VeChn, MI_VENC_ChnAttr_t *pstAttr);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstAttr	编码通道属性指针。	输入/输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 获取未创建的通道的属性，返回失败 MI_ERR_VENC_UNEXIST。
- 如果 pstAttr 为空，则返回失败。

➢ 举例

无。

➢ 相关主题

无。

1.3.13 MI_VENC_GetStream

- **描述**

  获取编码的码流。

- **语法**

  MI_S32 MI_VENC_GetStream(MI_VENC_CHN VeChn, MI_VENC_Stream_t *pstStream, MI_S32 s32MilliSec);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstStream	码流结构体指针。上层需要将该指针指向一块分配好的内存，而且其内的成员 MI_VENC_Pack_t *pstPack 同样需要自行配置相应 PackCount*sizeof(MI_VENC_Pack_t)的内存。	输入/输出
s32MilliSec	调用该 api 直到获取到码流的超时等待时间。取值范围：>=-1 •-1：阻塞。 •0：非阻塞。 •大于 0：超时等待时间，该时间为相对时间，单位 ms。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 如果通道未创建，返回失败。
- 如果 pstStream 为空，返回 MI_ERR_VENC_NULL_PTR。
- 如果 s32MilliSec 小于-1，返回 MI_ERR_VENC_ILLEGAL_PARAM。
- 支持超时方式获取。支持 select/poll 系统调用。
- s32MilliSec=0 时，则为非阻塞获取，即如果缓冲无数据，则返回失败 MI_ERR_VENC_BUF_EMPTY。
  s32MilliSec=-1 时，则为阻塞，即如果缓冲无数据，则会等待有数据时才返回获取成功。
  s32MilliSec>0 时，则为超时，即如果缓冲无数据，则会等待用户设定的超时时间，若在设定的时间内有数据则返回获取成功，否则返回超时失败。
- 码流结构体 MI_VENC_Stream_t 包含 4 个部分：
  1) 码流包信息指针 pstPack 指向一组 MI_VENC_Pack_t 的内存空间，该空间由调用者分配。大小为 u32PackCount x sizeof(MI_VENC_Pack_t)，可以在 select 之后通过调用 MI_VENC_Query 获得。
  2) 码流包个数 u32PackCount 指定 pstPack 中 MI_VENC_Pack_t 的个数。一张完整的帧是由 u32PackCount 个码流包构成。
  3) 序列号 u32Seq 表示帧序列号。
  4) 码流特征信息 stH264Info/stJpegInfo/stH265Info，数据类型为联合体，包含了不同编码协议对应的码流特征信息，码流特征信息的输出用于支持用户的上层应用。

- 此接口应当和 MI_VENC_ReleaseStream 配对起来使用，用户获取码流后系统不会主动释放码流缓存，需要用户及时释放已经获取的码流缓存，否则可能会导致码流 buffer 满，影响编码器编码，并且用户必须按先获取先释放的顺序释放已经获取的码流缓存。
- 建议用户使用 select 方式获取码流，且按照如下的流程：
  （1）调用 MI_VENC_Query 函数查询编码通道状态；
  （2）确保 u32CurPacks 和 u32LeftStreamFrames 同时大于 0；
  （3）调用 malloc 分配 u32CurPacks 个包信息结构体；
  （4）调用 MI_VENC_GetStream 获取编码码流；
  （5）调用 MI_VENC_ReleaseStream 释放码流缓存。
- 以 H.264 为例介绍 pstPack[0]存储的码流结构，比如 pstPack[0]对应信息如下：，
  pstPack[0].u32DataNum=3,
  pstPack[0].asackInfo[0].stPackType. eH264EType=E_MI_VENC_H264E_NALU_SPS，
  pstPack[0].asackInfo[1].stPackType.eH264EType=E_MI_VENC_H264E_NALU_PPS，
  pstPack[0].asackInfo[2]. stPackType.eH264EType=E_MI_VENC_H264E_NALU_SEI；
  即 pstPack[0]包含三个 NALU 包，包含 SPS/PPS/SEI。如下图所示：

```
MI_S32 VencGetH264Stream()
{
    MI_S32 i;
    MI_S32 s32Ret;
    MI_S32 s32VencFd;
    MI_U32 u32FrameIdx = 0;
    MI_VENC_CHN VeChn = 0;
    MI_VENC_ChnAttr_t stStat;
    MI_VENC_Stream_t stStream;
    fd_set read_fds;
    FILE* pFile=NULL;

    pFile = fopen("stream.h264","wb");
    if (pFile == NULL)
    {
        return E_MI_ERR_FAILED;
    }
    s32VencFd = MI_VENC_GetFd(VeChn);
    do{
        FD_ZERO(&read_fds);
        FD_SET(s32VencFd, &read_fds);
        s32Ret = select(s32VencFd+1, &read_fds, NULL, NULL, NULL);
        if (s32Ret < 0)
        {
            printf("select err\n");
            return E_MI_ERR_FAILED;
        }
        else if (0 == s32Ret)
        {
            printf("timeout\n");
            return E_MI_ERR_FAILED;
        }
        else
        {
            if (FD_ISSET(s32VencFd, &read_fds))
            {
                s32Ret = MI_VENC_Query(VeChn, &stStat);
                if (s32Ret != MI_SUCCESS)
                {
                    return E_MI_ERR_FAILED;
                }
```

➢ 举例

```
    /*************************************************
    suggest to check both u32CurPacks and u32LeftStreamFrames at
    the same time,for example:
                if (0 == stStat.u32CurPacks || 0 ==
stStat.u32LeftStreamFrames)
                {
                    continue;
                }
    *************************************************/
                if (0 == stStat.u32CurPacks)
                {
                    continue;
                }
                stStream.pstPack =
(MI_VENC_Pack_t*)malloc(sizeof(MI_VENC_Pack_t)*stStat.u32CurPac
ks);
                if (NULL == stStream.pstPack)
                {
```

➢ 相关主题

无。

1.3.14 MI_VENC_ReleaseStream

➢ 描述

释放码流缓存。

➢ 语法

MI_S32 MI_VENC_ReleaseStream(MI_VENC_CHN VeChn, MI_VENC_Stream_t *pstStream);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstStream	码流结构体指针。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 如果通道未创建，则返回错误码 MI_ERR_VENC_UNEXIST。
- 如果 pstStream 为空，则返回错误码 MI_ERR_VENC_NULL_PTR。
- 此接口应当和 MI_VENC_GetStream 配对起来使用，用户获取码流后必须及时释放已经获取的码流缓存，否则可能会导致码流 buffer 满，影响编码器编码，并且用户必须按先获取先释放的顺序释放已经获取的码流缓存。
- 在编码通道复位以后，所有未释放的码流包均无效，不能再使用或者释放这部分无效的码流缓存。
- 释放无效的码流会返回失败 MI_ERR_VENC_ILLEGAL_PARAM。

➢ 举例

请参见 MI_VENC_GetStream 的举例。

➢ 相关主题

无。

1.3.15 MI_VENC_InsertUserData

➢ 描述

插入用户数据。用户首先申请一块空间，填充自定义的信息，传入对应的指针和数据长度，在下一个 SEI 的 NAL（跟在下一个 I 帧之前）就会有相应的信息。

➢ 语法

MI_S32 MI_VENC_InsertUserData(MI_VENC_CHN VeChn, MI_U8 *pu8Data, MI_U32 u32Len);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pu8Data	用户数据指针。	输入
u32Len	用户数据长度。 取值范围：(0,1024]，以 byte 为单位。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 如果通道未创建，则返回失败。
- 如果 pu8Data 为空，则返回失败。
- 插入用户数据，只支持 H.264/H.265 编码协议。
- H.264 协议通道最多同时分配 4 块内存空间用于缓存用户数据 ,且每段用户数据大小不超过 1kbyte。如果用户插入的数据多余 4 块，或插入的一段用户数据大于 1kbyte 时，此接口会返回错误。每段用户数据以 SEI 包的形式被插入到下一个 I 帧之前。在某段用户数据包被编码发送之后，H.264 通道内缓存这段用户数据的内存空间被清零，用于存放新的用户数据。

```
MI_S32 VencInsertUserData()
{
    MI_S32 s32Ret;
    MI_VENC_CHN VeChn = 0;
    MI_U8 au8UserData[]="sigmastar2020";
    s32Ret = MI_VENC_InsertUserData(VeChn, au8UserData,
sizeof(au8UserData));
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_InsertUserData err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }

    return s32Ret;
}
```

- **举例**

- **相关主题**

  无。

1.3.16 MI_VENC_SetMaxStreamCnt

- **描述**

  设置码流最大缓存帧数。

- **语法**

  MI_S32 MI_VENC_SetMaxStreamCnt(MI_VENC_CHN VeChn, MI_U32 u32MaxStrmCnt);

- **参数**

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
u32MaxStrmCnt	最大码流缓存帧数。取值范围：>=1	输入

- **返回值**

  返回值
  - MI_OK 成功。
  - 非 MI_OK 失败，参照错误码。

- **依赖**

  - 头文件：mi_common.h、mi_venc.h
  - 库文件：libmi.a

※ 注意

- 此接口用于设置编码通道的码流 buffer 中能够缓存的最大码流帧数。
- 若缓存码流帧数已达到最大码流帧数，当前待编码图像因码流 buffer 满而直接丢掉不编码。
- 最大码流帧数在创建通道时由系统内部指定默认值，默认值为 3。
- 此接口在创建通道之后，销毁编码通道之前均可以被调用。在下一帧开始编码之前生效。此接口允许被多次调用。建议在创建编码通道成功之后，启动编码前进行设置，不建议在编码过程中动态调整。

➢ 举例

无。

➢ 相关主题

无。

1.3.17 MI_VENC_GetMaxStreamCnt

➢ 描述

获取码流最大缓存帧数。

➢ 语法

MI_S32 MI_VENC_GetMaxStreamCnt(MI_VENC_CHN VeChn, MI_U32 *pu32MaxStrmCnt);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pu32MaxStrmCnt	最大码流缓存帧数的指针。	输入/输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

无。

➢ **举例**

无。

➢ **相关主题**

无。

1.3.18 MI_VENC_RequestIdr

➢ **描述**

请求一张 IDR 帧。

➢ **语法**

MI_S32 MI_VENC_RequestIdr(MI_VENC_CHN VeChn, MI_BOOL bInstant);

➢ **参数**

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
bInstant	是否使能立即编码 IDR 帧: 0: 在尽可能短的时间内编出 IDR 帧; 1: 下帧立即编码 IDR 帧	输入

➢ **返回值**

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ **依赖**

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ **注意**

- 如果通道未创建，则返回失败。
- IDR 帧请求，只支持 H.264/H.265 编码协议。
- 由于此接口不受帧率控制影响，每调用一次即编出一个 IDR，调用频繁会影响码流帧率和码率的稳定，使用时需注意

```
MI_S32 VencRequestIdr()
{
    MI_S32 s32Ret;
    MI_VENC_CHN VeChn = 0;
    MI_BOOL bInstant;

    bInstant = TRUE;
    s32Ret = MI_VENC_RequestIdr(VeChn, bInstant);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_RequestIDR err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }

    return s32Ret;
}
```

➢ **举例**

➢ **相关主题**

无。

1.3.19 MI_VENC_EnableIdr

➢ **描述**

是否使能 IDR 帧。

➢ **语法**

MI_S32 MI_VENC_EnableIdr(MI_VENC_CHN VeChn, MI_BOOL bEnableIdr);

➢ **参数**

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
bEnableIDR	是否使能的标志。	输入

➢ **返回值**

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ **依赖**

※ **注意**

- 如果通道未创建，则返回失败。
- 若不使能 IDR 帧，则在下一帧之后都编不出 IDR 帧或 I 帧，直到再次使能为止。
- 本接口只支持 H.264/H.265 编码协议。

➢ 举例

```
MI_S32 VencEnableIdr()
{
    MI_S32 s32Ret;
    MI_VENC_CHN VeChn = 0;
    s32Ret = MI_VENC_EnableIdr(VeChn);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_EnableIdr err0x%x\n",s32Ret);
        return E_MI_ERR_FAILED;
    }

    return s32Ret;
}
```

➢ 相关主题

无。

1.3.20 MI_VENC_SetH264IdrPicId

➢ 描述

设置 IDR 帧的 idr_pic_id 属性。

➢ 语法

MI_S32 MI_VENC_SetH264IdrPicId(MI_VENC_CHN VeChn, MI_VENC_H264IdrPicIdCfg_t *pstH264eIdrPicIdCfg);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH264IdrPicIdCfg	idr_pic_id 的参数指针。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 如果通道未创建，则返回失败。

- idr_pic_id 的参数主要由两个参数决定：
  1) enH264eIdrPicIdMode：设置 IDR 帧 idr_pic_id 的模式，
     enH264eIdrPicIdMode=E_MI_VENC_H264E_IDR_PIC_ID_MODE_AUTO 表示由编码内部流程自动计算生成 idr_pic_id，
     enH264eIdrPicIdMode=E_MI_VENC_H264E_IDR_PIC_ID_MODE_USR 表示由用户设定 idr_pic_id 的值。
  2) u32H264eIdrPicId：设置 IDR 帧 idr_pic_id 的值，当 enH264eIdrPicIdMode=E_MI_VENC_H264E_IDR_PIC_ID_MODE_USR 时该值才有效。
- 设置 IDR 帧的 idr_pic_id，只支持 H.264 编码协议。
- 此接口在创建通道之后，销毁编码通道之前均可以被调用。在下一帧开始编码之前生效。此接口允许被多次调用。建议在创建编码通道成功之后，启动编码前进行设置，不建议在编码过程中动态调整。
- 暂不支持。

➢ 举例

无。

➢ 相关主题

无。

1.3.21 MI_VENC_GetH264IdrPicId

➢ 描述

获取 IDR 帧的 idr_pic_id 的配置属性。

➢ 语法

MI_S32 MI_VENC_GetH264IdrPicId(MI_VENC_CHN VeChn, MI_VENC_H264IdrPicIdCfg_t *pstH264eIdrPicIdCfg);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH264eIdrPicIdCfg	idr_pic_id 的参数指针。	输出

➢ 返回值

返回值：
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

- 依赖
  - 头文件：mi_common.h、mi_venc.h
  - 库文件：libmi.a

※ 注意

- 如果通道未创建，则返回失败。
- 此接口在创建通道之后，销毁编码通道之前均可以被调用。
- 暂不支持。

➢ 举例

无。

➢ 相关主题

无。

1.3.22 MI_VENC_GetFd

➢ 描述

获取编码通道对应的设备文件句柄，用于调用 select/poll 监听是否有编码数据，与轮询 CPU 相比有效降低了 cpu 使用率。

➢ 语法

MI_S32 MI_VENC_GetFd(MI_VENC_CHN VeChn);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

无。

➢ 举例

请参见 MI_VENC_GetStream 的举例。

- **相关主题**

  无。

1.3.23 MI_VENC_CloseFd

➢ 描述

关闭编码通道对应的设备文件句柄。

➢ 语法

MI_S32 MI_VENC_CloseFd(MI_VENC_CHN VeChn);

➢ 参数

参数名称	描述	输入/输出
VeChn	视频编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 错误码

无。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

无。

➢ 举例

无。

➢ 相关主题

无。

1.3.24 MI_VENC_SetRoiCfg

➢ 描述

设置 H.264/H.265 通道的 ROI 属性。

➢ 语法

MI_S32 MI_VENC_SetRoiCfg(MI_VENC_CHN VeChn, MI_VENC_RoiCfg_t *pstVencRoiCfg);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstVencRoiCfg	ROI 区域参数。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 芯片差异

不同 Chip 和编码协议对 ROI 参数的限制也不同，如下表：

Chip	Codec	Alignment				bAbsQp	
		u32Left	u32Top	u32Width	u32Height	TRUE	FALSE
328Q/329D/326D	H.264	16	16	16	16	support	support
	H.265	32	32	32	32	support	support
325/325DE/327DE	H.264	16	16	16	16	support	support
	H.265	32	32	32	32	support	support
336D/336Q/339G	H.264	16	16	16	16	No support	support
	H.265	32	32	32	32	No support	support
335/337DE	H.264	16	16	16	16	No support	support
	H.265	32	32	32	32	No support	support

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于设置 H.264/H.265 协议编码通道 ROI 区域的参数。ROI 参数主要由 5 个参数决定：
  1）u32Index：系统支持每个通道可设置 8 个 ROI 区域，系统内部按照 0～7 的索引号对 ROI 区域进行管理，u32Index 表示的用户设置 ROI 的索引号。ROI 区域之间可以互相叠加，且当发生叠加时，ROI 区域之间的优先级按照索引号 0～7 依次提高。
  2）bEnable：指定当前的 ROI 区域是否使能。
  3）bAbsQp：指定当前的 ROI 区域采用绝对 QP 方式或是相对 QP。

4）s32Qp :当 bAbsQp 为 true 时 ,s32Qp 表示 ROI 区域内部的所有宏块采用的 QP 值 ,当 bAbsQp 为 false 时，s32Qp 表示 ROI 区域内部的所有宏块采用的相对 QP 值。

5）stRect :指定当前的 ROI 区域的位置坐标和区域的大小。ROI 区域的起始点坐标必须在图像范围内。

- 本接口属于高级接口，系统默认没有 ROI 区域使能，用户必须调用此接口启动 ROI。
- 本接口可在编码通道创建之后，编码通道销毁之前设置。此接口在编码过程中被调用时，等到下一个帧时生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 建议用户在调用此接口之前，先调用 MI_VENC_GetRoiCfg 接口，获取当前通道的 ROI 配置，然后再进行设置。

```
MI_S32 VencSetRoi()
{
    MI_S32 s32Ret;
    MI_VENC_RoiCfg_t stRoiCfg;
    MI_S32 index=0;
    MI_VENC_CHN VeChnId=0;
    //...omit other thing
    s32Ret = MI_VENC_GetRoiCfg(VeChnId, index, &stRoiCfg);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_GetRoiCfg err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }
    stRoiCfg.bEnable = TRUE;
    stRoiCfg.bAbsQp  = FALSE;
    stRoiCfg.s32Qp = 10;
    stRoiCfg.stRect.u32Left = 16;
    stRoiCfg.stRect.u32Top = 16;
    stRoiCfg.stRect.u32Width = 16;
    stRoiCfg.stRect.u32Height = 16;
    stRoiCfg.u32Index = 0;
    s32Ret = MI_VENC_SetRoiCfg(VeChnId, &stRoiCfg);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_SetRoiCfg err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }

    return s32Ret;
}
```

➢ 举例

➢ 相关主题

无。

1.3.25 MI_VENC_GetRoiCfg

- **描述**

  获取 H.264/H.265 通道的 Roi 配置属性。

- **语法**

  MI_S32 MI_VENC_GetRoiCfg(MI_VENC_CHN VeChn, MI_U32 u32Index, MI_VENC_RoiCfg_t *pstVencRoiCfg);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
u32Index	H.264 协议编码通道 ROI 区域索引。	输入
pstVencRoiCfg	对应 ROI 区域的配置。	输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于获取 H.264 协议编码通道的 index 的 ROI 配置。
- 本接口可在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

➢ 举例

请参见 MI_VENC_GetRoiCfg 的举例。

➢ 相关主题

无。

1.3.26 MI_VENC_SetRoiBgFrameRate

➢ 描述

设置 H.264/H.265 通道的非 ROI 区域帧率属性。

➢ 语法

MI_S32 MI_VENC_SetRoiBgFrameRate(MI_VENC_CHN VeChn, MI_VENC_RoiBgFrameRate_t *pstRoiBgFrmRate);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入

pstRoiBgFrmRate	非 ROI 区域帧率控制参数。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于设置 H.264 协议编码通道非 ROI 区域帧率控制的参数。
- 本接口调用之前必须首先使能 ROI 区域。
- 本接口属于高级接口，系统默认没有使能非 ROI 区域帧率属性，用户必须调用此接口使能。
- 本接口可在编码通道创建之后，编码通道销毁之前设置。此接口在编码过程中被调用时，等到下一帧时生效。
- 设置通道帧率控制属性时，输入帧率 SrcFrmRate 大于输出帧率 DstFrmRate 且 DstFrmRate 大于等于 0 或同时等于-1。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 建议用户在调用此接口之前，先调用 MI_VENC_GetRoiBgFrameRate 接口，
- 获取当前通道的非 ROI 区域帧率配置，然后再进行设置。
- 设置非 ROI 区域的帧率低于 ROI 区域时，不建议使用跳帧参考或者 svc-t 编码。跳帧参考或 svc-t 编码场景下，如果设置了非 ROI 区域的帧率低于 ROI 区域，则实际编码的非 ROI 区域目标帧率可能大于用户配置的目标帧率，因为 base 层的非 ROI 区域不能编码为 pskip 块。例如在调用了此接口设置了非 ROI 区域帧率比例为 2:1 之前，还调用了 MI_VENC_SetRefParam 设置长参考帧，会导致实际编码的非 ROI 区域帧率比例低于 2:1。

➢ 举例

```
MI_S32 VencSetRoiFrameRate()
{
    MI_S32 s32Ret;
    MI_VENC_RoiBgFrameRate_t stRoiBgFrameRate;
    MI_S32 index = 0;
    MI_VENC_CHN VeChnId=0;

    //...omit other thing
    stRoiBgFrameRate.s32SrcFrmRate=30;
    stRoiBgFrameRate.s32DstFrmRate=15;
    s32Ret = MI_VENC_SetRoiBgFrameRate(VeChnId, &stRoiBgFrameRate);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_SetRoiBgFrameRate err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }

    return s32Ret;
}
```

➢ 相关主题

无。

1.3.27 MI_VENC_GetRoiBgFrameRate

➢ 描述

获取 H.264/H.265 通道的非 Roi 区域帧率配置属性。

➢ 语法

MI_S32 MI_VENC_GetRoiBgFrameRate(MI_VENC_CHN VeChn, MI_VENC_RoiBgFrameRate_t *pstRoiBgFrmRate);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstRoiBgFrmRate	非 ROI 区域帧率的配置。	输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口可在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

➢ 举例

请参见 MI_VENC_SetRoiBgFrameRate 的举例。

➢ 相关主题

无。

1.3.28 MI_VENC_SetH264SliceSplit

➢ 描述

设置 H.264 通道的 slice 分割属性。

➢ 语法

MI_S32 MI_VENC_SetH264SliceSplit(MI_VENC_CHN VeChn, MI_VENC_ParamH264SliceSplit_t *pstSliceSplit);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstSliceSplit	H.264 码流 slice 分割参数。	输入

➢ 返回值

返回值：
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意–

- 本接口用于设置 H.264 协议编码通道码流的分割方式
- Slice 分割属性主要由两个参数决定
  1) bSplitEnable：当前帧是否进行 slice 分割。
  2) u32SliceRowCount：slice 按照宏块行进行分割。u32SliceRowCount 表示每个 slice 占图像宏块行数。且当编码至图像的最后几行，不足 u32SliceRowCount 时，剩余的宏块行被划分为一个 slice。
- 系统默认 bSplitEnable 为 false，当设置 bSplitEnable 为 true 时，以 u32SliceRowCount 宏块行作为一个 slice 单独编码，每编码完成一个 slice 用户即可获取该 slice 的码流数据。建议 u32SliceRowCount 不要设置过小，u32SliceRowCount 越小用户收到通知越频繁，系统开销越大，且容易造成码流输出内存碎片，降低码流输出内存的使用效率。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。此接口在编码过程中被调用时，等到下一帧时生效。
- 建议用户在调用此接口之前，先调用 MI_VENC_GetH264SliceSplit 接口，获取当前通道的 slicesplit 配置，然后再进行设置。

➢ 举例

```
MI_S32 VencSetSliceSplit()
{
    MI_S32 s32Ret;
    MI_VENC_ParamH264SliceSplit_t stSlice;
    MI_VENC_CHN VeChnId = 0;

    //...omit other thing
    s32Ret = MI_VENC_GetH264SliceSplit(VeChnId, &stSlice);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_GetH264SliceSplit err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }
    stSlice.bSplitEnable = TRUE;
    stSlice.u32SliceRowCount = 8;
    s32Ret = MI_VENC_SetH264SliceSplit(VeChnId, &stSlice);
    if(MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_SetH264SliceSplit err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }

    return s32Ret;
}
```

➢ **相关主题**

无。

1.3.29 MI_VENC_GetH264SliceSplit

➢ **描述**

获取 H.264 通道的 slice 分割属性。

➢ **语法**

MI_S32 MI_VENC_GetH264SliceSplit(MI_VENC_CHN VeChn, MI_VENC_ParamH264SliceSplit_t *pstSliceSplit);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstSliceSplit	H.264 码流 slice 分割参数。	输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于获取 H.264 协议编码通道的 slice 分割属性。
- 本接口可在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

➢ 举例

请参见 MI_VENC_SetH264SliceSplit 的举例。

➢ 相关主题

无。

1.3.30 MI_VENC_SetH264InterPred

➢ 描述

设置 H.264 协议编码通道的帧间预测属性。

➢ 语法

MI_S32 MI_VENC_SetH264InterPred(MI_VENC_CHN VeChn, MI_VENC_ParamH264InterPred_t *pstH264InterPred);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH264InterPred	H.264 协议编码通道的帧间预测配置。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于设置 H.264 协议编码通道的帧间预测的配置
- 帧间预测的属性主要由七个参数决定：
  1）u32HWSize：帧间预测时的水平搜索窗的大小。
  2）u32VWSize：帧间预测时的垂直搜索窗的大小。
  3）bInter16x16PredEn：16x16 块帧间预测使能标识。
  4）bInter16x8PredEn：16x8 块帧间预测使能标识。
  5）bInter8x16PredEn：8x16 块帧间预测使能标识。
  6）bInter8x8PredEn：8x8 块帧间预测使能标识。
  7） bExtedgeEn：帧间预测扩边搜索使能。对于图像边界上的宏块进行帧间预测时，搜索窗的范围会超出图像的边界，此时，扩边搜索使能会对图像进行扩边，进行帧间预测。如果扩编搜索使能被关闭，那么对于超出图像范围的搜索窗，不会进行预测。
- 此接口属于高级接口，用户可以选择性调用，建议不调用，系统会有默认值。默认搜索窗的大小会根据图像的分辨率不同，其他 5 个预测使能开关全部默认使能。
- bInter16x16PredEn 的预测开关只能开启。
- 本接口可在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，等到下一帧时生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 建议用户在调用此接口之前，先调用 MI_VENC_GetH264InterPred 接口，获取当前编码通道的 InterPred 配置，然后再进行设置。

➢ 举例

```
MI_S32 VencSetInterPred()
{
    MI_S32 s32Ret;
    MI_VENC_ParamH264InterPred_t stInterPred;
    MI_VENC_CHN VeChnId = 0;

    //...omit other thing
    s32Ret = MI_VENC_GetH264InterPred(VeChnId, &stInterPred);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_GetH264InterPred err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }
    stInterPred.bExtedgeEn = TRUE;
    stInterPred.bInter16x16PredEn = TRUE;
    stInterPred.bInter16x8PredEn = FALSE;
    stInterPred.bInter8x16PredEn = FALSE;
    stInterPred.bInter8x8PredEn = FALSE;
    stInterPred.u32HWSize = 4;
    stInterPred.u32VWSize = 2;
    s32Ret = MI_VENC_SetH264InterPred(VeChnId, &stInterPred);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_SetH264InterPred err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }

    return s32Ret;
}
```

➢ **相关主题**

无。

1.3.31 MI_VENC_GetH264InterPred

➢ **描述**

获取 H.264 协议编码通道的帧间预测属性。

➢ **语法**

MI_S32 MI_VENC_GetH264InterPred(MI_VENC_CHN VeChn, MI_VENC_ParamH264InterPred_t *pstH264InterPred);

➢ **参数**

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH264InterPred	H.264 协议编码通道的帧间预测参数。	输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于获取 H.264 协议编码通道的帧间预测方式。
- 本接口可在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

➢ 举例

请参见 MI_VENC_SetH264InterPred 的举例。

➢ 相关主题

无。

1.3.32 MI_VENC_SetH264IntraPred

➢ 描述

设置 H.264 协议编码通道的帧内预测属性。

➢ 语法

MI_S32 MI_VENC_SetH264IntraPred(MI_VENC_CHN VeChn, MI_VENC_ParamH264IntraPred_t *pstH264IntraPred);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH264IntraPred	H.264 协议编码通道的帧内预测配置。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意–

- 本接口用于设置 H.264 协议编码通道的帧内预测的配置
- 帧间预测的属性主要由四个参数决定：
  1） bIntra16x16PredEn：16x16 块帧内预测使能标识。
  2）bIntraNxNPredEn：NxN 块帧内预测使能标识。其中，NxN 表示 4x4 和 8x8。
  3）bIpcmEn：IPCM 块帧内预测使能标识。
  4）constrained_intra_pred_flag：具体的含义，请参见 H.264 协议。
- 此接口属于高级接口，用户可以选择性调用，不建议调用，系统会有默认值。系统默认使能 bIntra16x16PredEn 和 bIntraNxNPredEn，bIpcmEn 对于不同芯片有不同默认值，constrained_intra_pred_flag 默认为 0。
- bIntra16x16PredEn、bIntraNxNPredEn 两种帧内预测使能开关必须有一个是开启的，系统不支持两个开关全部关闭。
- 本接口可在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，等到下一个 I 帧时生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 建议用户在调用此接口之前，先调用 MI_VENC_GetH264IntraPred 接口，获取当前编码通道的 IntraPred 配置，然后再进行设置。

➢ 举例

```
MI_S32 VencSetIntraPred()
{
    MI_S32 s32Ret;
    MI_VENC_ParamH264IntraPred_t stIntraPred;
    MI_VENC_CHN VeChnId = 0;

    //...omit other thing
    s32Ret = MI_VENC_GetH264IntraPred(VeChnId, &stIntraPred);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_GetH264IntraPred err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }
    stIntraPred.bIntra16x16PredEn = TRUE;
    stIntraPred.bIntraNxNPredEn      = FALSE;
    stIntraPred.bIpcmEn     = FALSE;
    stIntraPred.constrained_intra_pred_flag = 0;
    s32Ret = MI_VENC_SetH264IntraPred(VeChnId, &stIntraPred);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_SetH264IntraPred err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }

    return s32Ret;
}
```

➢ **相关主题**

无。

1.3.33 MI_VENC_GetH264IntraPred

➢ **描述**

获取 H.264 协议编码通道的帧内预测属性。

➢ **语法**

MI_S32 MI_VENC_GetH264IntraPred(MI_VENC_CHN VeChn, MI_VENC_ParamH264IntraPred_t *pstH264IntraPred);

➢ **参数**

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH264IntraPred	H.264 协议编码通道的帧内预测参数。	输出

➢ **返回值**

返回值 { MI_OK 成功。

非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于获取 H.264 协议编码通道的帧内预测方式。
- 本接口可在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

➢ 举例

请参见 MI_VENC_SetH264IntraPred 的举例。

➢ 相关主题

无。

1.3.34 MI_VENC_SetH264Trans

➢ 描述

设置 H.264 协议编码通道的变换、量化的属性。

➢ 语法

MI_S32 MI_VENC_SetH264Trans(MI_VENC_CHN VeChn, MI_VENC_ParamH264Trans_t *pstH264Trans);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH264Trans	H.264 协议编码通道的变换、量化属性。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h

- 库文件：mi_venc.ko、libmi_venc.so

※ 注意

- 本接口用于设置 H.264 协议编码通道的变换、量化的配置。
- 变换、量化属性主要由三个参数组成：
  1) u32IntraTransMode：帧内预测宏块的变换属性。u32IntraTransMode=0 表示支持对帧内预测宏块支持 4x4 变换和 8x8 变换 ;u32IntraTransMode=1 表示只支持对帧内预测宏块支持 4x4 变换；u32IntraTransMode=2 表示只支持对帧内预测宏块支持 8x8 变换。只有在编码通道协议为 highprofile 时，才能选择 8x8 变换，即在 baselineprofile 和 mainprofile 下，系统只支持 u32IntraTransMode=1 的配置。
  2) u32InterTransMode：帧间预测宏块的变换属性。u32InterTransMode=0 表示支持对帧间预测宏块支持 4x4 变换和 8x8 变换 ;u32InterTransMode=1 表示只支持对帧间预测宏块支持 4x4 变换；u32InterTransMode=2 表示只支持对帧间预测宏块支持 8x8 变换。只有在编码通道协议为 highprofile 时，才能选择 8x8 变换，即在 baselineprofile 和 mainprofile 下，系统只支持 u32InterTransMode=1 的配置。
  3) s32ChromaQpIndexOffset：具体含义请参见 H.264 协议关于 chroma_qp_index_offset 的解释。
- 此接口属于高级接口 ,用户可以选择性调用 ,不建议调用 ,系统会有默认值。系统会根据不同的 profile 设置默认参数。不同 profile 下系统默认 trans 参数如下表：

Profile	u32IntraTransMode	u32InterTransMode	bScalingListValid
Baseline/mainprofile	1	1	false
Highprofile	0	0	false

- 本接口可在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，等到下一个 I 帧时生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 建议用户在调用此接口之前，先调用 MI_VENC_GetH264Trans 接口，获取当前编码通道的 trans 配置，然后再进行设置。

➢ 举例

```
MI_S32 VencSetTrans()
{
    MI_S32 s32Ret;
    MI_VENC_ParamH264Trans_t stTrans;
    MI_VENC_CHN VeChnId = 0;

    //...omit other thing
    s32Ret = MI_VENC_GetH264Trans(VeChnId, &stTrans);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_GetH264Trans err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }
    stTrans.u32IntraTransMode = 2;
    stTrans.u32InterTransMode = 2;
    stTrans.s32ChromaQpIndexOffset = 2;
    s32Ret = MI_VENC_SetH264Trans(VeChnId, &stTrans);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_SetH264Trans err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }

    return s32Ret;
}
```

➢ **相关主题**

无。

1.3.35 MI_VENC_GetH264Trans

➢ **描述**

获取 H.264 协议编码通道的变换、量化属性。

➢ **语法**

MI_S32 MI_VENC_GetH264Trans(MI_VENC_CHN VeChn, MI_VENC_ParamH264Trans_t *pstH264Trans);

➢ **参数**

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH264Trans	H.264 协议编码通道的变换、量化参数。	输出

➢ **返回值**

返回值：
- MI_OK 成功。

非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于获取 H.264 协议编码通道的变换、量化配置。
- 本接口可在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

➢ 举例

请参见 MI_VENC_SetH264Trans 的举例。

➢ 相关主题

无。

1.3.36 MI_VENC_SetH264Entropy

➢ 描述

设置 H.264 协议编码通道的熵编码模式。

➢ 语法

MI_S32 MI_VENC_SetH264Entropy(MI_VENC_CHN VeChn, MI_VENC_ParamH264Entropy_t *pstH264EntropyEnc);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH264EntropyEnc	H.264 协议编码通道的熵编码模式。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h

- 库文件：libmi.a

※ 注意

- 本接口用于设置 H.264 协议编码通道的熵编码的配置。
- 变换、量化属性主要由两个参数组成：

1) u32EntropyEncModeI：I 帧的熵编码方式，u32EntropyEncModeI=0 表示 I 帧使用 cavlc 编码，u32EntropyEncModeI=1 表示 I 帧使用 cabac 编码方式。

2) u32EntropyEncModeP：P 帧的熵编码方式，u32EntropyEncModeP=0 表示 P 帧使用 cavlc 编码，u32EntropyEncModeP=1 表示 P 帧使用 cabac 编码方式。

- I 帧的熵编码方式与 P 帧的熵编码方式可以分别设置。
- Baseline profile 不支持 cabac 编码方式，支持 cavlc 编码方式，mainprofile 和 highprofile 支持 cabac 编码方式和 cavlc 编码方式。
- Cabac 编码方式相对于 cavlc 编码方式，需要更大的计算量，可是，会产生更少的码流。如果系统性能不够富裕，建议用户可以采取 I 帧 cavlc 编码，P 帧 cabac 编码。
- 此接口属于高级接口，用户可以选择性调用，不建议调用，系统会有默认值。系统会根据不同的 profile 设置默认参数。不同 profile 下系统默认熵编码参数如下表：

Profile	u32EntropyEncModeI	u32EntropyEncModeP
Baseline	0	0
Mainprofile/Highprofile	1	1

- 本接口可在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，等到下一个 I 帧时生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 建议用户在调用此接口之前，先调用 MI_VENC_GetH264Entropy 接口，获取当前编码通道的 Entropy 配置，然后再进行设置。

➢ 举例

```
MI_S32 VencSetEntropy()
{
    MI_S32 s32Ret;
    MI_VENC_ParamH264Entropy_t stEntropy;
    MI_VENC_CHN VeChnId = 0;

    //...omit other thing
    s32Ret = MI_VENC_GetH264Entropy(VeChnId, &stEntropy);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_GetH264Entropy err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }
    stEntropy.u32EntropyEncModeI = 0;
    stEntropy.u32EntropyEncModeP = 0;
    s32Ret = MI_VENC_SetH264Entropy(VeChnId, &stEntropy);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_SetH264Entropy err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }

    return s32Ret;
}
```

➢ **相关主题**

无。

1.3.37 MI_VENC_GetH264Entropy

➢ **描述**

获取 H.264 协议编码通道的熵编码属性。

➢ **语法**

MI_S32 MI_VENC_GetH264Entropy(MI_VENC_CHN VeChn, MI_VENC_ParamH264Entropy_t *pstH264EntropyEnc);

➢ **参数**

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH264EntropyEnc	H.264 协议编码通道的熵编码属性。	输出

➢ **返回值**

返回值：
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于获取 H.264 协议编码通道的熵编码配置。
- 本接口可在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

➢ 举例

请参见 MI_VENC_SetH264Entropy 的举例。

➢ 相关主题

无。

1.3.38 MI_VENC_SetH265InterPred

➢ 描述

设置 H.265 协议编码通道的帧间预测属性。

➢ 语法

MI_S32 MI_VENC_SetH265InterPred(MI_VENC_CHN VeChn, MI_VENC_ParamH265InterPred_t *pstH265InterPred);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH265InterPred	H.265 协议编码通道的帧间预测配置。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于设置 H.265 协议编码通道的帧间预测的配置
- 帧间预测的属性主要由七个参数决定：
  1）u32HWSize：帧间预测时的水平搜索窗的大小。
  2）u32VWSize：帧间预测时的垂直搜索窗的大小。
  3）bInter16x16PredEn：16x16 块帧间预测使能标识。
  4）bInter16x8PredEn：16x8 块帧间预测使能标识。
  5）bInter8x16PredEn：8x16 块帧间预测使能标识。
  6）bInter8x8PredEn：8x8 块帧间预测使能标识。
  7） bExtedgeEn：帧间预测扩边搜索使能。对于图像边界上的宏块进行帧间预测时，搜索窗的范围会超出图像的边界，此时，扩边搜索使能会对图像进行扩边，进行帧间预测。如果扩编搜索使能被关闭，那么对于超出图像范围的搜索窗，不会进行预测。
- 此接口属于高级接口，用户可以选择性调用，建议不调用，系统会有默认值。默认搜索窗的大小会根据图像的分辨率不同，其他 5 个预测使能开关全部默认使能。
- bInter16x16PredEn 的预测开关只能开启。
- 本接口可在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，等到下一帧时生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 建议用户在调用此接口之前，先调用 MI_VENC_GetH265InterPred 接口，获取当前编码通道的 InterPred 配置，然后再进行设置。

➢ 举例

请参考 MI_VENC_SetH264InterPred 的举例。

➢ 相关主题

无。

1.3.39 MI_VENC_GetH265InterPred

➢ 描述

获取 H.265 协议编码通道的帧间预测属性。

➢ 语法

MI_S32 MI_VENC_GetH265InterPred(MI_VENC_CHN VeChn, MI_VENC_ParamH265InterPred_t *pstH265InterPred);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH265InterPred	H.265 协议编码通道的帧间预测参数。	输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于获取 H.265 协议编码通道的帧间预测方式。
- 本接口可在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

➢ 举例

请参见 MI_VENC_SetH265InterPred 的举例。

➢ 相关主题

无。

1.3.40 MI_VENC_SetH265IntraPred

➢ 描述

设置 H.265 协议编码通道的帧内预测属性。

➢ 语法

MI_S32 MI_VENC_SetH265IntraPred(MI_VENC_CHN VeChn, MI_VENC_ParamH265IntraPred_t *pstH265IntraPred);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH265IntraPred	H.265 协议编码通道的帧内预测配置。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于设置 H.265 协议编码通道的帧内预测的配置
- 帧内预测的属性主要由三个参数决定：
  1) u32Intra32x32Penalty：32x32 块帧内预测被选中概率，值越大概率越低。
  2) u32Intra16x16Penalty：16x16 块帧内预测被选中概率，值越大概率越低。
  3) u32Intra8x8Penalty：8x8 块帧内预测被选中概率，值越大概率越低。
- 此接口属于高级接口，用户可以选择性调用，不建议调用，系统会有默认值。
- 本接口可在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，等到下一个 I 帧时生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 建议用户在调用此接口之前，先调用 MI_VENC_GetH265IntraPred 接口，获取当前编码通道的 IntraPred 配置，然后再进行设置。

➢ 举例

```
MI_S32 H265SetIntraPred()
{
    MI_S32 s32Ret;
    MI_VENC_ParamH265IntraPred_t stIntraPred;
    MI_VENC_CHN VeChnId = 0;

    //...omit other thing
    s32Ret = MI_VENC_GetH265IntraPred(VeChnId, &stIntraPred);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_GetH265IntraPred err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }
    stIntraPred.u32Intra32x32Penalty = 0;
    stIntraPred.u32Intra16x16Penalty = 0;
    stIntraPred.u32Intra8x8Penalty = 10;
    s32Ret = MI_VENC_SetH265IntraPred(VeChnId, &stIntraPred);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_SetH265IntraPred err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }

    return s32Ret;
}
```

➢ **相关主题**

无。

1.3.41 MI_VENC_GetH265IntraPred

➢ **描述**

获取 H.265 协议编码通道的帧内预测属性。

➢ **语法**

MI_S32 MI_VENC_GetH265IntraPred(MI_VENC_CHN VeChn, MI_VENC_ParamH265IntraPred_t *pstH265IntraPred);

➢ **参数**

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH265IntraPred	H.265 协议编码通道的帧内预测参数。	输出

➢ **返回值**

MI_OK 成功。

返回值

非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于获取 H.265 协议编码通道的帧内预测方式。
- 本接口可在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

➢ 举例

请参见 MI_VENC_SetH265IntraPred 的举例。

➢ 相关主题

无。

1.3.42 MI_VENC_SetH265Trans

➢ 描述

设置 H.265 协议编码通道的变换、量化的属性。

➢ 语法

MI_S32 MI_VENC_SetH265Trans(MI_VENC_CHN VeChn, MI_VENC_ParamH265Trans_t *pstH265Trans);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH265Trans	H.265 协议编码通道的变换、量化属性。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意–

- 本接口用于设置 H.265 协议编码通道的变换、量化的配置。
- 变换、量化属性主要由三个参数组成：
  1) u32IntraTransMode：帧内预测宏块的变换属性。u32IntraTransMode=0 表示支持对帧内预测宏块支持 4x4 变换和 8x8 变换；u32IntraTransMode=1 表示只支持对帧内预测宏块支持 4x4 变换；u32IntraTransMode=2 表示只支持对帧内预测宏块支持 8x8 变换。只有在编码通道协议为 highprofile 时，才能选择 8x8 变换，即在 baselineprofile 和 mainprofile 下，系统只支持 u32IntraTransMode=1 的配置。
  2) u32InterTransMode：帧间预测宏块的变换属性。u32InterTransMode=0 表示支持对帧间预测宏块支持 4x4 变换和 8x8 变换；u32InterTransMode=1 表示只支持对帧间预测宏块支持 4x4 变换；u32InterTransMode=2 表示只支持对帧间预测宏块支持 8x8 变换。只有在编码通道协议为 highprofile 时，才能选择 8x8 变换，即在 baselineprofile 和 mainprofile 下，系统只支持 u32InterTransMode=1 的配置。
  3) s32ChromaQpIndexOffset：具体含义请参见 H.265 协议关于 slice_cb_qp_offset 的解释。
- 此接口属于高级接口 ,用户可以选择性调用 ,不建议调用 ,系统会有默认值。系统会根据不同的 profile 设置默认参数。不同 profile 下系统默认 trans 参数如下表：

Profile	u32IntraTransMode	u32InterTransMode	bScalingListValid
Baseline/mainprofile	1	1	false
Highprofile	0	0	false

- 本接口可在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，等到下一个 I 帧时生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

建议用户在调用此接口之前，先调用 MI_VENC_GetH265Trans 接口，获取当前编码通道的 trans 配置，然后再进行设置。

➢ 举例

```
MI_S32 H265SetTrans()
{
    MI_S32 s32Ret;
    MI_VENC_ParamH265Trans_t stTrans;
    MI_VENC_CHN VeChnId = 0;

    //...omit other thing
    s32Ret = MI_VENC_GetH265Trans(VeChnId, &stTrans);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_GetH265Trans err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }
    stTrans.u32IntraTransMode = 2;
    stTrans.u32InterTransMode = 2;
    stTrans.s32ChromaQpIndexOffset = 2;
    s32Ret = MI_VENC_SetH265Trans(VeChnId, &stTrans);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_SetH265Trans err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }

    return s32Ret;
}
```

➢ **相关主题**

无。

1.3.43 MI_VENC_GetH265Trans

➢ **描述**

获取 H.265 协议编码通道的变换、量化属性。

➢ **语法**

MI_S32 MI_VENC_GetH265Trans(MI_VENC_CHN VeChn, MI_VENC_ParamH265Trans_t *pstH265Trans);

➢ **参数**

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH265Trans	H.265 协议编码通道的变换、量化参数。	输出

➢ **返回值**

返回值：
- MI_OK 成功。

非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于获取 H.265 协议编码通道的变换、量化配置。
- 本接口可在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

➢ 举例

请参见 MI_VENC_SetH265Trans 的举例。

➢ 相关主题

无。

1.3.44 MI_VENC_SetH264Dblk

➢ 描述

设置 H.264 协议编码通道的 Deblocking 类型。

➢ 语法

MI_S32 MI_VENC_SetH264Dblk(MI_VENC_CHN VeChn, MI_VENC_ParamH264Dblk_t *pstH264Dblk);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH264Dblk	H.264 协议编码通道的 Deblocking 参数。	输入

➢ 返回值

返回值：
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意–

- 本接口用于设置 H.264 协议编码通道的 Deblocking 的配置
- 变换、量化属性主要由三个参数组成：
  1）disable_deblocking_filter_idc：具体含义请参见 H.264 协议。
  2）slice_alpha_c0_offset_div2：具体含义请参见 H.264 协议。
  3）slice_beta_offset_div2：具体含义请参见 H.264 协议。
- 系统默认打开 deblocking 功能，默认 disable_deblocking_filter_idc=0，slice_alpha_c0_offset_div2=0，slice_beta_offset_div2=0。
- 如果用户想关闭 deblocking 功能，可以将 disable_deblocking_filter_idc 置为 1。
- 本接口可在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，等到下一帧时生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 建议用户在调用此接口之前 ,先调用 MI_VENC_GetH264Dblk 接口 ,获取当前编码通道的 Dblk 配置，然后再进行设置。

➢ 举例

```
MI_S32 VencSetDblk()
{
    MI_S32 s32Ret;
    MI_VENC_ParamH264Dblk_t stDblk;
    MI_VENC_CHN VeChnId = 0;

    //...omit other thing
    s32Ret = MI_VENC_GetH264Dblk(VeChnId, &stDblk);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_GetH264Dblk err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }
    stDblk.disable_deblocking_filter_idc = 0;
    stDblk.slice_alpha_c0_offset_div2 = 6;
    stDblk.slice_beta_offset_div2 = 5;
    s32Ret = MI_VENC_SetH264Dblk(VeChnId,&stDblk);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_SetH264Dblk err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }

    return s32Ret;
}
```

➢ 相关主题

无。

1.3.45 MI_VENC_GetH264Dblk

- 描述

  获取 H.264 协议编码通道的 dblk 类型。

- 语法

  MI_S32 MI_VENC_GetH264Dblk(MI_VENC_CHN VeChn, MI_VENC_ParamH264Dblk_t *pstH264Dblk);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH264Dblk	H.264 协议编码通道的 dblk 属性。	输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于获取 H.264 协议编码通道的 dblk 配置。
- 本接口可在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

➢ 举例

请参见 MI_VENC_SetH264Dblk 的举例。

➢ 相关主题

无。

1.3.46 MI_VENC_SetH265Dblk

➢ 描述

设置 H.265 协议编码通道的 Deblocking 类型。

➢ 语法

MI_S32 MI_VENC_SetH265Dblk(MI_VENC_CHN VeChn, MI_VENC_ParamH265Dblk_t *pstH265Dblk);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH265Dblk	H.265 协议编码通道的 Deblocking 参数。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意–

- 本接口用于设置 H.265 协议编码通道的 Deblocking 的配置
- 变换、量化属性主要由三个参数组成：
  1）disable_deblocking_filter_idc：具体含义请参见 H.265 协议关于 slice_deblocking_filter_disabled_flag 的解释。
  2）slice_tc_offset_div2：具体含义请参见 H.265 协议。
  3）slice_beta_offset_div2：具体含义请参见 H.265 协议。
- 系统默认打开 deblocking 功能，默认 disable_deblocking_filter_idc=0， slice_tc_offset_div2=0，slice_beta_offset_div2=0。
- 如果用户想关闭 deblocking 功能，可以将 disable_deblocking_filter_idc 置为 1。
- 本接口可在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，等到下一帧时生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 建议用户在调用此接口之前 ,先调用 MI_VENC_GetH265Dblk 接口 ,获取当前编码通道的 Dblk 配置，然后再进行设置。

➢ 举例

请参见 MI_VENC_SetH264Dblk 的举例。

➢ 相关主题

无。

1.3.47 MI_VENC_GetH265Dblk

➢ 描述

获取 H.265 协议编码通道的 dblk 类型。

➢ 语法

MI_S32 MI_VENC_GetH265Dblk(MI_VENC_CHN VeChn, MI_VENC_ParamH265Dblk_t

*pstH265Dblk);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH265Dblk	H.265 协议编码通道的 dblk 属性。	输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于获取 H.265 协议编码通道的 dblk 配置。
- 本接口可在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

➢ 举例

请参见 MI_VENC_SetH264Dblk 的举例。

➢ 相关主题

无。

1.3.48 MI_VENC_SetH264Vui

➢ 描述

设置 H.264 协议编码通道的 vui 参数。

➢ 语法

MI_S32 MI_VENC_SetH264Vui(MI_VENC_CHN VeChn, MI_VENC_ParamH264Vui_t *pstH264Vui);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH264Vui	H.264 协议编码通道的 Vui 参数。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于设置 H.264 协议编码通道的 VUI 的配置
- 本接口可在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，等到下一个 I 帧时生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 建议用户在调用此接口之前，先调用 MI_VENC_GetH264Vui 接口，获取当前编码通道的 Vui 配置，然后再进行设置。

➢ 举例

```
MI_S32 SetVui()
{
    MI_S32 s32Ret;
    MI_VENC_ParamH264Vui_t stVui;
    MI_VENC_CHN VeChnId = 0;

    //...omit other thing
    s32Ret = MI_VENC_GetH264Vui(VeChnId, &stVui);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_GetH264Vui err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }
    stVui.stVuiTimeInfo.u8TimingInfoPresentFlag = 1;
    s32Ret = MI_VENC_SetH264Vui(VeChnId,&stVui);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_SetH264Vui err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }

    return s32Ret;
}
```

➢ 相关主题

无。

1.3.49 MI_VENC_GetH264Vui

➢ 描述

获取 H.264 协议编码通道的 Vui 配置。

➢ 语法

MI_S32 MI_VENC_GetH264Vui(MI_VENC_CHN VeChn, MI_VENC_ParamH264Vui_t *pstH264Vui);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH264Vui	H.264 协议编码通道的 Vui 属性。	输出

➢ 返回值

返回值：
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于获取 H.264 协议编码通道的 Vui 配置。
- 本接口可在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

➢ 举例

请参见 MI_VENC_SetH264Vui 的举例

➢ 相关主题

无。

1.3.50 MI_VENC_SetH265Vui

➢ 描述

设置 H.265 协议编码通道的 vui 参数。

➢ 语法

MI_S32 MI_VENC_SetH265Vui(MI_VENC_CHN VeChn, MI_VENC_ParamH265Vui_t *pstH265Vui);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH265Vui	H.265 协议编码通道的 Vui 参数。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于设置 H.265 协议编码通道的 VUI 的配置
- 本接口可在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，等到下一个 I 帧时生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 建议用户在调用此接口之前，先调用 MI_VENC_GetH265Vui 接口，获取当前编码通道的 Vui 配置，然后再进行设置。

➢ 举例

请参见 MI_VENC_SetH264Vui 的举例

➢ 相关主题

无。

1.3.51 MI_VENC_GetH265Vui

➢ 描述

获取 H.265 协议编码通道的 Vui 配置。

➢ 语法

MI_S32 MI_VENC_GetH265Vui(MI_VENC_CHN VeChn, MI_VENC_ParamH265Vui_t *pstH265Vui);

- 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstH265Vui	H.265 协议编码通道的 Vui 属性。	输出

- 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

- 依赖

  - 头文件：mi_common.h、mi_venc.h
  - 库文件：libmi.a

※ 注意

  - 本接口用于获取 H.265 协议编码通道的 Vui 配置。
  - 本接口可在编码通道创建之后，编码通道销毁之前调用。
  - 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

- 举例

请参见 MI_VENC_SetH264Vui 的举例

- 相关主题

无。

1.3.52 MI_VENC_SetH265SliceSplit

- 描述

设置 H.265 通道的 slice 分割属性。

- 语法

MI_S32 MI_VENC_SetH265SliceSplit(MI_VENC_CHN VeChn, MI_VENC_ParamH265SliceSplit_t *pstSliceSplit);

- 参数

参数名称	描述	输入/输出

VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstSliceSplit	H.265 码流 slice 分割参数。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意–

- 本–接口用于设置 H.265 协议编码通道码流的分割方式
- Slice 分割属性主要由两个参数决定：

  1）bSplitEnable：当前帧是否进行 slice 分割。

  2）u32SliceRowCount：slice 按照宏块行进行分割。u32SliceRowCount 表示每个 slice 占图像宏块行数。且当编码至图像的最后几行，不足 u32SliceRowCount 时，剩余的宏块行被划分为一个 slice。
- 本接口属于高级接口，用户可以选择性调用，建议不调用，系统默认 bSplitEnable 为 false。本接口可在编码通道创建之后，编码通道销毁之前设置。此接口在编码过程中被调用时，等到下一帧时生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 建议用户在调用此接口之前，先调用 MI_VENC_GetH265SliceSplit 接口，获取当前通道的 slicesplit 配置，然后再进行设置。

➢ 举例

请参见 MI_VENC_SetH264SliceSplit 的举例。

➢ 相关主题

无。

1.3.53 MI_VENC_GetH265SliceSplit

➢ 描述

获取 H.265 通道的 slice 分割属性。

➢ 语法

MI_S32 MI_VENC_GetH265SliceSplit(MI_VENC_CHN VeChn, MI_VENC_ParamH265SliceSplit_t *pstSliceSplit);

- 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstSliceSplit	H.265 码流 slice 分割参数。	输出

- 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

- 依赖

  - 头文件：mi_common.h、mi_venc.h
  - 库文件：libmi.a

※ 注意

  - 本接口用于获取 H.265 协议编码通道的 slice 分割属性。
  - 本接口可在编码通道创建之后，编码通道销毁之前调用。
  - 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

- 举例

请参见 MI_VENC_SetH264SliceSplit 的举例。

- 相关主题

无。

1.3.54 MI_VENC_SetJpegParam

- 描述

设置 JPEG 协议编码通道的高级参数。

- 语法

MI_S32 MI_VENC_SetJpegParam(MI_VENC_CHN VeChn,MI_VENC_ParamJpeg_t *pstJpegParam);

- 参数

参数名称	描述	输入/输出

VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstJpegParam	JPEG 协议编码通道的高级参数集合。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于设置 JPEG 协议编码通道的高级参数。
- 高级参数主要由 4 个参数组成：
  1) u32Qfactor：量化表因子范围为[1,99]，u32Qfactor 越大，量化表中的量化系数越小，得到的图像质量会更好，同时，编码压缩率更低。同理 u32Qfactor 越小，量化表中的量化系数越大，得到的图像质量会更差，同时，编码压缩率更高。具体的 u32Qfactor 与量化表的关系请见 RFC2435 标准。
  2) au8YQt[64]，au8CbCrQt[64]：这两个参数对应两个量化表空间，用户可以通过这两个参数设置用户的量化表。
  3) u32McuPerEcs：每个 Ecs 中包含多少 Mcu。系统模式 u32MCUPerECS=0，表示当前帧的所有的 MCU 被编码为一个 ECS。u32MCUPerECS 的最小值为 0，最大值不超过 (picwidth+15)>>4x(picheight+15)>>4x2。
- 如果用户想使用自己的量化表，在设置量化表的同时，请将 Qfactor 设置为 50。
- 本接口可在编码通道创建之后，编码通道销毁之前设置。本接口在编码过程中被调用时，等到下一个帧编码时生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 建议用户在调用此接口之前，先调用 MI_VENC_GetJpegParam 接口，获取当前编码通道的 JpegParam 配置，然后再进行设置。

➢ **举例**

```
MI_S32 SetJpegParam()
{
    MI_S32 s32Ret;
    MI_VENC_ParamJpeg_t stParamJpeg;
    MI_VENC_CHN VeChnId = 0;
    int i;

    //...omit other thing
    s32Ret = MI_VENC_GetJpegParam(VeChnId, &stParamJpeg);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_GetJpegParam err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }
    stParamJpeg.u32MCUPerEcs = 100;
    for (i = 0; i < 64; i++)
    {
        stParamJpeg.au8YQt[i] = 16;
        stParamJpeg.au8CbCrQt[i] = 17;
    }
    s32Ret=MI_VENC_SetJpegParam(VeChnId, &stParamJpeg);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_SetJpegParam err0x%x\n",s32Ret);
        return E_MI_ERR_FAILED;
    }

    return s32Ret;
}
```

➢ **相关主题**

无。

1.3.55 MI_VENC_GetJpegParam

➢ **描述**

获取 JPEG 协议编码通道的高级参数配置。

➢ **语法**

MI_S32 MI_VENC_GetJpegParam(MI_VENC_CHN VeChn, MI_VENC_ParamJpeg_t *pstJpegParam);

➢ **参数**

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入

pstJpegParam	Jpeg 协议编码通道的高级参数配置。	输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于获取 JPEG 协议编码通道的高级参数配置。
- 本接口可在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

➢ 举例

请参见 MI_VENC_SetJpegParam 的举例。

➢ 相关主题

无。

1.3.56 MI_VENC_SetRcParam

➢ 描述

设置编码通道码率控制器的高级参数。

➢ 语法

MI_S32 MI_VENC_SetRcParam(MI_VENC_CHN VeChn, MI_VENC_RcParam_t *pstRcParam);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstRcParam	编码通道码率控制器的高级参数。	输入

➢ 返回值

- MI_OK 成功。

返回值

非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

➢ 芯片差异

不同 Chip 和不同编码协议支持的码控算法是不一样的，如下表：

Chip	H.264	H.265	JPEG
328Q/329D/326D	FIXQP/CBR/VBR	FIXQP/CBR/VBR	FIXQP/CBR
325/325DE/327DE	FIXQP/CBR/VBR	FIXQP/CBR/VBR	FIXQP/CBR
336D/336Q/339G	FIXQP/CBR/VBR/AVBR	FIXQP/CBR/VBR/AVBR	FIXQP/CBR
335/337DE	FIXQP/CBR/VBR/AVBR	FIXQP/CBR/VBR/AVBR	FIXQP/CBR

※ 注意

- 编码通道码率控制器的高级参数都有默认值，而不是必须调用这个接口才能启动编码通道。
- 建议用户先调用 MI_VENC_GetRcParam 接口，获取 RC 高级参数，然后修改相应参数，再调用本接口对高级参数进行设置。
- RC 高级参数现仅支持 H.264/H.265/Mjpeg 码率控制模式。
- 码率控制器的高级参数由以下参数组成：u32ThrdI[RC_TEXTURE_THR_SIZE]，u32ThrdP[RC_TEXTURE_THR_SIZE]：分别衡量 I 帧，P 帧的宏块复杂度的一组阈值。这组阈值按照从小到大的顺序依次排列，每个阈值的取值范围为[0,255]。这组阈值用于在进行宏块级码率控制时，根据图像复杂度对每个宏块的 Qp 进行适当的调整。对于 H.264，宏块级码率控制只有加方向（最大加 12），即如果当前宏块的图像复杂度处于某两阈值之间时，当前宏块的 Qp 值就在宏块行起始 Qp 值的基础上加上 x，x 取值如下表（C 表示图像复杂度）：

C Range	x
C<=u32Thrd[0]	0
u32Thrd[0]<C<=u32Thrd[1]	1
u32Thrd[1]<C<=u32Thrd[2]	2
u32Thrd[2]<C<=u32Thrd[3]	3
u32Thrd[3]<C<=u32Thrd[4]	4
u32Thrd[4]<C<=u32Thrd[5]	5
u32Thrd[5]<C<=u32Thrd[6]	6
u32Thrd[6]<C<=u32Thrd[7]	7
u32Thrd[7]<C<=u32Thrd[8]	8
u32Thrd[8]<C<=u32Thrd[9]	9
u32Thrd[9]<C<=u32Thrd[10]	10
u32Thrd[10]<C<=u32Thrd[11]	11

C Range	x
u32Thrd[11]<C	12

对于 H.265，宏块级码率控制既有加方向（最大加 8），也有减方向（最大减 4），即如果当前宏块的图像复杂度小于等于 u32Thr[3]阈值时，当前宏块的 Qp 值就在宏块行起始 Qp 值的基础上减去 x；如果当前宏块的图像复杂度大于 u32Thr[3]阈值时，当前宏块的 Qp 值就在宏块行起始 Qp 值的基础上加上 y，x，y 的取值如下表（C 表示图像复杂度）：

C Range	x or y
C<u32Thrd[0]	x=4
u32Thrd[0]<=C<u32Thrd[1]	x=3
u32Thrd[1]<=C<u32Thrd[2]	x=2
u32Thrd[2]<=C<u32Thrd[3]	x=1
u32Thrd[3]<=C<=u32Thrd[4]	x=y=0
u32Thrd[4]<C<=u32Thrd[5]	y=1
u32Thrd[5]<C<=u32Thrd[6]	y=2
u32Thrd[6]<C<=u32Thrd[7]	y=3
u32Thrd[7]<C<=u32Thrd[8]	y=4
u32Thrd[8]<C<=u32Thrd[9]	y=5
u32Thrd[9]<C<=u32Thrd[10]	y=6
u32Thrd[10]<C<=u32Thrd[11]	y=7
u32Thrd[11]<C	y=8

- u32RowQpDelta：在宏块级码率控制时，每一行宏块的起始 QP 相对于帧起始 QP 的波动幅度值。对于码率波动较严格的场景下，可以尝试将此参数调大，实现更加精确的码率控制，但可能会到导致某些帧图像内部的图像质量有差异。在高码率时，该值推荐为 0；中码率时推荐该值为 0 或 1；低码率时推荐该值为 2～5。
- CBR 参数如下：
  1) u32MinIprop&u32MaxIprop：CBR 高级参数，分别表示的是最小 IP 比例和最大 IP 比例。IP 比例表示 I 帧和 P 帧的 Bits 数的比例。这两个值用于钳位 IP 比例的范围。当 u32MinIprop 被调整较大时，会导致 I 帧清晰，P 帧模糊。当 u32MaxIprop 被调整较小时，会导致 I 帧模糊，P 帧清晰。在正常情况下不建议对 IP 大小比进行约束，避免带来呼吸效应和码率波动，默认 u32MinIprop 设为 1，u32MaxIprop 设为 20。在对 I 帧大小有约束的场景时，可以根据对 I 帧大小波动的依赖来设置 u32MinIprop 和 u32MaxIprop 的值。
  2) u32MaxQp&u32MinQp&u32MaxStartQp：CBR 高级参数，u32MaxQp、u32MinQp 表示的是当前帧的最大 QP 和最小 QP。这个钳位效果最强烈，所有其他对图像 QP 的调整，如宏块级码率控制，最终都会被约束到这个最大 QP 和最小 QP。u32MaxStartQp 表示的是帧级码率控制输出的最大 QP，取值范围是[u32MinQp,u32MaxQp]。宏块级码率控制可能会使某些宏块的 QP 大于或小于 u32MaxStartQp，但都会被钳位到[u32MinQp,u32MaxQp]。默认值 u32MinQp 为 10，u32MaxQp 为 51，u32MaxStartQp 为 51。在对质量无特殊需求下，建议不更改此组参数。

3) u32MaxPPDeltaQp&u32MaxIPDeltaQp：CBR 高级参数，表示的是连续两个 P 帧起始 Qp 之间的最大差值和连续 IP 帧起始 QP 之间的最大差值。当出现大运动，场景切换等变化时，保证码率平稳条件下，可能会导致连续 P 帧之间 QP 差异过大，导致图像马赛克等，可以通过调整这两个参数对 QP 变化进行钳位，避免出现图像质量波动。u32MaxPPDelta 默认值为 3；u32MaxIPdeltaQP 默认值为 5，可以根据场景对质量与码率要求的差异修改。
4) s32IPQPDelta：CBR 高级参数，表示的是平均 QP 值与当前 I 帧 QP 的差值，此参数可为负值。可用于调整 I 帧过大和呼吸效应。系统默认值为 2，增大此值，I 帧变清晰。u32RQRatio[8]：CBR 高级参数，表示的是码率稳定和质量稳定在进行码率控制中的权重。u32RQRatio[i]/100 表示质量稳定权重，1-u32RQRatio[i]/100 表示码率稳定权重，譬如：u32RQRatio[i]=75，表示质量稳定占 75%的权重，码率稳定占 25%的权重。CBR 提供了 8 种场景模式，分别是：Normal(正常场景),Move(运动场景),Still(静止场景),StillToMove(静止转为运动场景),MoveToStill(运动转为静止场景),SceneSwitch(场景切换),SharpMove(剧烈运动场景),Init(程序初始化)，分别对应了 u32RQRatio 中 0～7 的索引号。现在系统默认值为 u32RQRatio[]={75,75,75,50,50,20,30,0}。用户可以设置不同场景下，码率控制的权重，以满足特定的依赖，譬如：在大运动场景下，质量稳定的比例为 30%，码率稳定的比例为 70%，用户可以将质量比例继续调小，码率的波动变的更小，保留，暂不使用。

- VBR 参数如下：
  1) s32DeltaQP：VBR 高级参数，表示的是在 VBR 质量出现波动的过程中，帧与帧之间的最大的 QP 变化。这个参数可用于防止出现马赛克。系统默认为 2。
  2) s32ChangePos：VBR 高级参数，表示的是 VBR 开始调整 QP 时码率与最大码率的比值。系统默认为 90，如果在内容变化剧烈且要求不超出最大码率的场景，建议减小此值，同时增大 s32DeltaQP，但码率控制稳定时的码率偏小和质量偏差。
- AVBR 参数如下：
  1) s32ChangePos:AVBR 高级参数，表示的是 AVBR 开始调整 Qp 时码率与最大码率的比值。系统默认为 80，如果在内容变化剧烈且要求不超出最大码率的场景，建议减小此值，同时增大 s32DeltaQP，但码率控制稳定时的码率偏小和质量偏差。
  2) u32MaxQp&u32MinQp :AVBR 高级参数 ,u32MaxQp、u32MinQp 表示的是当前帧的最大 Qp 和最小 Qp。这个钳位效果最强烈，所有其他对图像 Qp 的调整，如宏块级码率控制，最终都会被约束到这个最大 Qp 和最小 Qp。默认值 u32MinQp 为 12，u32MaxQp 为 48 在对质量无特殊需求下，建议不更改此组参数。
  3) u32MaxIQp&u32MinIQp :AVBR 高级参数 ,u32MaxIQp、u32MinIQp 表示的是当前序列 IDR 帧的最大 Qp 和最小 Qp。这个钳位效果最强烈，所有其他对图像 Qp 的调整，如宏块级码率控制，最终都会被约束到这个最大 Qp 和最小 Qp。在对质量无特殊需求下，建议不更改此组参数。
- pRcParam：由用户制定的 RC 高级参数，保留，暂时没有使用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

```
MI_S32 VencSetRcParam()
{
    MI_S32 s32Ret;
    MI_VENC_RcParam_t stVencRcPara;
    MI_VENC_CHN VeChnId = 0;

    //...omit other thing
    s32Ret = MI_VENC_GetRcParam(VeChnId, &stVencRcPara);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_GetRcParam err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }

    stVencRcPara.stParamH264Cbr.s32IPQPDelta = 2;
    s32Ret = MI_VENC_SetRcParam(VeChnId, &stVencRcPara);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_SetRcParam err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }
    //...omit other thing

    return s32Ret;
}
```

- **举例**

- **相关主题**

  无。

1.3.57 MI_VENC_GetRcParam

- **描述**

  获取通道码率控制高级参数。

- **语法**

  MI_S32 MI_VENC_GetRcParam(MI_VENC_CHN VeChn, MI_VENC_RcParam_t *pstRcParam);

- **参数**

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstRcParam	通道码率控制参数指针。	输出

➢ 返回值

返回值	MI_OK 成功。
	非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于获取 H.264 编码通道码率控制的高级参数。各参数的含义请具体请参见 MI_VENC_RcParam_t。
- 如果 pstRcParam 为空，则返回失败。

➢ 举例

无。

➢ 相关主题

无。

1.3.58 MI_VENC_SetRefParam

➢ 描述

设置 H.264/H.265 编码通道高级跳帧参考参数。

➢ 语法

MI_S32 MI_VENC_SetRefParam(MI_VENC_CHN VeChn, MI_VENC_ParamRef_t *pstRefParam);

- 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstRefParam	H.264/H.265 编码通道高级跳帧参考参数。	输入

- 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

- 依赖

  - 头文件：mi_common.h、mi_venc.h
  - 库文件：libmi.a

※ 注意

  - 如果 pstRefParam 为空，则返回失败。
  - 创建 H.264/H.265 协议编码通道时，默认跳帧参考模式是 1 倍跳帧参考模式。如果用户需要修改编码通道的跳帧参考，建议在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
  - 本接口在编码过程中被调用时，等到下一个 I 帧时生效。
  - 如果用户设置 1 倍跳帧参考模式，对应的配置为：bEnablePred = TRUE，u32Enhance=0，u32Base=1；
    如果设置 2 倍跳帧参考模式，对应的配置为：bEnablePred = TRUE，u32Enhance=1，u32Base=1；
    如果设置 4 倍跳帧参考模式，对应的配置为：bEnablePred = TRUE，u32Enhance=1，u32Base=2。

- 举例

```
MI_S32 VencSetRefParam()
{
    MI_S32 s32Ret;
    MI_VENC_ParamRef_t stRefParam;
    MI_VENC_CHN VeChnId = 0;

    //...omit other thing

    stRefParam.u32Base = 1;
    stRefParam.u32Enhance = 3;
    stRefParam.bEnablePred = TRUE;
    s32Ret = MI_VENC_SetRefParam(VeChn, &stRefParam);
    if (MI_SUCCESS != s32Ret)
    {
        printf("MI_VENC_SetRefParam err0x%x\n", s32Ret);
        return E_MI_ERR_FAILED;
    }
    //...omit other thing

    return s32Ret;
}
```

➢ 相关主题

无。

1.3.59 MI_VENC_GetRefParam

➢ 描述

获取 H.264/H.265 编码通道高级跳帧参考参数。

➢ 语法

MI_S32 MI_VENC_GetRefParam(MI_VENC_CHN VeChn, MI_VENC_ParamRef_t *pstRefParam);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstRefParam	H.264/H.265 编码通道高级跳帧参考参数。	输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 如果 pstRefParam 为空，则返回失败。

➢ 举例

无。

➢ 相关主题

无。

1.3.60 MI_VENC_SetCrop

➢ 描述

设置通道的裁剪属性。

➢ 语法

MI_S32 MI_VENC_SetCrop(VENC_CHNVeChn, MI_VENC_CropCfg_t *pstCropCfg);

➢ 参数

参数名称	描述	输入/输出
VeChn	通道号	输入
pstCropCfg	通道裁剪属性。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于设置通道的裁剪属性。
- 本接口必须在通道创建之后，通道销毁前调用。
- 裁剪属性由两部分组成：
  1) bEnable：是否使能通道裁剪功能。
  2) stRect：裁剪区域属性，包括裁剪区域起始点坐标，以及裁剪区域的尺寸。

➢ 举例

无。

➢ 相关主题

无。

1.3.61 MI_VENC_GetCrop

➢ 描述

获取通道的裁剪属性。

➢ 语法

MI_S32 MI_VENC_GetCrop(MI_VENC_CHN VeChn, MI_VENC_CropCfg_t *pstCropCfg);

➢ 参数

参数名称	描述	输入/输出
VeChn	通道号	输入
pstCropCfg	通道裁剪属性	输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于获取通道的裁剪属性。
- 本接口必须在通道创建之后，通道销毁之前调用。

➢ 举例

无。

➢ 相关主题

无。

1.3.62 MI_VENC_SetFrameLostStrategy

➢ 描述

设置编码通道瞬时码率超过阈值时丢帧策略。

➢ 语法

MI_S32 MI_VENC_SetFrameLostStrategy(MI_VENC_CHN VeChn, MI_VENC_ParamFrameLost_t *pstFrmLostParam);

➢ 参数

参数名称	描述	输入/输出

VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstFrmLostParam	编码通道丢帧策略的参数。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意–

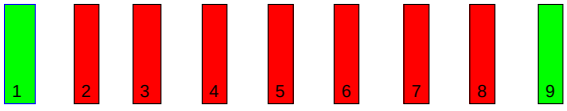
- 如果 pstFrmLostParam 为空，则返回失败。
- pstFrmLostParam 主要由四个参数决定:
  1) enFrmLostMode：丢帧策略模式。
  2) u32EncFrmGaps：丢帧间隔。
  3) bFrmLostOpen：丢帧开关。
  4) u32FrmLostBpsThr：丢帧阈值。
- 本接口属于高级接口，用户可以选择性调用，系统有默认值，默认在瞬时码率超出阈值时为丢帧。
- 本接口提供瞬时码率超过阈值时两种处理方式：丢帧和编码 pskip 帧。u32EncFrmGaps 控制是否均匀丢帧或均匀编码 pskip 帧。如下图所示：


->编码该帧时瞬时码率未超出阈值,且保留下来的帧

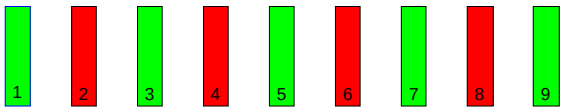

->编码该帧时瞬时码率超出阈值，且丢弃的帧


->编码该帧时瞬时码率超出阈值，且编码为 pskip 帧

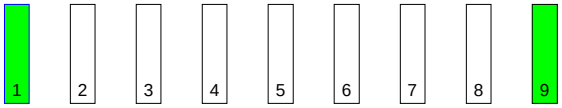
enFrmLostMode=E_MI_VENC_FRMLOST_NORMAL，u32EncFrmGaps=0



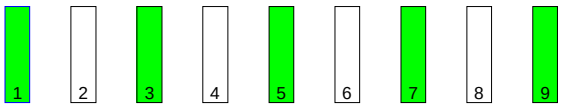
enFrmLostMode=E_MI_VENC_FRMLOST_NORMAL，u32EncFrmGaps=1



enFrmLostMode=E_MI_VENC_FRMLOST_PSKIP，u32EncFrmGaps=0



enFrmLostMode=E_MI_VENC_FRMLOST_PSKIP，u32EncFrmGaps=1



- 本接口可在编码通道创建之后，编码通道销毁之前设置。此接口在编码过程中被调用时，等到下一帧时生效。

➢ 举例

无。

➢ 相关主题

无。

1.3.63 MI_VENC_GetFrameLostStrategy

➢ 描述

获取编码通道瞬时码率超过阈值时丢帧策略。

➢ 语法

MI_S32 MI_VENC_GetFrameLostStrategy(MI_VENC_CHN VeChn, MI_VENC_ParamFrameLost_t *pstFrmLostParam);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstFrmLostParam	编码通道丢帧策略的参数。	输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 如果 pstFrmLostParam 为空，则返回失败。

➢ 举例

无。

➢ 相关主题

无。

1.3.64 MI_VENC_SetSuperFrameCfg

➢ 描述

设置编码超大帧配置。

➢ 语法

MI_S32 MI_VENC_SetSuperFrameCfg(MI_VENC_CHN VeChn, MI_VENC_SuperFrameCfg_t

*pstSuperFrmParam);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstSuperFrmParam	编码超大帧配置参数。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_comm_rc.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 如果通道未创建时，则返回失败。
- 本接口属于高级接口，用户可以选择性调用，系统默认值。系统默认值为 eSuperFrmMode=E_MI_VENC_SUPERFRM_NONE。
- 本接口可在编码通道创建之后，编码通道销毁之前设置。
- 超大帧模式选择 E_MI_VENC_SUPERFRM_NONE 时，MI 不做任何处理。
- 超大帧模式选择 E_MI_VENC_SUPERFRM_DISCARD 时，MI 会将超大帧丢弃，并继续编下一帧；如果是 SSC325,SSC327,SSC335,SSC337 等版本，因为内存不足的关系，丢弃超大帧后会强制申请 I 帧。
- 超大帧模式选择 E_MI_VENC_SUPERFRM_REENCODE 时，MI 会将这一帧 QP 提高 4 并重新编码，最多重复 4 次，如果还是超出阈值，就把这一帧丢弃。
- 超大帧处理主要是防止因为某一帧过大造成的画面卡顿。阈值单位是 bit，建议根据实际场景设定。

➢ 举例

无。

➢ 相关主题

无。

1.3.65 MI_VENC_GetSuperFrameCfg

➢ 描述

获取编码超大帧配置。

- 语法

MI_S32 MI_VENC_GetSuperFrameCfg(MI_VENC_CHN VeChn, MI_VENC_SuperFrameCfg_t *pstSuperFrmParam);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstSuperFrmParam	编码超大帧配置参数。	输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 如果 pstSuperFrmParam 为空，则返回失败。

➢ 举例

无。

➢ 相关主题

无。

1.3.66 MI_VENC_SetRcPriority

➢ 描述

设置码率控制的优先级类型。

➢ 语法

MI_S32 MI_VENC_SetRcPriority(MI_VENC_CHN VeChn, MI_VENC_RcPriority_e eRcPriority);

➢ 参数

参数名称	描述	输入/输出
VeChn	通道号	输入
enRcPriority	优先级类型枚举。	输入

➢ 返回值

- MI_OK 成功。

返回值

非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于设置码率控制是以目标码率还是以超大帧阈值为高优先级。
- 当码率控制以目标码率为高优先级时，在码率不足时可能编码出超大帧以补偿码率，这时超大帧不会重编。当码率控制以超大帧阈值为高优先级时，超大帧则会重编以降低比特数，结果可能导致码率不足。
- 本接口必须在编码通道创建之后，编码通道销毁之前调用。
- 本接口只支持 H.264 和 H.265 两种协议的码率控制模式。

➢ 举例

无。

➢ 相关主题

无。

1.3.67 MI_VENC_GetRcPriority

➢ 描述

获取码率控制的优先级类型。

➢ 语法

MI_S32 MI_VENC_GetRcPriority(MI_VENC_CHN VeChn, MI_VENC_RcPriority_e *peRcPriority);

➢ 参数

参数名称	描述	输入/输出
VeChn	通道号	输入
penRcPriority	优先级类型指针。	输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h

- 库文件：libmi.a

※ 注意

- 本接口必须在编码通道创建之后，编码通道销毁之前调用。

➢ 举例

无。

➢ 相关主题

无。

1.3.68 MI_VENC_AllocCustomMap

➢ 描述

分配智能编码所用的 Custom Map 的内存，Custom Map 包含 QP Map 和 Mode Map。

➢ 语法

MI_S32 MI_VENC_ AllocCustomMap (MI_VENC_CHN VeChn, MI_PHY *pstPhyAddr, MI_VOID **ppCpuAddr);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstPhyAddr	分配的物理地址的指针。	输入/输出
ppCpuAddr	分配的 CPU 虚拟地址的指针。	输入/输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

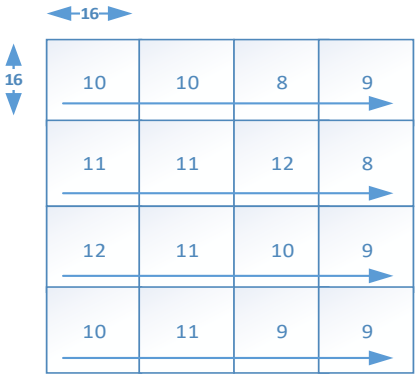
※ 注意

- 本接口用于分配 Custom Map 所需的内存，并返回分配好的内存的地址，包括物理地址和 CPU 地址，同时返回图像帧的大小。
- 如果地址分配失败，pstPhyAddr 或 ppCpuAddr 为空，则返回失败。
- 本接口可在编码通道创建之后，编码通道销毁之前设置，分配的内存会在通道销毁时候同步释放。
- Custom Map 包含 QP Map 和 Mode Map 两种，可以同时使用两种 Map,不同的 Map 使用不同的位来区分。

- QPMap 中 QP 的设置在 H.264 中以 16*16 块为单位，每一个 16*16 的 QP 值，采用用户设定的相应块的 QP 值。所有这些块的 QP 值组成 QP 表，该表中 QP 值的组织方式如下图示。QP 值依次填入为 QPMap 所分配的内存地址中。QP 值根据设定（参考 1.2.73 MI_VENC_SetAdvCustRcAttr）使用相对或绝对值填入，如果是绝对 QP 值，QP 的取值范围为[12,48]。如果是相对 QP 值，QP 的取值范围为[0,63]（实际生效的相对 QP 值为配置值减 32，即实际生效的范围为[-32，31]）。
  ModeMap 中 Mode 的设置在 H.264 中同样以 16*16 块为单位，每一个 16*16 的 Mode 值(0:not use,1:skip mode,2:intra mode)，采用用户设定的相应块的 Mode 值。所有这些块的 Mode 值组成 Mode 表，该表中 Mode 值的组织方式和 QP 值的组织方式一样，Mode 值依次填入为 ModeMap 所分配的内存地址中。
  每个 16*16 块可以配置 1 byte 的 Mode/QP 值，Mode 值占据所填写值的高 2 位，QP 值占据低 6 位。
  配置填写示例如下：

```
struct {
        uint8   qp      :   6;
        uint8   mode    :   2;

} H264CustomMapConf;
```

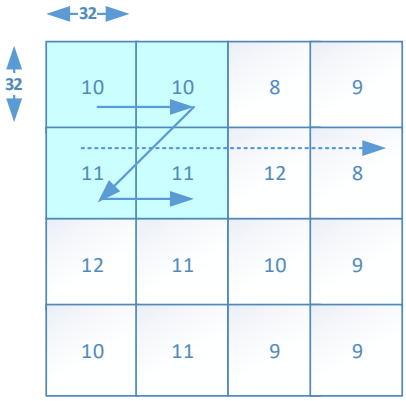


AVC

- QPMap 中 QP 的设置在 H.265 中以 4 个 32*32 块为单位，每一个 32*32 块的 QP 值，采用用户设定的相应块的 QP 值。所有这些块的 QP 值组成 QP 表，该表中 QP 值的组织方式如下图示， QP 值按照上下各两个的次序依次填入为 QPMap 所分配的内存地址中。QP 值同样根据设定使用相对或绝对值填入。
  ModeMap 中 Mode 的设置在 H.265 中同样以 4 个 32*32 块为单位，每 4 个 32*32 块的 Mode 值(0:not use,1:skip mode,2:intra mode)，采用用户设定的相应块的 Mode 值。所有这些块的 Mode 值组成 Mode 表，该表中 Mode 值的组织方式和 QP 值的组织方式一样，Mode 值依次填入为 ModeMap 所分配的内存地址中。
  以上 4 个 32*32 块中，可以配置 4bytes 的 QP/Mode 值。其中 Mode 值占 2 位，每个 32*32 的块对应的 QP 值占 6 位（共 24 位）。
  配置填写示例如下：

```
struct {
 uint32   sub_ctu_qp_0        :   6;
```

```
    uint32  mode                    :  2;
    uint32  sub_ctu_qp_1        :  6;
    uint32  reserved_0              :  2;
    uint32  sub_ctu_qp_2        :  6;
    uint32  reserved_1        :  2;
    uint32  sub_ctu_qp_3        :  6;
    uint32  reserved_2        :  2;
} H265CustomMapConf;
```



HEVC

- 举例

  无。

- 相关主题

  无。

1.3.69 MI_VENC_ApplyCustomMap

- 描述

  应用已经配置的 Custom Map。

- 语法

  MI_S32 MI_VENC_ApplyCustomMap (MI_VENC_CHN VeChn, MI_PHY PhyAddr);

- 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
PhyAddr	分配的虚拟物理地址。	输入

- 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于在配置自定义的 Custom Map 到内存后，应用相关的配置。
- 本接口可在初始化或修改 Custom Map 后使用。

➢ 举例

无。

➢ 相关主题

无。

1.3.70 MI_VENC_GetLastHistoStaticInfo

➢ 描述

获取最新的图像帧编码后的相关输出信息。

➢ 语法

MI_S32 MI_VENC_GetLastHistoStaticInfo (MI_VENC_CHN VeChn, MI_VENC_FrameHistoStaticInfo_t **ppFrmHistoStaticInfo);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
ppFrmHistoStaticInfo	保存的最新已编码帧相关输出的地址的指针。	输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于获取保存的最近已经编码的图像帧输出相关信息的地址，其中输出信息包括编码后的帧类型，块的数量，帧大小等等。
- 如果获取相关输出内容失败，则返回失败。

- 本接口可在编码通道创建之后，编码通道销毁之前设置，获取的相关输出内容会在通道销毁时候或者调用下面的释放接口时候释放。

➢ 举例

无。

➢ 相关主题

无。

1.3.71 MI_VENC_ReleaseHistoStaticInfo

➢ 描述

释放保存的最新的图像帧编码后的相关输出信息占用的内存。

➢ 语法

MI_S32 MI_VENC_ReleaseHistoStaticInfo (MI_VENC_CHN VeChn);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于释放之前保存的最近的已经编码的图像帧的输出相关信息占用的内存。

➢ 举例

无。

➢ 相关主题

无。

1.3.72 MI_VENC_SetAdvCustRcAttr

➢ 功能

设置自定义的高级码控相关属性配置。

➢ 语法

MI_S32 MI_VENC_ SetAdvCustRcAttr (MI_VENC_CHN Vehn, MI_VENC_AdvCustRcAttr_t sAdvCustRcAttr);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
sAdvCustRcAttr	自定义的高级码控相关功能的开关参数配置。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 本接口用于设置自定义的高级码控相关属性配置。
- 本接口在编码通道创建之后，图像开始接收之前设置。
- 设置了自定义的高级码控功能的开关后，才能使用 MI_VENC_AllocCustomMap 和 MI_VENC_GetLastHistoStaticInfo 来启动相关功能。

➢ 举例

无。

➢ 相关主题

无。

1.3.73 MI_VENC_SetInputSourceConfig

➢ 描述

设置 H.264/H.265 的输入配置信息。

➢ 语法

MI_S32 MI_VENC_SetInputSourceConfig(MI_VENC_CHN VeChn, MI_VENC_InputSourceConfig_t *pstInputSourceConfig);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstInputSourceConfig	输入配置信息	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

➢ 芯片差异

不同 Chip 在使用 Input ring mode 的时候，需不需要调用此接口的限制是不一样的，如下表：

Chip	使用 ring mode 时需要调用此接口？
328Q/329D/326D	N
325/325DE/327DE	N
336D/336Q/339G	N
335/337DE	Y

※ 注意

- 如果通道未创建时，则返回失败。
- 本接口在编码通道创建之后，图像开始接收之前设置。
- 允许对最多一个通道设置成 input ring mode。
- 如果设置了 E_MI_VENC_INPUT_MODE_RING_ONE_FRM/E_MI_VENC_INPUT_MODE_RING_HALF_FRM，那 APP 在调用 MI_SYS_BindChnPort2 需要设置 E_MI_SYS_BIND_TYPE_HW_RING 和相应 ring buffer 高度。比如分辨率为 1920x1080，如果设置为 E_MI_VENC_INPUT_MODE_RING_ONE_FRM，则

ring buffer 高度需设置 1080;如果为 E_MI_VENC_INPUT_MODE_RING_HALF_FRM，则 ring buffer 高度需设置 540。

➢ 举例

无。

➢ 相关主题

无。

1.3.74 MI_VENC_SetSmartDetInfo

➢ 描述

用于第三方智能侦测算法向 VENC 提供智能编码所需的统计信息。

➢ 语法

MI_S32 MI_VENC_ SetSmartDetInfo(MI_VENC_CHN VeChn, MI_VENC_SmartDetInfo_t *pstSmartDetInfo);

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstSmartDetInfo	智能侦测算法相关的统计信息。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 如果通道未创建时，则返回失败。
- 本接口在编码通道创建之后，且图像开始接收之后设置。
- 本接口参数设置只在 SSC33X 生效。

➢ 举例

无。

- **相关主题**

  无。

1.3.75 MI_VENC_SetIntraRefresh

- **描述**

  设置 H.264/H.265 的 P 帧刷新 Islice，开启 IntraRefresh 模式后，只会在编码开始位置有一张 IDR，后续只有 P 帧，开启 IntraRefresh 模式的好处是不改变 IDR 帧质量，把原本 IDR 编码的 Intra LCU/宏块分散在若干个 P 帧中，使每帧的大小相对平均。

  优点：
  1：码率非常平稳，对网络冲击小，适合无线传输环境。
  2：编码，解码及网络延迟非常小
  3：不会降低 I 帧质量，不会引起严重呼吸效应
  缺点：
  4：只支持 normalP 的 GOP 结构，其他 GOP 结构不支持

  5：该技术主要针对码率平稳要求高的场景，并不能够降低太多码率，对低码率场景优化不大，仅支持宏块/LCU 按行刷新

- **语法**

  MI_S32 MI_VENC_SetIntraRefresh(MI_VENC_CHN VeChn, MI_VENC_IntraRefresh_t *pstIntraAttr)

- **参数**

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstIntraAttr	输入 Intra 配置信息	输入

- **返回值**

  返回值：
  - MI_OK 成功。
  - 非 MI_OK 失败，参照错误码。

- **依赖**

  - 头文件：mi_common.h、mi_venc.h
  - 库文件：libmi.a

- 芯片差异

  目前 h264 和 h265 support intra refresh 的 chip 如下

Chip	是否支持 intra refresh
336D/336Q/339G	Y
335/337DE	Y

※ 注意

- 如果通道未创建时，则返回失败。
- 本接口在编码通道创建之后，图像开始接收之前设置。

➢ 举例

参考 mi_demo 里面 st_main_venc.c
注意 u32RefreshLineNum 是取值范围是：
IntraRefreshLine * Gop >MbH 例如：
1080P gop 30 mb_height:16 那取值范围是 1088 / 16 = 68 个MB行 68/gop = 2.26
取值要大2<=68为了有降低码率， u32RefreshLineNum 取值建议不要取值太大，
不然后回导致整个gop的P frame都刷到Islice.

➢ 参考代码

```
MI_VENC_IntraRefresh_t stIntraAttr;

stIntraAttr.bEnable = conf->gdr.enable;

stIntraAttr.u32RefreshLineNum = conf->gdr.refreshmaxNum;

stIntraAttr.u32ReqIQp = conf->gdr.requst_IQP;

s32Ret = MI_VENC_SetIntraRefresh(conf->venc_chn, &stIntraAttr);
```

➢ 相关主题

无。

1.3.76 MI_VENC_GetIntraRefresh

➢ 描述

获取 H.264/H.265 的 Intra Refresh 参数

➢ 语法

MI_S32 MI_VENC_GetIntraRefresh(MI_VENC_CHN VeChn, MI_VENC_IntraRefresh_t *pstIntraAttr)

➢ 参数

参数名称	描述	输入/输出
VeChn	编码通道号。 取值范围：[0,VENC_MAX_CHN_NUM)。	输入
pstIntraAttr	输出 Intra 配置信息	输出

➢ 返回值

- MI_OK 成功。

返回值

非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

➢ 芯片差异

目前 h264 和 h265 encoder support intra refresh 的 chip 如下

Chip	是否支持 intra refresh
336D/336Q/339G	Y
335/337DE	Y

※ 注意

- 如果通道未创建时，则返回失败。
- 本接口在编码通道创建之后，图像开始接收之前设置。

➢ 举例

无。

➢ 参考代码

无。

➢ 相关主题

无。

1.3.77 MI_VENC_InitDev

➢ 描述

初始化 venc 设备。

➢ 语法

MI_S32 MI_VENC_InitDev(MI_VENC_InitParam_t *pstInitParam);

➢ 参数

参数名称	描述	输入/输出
pstInitParam	设备初始化参数。	输入

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 此函数必须在 MI_VENC_CreateChn 前调用，否则返回失败。
- 如果本接口在 MI_VENC_CreateChn 之前没有调用，内部会使用默认的参数初始化设备。
- 本接口必须和 MI_VENC_DeInitDev 成对使用，不可单独重复调用，否则返回失败。
- 本接口参数设置只在 SSC33X 生效。

➢ 举例

无。

➢ 相关主题

无。

1.3.78 MI_VENC_DeInitDev

➢ 描述

反初始化 venc 设备。

➢ 语法

MI_S32 MI_VENC_DeInitDev(void);

➢ 参数

参数名称	描述	输入/输出

➢ 返回值

返回值
- MI_OK 成功。
- 非 MI_OK 失败，参照错误码。

➢ 依赖

- 头文件：mi_common.h、mi_venc.h
- 库文件：libmi.a

※ 注意

- 此函数必须在初始化设备后调用，否则返回失败。
- 如果本接口在 app 退出前没有调用，内部会自动反初始化设备。
- 本接口必须和 MI_VENC_InitDev 成对使用，不可单独重复调用，否则返回失败。
- 本接口参数设置只在 SSC33X 生效。

➢ 举例

无。

➢ 相关主题

无。

2. VENC 数据类型

相关数据类型、数据结构定义如下：

VENC_MAX_CHN_NUM	定义最大通道数
RC_TEXTURE_THR_SIZE	定义纹理级码控的阈值个数
MI_VENC_H264eNaluType_e	定义 H.264 码流 NALU 类型
MI_VENC_H264eRefSliceType_e	定义获取的 H.264 码流属于何种跳帧参考模式下的参考帧
MI_VENC_H264eRefType_e	定义 H.264 跳帧参考码流的帧类型以及参考属性
MI_VENC_JpegePackType_e	定义 JPEG 码流的 PACK 类型
MI_VENC_H265eNaluType_e	定义 H.265 码流 NALU 类型
MI_VENC_DataType_t	定义码流结果联合体
MI_VENC_PackInfo_t	定义当前码流包数据中包含的其他类型码流包数据的结构体
MI_VENC_Pack_t	定义帧码流包结构体
MI_VENC_StreamInfoH264_t	定义 H.264 协议码流特征信息
MI_VENC_StreamInfoJpeg_t	定义 JPEG 协议码流特征信息
MI_VENC_StreamInfoH265_t	定义 H.265 协议码流特征信息
MI_VENC_Stream_t	定义帧码流类型结构体
MI_VENC_StreamBufInfo_t	定义码流 buffer 信息的结构体
MI_VENC_AttrH264_t	定义 H.264 编码器属性结构体
MI_VENC_AttrJpeg_t	定义 JPEG 抓拍编码器属性结构体
MI_VENC_AttrH265_t	定义 H.265 编码器属性结构体
MI_VENC_Attr_t	定义编码器属性结构体
MI_VENC_ChnAttr_t	定义编码通道属性结构体
MI_VENC_ChnStat_t	定义编码通道的状态结构体
MI_VENC_ParamH264SliceSplit_t	定义 H.264 编码通道 slice 分割属性
MI_VENC_ParamH264InterPred_t	定义 H.264 编码通道帧间预测属性
MI_VENC_ParamH264Trans_t	定义 H.264 编码通道变换、量化属性
MI_VENC_ParamH264Entropy_t	定义 H.264 编码通道熵编码属性
MI_VENC_ParamH265InterPred_t	定义 H.264 编码通道帧间预测属性
MI_VENC_ParamH265IntraPred_t	定义 H.264 编码通道帧内预测属性
MI_VENC_ParamH265Trans_t	定义 H.264 编码通道变换、量化属性
MI_VENC_ParamH264Dblk_t	定义 H.264 编码通道 Deblocking 属性
MI_VENC_ParamH264Vui_t	定义 H.264 编码通道 VUI 属性
MI_VENC_ParamH264VuiAspectRatio_t	定义 H.264 协议编码通道 Vui 中 AspectRatio 信息的结构体
MI_VENC_ParamH264VuiTimeInfo_t	定义 H.264 协议编码通道 Vui 中 TIME_INFO 信息的结构体
MI_VENC_ParamH264VuiVideoSignal_t	定义 H.264 协议编码通道 Vui 中 VIDEO_SIGNAL 信息的结构体

MI_VENC_ParamJpeg_t	定义 JPEG 编码参数集合
MI_VENC_RoiCfg_t	定义编码通道感兴趣区域编码属性
MI_VENC_RoiBgFrameRate_t	定义非 Roi 区域的帧率属性
MI_VENC_RcAttr_t	定义编码通道码率控制器属性
MI_VENC_RcMode_e	定义编码通道码率控制器模式
MI_VENC_AttrH264Cbr_t	定义 H.264 编码通道 CBR 属性结构
MI_VENC_AttrH264Vbr_t	定义 H.264 编码通道 VBR 属性结构
MI_VENC_AttrH264FixQp_t	定义 H.264 编码通道 Fixqp 属性结构
MI_VENC_AttrH264Abr_t	定义 H.264 编码通道 ABR 属性结构
MI_VENC_SuperFrmMode_e	定义码率控制中超大帧处理模式
MI_VENC_AttrH265Cbr_t	定义 H.265 编码通道 CBR 属性结构
MI_VENC_AttrH265Vbr_t	定义 H.265 编码通道 VBR 属性结构
MI_VENC_AttrH265FixQp_t	定义 H.265 编码通道 Fixqp 属性结构
MI_VENC_ParamH264Vbr_t	定义 H264 协议编码通道 VBR 码率控制模式高级参数配置
MI_VENC_ParamH264Cbr_t	定义 H264 协议编码通道 CBR 新版码率控制模式高级参数配置
MI_VENC_ParamH265Vbr_t	定义 H265 协议编码通道 VBR 码率控制模式高级参数配置
MI_VENC_ParamH265Cbr_t	定义 H265 协议编码通道 CBR 新版码率控制模式高级参数配置
MI_VENC_RcParam_t	定义编码通道的码率控制高级参数
MI_VENC_CropCfg_t	定义通道截取（Clip）参数
MI_VENC_RecvPicParam_t	接收指定帧数图像编码
MI_VENC_H264eIdrPicIdMode_e	设置 IDR 帧或 I 帧的 idr_pic_id 的模式
MI_VENC_H264IdrPicIdCfg_t	IDR 帧或 I 帧的 idr_pic_id 参数
MI_VENC_FrameLostMode_e	定义编码通道瞬时码率超过阈值时的丢帧模式
MI_VENC_ParamFrameLost_t	定义编码通道瞬时码率超过阈值时的丢帧策略
MI_VENC_SuperFrameCfg_t	超大帧处理策略参数
MI_VENC_RcPriority_e	码率控制优先级枚举
MI_VENC_ModParam_t	定义编码模块参数
MI_VENC_ModType_e	定义模块参数类型
MI_VENC_ParamModVenc_t	定义 mi_venc.ko 模块参数
MI_VENC_ParamModH264e_t	定义 mi_h264e.ko 模块参数
MI_VENC_ParamModH265e_t	定义 mi_h265.ko 模块参数
MI_VENC_ParamModJpege_t	定义 mi_jpege.ko 模块参数
MI_VENC_AdvCustRcAttr_t	自定义的高级码控相关功能的开关参数
MI_VENC_FrameHistoStaticInfo_t	定义图像帧编码的相关配置属性信息
MI_VENC_InputSourceConfig_t	定义 H.264/H.265 编码通道输入配置参数结构体
MI_VENC_InputSrcBufferMode_e	定义 H.264/H.265 输入 buffer 模式
MI_VENC_AttrH264Avbr_t	定义 H.264 编码通道 AVBR 属性结构
MI_VENC_AttrH265Avbr_t	定义 H.265 编码通道 AVBR 属性结构
MI_VENC_ParamH264Avbr_t	定义 H264 协议编码通道 AVBR 码率控制模式高级参数配置
MI_VENC_ParamH265Avbr_t	定义 H265 协议编码通道 AVBR 码率控制模式高级参数配置

VENC_MAX_SAD_RANGE_NUM	定义智能检测相关之统计 SAD 值落入区间范围的最大区间个数
MI_VENC_SmartDetType_e	定义智能侦测类型
MI_VENC_MdInfo_t	定义运动侦测相关统计信息
MI_VENC_SmartDetInfo_t	定义智能侦测算法所需的统计信息
MI_VENC_IntraRefresh_t	支持 P 帧刷 Islice 的控制参数
MI_VENC_InitParam_t	Venc 设备初始化参数

2.1. VENC_MAX_CHN_NUM

➢ 说明

定义最大通道个数。

➢ 定义

```
#define   VENC_MAX_CHN_NUM      16
```

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.2. RC_TEXTURE_THR_SIZE

➢ 说明

定义 RC 宏块复杂度的阈值的个数。

➢ 定义

```
#define   RC_TEXTURE_THR_SIZE   1
```

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.3. MI_VENC_H264eNaluType_e

➢ 说明

定义 H.264 码流 NALU 类型。

➢ 定义

```
typedef enum
{
    E_MI_VENC_H264E_NALU_PSLICE    =  1,
    E_MI_VENC_H264E_NALU_ISLICE    =  5,
    E_MI_VENC_H264E_NALU_SEI       =  6,
    E_MI_VENC_H264E_NALU_SPS  =  7,
    E_MI_VENC_H264E_NALU_PPS  =  8,
    E_MI_VENC_H264E_NALU_IPSLICE   =  9,
```

```
    E_MI_VENC_H264E_NALU_PREFIX = 14,
    E_MI_VENC_H264E_NALU_MAX
}MI_VENC_H264eNaluType_e;
```

➢ 成员

成员名称	描述
E_MI_VENC_H264E_NALU_PSLICE	PSLICE 类型。
E_MI_VENC_H264E_NALU_ISLICE	ISLICE 类型。
E_MI_VENC_H264E_NALU_SEI	SEI 类型。
E_MI_VENC_H264E_NALU_SPS	SPS 类型。
E_MI_VENC_H264E_NALU_PPS	PPS 类型。
E_MI_VENC_H264E_NALU_PREFIX	PREFIX 类型
E_MI_VENC_H264E_NALU_IPSLICE	P 帧刷 ISLICE 类型（暂不支持）。

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.4. MI_VENC_H264eRefSliceType_e

➢ 说明

定义获取的 H.264 码流属于何种跳帧参考模式下的参考帧。

➢ 定义

```
typedef enum
{
    E_MI_VENC_H264E_REFSLICE_FOR_1X = 1,
    E_MI_VENC_H264E_REFSLICE_FOR_2X = 2,
    E_MI_VENC_H264E_REFSLICE_FOR_4X,
    E_MI_VENC_H264E_REFSLICE_FOR_MAX = 5
}MI_VENC_H264eRefSliceType_e;
```

➢ 成员

成员名称	描述
E_MI_VENC_H264E_REFSLICE_FOR_1X	1 倍跳帧参考时的参考帧。
E_MI_VENC_H264E_REFSLICE_FOR_2X	2 倍跳帧参考时的参考帧或 4 倍跳帧参考时用于 2 倍跳帧参考的参考帧。
E_MI_VENC_H264E_REFSLICE_FOR_4X	4 倍跳帧参考时的参考帧。
E_MI_VENC_H264E_REFSLICE_MAX	非参考帧。

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.5. MI_VENC_H264eRefType_e

➢ 说明

定义 H.264 跳帧参考码流的帧类型以及参考属性。

➢ 定义

```
typedef enum
{
   E_MI_VENC_BASE_IDR = 0,
   E_MI_VENC_BASE_P_REFTOIDR,
   E_MI_VENC_BASE_P_REFBYBASE,
   E_MI_VENC_BASE_P_REFBYENHANCE,
   E_MI_VENC_ENHANCE_P_REFBYENHANCE,
   E_MI_VENC_ENHANCE_P_NOTFORREF,
   E_MI_VENC_REF_TYPE_MAX
}MI_VENC_H264eRefType_e;
```

➢ 成员

成员名称	描述
E_MI_VENC_BASE_IDR	base 层中的 IDR 帧。
E_MI_VENC_BASE_P_REFTOIDR	Base 层中的 P 帧，用于参考 I 帧
E_MI_VENC_BASE_P_REFBYBASE	base 层中的 P 帧，用于 base 层中其他帧的参考。
E_MI_VENC_BASE_P_REFBYENHANCE	base 层中的 P 帧，用于 enhance 层中的帧的参考。
E_MI_VENC_ENHANCE_P_REFBYENHANCE	enhance 层中的 P 帧，用于 enhance 层中其他帧的参考。
E_MI_VENC_ENHANCE_P_NOTFORREF	enhance 层中的 P 帧，不用于参考

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.6. MI_VENC_JpegePackType_e

➢ 说明

定义 JPEG 码流的 PACK 类型。

➢ 定义

```
typedef enum
{
    E_MI_VENC_JPEGE_PACK_ECS = 5,
    E_MI_VENC_JPEGE_PACK_APP = 6,
    E_MI_VENC_JPEGE_PACK_VDO = 7,
    E_MI_VENC_JPEGE_PACK_PIC = 8,
    E_MI_VENC_JPEGE_PACK_MAX
}MI_VENC_JpegePackType_e;
```

➢ 成员

成员名称	描述
E_MI_VENC_JPEGE_PACK_ECS	ECS 类型。
E_MI_VENC_JPEGE_PACK_APP	APP 类型。
E_MI_VENC_JPEGE_PACK_VDO	VDO 类型。
E_MI_VENC_JPEGE_PACK_PIC	PIC 类型。

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.7. MI_VENC_H265eNaluType_e

➢ 说明

定义 H.265 码流 NALU 类型。

➢ 定义

```
typedef enum
{
    E_MI_VENC_H265E_NALU_PSLICE = 1,
    E_MI_VENC_H265E_NALU_ISLICE = 19,
    E_MI_VENC_H265E_NALU_VPS = 32,
    E_MI_VENC_H265E_NALU_SPS = 33,
    E_MI_VENC_H265E_NALU_PPS = 34,
    E_MI_VENC_H265E_NALU_SEI = 39,
    E_MI_VENC_H265E_NALU_MAX
} MI_VENC_H265eNaulType_e;
```

➢ 成员

成员名称	描述
E_MI_VENC_H265E_NALU_PSLICE	PSLICE 类型。
E_MI_VENC_H265E_NALU_ISLICE	ISLICE 类型。
E_MI_VENC_H265E_NALU_VPS	VPS 类型。
E_MI_VENC_H265E_NALU_SPS	SPS 类型。
E_MI_VENC_H265E_NALU_PPS	PPS 类型。
E_MI_VENC_H265E_NALU_SEI	SEI 类型。

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.8. MI_VENC_Rect_t

➢ 说明

定义编码描述的矩形框。

➢ 定义

```
typedef struct MI_VENC_Rect_s
{
    MI_U32 u32Left;
    MI_U32 u32Top;
    MI_U32 u32Width;
    MI_U32 u32Height;
}MI_VENC_Rect_t;
```

➢ 成员

成员名称	描述
u32Left	矩形框左侧与实际画面左侧的距离，单位为 pixcel
u32Top	矩形框上边缘与实际画面上边缘的距离，单位为 pixcel
u32Width	矩形框的宽度，单位为 pixcel
u32Height	矩形框的高度，单位为 pixcel

※ 注意事项

矩形框不能超出编码实际画面的范围。

➢ 相关数据类型及接口

MI_VENC_SetRoiCfg
MI_VENC_SetCrop
MI_VENC_SetRoiBgFrameRate

2.9. MI_VENC_DataType_t

➢ 说明

定义码流结果类型。

➢ 定义

```
typedef union MI_VENC_DataType_s
{
  MI_VENC_H264eNaluType_e  eH264EType;
  MI_VENC_JpegePackType_e  eJPEGEType;
  MI_VENC_H265eNaulType_e  eH265EType;
}MI_VENC_DataType_t;
```

➢ 成员

成员名称	描述
enH264EType	H.264 码流包类型。
enJPEGEType	JPEG 码流包类型。
enH265EType	H.265 码流包类型。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_H264eNaluType_e
MI_VENC_JpegePackType_e

2.10. MI_VENC_PackInfo_t

➢ 说明

定义当前码流包数据中包含的其他类型码流包数据的结构体。

➢ 定义

```
typedef struct MI_VENC_PackInfo_s
{
  MI_VENC_DataType_t  stPackType;
  MI_U32   u32PackOffset;
  MI_U32   u32PackLength;
  MI_U32   u32SliceId;
}MI_VENC_PackInfo_t;
```

➢ 成员

成员名称	描述
u32PackType	当前码流包数据包含其他码流包的类型。
u32PackOffset	当前码流包数据包含其他码流包数据的偏移。
u32PackLength	当前码流包数据包含其他码流包数据的大小。
u32SliceId	当前码流包数据包含其他码流包的 slice id。

2.11. MI_VENC_Pack_t

➢ 说明

定义帧码流包结构体。

➢ 定义

```
typedef struct MI_VENC_Pack_s
{
    MI_PHY phyAddr;
    MI_U8 *pu8Addr;
    MI_U32 u32Len;
    MI_U64 u64PTS;
    MI_BOOL bFrameEnd;
    MI_VENC_DataType_t stDataType;
    MI_U32 u32Offset;
    MI_U32 u32DataNum;
    MI_VENC_PackInfo_t asackInfo[8];
} MI_VENC_Pack_t;
```

➢ 成员

成员名称	描述
pu8Addr	码流包首地址。
phyAddr	码流包物理地址。
u32Len	码流包长度。
stDataType	码流类型，支持 H.264/JPEG/MPEG-4 协议类型的数据包。
u64PTS	时间戳。单位：us。
bFrameEnd	帧结束标识。 取值范围： TRUE：该码流包是该帧的最后一个包。FALSE：该码流包不是该帧的最后一个包。
u32Offset	码流包中有效数据与码流包首地址 pu8Addr 的偏移。
u32DataNum	当前码流包（当前包的类型由 DataType 指定）数据中包含其他类型码流包的个数。
asackInfo[8]	当前码流包数据中包含其他类型码流包数据信息。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_DataType_t

2.12. MI_VENC_StreamInfoH264_t

➢ 说明

定义 H.264 协议码流特征信息。

➢ 定义

```
typedef struct MI_VENC_StreamInfoH264_s
{
  MI_U32   u32PicBytesNum;
  MI_U32   u32PSkipMbNum;
  MI_U32   u32IpcmMbNum;
  MI_U32   u32Inter16x8MbNum;
  MI_U32   u32Inter16x16MbNum;
  MI_U32   u32Inter8x16MbNum;
  MI_U32   u32Inter8x8MbNum;
  MI_U32   u32Intra16MbNum;
  MI_U32   u32Intra8MbNum;
  MI_U32   u32Intra4MbNum;
  MI_VENC_H264eRefSliceType_e   eRefSliceType;
  MI_VENC_H264eRefType_e    eRefType;
  MI_U32   u32UpdateAttrCnt;
  MI_U32   u32StartQp;
}MI_VENC_StreamInfoH264_t;
```

➢ 成员

成员名称	描述
u32PicBytesNum	编码当前帧的字节（BYTE）数
u32PSkipMbNum	编码当前帧中采用跳跃（SKIP）编码模式的宏块数
u32IpcmMbNum	编码当前帧中采用 IPCM 编码模式的宏块数
u32Inter16x8MbNum	编码当前帧中采用 Inter16x8 预测模式的宏块数
u32Inter16x16MbNum	编码当前帧中采用 Inter16x16 预测模式的宏块数
u32Inter8x16MbNum	编码当前帧中采用 Inter8x16 预测模式的宏块数
u32Inter8x8MbNum	编码当前帧中采用 Inter8x8 预测模式的宏块数
u32Intra16MbNum	编码当前帧中采用 Intra16 预测模式的宏块数
u32Intra8MbNum	编码当前帧中采用 Intra8 预测模式的宏块数

成员名称	描述
u32Intra4MbNum	编码当前帧中采用 Intra4 预测模式的宏块数
enRefSliceType	编码当前帧属于何种跳帧参考模式下的参考帧
enRefType	高级跳帧参考下的编码帧类型
u32UpdateAttrCnt	通道属性或参数(包含 RC 参数)被设置的次数
u32StartQp	编码采用的起始 Qp 值

- 注意事项

  保存 H.264 码流时，可只选择相应跳帧参考模式下的参考帧：

  当跳帧参考模式是 1 倍跳帧参考模式时，可只保存 enRefSliceType 等于

  E_MI_VENC_H264E_REFSLICE_FOR_1X 的码流。

  当跳帧参考模式是 2 倍跳帧参考模式时，可只保存 enRefSliceType 等于

  E_MI_VENC_H264E_REFSLICE_FOR_2X 的码流。

  当跳帧参考模式是 4 倍跳帧参考模式时，可只保存 enRefSliceType 等于 E_MI_VENC_H264E_REFSLICE_FOR_4X 的码流，或同时保存 enRefSliceType 等于 E_MI_VENC_H264E_REFSLICE_FOR_2X 和 E_MI_VENC_H264E_REFSLICE_FOR_4X 的码流。

- 相关数据类型及接口

  MI_VENC_DataType_t

  MI_VENC_H264eRefSliceType_e

2.13. MI_VENC_StreamInfoJpeg_t

- 说明

  定义 JPEG 协议码流特征信息。

- 定义

```
typedef struct MI_VENC_StreamInfoJpeg_s
{
  MI_U32   u32PicBytesNum;
  MI_U32   u32UpdateAttrCnt;
  MI_U32   u32Qfactor;
}MI_VENC_StreamInfoJpeg_t;
```

- 成员

成员名称	描述
u32PicBytesNum	一帧 jpeg 码流大小，以字节（byte）为单位。
u32UpdateAttrCnt	通道属性或参数(包含 RC 参数)被设置的次数。
u32Qfactor	编码当前帧的 Qfactor。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_DataType_t

2.14. MI_VENC_StreamInfoH265_t

➢ 说明

定义 H.265 协议码流特征信息。

➢ 定义

```
typedef struct MI_VENC_StreamInfoH265_s
{
  MI_U32   u32PicBytesNum;
  MI_U32   u32Inter64x64CuNum;
  MI_U32   u32Inter32x32CuNum;
  MI_U32   u32Inter16x16CuNum;
  MI_U32   u32Inter8x8CuNum;
  MI_U32   u32Intra32x32CuNum;
  MI_U32   u32Intra16x16CuNum;
  MI_U32   u32Intra8x8CuNum;
  MI_U32   u32Intra4x4CuNum;
  MI_VENC_H265eRefType_e       eRefType;
  MI_U32   u32UpdateAttrCnt;
  MI_U32   u32StartQp;
}MI_VENC_StreamInfoH265_t;
```

➢ 成员

成员名称	描述
u32PicBytesNum	编码当前帧的字节（BYTE）数
u32Inter64x64CuNum	编码当前帧中采用 Inter64x64 预测模式的 CU 块数
u32Inter32x32CuNum	编码当前帧中采用 Inter32x32 预测模式的 CU 块数
u32Inter16x16CuNum	编码当前帧中采用 Inter16x16 预测模式的 CU 块数
u32Inter8x8CuNum	编码当前帧中采用 Inter8x8 预测模式的 CU 块数
u32Intra32x32CuNum	编码当前帧中采用 Intra32x32 预测模式的 CU 块数
u32Intra16x16CuNum	编码当前帧中采用 Intra16x16 预测模式的 CU 块数
u32Intra8x8CuNum	编码当前帧中采用 Intra8x8 预测模式的 CU 块数
u32Intra4x4CuNum	编码当前帧中采用 Intra4x4 预测模式的 CU 块数
eRefType	高级跳帧参考下的编码帧类型
u32UpdateAttrCnt	通道属性或参数(包含 RC 参数)被设置的次数
u32StartQp	编码采用的起始 Qp 值

※ 注意事项

enRefType 请参见 MI_VENC_StreamInfoH264_t 中关于 enRefType 变量的说明。

➢ 相关数据类型及接口

MI_VENC_DataType_t
MI_VENC_H264eRefSliceType_e
MI_VENC_H264eRefType_e

2.15. MI_VENC_Stream_t

➢ 说明

定义帧码流类型结构体。

➢ 定义

```
typedef struct MI_VENC_Stream_s
{
    MI_VENC_Pack_t *pstPack;
    MI_U32 u32PackCount;
    MI_U32 u32Seq;
    MI_SYS_BUF_HANDLE hMiSys;
    union
    {
        MI_VENC_StreamInfoH264_t stH264Info;
        MI_VENC_StreamInfoJpeg_t stJpegInfo;
        MI_VENC_StreamInfoH265_t stH265Info;
    };
} MI_VENC_Stream_t;
```

➢ 成员

成员名称	描述
pstPack	帧码流包结构。
u32PackCount	一帧码流的所有包的个数。
u32Seq	码流序列号。。
hMiSys	Buffer 句柄。
stH624Info/stJpegInfo/ /stH265Info	码流特征信息。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_Pack_t
MI_VENC_GetStream

2.16. MI_VENC_StreamBufInfo_t

➢ 说明

定义码流 buffer 信息的结构体。

➢ 定义

```
typedef struct MI_VENC_StreamBufInfo_s
{
  MI_PHY   phyAddr;
  void*pUserAddr;
  MI_U32   u32BufSize;
}MI_VENC_StreamBufInfo_t;
```

➢ 成员

成员名称	描述
u32PhyAddr	码流 buffer 的物理地址。
pUserAddr	码流 buffer 的虚拟地址。
u32BufSize	码流 buffer 的大小。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_GetStreamBufInfo

2.17. MI_VENC_AttrH264_t

➢ 说明

定义 H.264 编码属性结构体。

➢ 定义

```
typedef struct MI_VENC_AttrH264_s
{
  MI_U32   u32MaxPicWidth;
  MI_U32   u32MaxPicHeight;
  MI_U32   u32BufSize;
  MI_U32   u32Profile;
  MI_BOOL bByFrame;
  MI_U32   u32PicWidth;
  MI_U32   u32PicHeight;
  MI_U32   u32BFrameNum;
  MI_U32   u32RefNum;
}MI_VENC_AttrH264_t;
```

➢ 成员

成员名称	描述
u32MaxPicWidth	编码图像最大宽度。取值范围：[MIN_WIDTH,MAX_WIDTH]，以像素为单位。必须是 MIN_ALIGN 的整数倍。 静态属性。
u32PicWidth	编码图像宽度。 取值范围：[MIN_WIDTH,u32MaxPicWidth]，以像素为单位。 必须是 MIN_ALIGN 的整数倍。 动态属性。
u32MaxPicHeight	编码图像最大高度。取值范围：[MIN_HEIGHT,MAX_HEIGHT]，以像素为单位。必须是 MIN_ALIGN 的整数倍 静态属性。
u32PicHeight	编码图像高度。 取值范围：[MIN_HEIGHT,u32MaxPicHeight]，以像素为单位。 必须是 MIN_ALIGN 的整数倍 动态属性。
u32BufSize	码流 buffer 大小。 取值范围：[Min,Max]，以 byte 为单位。 必须是 64 的整数倍。 推荐值：一幅最大编码图像大小。推荐值为 u32MaxPicWidthx u32MaxPicHeightx1.5byte。 Min：一幅最大编码图像大小的 1/2。 Max：无限制，但是会消耗更多的内存。 静态属性。
bByFrame	帧/包模式获取码流。取值范围：{TRUE,FALSE}。 TRUE：按帧获取。 FALSE：按包获取。 静态属性。
u32Profile	编码的等级。取值范围：[0,3]。0：Baseline。1：MainProfile。2：HighProfile。3：Svc-T。 动态属性。
u32BFrameNum	编码支持 B 帧的个数。取值范围：[0,Max]。Max：无限制。静态属性，保留字段，暂不支持。
u32RefNum	编码支持参考帧的个数。取值范围：[1,2]。静态属性，保留字段，暂不支持。

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.18. MI_VENC_AttrJpeg_t

➢ 说明

定义 JPEG 抓拍属性结构体。

➢ 定义

```
typedef struct MI_VENC_AttrJpeg_s
{
  MI_U32   u32MaxPicWidth;
  MI_U32   u32MaxPicHeight;
  MI_U32   u32BufSize;
  MI_BOOL bByFrame;
  MI_U32   u32PicWidth;
  MI_U32   u32PicHeight;
  MI_BOOL bSupportDCF;
  MI_U32 u32RestartMakerPerRowCnt;
}MI_VENC_AttrJpeg_t;
```

➢ 成员

成员名称	描述
u32MaxPicWidth	编码图像最大宽度。 取值范围：[MIN_WIDTH,MAX_WIDTH]，以像素为单位。 必须是 MIN_ALIGN 的整数倍。 静态属性。
u32PicWidth	编码图像宽度。 取值范围：[MIN_WIDTH,u32MaxPicWidth]，以像素为单位。 必须是 MIN_ALIGN 的整数倍。 动态属性。
u32MaxPicHeight	编码图像最大高度。 取值范围：[MIN_HEIGHT,MAX_HEIGHT]，以像素为单位。 必须是 MIN_ALIGN 的整数倍 静态属性。
u32PicHeight	编码图像高度。 取值范围：[MIN_HEIGHT,u32MaxPicHeight]，以像素为单位。 必须是 MIN_ALIGN 的整数倍 动态属性。

成员名称	描述
u32BufSize	配置 buffer 大小。 取值范围：不小于图像最大宽高 16 对齐后的乘积。 必须是 64 的整数倍。 静态属性。
bByFrame	获取码流模式,帧或包。取值范围： {TRUE,FALSE}。 TRUE：按帧获取。 FALSE：按包获取。 静态属性。
bSupportDCF	是否支持 Jpeg 的缩略图。 静态属性。
u32RestartMakerPerRowCnt	每隔一定行数就重新开始一个 marker。

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.19. MI_VENC_AttrH265_t

➢ 说明

定义 H.265 编码属性结构体。

➢ 定义

```
typedef struct MI_VENC_AttrH265_s
{
  MI_U32   u32MaxPicWidth;
  MI_U32   u32MaxPicHeight;
  MI_U32   u32BufSize;
  MI_U32   u32Profile;
  MI_BOOL bByFrame;
  MI_U32   u32PicWidth;
  MI_U32   u32PicHeight;
  MI_U32   u32BFrameNum;
  MI_U32   u32RefNum;
}MI_VENC_AttrH265_t;
```

➢ 成员

成员名称	描述
u32MaxPicWidth	编码图像最大宽度。取值范围：[MIN_WIDTH,MAX_WIDTH]，以像素为单位。必须是 MIN_ALIGN 的整数倍。 静态属性。
u32PicWidth	编码图像宽度。 取值范围：[MIN_WIDTH,u32MaxPicWidth]，以像素为单位。必须是 MIN_ALIGN 的整数倍。动 态属性。
u32MaxPicHeight	编码图像最大高度。取值范围：[MIN_HEIGHT,MAX_HEIGHT]，以像素为单位。必须是 MIN_ALIGN 的整数倍 静态属性。
u32PicHeight	编码图像高度。 取值范围：[MIN_HEIGHT,u32MaxPicHeight]，以像素为单位。必须是 MIN_ALIGN 的整数倍 动态属性。
u32BufSize	码流 buffer 大小。 取值范围：[Min,Max]，以 byte 为单位。 必须是 64 的整数倍。 推荐值：一幅最大编码图像大小。推荐值为 u32MaxPicWidth u32MaxPicHeight 1.5byte。 Min：一幅最大编码图像大小的 1/2。 Max：无限制，但是会消耗更多的内存。 静态属性。
bByFrame	帧/包模式获取码流。取值范围：{TRUE,FALSE}。 TRUE：按帧获取。 FALSE：按包获取。 静态属性。
u32Profile	编码的等级。 取值范围：0。 0：MainProfile。 动态属性。
u32BFrameNum	编码支持 B 帧的个数。取值范围：[0,Max]。Max：无限制。静态属性， 保留字段，暂不支持。
u32RefNum	编码支持参考帧的个数。 取值范围：[1,2]。 静态属性，保留字段，暂不支持。

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.20. MI_VENC_Attr_t

➢ 说明

定义编码器属性结构体。

➢ 定义

```
typedef struct MI_VENC_Attr_s
{
  MI_VENC_ModType_e        eType;
  union
  {
    MI_VENC_AttrH264_t stAttrH264e;
    MI_VENC_AttrJpeg_t       stAttrJpeg;
    MI_VENC_AttrH265_t stAttrH265e;
  };
}MI_VENC_Attr_t;
```

➢ 成员

成员名称	描述
eType	编码协议类型。
stAttrH264e/stAttrMjpeg/stAttrJpeg/ /stAttrH265e	某种协议的编码器属性。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_CreateChn

2.21. MI_VENC_ChnAttr_t

➢ 说明

定义编码通道属性结构体。

➢ 定义

```
typedef struct MI_VENC_ChnAttr_s
{
  MI_VENC_Attr_t    stVeAttr;
  MI_VENC_RcAttr_t  stRcAttr;
}MI_VENC_ChnAttr_t;
```

➢ 成员

成员名称	描述
stVeAttr	编码器属性。
stRcAttr	码率控制器属性。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_CreateChn

2.22. MI_VENC_ChnStat_t

➢ 说明

定义编码通道的状态结构体。

➢ 定义

```
typedef struct MI_VENC_CHN Stat_s
{
  MI_U32  u32LeftPics;
  MI_U32  u32LeftStreamBytes;
  MI_U32  u32LeftStreamFrames;
  MI_U32  u32LeftStreamMillisec;
  MI_U32  u32CurPacks;
  MI_U32  u32LeftRecvPics;
  MI_U32  u32LeftEncPics;
  MI_U32  u32FrmRateNum;
  MI_U32  u32FrmRateDen;
  MI_U32  u32Bitrate;
}MI_VENC_ChnStat_t;
```

➢ 成员

成员名称	描述
u32LeftPics	待编码的图像数。
u32LeftStreamBytes	码流 buffer 剩余的 byte 数。
u32LeftStreamFrames	码流 buffer 剩余的帧数。

成员名称	描述
u32LeftStreamMillisec	码流 buffer 剩余的时长，单位为 ms
u32CurPacks	当前帧的码流包个数。
u32LeftRecvPics	剩余待接收的帧数，在用户设置 MI_VENC_StartRecvPicEx 后有效。
u32LeftEncPics	剩余待编码的帧数，在用户设置 MI_VENC_StartRecvPicEx 后有效。
u32FrmRateNum	最近 1s 内统计的帧率分子，以整数为单位。
u32FrmRateDen	最近 1s 内统计的帧率分母，以整数为单位。
u32Bitrate	最近 1s 内统计的码率，单位为 kbps

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_Query

2.23. MI_VENC_ParamH264SliceSplit_t

➢ 说明

定义 H.264 协议编码通道 SLICE 分割结构体。

➢ 定义

```
typedef struct MI_VENC_ParamH264SliceSplit_s
{
  MI_BOOL bSplitEnable;
  MI_U32   u32SliceRowCount;
}MI_VENC_ParamH264SliceSplit_t;
```

➢ 成员

成员名称	描述
bSplitEnable	Slice 分割是否使能。
u32SliceRowCount	表示每个 slice 占的宏块行数。最小值为：1，最大值为：(图像高+15)/16。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH264SliceSplit

MI_VENC_GetH264SliceSplit

2.24. MI_VENC_ParamH265SliceSplit_t

➢ 说明

定义 H.265 协议编码通道 SLICE 分割结构体。

➢ 定义

```
typedef struct MI_VENC_ParamH265SliceSplit_s
{
  MI_BOOL bSplitEnable;
  MI_U32   u32SliceRowCount;
}MI_VENC_ParamH265SliceSplit_t;
```

➢ 成员

成员名称	描述
bSplitEnable	Slice 分割是否使能。
u32SliceRowCount	表示每个 slice 占的宏块行数。最小值为：1，最大值为：(图像高+15)/16。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH265SliceSplit
MI_VENC_GetH265SliceSplit

2.25. MI_VENC_ParamH264InterPred_t

➢ 说明

定义 H.264 协议编码通道帧间预测结构体。

➢ 定义

```
typedef struct MI_VENC_ParamH264InterPred_s
{
  /*searchwindow*/
  MI_U32   u32HWSize;
  MI_U32   u32VWSize;
  MI_BOOL bInter16x16PredEn;
  MI_BOOL bInter16x8PredEn;
  MI_BOOL bInter8x16PredEn;
  MI_BOOL bInter8x8PredEn;
  MI_BOOL bInter8x4PredEn;
  MI_BOOL bInter4x8PredEn;
  MI_BOOL bInter4x4PredEn;
  MI_BOOL bExtedgeEn;
}MI_VENC_ParamH264InterPred_t;
```

➢ 成员

成员名称	描述
u32HWSize	水平搜索窗大小。
u32VWSize	垂直搜索窗大小。
bInter16x16PredEn	16x16 帧间预测使能开关，默认使能。
bInter16x8PredEn	16x8 帧间预测使能开关，默认使能。
bInter8x16PredEn	8x16 帧间预测使能开关，默认使能。
bInter8x8PredEn	8x8 帧间预测使能开关，默认使能。
bInter8x4PredEn	8x4 帧间预测使能开关，默认使能。
bInter4x8PredEn	4x8 帧间预测使能开关，默认使能。
bInter4x4PredEn	4x4 帧间预测使能开关，默认使能。
bExtedgeEn	当搜索遇见图像边界时，超出了图像范围，是否进行补搜的使能开关，默认使能。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH264InterPred
MI_VENC_GetH264InterPred

2.26. MI_VENC_ParamH264IntraPred_t

➢ 说明

定义 H.264 协议编码通道帧内预测结构体。

➢ 定义

```
typedef struct MI_VENC_ParamH264IntraPred_s
{
  MI_BOOL bIntra16x16PredEn;
  MI_BOOL bIntraNxNPredEn;
  MI_BOOL bConstrainedIntraPredFlag;
  MI_BOOL bIpcmEn;
  MI_U32   u32Intra16x16Penalty;
  MI_U32   u32Intra4x4Penalty;
  MI_BOOL bIntraPlanarPenalty;
}MI_VENC_ParamH264IntraPred_t;
```

➢ 成员

成员名称	描述
bIntra16x16PredEn	16x16 帧内预测使能，默认使能。 取值范围：0 或 1。 0：不使能； 1：使能。
bIntraNxNPredEn	NxN 帧内预测使能，默认使能。 取值范围：0 或 1。 0：不使能；1：使能。
bConstrainedIntraPredFlag	默认为 0。取值范围：0 或 1。请参见 H.264 协议关于 constrained_intra_pred_flag 的解释。
bIpcmEn	IPCM 预测使能，默认值根据不同的芯片有差异。 取值范围：0 或 1。
u32Intra16x16Penalty	默认为 0。
u32Intra4x4Penalty	默认为 0。
bIntraPlanarPenalty	默认为 0。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH264IntraPred
MI_VENC_GetH264IntraPred

2.27. MI_VENC_ParamH264Trans_t

➢ 说明

定义 H.264 协议编码通道变换、量化结构体。

➢ 定义

```
typedef struct MI_VENC_ParamH264Trans_s
{
  MI_U32   u32IntraTransMode;
  MI_U32   u32InterTransMode;
  MI_S32 s32ChromaQpIndexOffset;
}MI_VENC_ParamH264Trans_t;
```

➢ 成员

成员名称	描述
u32IntraTransMode	帧内预测的变换模式： •0：支持 4x4，16x16 变换，highprofile 支持。 - 1：4x4 变换，baseline,main,highprofile 均支持。 2：8x8 变换，highprofile 支持。系统根据通道协议类型默认 TransMode 的选择。
u32InterTransMode	帧间预测的变换模式： •0：支持 4x4，8x8 变换，highprofile 支持。 - 1：4x4 变换，baseline,main,highprofile 均支持。 - 2：8x8 变换，highprofile 支持。系统根据通道协议类型默认 TransMode 的选择。
s32ChromaQpIndexOffset	具体含义请参见 H.264 协议关于 slice_qp_offset_index 的解释。系统默认值为 0。取值范围：[-12,12]。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH264Trans

MI_VENC_GetH264Trans

2.28. MI_VENC_ParamH264Entropy_t

➢ 说明

定义 H.264 协议编码通道熵编码结构体。

➢ 定义

```
typedef struct MI_VENC_ParamH264Entropy_s
{
  MI_U32   u32EntropyEncModeI;
  MI_U32   u32EntropyEncModeP;
}MI_VENC_ParamH264Entropy_t;
```

➢ 成员

成员名称	描述
u32EntropyEncModeI	I 帧熵编码模式。 0：cavlc 1：cabac >=2 没有意义。

成员名称	描述
	Baseline 不支持 cabac。
u32EntropyEncModeP	P 帧熵编码模式。 - 0：cavlc - 1：cabac >=2 没有意义。 Baseline 不支持 cabac。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH264Entropy
MI_VENC_GetH264Entropy

2.29. MI_VENC_ParamH265InterPred_t

➢ 说明

定义 H.265 协议编码通道帧间预测结构体。

➢ 定义

```
typedef struct MI_VENC_ParamH265InterPred_s
{
  /*search window*/
  MI_U32   u32HWSize;
  MI_U32   u32VWSize;
  MI_BOOL bInter16x16PredEn;
  MI_BOOL bInter16x8PredEn;
  MI_BOOL bInter8x16PredEn;
  MI_BOOL bInter8x8PredEn;
  MI_BOOL bInter8x4PredEn;
  MI_BOOL bInter4x8PredEn;
  MI_BOOL bInter4x4PredEn;
  MI_U32   u32Inter32x32Penalty;
  MI_U32   u32Inter16x16Penalty;
  MI_U32   u32Inter8x8Penalty;
  MI_BOOL bExtedgeEn;
}MI_VENC_ParamH265InterPred_t;
```

➢ 成员

成员名称	描述
u32HWSize	水平搜索窗大小。
u32VWSize	垂直搜索窗大小。

成员名称	描述
bInter16x16PredEn	16x16 帧间预测使能开关，默认使能。
bInter16x8PredEn	16x8 帧间预测使能开关，默认使能。
bInter8x16PredEn	8x16 帧间预测使能开关，默认使能。
bInter8x8PredEn	8x8 帧间预测使能开关，默认使能。
bInter8x4PredEn	8x4 帧间预测使能开关，默认使能。
bInter4x8PredEn	4x8 帧间预测使能开关，默认使能。
bInter4x4PredEn	4x4 帧间预测使能开关，默认使能。
u32Inter32x32Penalty	32x32 帧间预测选中的概率，值越大选中的概率越低。 默认为 0。取值范围[0,65535]
u32Inter16x16Penalty	16x16 帧间预测选中的概率，值越大选中的概率越低。 默认为 0。取值范围[0,65535]
u32Inter8x8Penalty	8x8 帧间预测选中的概率，值越大选中的概率越低。 默认为 0。取值范围[0,65535]
bExtedgeEn	当搜索遇见图像边界时，超出了图像范围，是否进行补搜的使能开关，默认使能。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH265InterPred
MI_VENC_GetH265InterPred

2.30. MI_VENC_ParamH265IntraPred_t

➢ 说明

定义 H.265 协议编码通道帧内预测结构体。

➢ 定义

```
typedef struct MI_VENC_ParamH265IntraPred_s
{
  MI_BOOL bIntra32x32PredEn;
  MI_BOOL bIntra16x16PredEn;
  MI_BOOL bIntra8x8PredEn;
  MI_BOOL bConstrainedIntraPredFlag;
  MI_U32   u32Intra32x32Penalty;
  MI_U32   u32Intra16x16Penalty;
  MI_U32   u32Intra8x8Penalty;
}MI_VENC_ParamH265IntraPred_t;
```

➢ 成员

成员名称	描述
bIntra32x32PredEn	32x32 帧间预测使能开关，默认使能。
bIntra16x16PredEn	16x16 帧间预测使能开关，默认使能。
bIntra8x8PredEn	8x8 帧间预测使能开关，默认使能。
bConstrainedIntraPredFlag	默认为 0。取值范围：0 或 1。请参见 H.265 协议关于 constrained_intra_pred_flag 的解释。
u32Intra32x32Penalty	32x32 块帧内预测被选中概率，值越大概率越低 默认为 0。
u32Intra16x16Penalty	16x16 块帧内预测被选中概率，值越大概率越低 默认为 0。
u32Intra8x8Penalty	8x8 块帧内预测被选中概率，值越大概率越低 默认为 0。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH265IntraPred
MI_VENC_GetH265IntraPred

2.31. MI_VENC_ParamH265Trans_t

➢ 说明

定义 H.265 协议编码通道变换、量化结构体。

➢ 定义

```
typedef struct MI_VENC_ParamH265Trans_s
{
  MI_U32   u32IntraTransMode;
  MI_U32   u32InterTransMode;
  MI_S32 s32ChromaQpIndexOffset;
}MI_VENC_ParamH265Trans_t;
```

➢ 成员

成员名称	描述
u32IntraTransMode	帧内预测的变换模式： •0：支持 4x4，16x16，32x32 变换，highprofile 支持。 - 1：4x4 变换，baseline,main,highprofile 均支持。

成员名称	描述
	- 2：8x8 变换，highprofile 支持。系统根据通道协议类型默认 TransMode 的选择。
u32InterTransMode	帧间预测的变换模式： •0：支持 4x4，8x8 变换，highprofile 支持。 - 1：4x4 变换，baseline,main,highprofile 均支持。 - 2：8x8 变换，highprofile 支持。系统根据通道协议类型默认 TransMode 的选择。
s32ChromaQpIndexOffset	具体含义请参见 H.265 协议关于 slice_cb_qp_offset 的解释。 系统默认值为 0。取值范围：[-12,12]。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH265Trans
MI_VENC_GetH265Trans

2.32. MI_VENC_ParamH264Dblk_t

➢ 说明

定义 H.264 协议编码通道 Dblk 结构体。

➢ 定义

```
typedef struct MI_VENC_ParamH264Dblk_s
{
  MI_U32   disable_deblocking_filter_idc;
  MI_S32   slice_alpha_c0_offset_div2;
  MI_S32   slice_beta_offset_div2;
}MI_VENC_ParamH264Dblk_t;
```

➢ 成员

成员名称	描述
disable_deblocking_filter_idc	取值范围[0,2],默认值 0，具体含义请参见 H.264 协议。
slice_alpha_c0_offset_div2	取值范围[-6,6],默认值 0，具体含义请参见 H.264 协议。
slice_beta_offset_div2	取值范围[-6,6],默认值 0，具体含义请参见 H.264 协议。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH264Dblk
MI_VENC_GetH264Dblk

2.33. MI_VENC_ParamH265Dblk_t

➢ 说明

定义 H.265 协议编码通道 Dblk 结构体。

➢ 定义

```
typedef struct MI_VENC_ParamH265Dblk_s
{
    MI_U32 disable_deblocking_filter_idc;      //special naming for CODEC ISO SPEC.
    MI_S32 slice_tc_offset_div2;       //special naming for CODEC ISO SPEC.
    MI_S32 slice_beta_offset_div2;       //special naming for CODEC ISO SPEC.
} MI_VENC_ParamH265Dblk_t;
```

➢ 成员

成员名称	描述
disable_deblocking_filter_idc	取值范围[0,2],默认值 0，具体含义请参见 H.265 协议关于 slice_deblocking_filter_disabled_flag 的解释。
slice_tc_offset_div2	取值范围[-6,6],默认值 0，具体含义请参见 H.265 协议。
slice_beta_offset_div2	取值范围[-6,6],默认值 0，具体含义请参见 H.265 协议。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH265Dblk

MI_VENC_GetH265Dblk

2.34. MI_VENC_ParamH264Vui_t

➢ 说明

定义 H.264 协议编码通道 Vui 结构体。

➢ 定义

```
typedef struct MI_VENC_ParamH264Vui_s
{
  MI_VENC_ParamH264VuiAspectRatio_t    stVuiAspectRatio;
  MI_VENC_ParamH264VuiTimeInfo_t       stVuiTimeInfo;
  MI_VENC_ParamH264VuiVideoSignal_t        stVuiVdeoSignal;
}MI_VENC_ParamH264Vui_t;
```

➢ 成员

成员名称	描述
stVuiAspectRatio	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍。
stVuiTimeInfo	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍。
stVuiVideoSignal	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH264Vui

MI_VENC_GetH264Vui

2.35. MI_VENC_ParamH264VuiAspectRatio_t

➢ 说明

定义 H.264 协议编码通道 Vui 中 AspectRatio 信息的结构体。

➢ 定义

```
typedef struct MI_VENC_ParamH264VuiAspectRatio_s
{
  MI_U8        u8AspectRatioInfoPresentFlag;
  MI_U8        u8AspectRatioIdc;
  MI_U8        u8OverscanInfoPresentFlag;
  MI_U8        u8OverscanAppropriateFlag;
  MI_U16       u16SarWidth;
  MI_U16       u16SarHeight;
}MI_VENC_ParamH264VuiAspectRatio_t;
```

➢ 成员

成员名称	描述
u8AspectRatioInfoPresentFlag	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍，系统默认为 0。取值范围：0 或 1。
u8AspectRatioIdc	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：[0,255],17~254 保留。
u8OverscanInfoPresentFlag	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍，系统默认为 0。取值范围：0 或 1。
u8OverscanAppropriateFlag	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍，系统默认为 0。取值范围：0 或 1。
u16SarWidth	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：(0,65535],并且与 u16SarHeight 互质。
u16SarHeight	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：(0,65535],并且与 u16SarWidth 互质。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH264Vui

2.36. MI_VENC_ParamH264VuiTimeInfo_t

➢ 说明

定义 H.264 协议编码通道 Vui 中 TIME_INFO 信息的结构体。

➢ 定义

```
typedef struct MI_VENC_ParamH264VuiTimeInfo_s
{
  MI_U8         u8TimingInfoPresentFlag;
  MI_U8         u8FixedFrameRateFlag;
  MI_U32   u32NumUnitsInTick;
  MI_U32   u32TimeScale;
}MI_VENC_ParamH264VuiTimeInfo_t;
```

➢ 成员

成员名称	描述
u8TimingInfoPresentFlag	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍，系统默认为 0。取值范围：0 或 1。
u32NumUnitsInTick	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：大于 0。
u32TimeScale	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍，系统默认为 60。取值范围：大于 0。
u8FixedFrameRateFlag;	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：0 或 1。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH264Vui

MI_VENC_GetH264Vui

2.37. MI_VENC_ParamH264VuiVideoSignal_t

➢ 说明

定义 H.264 协议编码通道 Vui 中 VIDEO_SIGNAL 信息的结构体。

➢ 定义

```
typedef struct MI_VENC_ParamVuiVideoSignal_s
{
  MI_U8    u8VideoSignalTypePresentFlag;
  MI_U8    u8VideoFormat;
  MI_U8    u8VideoFullRangeFlag;
  MI_U8    u8ColourDescriptionPresentFlag;
  MI_U8    u8ColourPrimaries;
  MI_U8    u8TransferCharacteristics;
  MI_U8    u8MatrixCoefficients;
}MI_VENC_ParamH264VuiVideoSignal_t;
```

➢ 成员

成员名称	描述
u8VideoSignalTypePresentFlag	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：0 或 1。
u8VideoFormat	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍，系统默认为 5。取值范围：[0,7]。
u8VideoFullRangeFlag	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：0 或 1。
u8ColourDescriptionPresentFlag;	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：0 或 1。
u8ColourPrimaries	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：[0,255]。
u8TransferCharacteristics	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：[0,255]。
u8MatrixCoefficients	具体含义请参见 H.264 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：[0,255]。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH264Vui

2.38. MI_VENC_ParamH265Vui_t

➢ 说明

定义 H.265 协议编码通道 Vui 结构体。

➢ 定义

```
typedef struct MI_VENC_ParamH265Vui_s
{
  MI_VENC_ParamH265VuiAspectRatio_t    stVuiAspectRatio;
  MI_VENC_ParamH265VuiTimeInfo_t       stVuiTimeInfo;
  MI_VENC_ParamH265VuiVideoSignal_t    stVuiVdeoSignal;
}MI_VENC_ParamH265Vui_t;
```

➢ 成员

成员名称	描述
stVuiAspectRatio	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍。
stVuiTimeInfo	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍。
stVuiVideoSignal	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH265Vui
MI_VENC_GetH265Vui

2.39. MI_VENC_ParamH265VuiAspectRatio_t

➢ 说明

定义 H.265 协议编码通道 Vui 中 AspectRatio 信息的结构体。

➢ 定义

```
typedef struct MI_VENC_ParamH265VuiAspectRatio_s
{
  MI_U8       u8AspectRatioInfoPresentFlag;
  MI_U8       u8AspectRatioIdc;
  MI_U8       u8OverscanInfoPresentFlag;
  MI_U8       u8OverscanAppropriateFlag;
  MI_U16      u16SarWidth;
  MI_U16      u16SarHeight;
}MI_VENC_ParamH265VuiAspectRatio_t;
```

➢ 成员

成员名称	描述
u8AspectRatioInfoPresentFlag	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍，系统默认为 0。取值范围：0 或 1。
u8AspectRatioIdc	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：[0,255],17~254 保留。
u8OverscanInfoPresentFlag	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍，系统默认为 0。取值范围：0 或 1。
u8OverscanAppropriateFlag	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍，系统默认为 0。取值范围：0 或 1。
u16SarWidth	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：(0,65535],并且与 u16SarHeight 互质。
u16SarHeight	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：(0,65535],并且与 u16SarWidth 互质。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH265Vui

2.40. MI_VENC_ParamH265VuiTimeInfo_t

➢ 说明

定义 H.265 协议编码通道 Vui 中 TIME_INFO 信息的结构体。

➢ 定义

```
typedef struct MI_VENC_ParamH265VuiTimeInfo_s
{
  MI_U8        u8TimingInfoPresentFlag;
  MI_U8        u8FixedFrameRateFlag;
  MI_U32       u32NumUnitsInTick;
  MI_U32       u32TimeScale;
}MI_VENC_ParamH265VuiTimeInfo_t;
```

➢ 成员

成员名称	描述
u8TimingInfoPresentFlag	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍，系统默认为 0。取值范围：0 或 1。
u32NumUnitsInTick	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：大于 0。
u32TimeScale	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍，系统默认为 60。取值范围：大于 0。
u8FixedFrameRateFlag;	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：0 或 1。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH265Vui
MI_VENC_GetH265Vui

2.41. MI_VENC_ParamH265VuiVideoSignal_t

➢ 说明

定义 H.265 协议编码通道 Vui 中 VIDEO_SIGNAL 信息的结构体。

➢ 定义

```
typedef struct MI_VENC_ParamVuiVideoSignal_s
{
  MI_U8    u8VideoSignalTypePresentFlag;
  MI_U8    u8VideoFormat;
  MI_U8    u8VideoFullRangeFlag;
  MI_U8    u8ColourDescriptionPresentFlag;
  MI_U8    u8ColourPrimaries;
  MI_U8    u8TransferCharacteristics;
  MI_U8    u8MatrixCoefficients;
}MI_VENC_ParamH265VuiVideoSignal_t;
```

➢ 成员

成员名称	描述
u8VideoSignalTypePresentFlag	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：0 或 1。
u8VideoFormat	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍，系统默认为 5。取值范围：[0,7]。
u8VideoFullRangeFlag	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：0 或 1。
u8ColourDescriptionPresentFlag;	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：0 或 1。
u8ColourPrimaries	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：[0,255]。
u8TransferCharacteristics	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：[0,255]。
u8MatrixCoefficients	具体含义请参见 H.265 协议中 Annex E Video usablility information 的介绍，系统默认为 1。取值范围：[0,255]。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH265Vui

2.42. MI_VENC_ParamJpeg_t

➢ 说明

定义 JPEG 协议编码通道高级参数结构体。

➢ 定义

```
typedef struct MI_VENC_ParamJpeg_s
{
    MI_U32 u32Qfactor;
    MI_U8 au8YQt[64];
    MI_U8 au8CbCrQt[64];
    MI_U32 u32McuPerEcs;
} MI_VENC_ParamJpeg_t;
```

➢ 成员

成员名称	描述
u32Qfactor	具体含义请参见 RFC2435 协议，系统默认为 90。取值范围：[1,99]。
au8YQt	Y 量化表。取值范围：[1,255]。
au8CbCrQt	CbCr 量化表。取值范围：[1,255]。
u32MCUPerEcs	每个 ECS 中包含多少个 MCU，系统默认为 0，表示不划分 Ecs。u32MCUPerECS：[0,(picwidth+15)>>4x(picheight+15)>>4x2]

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetJpegParam
MI_VENC_GetJpegParam

2.43. MI_VENC_RoiCfg_t

➢ 说明

定义编码感兴趣区域信息。

➢ 定义

```
typedef struct MI_VENC_RoiCfg_s
{
  MI_U32   u32Index;
  MI_BOOL bEnable;
  MI_BOOL bAbsQp;
  MI_S32 s32Qp;
  MI_VENC_Rect_t stRect;
}MI_VENC_RoiCfg_t;
```

➢ 成员

成员名称	描述
u32Index	ROI 区域的索引，系统支持的索引范围为[0,7]，不支持超出这个范围的索引。
bEnable	是否使能这个 ROI 区域。
bAbsQp	ROI 区域 QP 模式。 - FALSE：相对 QP - TURE：绝对 QP
s32Qp	QP 值，当 QP 模式为 MI_FALSE 时，s32Qp 为 QP 偏移，s32Qp 范围[-51,51]，当 QP 模式为 TRUE 时，s32Qp 为宏块 QP 值，s32Qp 范围[0,51]。
stRect	ROI 区域。

※ 注意事项

从 SSC33X 开始，ROI 区域的 QP 模式仅支持相对 QP（bAbsQp = MI_FALSE）；同时，s32Qp 值的可支持范围修改为[-32,31]。

➢ 相关数据类型及接口

MI_VENC_SetRoiCfg

MI_VENC_GetRoiCfg

2.44. MI_VENC_RoiBgFrameRate_t

➢ 说明

定义非编码感兴趣区域帧率。

➢ 定义

```
typedef struct MI_VENC_RoiBgFrameRate_s
{
   MI_S32 s32SrcFrmRate;
   MI_S32 s32DstFrmRate;
}MI_VENC_RoiBgFrameRate_t;
```

➢ 成员

成员名称	描述
s32SrcFrmRate	非 Roi 区域的源帧率。
s32DstFrmRate	非 Roi 区域的目标帧率。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetRoiBgFrameRate

MI_VENC_GetRoiBgFrameRate

2.45. MI_VENC_ParamRef_t

➢ 说明

定义 H.264/H.265 编码的高级跳帧参考参数。

➢ 定义

```
typedef struct MI_VENC_ParamRef_s
{
  MI_U32   u32Base;
  MI_U32   u32Enhance;
  MI_BOOL bEnablePred;
}MI_VENC_ParamRef_t;
```

➢ 成员

成员名称	描述
u32Base	base 层的周期。取值范围：（0，+∞）。
u32Enhance	enhance 层的周期。取 值范围：[0,255]。
bEnablePred	代表 base 层的帧是否被 base 层其他帧用作参考。当为 MI_FALSE 时，base 层的所有帧都参考 IDR 帧。

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.46. MI_VENC_RcAttr_t

➢ 说明

定义编码通道码率控制器属性。

➢ 定义

```
typedef struct MI_VENC_RcAttr_s
{
    MI_VENC_RcMode_e  eRcMode;
    union
    {
     MI_VENC_AttrH264Cbr_t       stAttrH264Cbr;
     MI_VENC_AttrH264Vbr_t       stAttrH264Vbr;
     MI_VENC_AttrH264FixQp_t     stAttrH264FixQp;
     MI_VENC_AttrH264Abr_t       stAttrH264Abr;
     MI_VENC_AttrH264Avbr_t stAttrH264Avbr;
     MI_VENC_AttrMjpegCbr_t      stAttrMjpegCbr;
     MI_VENC_AttrMjpegFixQp_t    stAttrMjpegFixQp;
     MI_VENC_AttrH265Cbr_t       stAttrH265Cbr;
     MI_VENC_AttrH265Vbr_t       stAttrH265Vbr;
     MI_VENC_AttrH265FixQp_t     stAttrH265FixQp;
```

```
        MI_VENC_AttrH265Avbr_t stAttrH265Avbr;
    };
    MI_VOID*    pRcAttr;
}MI_VENC_RcAttr_t;
```

➢ 成员

成员名称	描述
enRcMode	RC 模式。
stAttrH264Cbr	H.264 协议编码通道 Cbr 模式属性。
stAttrH264Vbr	H.264 协议编码通道 Vbr 模式属性。
stAttrH264FixQp	H.264 协议编码通道 Fixqp 模式属性。
stAttrH264Abr	H.264 协议编码通道 Abr 模式属性（暂不支持）。
stAttrH264Avbr	H.264 协议编码通道 Avbr 模式属性。
stAttrH265Cbr	H.265 协议编码通道 Cbr 模式属性。
stAttrH265Vbr	H.265 协议编码通道 Vbr 模式属性。
stAttrH265FixQp	H.265 协议编码通道 Fixqp 模式属性。
stAttrH265Avbr	H.265 协议编码通道 Avbr 模式属性。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_CreateChn

2.47. MI_VENC_RcMode_e

➢ 说明

定义编码通道码率控制器模式。

➢ 定义

```
typedef enum
{
    E_MI_VENC_RC_MODE_H264CBR = 1,
    E_MI_VENC_RC_MODE_H264VBR,
    E_MI_VENC_RC_MODE_H264ABR,
    E_MI_VENC_RC_MODE_H264FIXQP,
    E_MI_VENC_RC_MODE_H264AVBR,
    E_MI_VENC_RC_MODE_MJPEGCBR,
    E_MI_VENC_RC_MODE_MJPEGFIXQP,
    E_MI_VENC_RC_MODE_H265CBR,
    E_MI_VENC_RC_MODE_H265VBR,
    E_MI_VENC_RC_MODE_H265FIXQP,
    E_MI_VENC_RC_MODE_H265AVBR,
    E_MI_VENC_RC_MODE_MAX,
}MI_VENC_RcMode_e;
```

➢ 成员

成员名称	描述
E_MI_VENC_RC_MODE_H264CBR	H.264 CBR 模式。
E_MI_VENC_RC_MODE_H264VBR	H.264 VBR 模式。
E_MI_VENC_RC_MODE_H264ABR	H.264 ABR 模式（暂时不提供）。
E_MI_VENC_RC_MODE_H264FIXQP	H.264 FixQp 模式。
E_MI_VENC_RC_MODE_H264AVBR	H.264 AVBR 模式
E_MI_VENC_RC_MODE_MJPEGCBR	MJPEG CBR 模式
E_MI_VENC_RC_MODE_MJPEGFIXQP	MJPEG FixQp 模式
E_MI_VENC_RC_MODE_H265CBR	H.265 CBR 模式。
E_MI_VENC_RC_MODE_H265VBR	H.265 VBR 模式。
E_MI_VENC_RC_MODE_H265FIXQP	H.265 FixQp 模式。
E_MI_VENC_RC_MODE_H265AVBR	H.265 AVBR 模式

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_CreateChn

2.48. MI_VENC_AttrH264Cbr_t

➢ 说明

定义 H.264 编码通道 CBR 属性结构。

➢ 定义

```
typedef struct MI_VENC_AttrH264Cbr_s
{
  MI_U32   u32Gop;
  MI_U32   u32StatTime;
  MI_U32   u32SrcFrmRateNum;
  MI_U32   u32SrcFrmRateDen;
  MI_U32   u32BitRate;
  MI_U32   u32FluctuateLevel;
}MI_VENC_AttrH264Cbr_t;
```

➢ 成员

成员名称	描述
u32Gop	H.264gop 值。取值范围：[1,65536]。
u32StatTime	CBR 码率统计时间，以秒为单位。取值范围：[1,60]。
u32SrcFrmRateNum	编码器帧率分子，以整数为单位。
u32SrcFrmRateDen	编码器帧率分母，以整数为单位。
u32BitRate	平均 bitrate，以 kbps 为单位。取值范围：[2,102400]。
u32FluctuateLevel	最大码率相对平均码率的波动等级，保留，暂不使用。 取值范围：[0,5]。 推荐使用波动等级 0。

※ 注意事项

SrcFrmRate 应该设置为输入编码器的实际帧率，RC 需要根据 SrcFrmRate 进行码率控制。

u32SrcFrmRateNum/u32SrcFrmRateDen 范围在(0,30]之间。

➢ 相关数据类型及接口

MI_VENC_CreateChn

2.49. MI_VENC_AttrH264Vbr_t

➢ 说明

定义 H.264 编码通道 VBR 属性结构。

➢ 定义

```
typedef struct MI_VENC_AttrH264Vbr_s
{
   MI_U32   u32Gop;
   MI_U32   u32StatTime;
   MI_U32   u32SrcFrmRateNum;
   MI_U32   u32SrcFrmRateDen;
   MI_U32   u32MaxBitRate;
   MI_U32   u32MaxQp;
   MI_U32   u32MinQp;
}MI_VENC_AttrH264Vbr_t;
```

➢ 成员

成员名称	描述
u32Gop	H.264gop 值。取值范围：[1,65536]。
u32StatTime	VBR 码率统计时间，以秒为单位。取值范围：[1,60]。
u32SrcFrmRateNum	编码器帧率分子，以整数为单位。
u32SrcFrmRateDen	编码器帧率分母，以整数为单位。
u32MaxBitRate	编码器输出最大码率，以 kbps 为单位。取值范围：[2,102400]。
u32MaxQp	编码器支持图像最大 QP。取值范围：(u32MinQp,51]。
u32MinQp	编码器支持图像最小 QP。取值范围：[0,51]。

※ 注意事项

请参见 MI_VENC_AttrH264Cbr_t 关于 u32SrcFrmRateNum 和 u32SrcFrmRateDen 的说明。

➢ 相关数据类型及接口

MI_VENC_CreateChn

2.50. MI_VENC_AttrH264FixQp_t

➢ 说明

定义 H.264 编码通道 Fixqp 属性结构。

➢ 定义

```
typedef struct MI_VENC_AttrH264FixQp_s
{
  MI_U32  u32Gop;
  MI_U32  u32SrcFrmRateNum;
  MI_U32  u32SrcFrmRateDen;
  MI_U32  u32IQp;
  MI_U32  u32PQp;
}MI_VENC_AttrH264FixQp_t;
```

➢ 成员

成员名称	描述
u32Gop	H.264gop 值。取值范围：[1,65536]。
u32SrcFrmRateNum	编码器帧率分子，以整数为单位。
u32SrcFrmRateDen	编码器帧率分母，以整数为单位。
u32IQp	I 帧所有宏块 Qp 值。取值范围：[0,51]。
u32PQp	P 帧所有宏块 Qp 值。取值范围：[0,51]。

※ 注意事项

请参见 MI_VENC_AttrH264Cbr_t 关于 u32SrcFrmRateNum 和 u32SrcFrmRateDen 的说明。

➢ 相关数据类型及接口

MI_VENC_CreateChn

2.51. MI_VENC_AttrH264Abr_t

➢ 说明

定义 H.264 编码通道 ABR 属性结构。

➢ 定义

```
typedef struct MI_VENC_AttrH264Abr_s
{
  MI_U32   u32Gop; /*the interval of ISLICE.*/
  MI_U32   u32StatTime; /*theratestatistictime,theunitissenconds(s)*/
  MI_U32   u32SrcFrmRateNum;    /*theinputframerateofthevencchnnel*/
  MI_U32   u32SrcFrmRateDen;    /*thetargetframerateofthevencchnnel*/
  MI_U32   u32AvgBitRate;       /*averagebitrate*/
  MI_U32   u32MaxBitRate;    /*themaxbitrate*/
}MI_VENC_AttrH264Abr_t;
```

➢ 成员

成员名称	描述
u32Gop	H.264gop 值。取值范围：[1,65536]。
u32StatTime	ABR 码率统计时间，以秒为单位。取值范围：[1,60]。
u32SrcFrmRateNum	编码器帧率分子，以整数为单位。
u32SrcFrmRateDen	编码器帧率分母，以整数为单位。
u32AvgBitRate	平均码率。 取值范围[2000,u32MaxBitRate）

成员名称	描述
u32MaxBitRate	最大码率。取值范围[2000,102400000]

※ 注意事项

请参见 MI_VENC_AttrH264Cbr_t 关于 u32SrcFrmRateNum 和 u32SrcFrmRateDen 的说明。

➢ 相关数据类型及接口

MI_VENC_CreateChn

2.52. MI_VENC_AttrH264Avbr_t

➢ 说明

定义 H.264 编码通道 AVBR 属性结构。

➢ 定义

```
typedef struct MI_VENC_AttrH264Avbr_s
{
    MI_U32 u32Gop;
    MI_U32 u32StatTime;
    MI_U32 u32SrcFrmRateNum;
    MI_U32 u32SrcFrmRateDen;
    MI_U32 u32MaxBitRate;
    MI_U32 u32MaxQp;
    MI_U32 u32MinQp;
} MI_VENC_AttrH264Avbr_t;
```

➢ 成员

成员名称	描述
u32Gop	gop 值。取值范围：[1,65536]。
u32StatTime	码率统计时间，以秒为单位。取值范围：[1,60]。
u32SrcFrmRateNum	编码器帧率分子，以整数为单位。
u32SrcFrmRateDen	编码器帧率分母，以整数为单位。
u32MaxBitRate	编码器输出最大码率，以 kbps 为单位。取值范围：[2,102400]。
u32MaxQp	编码器支持图像最大 QP。取值范围：(u32MinQp,48]。
u32MinQp	编码器支持图像最小 QP。取值范围：[12,48]。

※ 注意事项

请参见 MI_VENC_AttrH264Cbr_t 关于 u32SrcFrmRateNum 和 u32SrcFrmRateDen 的说明。

➢ 相关数据类型及接口

MI_VENC_CreateChn

2.53. MI_VENC_AttrMjpegCbr_t

➢ 说明

定义 MJPEG 编码通道 CBR 属性结构。

➢ 定义

```
typedef struct MI_VENC_AttrMjpegCbr_s
{
  MI_U32   u32BitRate;
  MI_U32   u32SrcFrmRateNum;
  MI_U32   u32SrcFrmRateDen;
} MI_VENC_AttrMjpegCbr_t;
```

➢ 成员

成员名称	描述
u32BitRate	码率，取值范围[2000, 102400000]
u32SrcFrmRateNum	编码器帧率分子，以整数为单位。
u32SrcFrmRateDen	编码器帧率分母，以整数为单位。

※ 注意事项

请参见 MI_VENC_AttrH264Cbr_t 关于 u32SrcFrmRateNum 和 u32SrcFrmRateDen 的说明。

➢ 相关数据类型及接口

MI_VENC_CreateChn

2.54. MI_VENC_AttrMjpegFixQp_t

➢ 说明

定义 MJPEG 编码通道 FixQp 属性结构。

➢ 定义

```
typedef struct MI_VENC_AttrMjpegFixQp_s
{
    MI_U32 u32SrcFrmRateNum;
    MI_U32 u32SrcFrmRateDen;
    MI_U32 u32Qfactor;
} MI_VENC_AttrMjpegFixQp_t;
```

➢ 成员

成员名称	描述
u32Qfactor;	Qfactor 值，取值范围[10, 90]
u32SrcFrmRateNum	编码器帧率分子，以整数为单位。
u32SrcFrmRateDen	编码器帧率分母，以整数为单位。

※ 注意事项

请参见 MI_VENC_AttrH264Cbr_t 关于 u32SrcFrmRateNum 和 u32SrcFrmRateDen 的说明。

➢ 相关数据类型及接口

MI_VENC_CreateChn

2.55. MI_VENC_AttrH265Cbr_t

➢ 说明

定义 H.265 编码通道 CBR 属性结构。

➢ 定义

```
typedef struct MI_VENC_AttrH265Cbr_s
{
  MI_U32   u32Gop;
  MI_U32   u32StatTime;
  MI_U32   u32SrcFrmRateNum;
  MI_U32   u32SrcFrmRateDen;
  MI_U32   u32BitRate;
  MI_U32   u32FluctuateLevel;
}MI_VENC_AttrH265Cbr_t;
```

➢ 成员

成员名称	描述
u32Gop	H.265gop 值。取值范围： [1,65536]。
u32StatTime	CBR 码率统计时间，以秒为单位。取值 范围：[1,60]。
u32SrcFrmRateNum	编码器帧率分子，以整数为单位。
u32SrcFrmRateDen	编码器帧率分母，以整数为单位。
u32BitRate	平均 bitrate，以 kbps 为单位。取 值范围：[2,102400]。
u32FluctuateLevel	最大码率相对平均码率的波动等级，保留，暂 不使用。 取值范围：[0,5]。 推荐使用波动等级 0。

※ 注意事项

请参见 MI_VENC_AttrH264Cbr_t 关于 u32SrcFrmRate 和 fr32DstFrmRate 的说明。

➢ 相关数据类型及接口

MI_VENC_CreateChn

2.56. MI_VENC_AttrH265Vbr_t

➢ 说明

定义 H.265 编码通道 VBR 属性结构。

➢ 定义

```
typedef struct MI_VENC_AttrH265Vbr_s
{
  MI_U32   u32Gop;
  MI_U32   u32StatTime;
  MI_U32   u32SrcFrmRateNum;
  MI_U32   u32SrcFrmRateDen;
  MI_U32   u32MaxBitRate;
  MI_U32   u32MaxQp;
  MI_U32   u32MinQp;
}MI_VENC_AttrH265Vbr_t;
```

➢ 成员

成员名称	描述
u32Gop	H.265gop 值。取值范围：[1,65536]。
u32StatTime	VBR 码率统计时间，以秒为单位。取值范围：[1,60]。
u32SrcFrmRateNum	编码器帧率分子，以整数为单位。
u32SrcFrmRateDen	编码器帧率分母，以整数为单位。
u32MaxBitRate	编码器输出最大码率，以 kbps 为单位。取值范围：[2,102400]。
u32MaxQp	编码器支持图像最大 QP。取值范围：(u32MinQp,51]。
u32MinQp	编码器支持图像最小 QP。取值范围：[0,51]。

※ 注意事项

请参见 MI_VENC_AttrH264Cbr_t 关于 u32SrcFrmRate 和 fr32DstFrmRate 的说明。

➢ 相关数据类型及接口

MI_VENC_CreateChn

2.57. MI_VENC_AttrH265FixQp_t

➢ 说明

定义 H.265 编码通道 Fixqp 属性结构。

➢ 定义

```
typedef struct MI_VENC_AttrH265FixQp_s
{
  MI_U32   u32Gop;
  MI_U32   u32SrcFrmRateNum;
  MI_U32   u32SrcFrmRateDen;
  MI_U32   u32IQp;
  MI_U32   u32PQp;
}MI_VENC_AttrH265FixQp_t;
```

➢ 成员

成员名称	描述
u32Gop	H.265gop 值。取值范围：[1,65536]。
u32SrcFrmRateNum	编码器帧率分子，以整数为单位。
u32SrcFrmRateDen	编码器帧率分母，以整数为单位。

成员名称	描述
u32IQp	I 帧所有宏块 Qp 值。取 值范围：[0,51]。
u32PQp	P 帧所有宏块 Qp 值。取 值范围：[0,51]。

※ 注意事项

请参见 MI_VENC_AttrH264Cbr_t 关于 u32SrcFrmRate 和 fr32DstFrmRate 的说明。

➢ 相关数据类型及接口

MI_VENC_CreateChn

2.58. MI_VENC_AttrH265Avbr_t

➢ 说明

定义 H.265 编码通道 AVBR 属性结构。

➢ 定义

```
typedef struct MI_VENC_AttrH265Avbr_s
{
    MI_U32 u32Gop;
    MI_U32 u32StatTime;
    MI_U32 u32SrcFrmRateNum;
    MI_U32 u32SrcFrmRateDen;
    MI_U32 u32MaxBitRate;
    MI_U32 u32MaxQp;
    MI_U32 u32MinQp;
} MI_VENC_AttrH265Avbr_t;
```

➢ 成员

成员名称	描述
u32Gop	gop 值。取值范围： [1,65536]。
u32StatTime	码率统计时间，以秒为单位。取值范围： [1,60]。
u32SrcFrmRateNum	编码器帧率分子，以整数为单位。
u32SrcFrmRateDen	编码器帧率分母，以整数为单位。
u32MaxBitRate	编码器输出最大码率，以 kbps 为单位。取值范围： [2,102400]。
u32MaxQp	编码器支持图像最大 QP。取值 范围：(u32MinQp,48]。
u32MinQp	编码器支持图像最小 QP。取 值范围：[12,48]。

※ 注意事项

请参见 MI_VENC_AttrH264Cbr_t 关于 u32SrcFrmRateNum 和 u32SrcFrmRateDen 的说明。

➢ 相关数据类型及接口

MI_VENC_CreateChn

2.59. MI_VENC_SuperFrmMode_e

➢ 说明

定义码率控制中超大帧处理模式。

➢ 定义

```
typedef enum
{
  E_MI_VENC_SUPERFRM_NONE,
  E_MI_VENC_SUPERFRM_DISCARD,
  E_MI_VENC_SUPERFRM_REENCODE,
  E_MI_VENC_SUPERFRM_MAX
}MI_VENC_SuperFrmMode_e;
```

➢ 成员

成员名称	描述
E_MI_VENC_SUPERFRM_NONE	无特殊策略。
E_MI_VENC_SUPERFRM_DISCARD	丢弃超大帧。
E_MI_VENC_SUPERFRM_REENCODE	重编超大帧。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_GetRcParam

2.60. MI_VENC_ParamH264Vbr_t

➢ 说明

定义 H264 协议编码通道 VBR 码率控制模式高级参数配置。

➢ 定义

```
typedef struct MI_VENC_ParamH264Vbr_s
{
  MI_S32   s32IPQPDelta;
  MI_S32   s32ChangePos;
  MI_U32   u32MaxIQp;
  MI_U32   u32MinIQp;
  MI_U32   u32MaxIPProp;
}MI_VENC_ParamH264Vbr_t;
```

➢ 成员

成员名称	描述
s32IPQPDelta	IPQP 变化值。取值范围：[-12,12]。
s32ChangePos	VBR 开始调整 Qp 时的码率相对于最大码率的比例。取值范围：[50,100]。
u32MaxIQp	I 帧的最大 QP。用于控制 I 帧的最小 bits 数。取值范围：(MinQp,MaxQp]。
u32MinIQp	I 帧的最小 QP。用于控制 I 帧的最大 bits 数。取值范围：[MinQp,MaxQp)。
u32MaxIPProp	最大 IP 帧码率的比值，取值范围[5,100]。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_GetRcParam
MI_VENC_SetRcParam

2.61. MI_VENC_ParamH264Avbr_t

➢ 说明

定义 H264 协议编码通道 AVBR 码率控制模式高级参数配置。

- 定义

```
typedef struct MI_VENC_ParamH264Avbr_s
{
    MI_S32 s32IPQPDelta;
    MI_S32 s32ChangePos;
    MI_U32 u32MinIQp;
    MI_U32 u32MaxIPProp;
    MI_U32 u32MaxIQp;
    MI_U32 u32MaxISize;
    MI_U32 u32MaxPSize;
    MI_U32 u32MinStillPercent;
    MI_U32 u32MaxStillQp;
    MI_U32 u32MotionSensitivity;
} MI_VENC_ParamH264Avbr_t;
```

- 成员

成员名称	描述
s32IPQPDelta	IPQP 变化值。取值范围：[-12,12]。 默认值：0
s32ChangePos	AVBR 开始调整 Qp 时的码率相对于最大码率的比例。取值范围：[50,100]。 默认值：80
u32MinIQp	I 帧的最小 QP。用于控制 I 帧的最大 bits 数。取值范围：[MinQp,MaxQp)。 默认值：12
u32MaxIPProp	最大 IP 帧码率的比值，取值范围[5,100]。
u32MaxIQp	I 帧的最大 QP。用于控制 I 帧的最小 bits 数。取值范围：(MinQp,MaxQp]。 默认值：48
u32MaxISize	I 帧的最大 size。rate control 会尽量让 target size 不超过 max size。若为 0，则代表未加限制。 取值范围：[0, 800*1024] 默认值：0
u32MaxPSize	P 帧的最大 size。rate control 会尽量让 target size 不超过 max size。若为 0，则代表未加限制。 取值范围：[0, 800*1024] 默认值：0
u32MinStillPercent	静止状态下目标码率的最小百分比。若此变量设置为 100，AVBR 将不会在判别为静止时主动调低目标码率，AVBR 的表现将和 VBR 一致。 取值范围：[5,100] 默认值：25

成员名称	描述
u32MaxStillQp	静止状态最大 I 帧 QP。 取值范围：[MinIQp,MaxIQp] 暂不支持
u32MotionSensitivity	根据画面运动程度调整码率的灵敏度 取值范围：[0,100] 暂不支持

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_GetRcParam
MI_VENC_SetRcParam

2.62. MI_VENC_ParamMjpegCbr_t

➢ 说明

定义 MJPEG 协议编码通道 CBR 码率控制模式高级参数配置。

➢ 定义

```
typedef struct MI_VENC_ParamMjpegCbr_s
{
   MI_U32   u32MaxQfactor;
   MI_U32   u32MinQfactor;
} MI_VENC_ParamMjpegCbr_t;
```

➢ 成员

成员名称	描述
u32MaxQfactor	帧最大 Qfactor。用于控制图像质量。 取值范围：(u32MinQfactor,90]。
u32MinQfactor	帧最小 Qfactor。用于控制图像质量。 取值范围：[10, u32MaxQfactor)。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_GetRcParam
MI_VENC_SetRcParam

2.63. MI_VENC_ParamH264Cbr_t

➢ 说明

定义 H264 协议编码通道 CBR 新版码率控制模式高级参数配置。

➢ 定义

```
typedef struct MI_VENC_ParamH264Cbr_s
{
  MI_U32   u32MaxQp;
  MI_U32   u32MinQp;
  MI_S32   s32IPQPDelta;
  MI_U32   u32MaxIQp;
  MI_U32   u32MinIQp;
  MI_U32 u32MaxIPProp;
}MI_VENC_ParamH264Cbr_t;
```

➢ 成员

成员名称	描述
u32MaxQp	帧最大 QP，用于钳位质量。取值范围：(u32MinQp,51]。
u32MinQp	帧最小 QP，用于钳位码率波动。取值范围：[0,u32MaxQp)。
s32IPQPDelta	IPQP 变化值。取值范围：[-12,12]。
u32MaxIQp	I 帧的最大 QP。用于控制 I 帧的最小 bits 数。取值范围：(u32MinIQp,51]。
u32MinIQp	I 帧的最小 QP。用于控制 I 帧的最大 bits 数。取值范围：[0,u32MaxIQp)。
u32MaxIPProp	最大 IP 帧码率的比值，取值范围[5,100]。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_GetRcParam

MI_VENC_SetRcParam

2.64. MI_VENC_ParamH265Vbr_t

➢ 说明

定义 H265 协议编码通道 VBR 码率控制模式高级参数配置。

➢ 定义

```
typedef struct MI_VENC_ParamH265Vbr_s
{
  MI_S32   s32IPQPDelta;
  MI_S32   s32ChangePos;
  MI_U32   u32MaxIQp;
  MI_U32   u32MinIQP;
  MI_U32   u32MaxIPProp;
}MI_VENC_ParamH265Vbr_t;
```

➢ 成员

成员名称	描述
s32IPQPDelta	IPQP 变化值。取值范围：[-12,12]。
s32ChangePos	VBR 开始调整 Qp 时的码率相对于最大码率的比例。取值范围：[50,100]。
u32MaxIQp	I 帧的最大 QP。用于控制 I 帧的最小 bits 数。取值范围：(MinQp,MaxQp]。
u32MinIQP	I 帧的最小 QP。用于控制 I 帧的最大 bits 数。取值范围：[MinQp,MaxQp)。
u32MaxIPProp	最大 IP 帧码率的比值，取值范围[5,100]。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_GetRcParam
MI_VENC_SetRcParam

2.65. MI_VENC_ParamH265Avbr_t

➢ 说明

定义 H265 协议编码通道 AVBR 码率控制模式高级参数配置。

- 定义

```
typedef struct MI_VENC_ParamH265Avbr_s
{
    MI_S32 s32IPQPDelta;
    MI_S32 s32ChangePos;
    MI_U32 u32MinIQp;
    MI_U32 u32MaxIPProp;
    MI_U32 u32MaxIQp;
    MI_U32 u32MaxISize;
    MI_U32 u32MaxPSize;
    MI_U32 u32MinStillPercent;
    MI_U32 u32MaxStillQp;
    MI_U32 u32MotionSensitivity;
} MI_VENC_ParamH265Avbr_t;
```

- 成员

成员名称	描述
s32IPQPDelta	IPQP 变化值。取值范围：[-12,12]。 默认值：0
s32ChangePos	AVBR 开始调整 Qp 时的码率相对于最大码率的比例。取值范围：[50,100]。 默认值：80
u32MinIQp	I 帧的最小 QP。用于控制 I 帧的最大 bits 数。取值范围：[MinQp,MaxQp)。 默认值：12
u32MaxIPProp	最大 IP 帧码率的比值，取值范围[5,100]。
u32MaxIQp	I 帧的最大 QP。用于控制 I 帧的最小 bits 数。取值范围：(MinQp,MaxQp]。 默认值：48
u32MaxISize	I 帧的最大 size。rate control 会尽量让 target size 不超过 max size。若为 0，则代表未加限制。 取值范围：[0, 800*1024] 默认值：0
u32MaxPSize	P 帧的最大 size。rate control 会尽量让 target size 不超过 max size。若为 0，则代表未加限制。 取值范围：[0, 800*1024] 默认值：0
u32MinStillPercent	静止状态下最小码率相对于调节阈值码率的百分比。若此变量设置为 100，AVBR 将不会在判别为静止时主动调低目标码率，AVBR 的表现将和 VBR 一致。 取值范围：[5,100] 默认值：25

成员名称	描述
u32MaxStillQp	静止状态最大 I 帧 QP。 取值范围：[MinIQp,MaxIQp] 暂不支持
u32MotionSensitivity	根据画面运动程度调整码率的灵敏度 取值范围：[0,100] 暂不支持

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_GetRcParam
MI_VENC_SetRcParam

2.66. MI_VENC_ParamH265Cbr_t

➢ 说明

定义 H265 协议编码通道 CBR 新版码率控制模式高级参数配置。

➢ 定义

```
typedef struct MI_VENC_ParamH265Cbr_s
{
  MI_U32   u32MaxQp;
  MI_U32   u32MinQp;
  MI_S32   s32IPQPDelta;
  MI_U32   u32MaxIQp;
  MI_U32   u32MinIQp;
  MI_U32   u32MaxIPProp;
}MI_VENC_ParamH265Cbr_t;
```

➢ 成员

成员名称	描述
u32MaxQp	帧最大 QP，用于钳位质量。取 值范围：(u32MinQp,51]。
u32MinQp	帧最小 QP，用于钳位码率波动。取 值范围：[0,u32MaxQp)。
s32IPQPDelta	IPQP 变化值。取值范围： [-12,12]。
u32MaxIQp	I 帧的最大 QP。用于控制 I 帧的最小 bits 数。取值范 围：(u32MinIQp,51]。
u32MinIQp	I 帧的最小 QP。用于控制 I 帧的最大 bits 数。取值范

成员名称	描述
	围：[0,u32MaxIQp)。
u32MaxIPProp	最大 IP 帧码率的比值，取值范围[5,100]。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_GetRcParam

MI_VENC_SetRcParam

2.67. MI_VENC_RcParam_t

➢ 说明

定义编码通道的码率控制高级参数。

➢ 定义

```
typedef struct MI_VENC_RcParam_s
{
   MI_U32   u32ThrdI[RC_TEXTURE_THR_SIZE];
   MI_U32   u32ThrdP[RC_TEXTURE_THR_SIZE];
   MI_U32   u32RowQpDelta;
   union
   {
      MI_VENC_ParamH264Cbr_t       stParamH264Cbr;
      MI_VENC_ParamH264Vbr_t       stParamH264VBR;
      MI_VENC_ParamH264Avbr_t stParamH264Avbr;
      MI_VENC_ParamMjpegCbr_t stParamMjpegCbr;
      MI_VENC_ParamH265Cbr_t       stParamH265Cbr;
      MI_VENC_ParamH265Vbr_t stParamH265Vbr;
      MI_VENC_ParamH265Avbr_t stParamH265Avbr;
   };
   MI_VOID*pRcParam;
}MI_VENC_RcParam_t;
```

➢ 成员

成员名称	描述
u32ThrdI	I 帧宏块级码率控制的 mad 门限。取值范围：[0,255]。
u32ThrdP	P 帧宏块级码率控制的 mad 门限。取值范围：[0,255]。
u32RowQpDelta	行级码率控制。取值范

成员名称	描述
	围：[0,10]。
stParamH264Cbr	H.264 通道 CBR（ConstantBitRate）码率控制模式高级参数。
stParamH264Vbr	H.264 通道 VBR（VariableBitRate）码率控制模式高级参数。
stParamH264Avbr	H.264 通道 AVBR（AdaptiveVariableBitRate）码率控制模式高级参数。
stParamH265Cbr	H.265 通道 CBR（ConstantBitRate）码率控制模式高级参数。
stParamH265Vbr	H.265 通道 VBR（VariableBitRate）码率控制模式高级参数。
stParamH265Avbr	H.265 通道 AVBR（AdaptiveVariableBitRate）码率控制模式高级参数。
pRcParam	保留参数，目前没有使用到。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetRcParam

MI_VENC_GetRcParam

2.68. MI_VENC_CropCfg_t

➢ 说明

定义通道截取（Crop）参数。

➢ 定义

```
typedef struct MI_VENC_CropCfg_s
{
    MI_BOOL bEnable; /* Crop region enable */
    MI_VENC_Rect_t stRect; /* Crop region, note: s32X must be multi of 16 */
} MI_VENC_CropCfg_t;
```

➢ 成员

成员名称	描述
bEnable	是否进行裁剪。 TRUE：使能裁剪。 FALSE：不使能裁剪。
stRect	裁剪的区域。 stRect. u32Left ：H.264 必须 16 像素对齐，H.265 必

成员名称	描述
	须 32 像素对齐。 stRect. u32Top ：H.264/H.265 皆必须 2 像素对齐。 stRect.u32Width，s32Rect.u32Height：H.264 必须 16 像素对齐，H.265 必须 32 像素对齐。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetCrop
MI_VENC_GetCrop

2.69. MI_VENC_RecvPicParam_t

➢ 说明

接收指定帧数图像编码。

➢ 定义

```
typedef struct MI_VENC_RecvPicParam_s
{
    MI_S32 s32RecvPicNum;
}MI_VENC_RecvPicParam_t;
```

➢ 成员

成员名称	描述
s32RecvPicNum	编码通道连续接收并编码的帧数。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_StartRecvPicEx

2.70. MI_VENC_H264eIdrPicIdMode_e

➢ 说明

设置 IDR 帧或 I 帧的 idr_pic_id 的模式。

➢ 定义

```
typedef enum
{
    E_MI_VENC_H264E_IDR_PIC_ID_MODE_USR,
}MI_VENC_H264eIdrPicIdMode_e;
```

➢ 成员

成员名称	描述
E_MI_VENC_H264E_IDR_PIC_ID_MODE_USR	用户模式；即 idr_pic_id 由用户来设置。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH264IdrPicId

2.71. MI_VENC_H264IdrPicIdCfg_t

➢ 说明

IDR 帧或 I 帧的 idr_pic_id 参数。

➢ 定义

```
typedef struct MI_VENC_H264IdrPicIdCfg_s
{
   MI_VENC_H264eIdrPicIdMode_e   eH264eIdrPicIdMode;
   MI_U32   u32H264eIdrPicId;
}MI_VENC_H264IdrPicIdCfg_t;
```

➢ 成员

成员名称	描述
enH264eIdrPicIdMode	设置 idr_pic_id 的模式。
u32H264eIdrPicId	idr_pic_id 的值。取值范围为：[0,65535]。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetH264IdrPicId

2.72. MI_VENC_FrameLostMode_e

➢ 说明

瞬时码率超过阈值时的丢帧模式。

➢ 定义

```
typedef enum
{
  E_MI_VENC_FRMLOST_NORMAL,
  E_MI_VENC_FRMLOST_PSKIP,
  E_MI_VENC_FRMLOST_MAX,
}MI_VENC_FrameLostMode_e;
```

➢ 成员

成员名称	描述
E_MI_VENC_FRMLOST_NORMAL	瞬时码率超过阈值时正常丢帧。
E_MI_VENC_FRMLOST_PSKIP	瞬时码率超过阈值时编码 pskip 帧。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetFrameLostStrategy
MI_VENC_GetFrameLostStrategy

2.73. MI_VENC_ParamFrameLost_t

➢ 说明

瞬时码率超过阈值时的丢帧策略参数。

➢ 定义

```
typedef struct MI_VENC_ParamFrameLost_s
{
    MI_BOOL bFrmLostOpen;
    MI_U32 u32FrmLostBpsThr;
    MI_VENC_FrameLostMode_e eFrmLostMode;
    MI_U32 u32EncFrmGaps;
} MI_VENC_ParamFrameLost_t;
```

➢ 成员

成员名称	描述
bFrmLostOpen	瞬时码率超过阈值时丢帧开关。
u32FrmLostBpsThr	丢帧阈值。（单位为 bit/s）
enFrmLostMode	瞬时码率超过阈值时丢帧模式。
u32EncFrmGaps	丢帧间隔，默认为 0。取 值范围：[0,65535]

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetFrameLostStrategy

2.74. MI_VENC_SuperFrameCfg_t

➢ 说明

超大帧处理策略参数。

➢ 定义

```
typedef struct MI_VENC_SuperFrameCfg_s
{
  MI_VENC_SuperFrmMode_e  eSuperFrmMode;
  MI_U32   u32SuperIFrmBitsThr;
  MI_U32   u32SuperPFrmBitsThr;
  MI_U32   u32SuperBFrmBitsThr;
}MI_VENC_SuperFrameCfg_t;
```

➢ 成员

成员名称	描述
eSuperFrmMode	超大帧处理模式。
u32SuperIFrmBitsThr	I 帧超大阈值，默认为 500000。取值范围：大于等于 0
u32SuperPFrmBitsThr	P 帧超大阈值，默认为 500000。取值范围：大于等于 0
u32SuperBFrmBitsThr	B 帧超大阈值，默认为 500000。取值范围：大于等于 0

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetSuperFrameCfg

2.75. MI_VENC_RcPriority_e

➢ 说明

码率控制优先级枚举。

➢ 定义

```
typedef enum
{
  E_MI_VENC_RC_PRIORITY_BITRATE_FIRST=1,
  E_MI_VENC_RC_PRIORITY_FRAMEBITS_FIRST,
  E_MI_VENC_RC_PRIORITY_MAX,
}MI_VENC_RcPriority_e;
```

➢ 成员

成员名称	描述
E_MI_VENC_RC_PRIORITY_BITRATE_FIRST	目标码率优先。
E_MI_VENC_RC_PRIORITY_FRAMEBITS_FIRST	超大帧阈值优先。

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.76. MI_VENC_ModParam_t

➢ 说明

编码相关模块参数。

➢ 定义

```
typedef struct MI_VENC_ModParam_s
{
    MI_VENC_ModType_e eVencModType;
    union
    {
        //MI_VENC_ParamModVenc_t stVencModParam; //not defined yet
        //MI_VENC_ParamModH264e_t stH264eModParam; //not defined yet
        MI_VENC_ParamModH265e_t stH265eModParam;
        MI_VENC_ParamModJpege_t stJpegeModParam;
    };
} MI_VENC_ModParam_t;
```

➢ 成员

成员名称	描述
enVencModType	设置或者获取模块参数的类型。
stH265eModParam/ stJpegeModParam	mi_h265.ko/ mi_jpege.ko 模块参数结构。

※ 注意事项

无。

➢ 相关数据类型及接口

无

2.77. MI_VENC_ModType_e

➢ 说明

编码相关模块参数类型。

➢ 定义

```
typedef enum
{
  E_MI_VENC_MODTYPE_VENC=1,
  E_MI_VENC_MODTYPE_H264E,
  E_MI_VENC_MODTYPE_H265E,
  E_MI_VENC_MODTYPE_JPEGE,
  E_MI_VENC_MODTYPE_MAX
}MI_VENC_ModType_e;
```

➢ 成员

成员名称	描述
E_MI_VENC_MODTYPE_VENC	mi_venc.ko 模块参数类型
E_MI_VENC_MODTYPE_H264E	mi_h264e.ko 模块参数类型
E_MI_VENC_MODTYPE_H265E	mi_h265e.ko 模块参数类型
E_MI_VENC_MODTYPE_JPEGE	mi_jpege.ko 模块参数类型

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_ModParam_t

2.78. MI_VENC_ParamModH265e_t

➢ 说明

mi_h265.ko 模块参数。

➢ 定义

```
typedef struct MI_VENC_ParamModH265e_s
{
    MI_U32   u32OneStreamBuffer;
    MI_U32   u32H265eMiniBufMode;
}MI_VENC_ParamModH265e_t;
```

➢ 成员

成员名称	描述
u32OneStreamBuffer	mi_h265.ko 模块中 u32OneStreamBuffer 参数。 0：多包模式 1：单包模式
u32H265eMiniBufMode	mi_h265.ko 模块中 u32H265eMiniBufMode 参数。 0：码流 buffer 根据分辨率分配 1：码流 buffer 下限为 32k，用户保证合理

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_ModParam_t

2.79. MI_VENC_ParamModJpege_t

➢ 说明

mi_jpeg.ko 模块中参数。

➢ 定义

```
typedef struct MI_VENC_ParamModJpege_s
{
    MI_U32   u32OneStreamBuffer;
    MI_U32   u32JpegeMiniBufMode;
}MI_VENC_ParamModJpege_t;
```

➢ 成员

成员名称	描述
u32OneStreamBuffer	mi_jpeg.ko 模块中 u32OneStreamBuffer 参数。 0：多包模式 1：单包模式
u32JpegeMiniBufMode	mi_jpeg.ko 模块中 u32JpegeMiniBufMode 参数。 0：码流 buffer 根据分辨率分配 1：码流 buffer 下限为 32k，用户保证合理

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_ModParam_t

2.80. MI_VENC_AdvCustRcAttr_t

➢ 说明

自定义的高级码控相关功能的开关参数。

➢ 定义

```
typedef struct MI_VENC_AdvCustRcAttr_s
{
    MI_BOOL bEnableQPMap;
    MI_BOOL bAbsQP;
    MI_BOOL bEnableModeMap;
    MI_BOOL bEnabelHistoStaticInfo;
}MI_VENC_AdvCustRcAttr_t;
```

➢ 成员

成员名称	描述
bEnableQPMap	使能 QPMap 功能的开关。 TRUE：开。 FALSE：关。
bAbsQP	配置 QPMap 是否使用绝对 QP 值。 TRUE：是。 FALSE：否，使用相对 QP 值。
bEnableModeMap	使能 ModeMap 功能的开关。 TRUE：开。 FALSE：关。
bEnabelHistoStaticInfo	使能获取帧编码后相关统计数据的开关。 TRUE：开。 FALSE：关。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetAdvCustRcAttr

2.81. MI_VENC_FrameHistoStaticInfo_t

➢ 说明

图像帧编码的相关配置属性信息。

➢ 定义

```
typedef struct MI_VENC_FrameHistoStaticInfo_s
{
    MI_U8    u8PicSkip;
    MI_U16   u16PicType;
    MI_U32   u32PicPoc;
    MI_U32   u32PicSliNum;
    MI_U32   u32PicNumIntra;
    MI_U32   u32PicNumMerge;
    MI_U32   u32PicNumSkip;
    MI_U32   u32PicAvgCtuQp;
    MI_U32   u32PicByte;
    MI_U32   u32GopPicIdx;
    MI_U32   u32PicNum;
    MI_U32   u32PicDistLow;
    MI_U32   u32PicDistHigh;
} MI_VENC_FrameHistoStaticInfo_t;
```

➢ 成员

成员名称	描述
u8PicSkip	图像帧 skip flag。
u16PicType	编码帧的类型： 0：I 1：P 2：B
u32PicPoc	当前编码帧的图片顺序计数（POC）。
u32PicSliNum	片段（Slice Segment）总数。
u32PicNumIntra	当前编码帧的 Intra 块的数量（(8x8 unit）。
u32PicNumMerge	当前编码帧的 Merge 块的数量（(8x8 unit）。
u32PicNumSkip	当前编码帧的 Skip 块的数量（(8x8 unit）。
u32PicAvgCtuQp	当前编码帧所有 CTU 的平均 QP 值。
u32PicByte	当前编码图像帧的大小（bytes）。
u32GopPicIdx	GOP 中该图像帧的 index。
u32PicNum	当前已经编码的图片帧数。
u32PicDistLow	SSD 的低 32bit。
u32PicDistHigh	SSD 的高 32bit。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_GetLastHistoStaticInfo

2.82. MI_VENC_InputSourceConfig_t

➢ 说明

输入配置参数。

➢ 定义

```
typedef struct MI_VENC_InputSourceConfig_s
{
    MI_VENC_InputSrcBufferMode_e eInputSrcBufferMode;
}MI_VENC_InputSourceConfig_t;
```

➢ 成员

成员名称	描述
eInputSrcBufferMode	输入模式。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_InputSrcBufferMode_e,MI_VENC_SetInputSourceConfig

2.83. MI_VENC_InputSrcBufferMode_e

➢ 说明

定义 H.264 和 H.265 输入 buffer 模式枚举。

➢ 定义

```
typedef enum
{
    E_MI_VENC_INPUT_MODE_NORMAL_FRMBASE = 0, /*Handshake with input by about 3 buffers
                                                in frame mode*/
    E_MI_VENC_INPUT_MODE_RING_ONE_FRM, /*Handshake with input by one buffer in ring
                                         mode*/
    E_MI_VENC_INPUT_MODE_RING_HALF_FRM, /*Handshake with input by half buffer in ring
                                          mode*/
    E_MI_VENC_INPUT_MODE_MAX
} MI_VENC_InputSrcBufferMode_e;
```

➢ 成员

成员名称	描述
E_MI_VENC_INPUT_MODE_NORMAL_FRMBASE	默认与前级通过三张左右 frame 交互
E_MI_VENC_INPUT_MODE_RING_ONE_FRM	与前级通过一张 frame 以 ring 的形式交互
E_MI_VENC_INPUT_MODE_RING_HALF_FRM	与前级通过半张 frame 以 ring 的形式交互

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_InputSourceConfig_t,MI_VENC_SetInputSourceConfig

2.84. VENC_MAX_SAD_RANGE_NUM

➢ 说明

定义智能检测相关之统计 SAD 值落入区间范围的最大区间个数。

➢ 定义

```
#define   VENC_MAX_SAD_RANGE_NUM        16
```

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.85. MI_VENC_SmartDetType_e

➢ 说明

定义智能侦测类型。

➢ 定义

```
typedef enum
{
  E_MI_VENC_MD_DET=1,
  E_MI_VENC_ROI_DET,
  E_MI_VENC_SMART_DET_MAX,
} MI_VENC_SmartDetType_e;
```

➢ 成员

成员名称	描述
E_MI_VENC_MD_DET	运动侦测类型
E_MI_VENC_ROI_DET	ROI 侦测类型

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.86. MI_VENC_MdInfo_t

➢ 说明

定义运动侦测相关统计信息。

➢ 定义

```
typedef struct MI_VENC_MdInfo_s
{
  MI_U8     u8SadRangeRatio[VENC_MAX_SAD_RANGE_NUM];
} MI_VENC_MdInfo_t;
```

➢ 成员

成员名称	描述
u8SadRangeRatio	每一帧中所有 MB 之 SAD 落入每个区间的占比。

※ 注意事项

SAD 值的取值范围为[0, 255]，所有 SAD 值被分为 16 个区间：

[0,10)，[10,15)，[15,20)，[20,25)，[25,30)，[30,40)，[40,50)，[50,60)，[60,70)，[70,80)， [80,90)，90,100)，[100,120)，[120,140)，[140,160)，[160, 255]。

此处统计一帧中所有 MB(8*8)落入以上区间的占总的 MB 数量的比例。

➢ 相关数据类型及接口

VENC_MAX_SAD_RANGE_NUM

2.87. MI_VENC_SmartDetInfo_t

➢ 说明

定义智能侦测算法所需的统计信息。

➢ 定义

```
typedef struct MI_VENC_SmartDetInfo_s
{
    MI_VENC_SmartDetType_e eSmartDetType;
    union
    {
      MI_VENC_MdInfo_t   stMdInfo;
      MI_BOOL            bRoiExist;
    };
    MI_U8                u8ProtectFrmNum;
} MI_VENC_SmartDetInfo_t;
```

➢ 成员

成员名称	描述
eSmartDetType	智能侦测类型
stMdInfo	运动侦测统计信息
bRoiExist	ROI 区域是否存在信息 MI_TRUE: 当前帧有 ROI 存在; MI_FALSE: 当前帧无 ROI 存在;
u8ProtectFrmNum	保护帧数

※ 注意事项

无。

2.88. MI_VENC_IntraRefresh_t

➢ 说明

支持 P 帧刷 Islice 的控制参数。

➢ 定义

```
typedef struct MI_VENC_IntraRefresh_s
{
    MI_BOOL bEnable;
    MI_U32 u32RefreshLineNum;
    MI_U32 u32ReqIQp;
}MI_VENC_IntraRefresh_t;
```

➢ 成员

成员名称	描述
bEnable	Enable venc Intra Refresh 模式
u32RefreshLineNum	每个 gop 序列里面需要刷新的 Islice 总数
u32ReqIQp	是不是要强制编码 I 帧

※ 注意事项

只支持 H265,H264.

➢ 相关数据类型及接口

MI_VENC_SetIntraRefresh

2.89. MI_VENC_InitParam_t

➢ 说明

Venc 设备初始化参数。

➢ 定义

```
typedef struct MI_VENC_InitParam_s
{
    MI_U32 u32MaxWidth;
    MI_U32 u32MaxHeight;
}MI_VENC_InitParam_t;
```

➢ 成员

成员名称	描述
u32MaxWidth	设备使用的最大分辨率的宽度
u32MaxHeight	设备使用的最大分辨率的高度

※ 注意事项

- SSC33X 系列根据实际需要设置最大宽高。

➢ 相关数据类型及接口

MI_VENC_InitDev

3. 错误码

视频编码 API 错误码如表 3-1 所示。

表 3-1 视频编码 API 错误码

错误代码	宏定义	描述
0xa0022000	MI_VENC_OK	success
0xa0022001	MI_ERR_VENC_INVALID_DEVID	invalid device ID
0xa0022002	MI_ERR_VENC_INVALID_CHNID	invalid channel ID
0xa0022003	MI_ERR_VENC_ILLEGAL_PARAM	at lease one parameter is illegal
0xa0022004	MI_ERR_VENC_EXIST	channel exists
0xa0022005	MI_ERR_VENC_UNEXIST	channel unexist
0xa0022006	MI_ERR_VENC_NULL_PTR	using a NULL point
0xa0022007	MI_ERR_VENC_NOT_CONFIG	try to enable or initialize device or channel, before configuring attribute
0xa0022008	MI_ERR_VENC_NOT_SUPPORT	operation is not supported by NOW
0xa0022009	MI_ERR_VENC_NOT_PERM	operation is not permitted
0xa002200C	MI_ERR_VENC_NOMEM	failure caused by malloc memory
0xa002200D	MI_ERR_VENC_NOBUF	failure caused by malloc buffer
0xa002200E	MI_ERR_VENC_BUF_EMPTY	no data in buffer
0xa002200F	MI_ERR_VENC_BUF_FULL	no buffer for new data
0xa0022010	MI_ERR_VENC_NOTREADY	System is not ready
0xa0022011	MI_ERR_VENC_BADADDR	bad address
0xa0022012	MI_ERR_VENC_BUSY	resource is busy
0xa0022013	MI_ERR_VENC_CHN_NOT_STARTED	channel not start
0xa0022014	MI_ERR_VENC_CHN_NOT_STOPPED	channel not stop
0xa002201F	MI_ERR_VENC_UNDEFINED	unexpected error