MI VENC API

Version 2.07

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

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、视频编码通道的 reset、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。

1.2. 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
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

API 名	功能
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
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

API 名	功能
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_GetRcPriority	获取码率控制的优先级类型

1.2.1 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

※ **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

➢ **举例**

无。

➢ **相关主题**

无。

1.2.2 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
- MI_ERR_VENC_NOT_PERM。
- 可以设置 mi_venc.ko、mi_h264e.ko、mi_h265e.ko、mi_jpge.ko 模块的参数。

➢ 举例

无。

➢ 相关主题

无。

1.2.3 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。

➢ 举例

无。

➢ 相关主题

无。

1.2.4 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

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_comm_rc.h、mi_venc.h
- 库文件：libmi.a

※ 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：

  MaxPicWidthe∈[MIN_WIDTH,MAX_WIDTH]

  MaxPicHeighte∈[MIN_HEIGHT,MAX_HEIGHT]

  PicWidthe∈[MIN_WIDTH,MaxPicwidth]

  PicHeighte∈[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）。
- 编码器属性必须设置编码码流 buffer 深度，获取码流方式等，表 6-5 详细描述了各种协议的各项属性的特性。注：计算 H.264/MJPEG/JPEG 的 buffer 深度时 MaxPicwidth 和 MaxPicheight 需 16 对齐后进行计算，计算 H.265 的 buffer 深度时 MaxPicwidth 和 MaxPicheight 需 64 对齐后进行计算。

表 6-5 编码器属性的约束

编码协议	编码方式	码流 buffer 深度	获取码流模式	编码 profile
H.264	Frame	当 H264eMiniBufMode=0 时，Buffer≥MaxPicwidthx MaxPicheightx3/4；当 H264eMiniBufMode=0 时，Buffer≥32*1024byte；	Frame/ Slice	Baseline Mainprofile Highprofile
JPEG	Frame	JpegeMiniBufMode=0 时，Buffer ≥ MaxPicwidthxMaxPiche ightx1；JpegeMiniBufMode=0 时，Buffer≥32*1024byte；	Frame/Ecs	Baseline
MPEG-4	Frame	Buffer ≥ MaxPicwidthxMaxPich eightx3/4	Frame/Pa cket	Simpleprofile
H.265	Frame	H265eMiniBufMode=0 时，Buffer ≥ MaxPicwidthxMaxPiche ightx3/4；H265eMiniBufMode=1 时，	Frame/Slice	Mainprofile

		Buffer≥32*1024byte;		

- 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 :3840x2560(4k*2k)、1280x720(720P)、960x540、640x360、704x576、704x480、352x288、352x240。
- 编码器属性中除通道宽高都是静态属性，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/MPEG-4）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/MPEG-4），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。表 6-6 介绍了三种 RC 模式的公共的属性。

表 6-6 码率控制器属性的约束

RC 模式	GOP	StatTime(s)	FrmRate(SrcFrmRate/DstFrmRate)
CBR	≥1	≥1	SrcFrmRate≥DstFrmRate
VBR	≥1	≥1	SrcFrmRate≥DstFrmRate
FIXQP	≥1	≥1	SrcFrmRate≥DstFrmRate

- SrcFrmRate 应该设置为产生 TimeRef 的实际帧率，RC 需要根据 SrcFrmRate 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 SrcFrmRate 设置为 VI 的实际输出帧率，因为 TimeRef 是在 VI 输出的时候产生。假设 VI 的输出帧率是 30，如果 VENC 不进行帧率控制，SrcFrmRate 应该设置为 30，如果 VENC 进行帧率控制，VENC 帧率控制中的源帧率和 SrcFrmRate 都应该设置为 30。设置目标帧率 DstFrmRate 时，目标帧率类型定义为分数类型 MI_FR32 类型。MI_FR32 实际为 MI_U32 型，高 16 位用于表示分母，低 16 位表示分子。如果用户设置目标帧率为整数，设置分母高 16 位为 0 即可表示整数。例如设置 SrcFrmRate 等于 25，DstFrmRate 等于 12，则表示将从 25 帧输入图像中取出 12 帧进行编码，其余 13 帧将丢掉。如果设置 SrFrmRate 等于 25，DstFrmRate 等于 15/2，则表示要求编码器 2 秒钟编码 15 帧。
- DstFrmRate 的设置方式如下：如果要设置为整数帧率 25，则可采用以下方式 DstFrmRate=25；如果要设置为分数帧率 15/2，则采用以下方式：DstFrmRate=15+（2<<16）。
- 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表 6-7 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。

表 6-7 典型平均比特率配置

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_VOID)
{
   MI_S32
   s32Ret;VI_CHNViCh
   n=0;MI_VENC_CHN
   VeChn=0;
   MI_MI_VENC_CHN
   Attr_tstAttr;MPP_CHN_SstSrc
   Chn,stDestChn;

   /*seth264chnnelvideoencodeattribute*/stAttr.stVeA
   ttr.enType
                          =PT_H264;stAttr.stVeAttr.st
   AttrH264e.u32PicWidth=u32PicWidth;stAttr.stVeAttr
   .stAttrH264e.u32PicHeight=u32PicHeigh;

   stAttr.stVeAttr.stAttrH264e.u32MaxPicWidth=u32MaxPicWid
   th;stAttr.stVeAttr.stAttrH264e.u32MaxPicHeight=u32MaxPi
   cHeigh;

   stAttr.stVeAttr.stAttrH264e.u32Profile=2;
   ……//omitothervideoencodeassignmentshere.
   /*seth264chnnelratecontrolattribute*/stAttr.stRcAttr.en
   RcMode
                                  =E_MI_VENC_RC_MODE_H264CB
   R;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;
    ……//omitotherratecontrolassignmentshere.
    s32Ret=MI_VENC_CreateChn(VeChn,&stAttr);if
    (MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_CreateChnerr0x%x\n",s32Ret);r
        eturnMI_FAILURE;
    }
    s32Ret=MI_VENC_StartRecvPic(VeChn);if
    (s32Ret!=MI_SUCCESS)
    {
        printf("MI_VENC_StartRecvPicerr0x%x\n",s32Ret);
        returnMI_FAILURE;
    }
    ……//omitothercodehere.
    returnMI_SUCCESS;
}
```

- **相关主题**

  无。

1.2.5 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_VOID)
{
   MI_S32
   s32Ret;MI_VENC_CH
   N VeChn=0;
    s32Ret=MI_VENC_StopRecvPic(VeChn);if
    (s32Ret!=MI_SUCCESS)
    {
        printf("MI_VENC_StopRecvPicerr0x%x\n",s32Ret);r
        eturnMI_FAILURE;
    }
    s32Ret=MI_VENC_DestroyChn(VeChn);
    if(s32Ret!=MI_SUCCESS)
    {
        printf("MI_VENC_DestroyChnerr0x%x\n",s32Ret);
        returnMI_FAILURE;
    }
    returnMI_SUCCESS;
}
```

➢ 相关主题

无。

1.2.6 MI_VENC_ResetChn

➢ 描述

复位通道。

➢ 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

```
MI_S32 ResetVenc(MI_VOID)
{
   MI_S32 s32Ret;
   VI_CHNViChn=0;MI_
   VENC_CHN VeChn=0;
   MI_VENC_CHN
   Attr_tstAttr;MPP_CHN_SstSrc
   Chn,stDestChn;

   /*seth264chnnelvideoencodeattribute*/stAttr.stVeAttr.en
   Type
                          =PT_H264;stAttr.stVeAttr.stAttrH
   264e.u32PicWidth=u32PicWidth;stAttr.stVeAttr.stAttrH264
   e.u32PicHeight=u32PicHeigh;stAttr.stVeAttr.stAttrH264e.
   u32MaxPicWidth=u32MaxPicWidth;stAttr.stVeAttr.stAttrH26
   4e.u32MaxPicHeight=u32MaxPicHeigh;

   stAttr.stVeAttr.stAttrH264e.u32Profile=2;
   ……//omitothervideoencodeassignmentshere.
```

```
    /*seth264chnnelratecontrolattribute*/stAttr.stRcAttr
    .enRcMode
                                    =E_MI_VENC_RC_MODE_H26
    4CBR;stAttr.stRcAttr.stAttrH264Cbr.u32BitRate
                                    =10*1024;stAttr.stRcAt
    tr.stAttrH264Cbr.fr32DstFrmRate            =30;
    stAttr.stRcAttr.stAttrH264Cbr.u32SrcFrmRate=30;
    stAttr.stRcAttr.stAttrH264Cbr.u32Gop       =30;
    stAttr.stRcAttr.stAttrH264Cbr.u32FluctuateLevel=1;
    stAttr.stRcAttr.stAttrH264Cbr.u32StatTime  =1;
    ……//omitotherratecontrolassignmentshere.
    s32Ret=MI_VENC_CreateChn(VeChn,&stAttr);if
    (MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_CreateChnerr0x%x\n",s32Ret);r
        eturnMI_FAILURE;
    }
    s32Ret=MI_VENC_StartRecvPic(VeChn);if
    (s32Ret!=MI_SUCCESS)
    {
        printf("MI_VENC_StartRecvPicerr0x%x\n",s32Ret);
        returnMI_FAILURE;
    }
    s32Ret=MI_VENC_StopRecvPic(VeChn);if
    (s32Ret!=MI_SUCCESS)
    {
        printf("MI_VENC_StopRecvPicerr0x%x\n",s32Ret);r
        eturnMI_FAILURE;
    }
    s32Ret=MI_VENC_ReSetChn(VeChn);if
    (s32Ret!=MI_SUCCESS)
    {
        printf("MI_VENC_ReSetChnerr0x%x\n",s32Ret);r
        eturnMI_FAILURE;
    }
    ……//omitothercodehere.
    returnMI_SUCCESS;
}
```

➢ **相关主题**

无。

1.2.7 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。

1.2.8 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。

- **举例**

```
MI_S32 JpegSnapProcess(MI_VOID)
{
   MI_S32
   s32Ret;MI_VENC_CH
   N VeChn=0;
   MI_VENC_CHN
   Attr_tstAttr;MI_VENC_RecvPicPa
   ram_tstRecvParam;
   /*setjpegchnnelvideoencodeattribute*/stAttr.stVeAttr.e
   nType
                          =PT_JPEG;stAttr.stVeAttr.stAttrJ
   peg.u32PicWidth=u32PicWidth;stAttr.stVeAttr.stAttrJpeg
   .u32PicHeight=u32PicHeigh;stAttr.stVeAttr.stAttrJpeg.u
   32MaxPicWidth=u32MaxPicWidth;stAttr.stVeAttr.stAttrJpe
   g.u32MaxPicHeight=u32MaxPicHeigh;
   ……//omitothervideoencodeassignmentshere.
   //createjpegchannel
    s32Ret=MI_VENC_CreateChn(VeChn,&stAttr);if
    (MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_CreateChnerr0x%x\n",s32Ret);r
        eturnMI_FAILURE;
    }
   //startsnappingstRecvParam.
   s32RecvPicNum=2;
    s32Ret=MI_VENC_StartRecvPicEx(VeChn,&stRecvParam);i
   f(s32Ret!=MI_SUCCESS)
   {
        printf("MI_VENC_StartRecvPicExerr0x%x\n",s32Ret);
        returnMI_FAILURE;
    }
   ……//waituntilallpictureshavebeenencoded.
   s32Ret=MI_VENC_StopRecvPic(VeChn);if
    (s32Ret!=MI_SUCCESS)
   {
        printf("MI_VENC_StopRecvPicerr0x%x\n",s32Ret);r
        eturnMI_FAILURE;
```

```
    }
    //destroyjpegchannel
    s32Ret=MI_VENC_DestroyChn(VeChn);if
    (s32Ret!=MI_SUCCESS)
    {
        printf("MI_VENC_DestroyChnerr0x%x\n",s32Ret);r
        eturnMI_FAILURE;
    }
    returnMI_SUCCESS;
}
```

➢ **相关主题**

无。

1.2.9 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。

1.2.10 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：

在编码通道状态结构体中，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。
在编码通道状态结构体中，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。
在编码通道状态结构体中，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。
在编码通道状态结构体中，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。

- **在按包获取时当前帧可能不是一个完整帧（被取走一部分），按帧获取时表示**当前一个完整帧的包个数（如果没有一帧数据则为 0）。**用户在需要按帧获取码**流时，需要查询一个完整帧的包个数，在这种情况下，通常可以在 select 成功后执行 query 操作，此时 u32CurPacks 是当前完整帧中包的个数。
- **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。

1.2.11 MI_VENC_SetChnAttr

➢ **描述**

设置编码通道属性。

➢ **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，参见表 6-5。
- 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，关闭 OSD，并且清空码流 buffer 和缓存图像队列。
- 重设码流分辨率时 OSD 功能使用推荐以下接口调用顺序：

  1）MI_VENC_StopRecvPic 编码通道停止接收图像；

  2）MI_RGN_DetachFrmChn 把 OSD 区域从编码通道撤出；

  3）MI_VENC_SetChnAttr 重设编码码流分辨率；

  4）MI_RGN_AttachToChn 把 OSD 区域叠加到编码通道上；

  5）MI_VENC_StartRecvPic 开始接收图像编码。

➢ 举例

无。

➢ 相关主题

无。

1.2.12 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。

1.2.13 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。	输出
s32MilliSec	获取码流超时时间。 取值范围[-1,+) •-1：阻塞。 •0：非阻塞。 •大于 0：超时时间。	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

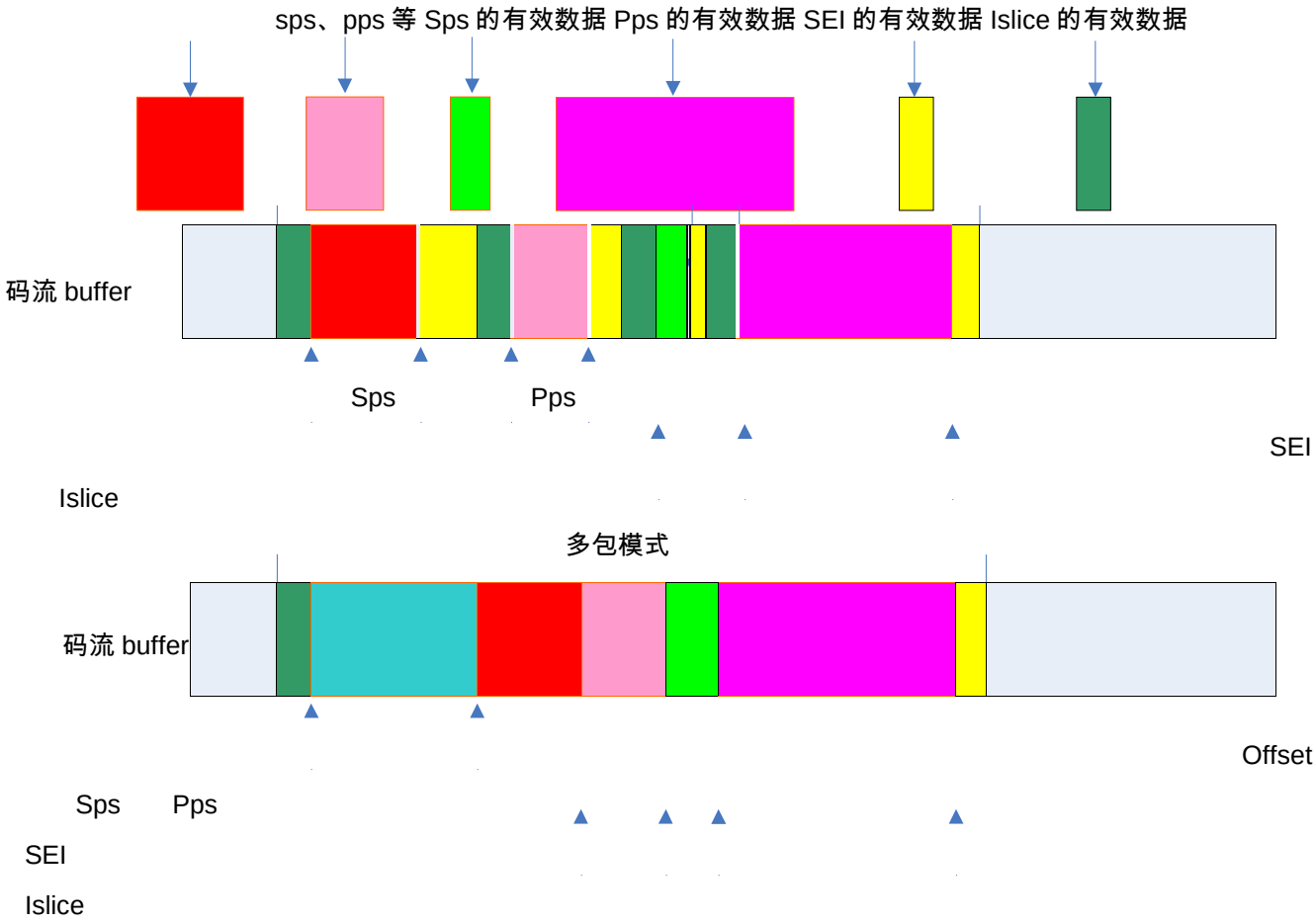
- 如果通道未创建，返回失败。
- 如果 pstStream 为空，返回 MI_ERR_VENC_NULL_PTR。
- 如果 s32MilliSec 小于-1，返回 MI_ERR_VENC_ILLEGAL_PARAM。
- 支持超时方式获取。支持 select/poll 系统调用。
- s32MilliSec=0 时，则为非阻塞获取，即如果缓冲无数据，则返回失败 MI_ERR_VENC_BUF_EMPTY。
  s32MilliSec=-1 时，则为阻塞，即如果缓冲无数据，则会等待有数据时才返回获取成功。
  s32MilliSec>0 时，则为超时，即如果缓冲无数据，则会等待用户设定的超时时间，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。
- 支持按包或按帧方式获取码流。如果按包获取，则：对于 H.264 和 H.265 编码协议，每次获取的是一个 NAL 单元。对于 JPEG 编码协议（包括 JPEG 抓拍和 MJPEG），每次获取的是一个 ECS 或图像参数码流包。对于 MPEG-4 协议编码通道，每次获取的是一个 Packet。
- 码流结构体 MI_VENC_Stream_t 包含 4 个部分 :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。如果是按包获取，则此空间不小于 sizeof(MI_VENC_Pack_t)的大小；如果按帧获取，则此空间不小于 Nsizeof(MI_VENC_Pack_t)的大小，其中 N 代表当前帧之中的包的个数，可以在 select 之后通过查询接口获得。码流包个数 u32PackCount 在输入时，此值指定 pstPack 中 MI_VENC_Pack_t 的个数。按包获取时，u32PackCount 必须不小于 1；按帧获

取时，u32PackCount 必须不小于当前帧的包个数。在函数调用成功后，u32PackCount 返回实际填充 pstPack 的包的个数。

- 序列号 u32Seq 按帧获取时是帧序列号；按包获取时为包序列号。码流特征信息，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 stH264Info/stJpegInfo/stMpeg4Info/stH265Info，码流特征信息的输出用于支持用户的上层应用。如果用户长时间不获取码流，码流缓冲区就会满。一个编码通道如果发生码流缓冲区满，就会不再启动编码，直到用户获取码流，从而有足够的码流缓冲可以用于编码时，才开始继续编码。建议用户获取码流接口调用与释放码流的接口调用成对出现，且尽快释放码流，防止出现由于用户态获取码流，释放不及时而导致的码流 buffer 满，停止编码。当 OneStreamBuffer=1 时，用户获取一帧码流时只会得到一个码流包（不包含用户数据）的地址，即 u32PackCount 为 1，地址为 pstPack[0].pu8Addr。而在 MI_VENC_Pack_t 结构体中增加 u32Offset，用来指出一帧码流中有效数据的地址和 pstPack[0].pu8Addr 的偏移。
- 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。如图 6-7 所示(以 H.264 为例),pstPack[0].u32DataNum=3, 即 3 种 NAL 包，分别为 SPS、PPS、SEI。
  pstPack[0].stPackInfo[0].u32PackType.enH264EType=E_MI_VENC_H264E_NALU_SPS，
  pstPack[0].stPackInfo[1].u32PackType.enH264EType=E_MI_VENC_H264E_NALU_PPS，
  pstPack[0].stPackInfo[2].u32PackType.enH264EType=E_MI_VENC_H264E_NALU_SEI ;其他协议以此类推。

图 6-7 码流包结构图

无效数据包头类型,例如：



两种模式可通过各协议 ko 加载时设置模块参数 OneStreamBuffer 来选择。OneStreamBuffer=1 表示单包模式；OneStreamBuffer=0 表示非单包模式，系统默认 OneStreamBuffer=0。

- 举例

```
MI_S32 VencGetH264Stream(MI_VOID)
{
    MI_S32 i;
    MI_S32
    s32Ret;MI_S32
    s32VencFd;
    MI_U32
    u32FrameIdx=0;MI_VEN
    C_CHN VeChn=0;
```

```
MI_VENC_CHN
Stat_tstStat;MI_VENC
_Stream_tstStream;fd
_setread_fds;
FILE*pFile=NULL;
pFile=fopen("stream.h264","wb"
);if(pFile==NULL)
{
    returnMI_FAILURE;
}
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("selecterr\n"
        );returnMI_FAILURE;
    }
    elseif(0==s32Ret)
    {
        printf("timeout\n")
        ;returnMI_FAILURE;
    }
    else
    {
        if(FD_ISSET(s32VencFd,&read_fds))
        {
            s32Ret=MI_VENC_Query(VeChn,&stStat);if
            (s32Ret!=MI_SUCCESS)
            {
               returnMI_FAILURE;
            }
            /************************************************
             *****suggesttocheckbothu32CurPacksandu32LeftStreamFram
             esat
```

```
thesametime,forexample:
          if(0==stStat.u32CurPacks||0==stStat.u32LeftStreamFrames)
           {
              continue;
           }
          ************************************************
          *****/if(0==stStat.u32CurPacks)
          {
               continue;
          }
              stStream.pstPack=(MI_VENC_Pack_t*)malloc(sizeo
                  f(MI_VENC_Pack_t)*stStat.u32CurPacks);
              if(NULL==stStream.pstPack)
              {
                 returnMI_FAILURE;
              }
              stStream.u32PackCount=stStat.u32CurPacks;
              s32Ret=MI_VENC_GetStream(VeChn,&stStream,-1);if
              (MI_SUCCESS!=s32Ret)
              {
                  free(stStream.pstPack
                  );stStream.pstPack=NU
                  LL;returnMI_FAILURE;
              }
              for(i=0;i<stStream.u32PackCount;i++)
              {
                  fwrite(stStream.pstPack[i].pu8A
ddr+stStream.pstPack[i].u32Offset,
                         1,stStream.pstPack[i].u32Le
n-stStream.pstPack[i].u32Offset,pFile);
              }
              s32Ret=MI_VENC_ReleaseStream(VeChn,&stStream);i
              f(MI_SUCCESS!=s32Ret)
              {
                  free(stStream.pstPack
                  );stStream.pstPack=NU
                  LL;returnMI_FAILURE;
              }
```

```
                free(stStream.pstPack
                );stStream.pstPack=NU
                LL;
            }
        }
        u32FrameIdx++;
    }while(u32FrameIdx<0xff);
    fclose(pFile);re
    turnMI_SUCCESS;
}
```

更详细的内容，请参考 sample **代码。**

➢ **相关主题**

无。

1.2.14 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。

1.2.15 MI_VENC_GetStreamBufInfo

➢ 描述

获取码流 buffer 的物理地址和大小。

➢ 语法

MI_S32 MI_VENC_GetStreamBufInfo(MI_VENC_CHN VeChn,MI_VENC_StreamBufInfo_t*pstStreamBufInfo)

➢ 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
pstStreamBufInfo	码流 buffer 信息结构体指针。	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，则返回错误码 MI_ERR_VENC_UNEXIST。
- 如果 pstStreamBufInfo 为空，则返回错误码 MI_ERR_VENC_NULL_PTR。

- 当用户需要使用一帧码流的物理地址时，应先调用该接口来获取码流 buffer 的起始物理地址及其大小，判断一帧码流是否有折回。当 MI_VENC_Stream_t 结构体中 pstPack[0].u32PhyAddr+u32Len 大于 pstStreamBufInfo->u32PhyAddr+pstStreamBufInfo->u32BufSize 则折回，通过获取一帧码流因折回而得到的两个物理地址，从而正确使用一帧码流的物理地址，其中 u32Len 为一帧码流的长度。

➢ 举例

```
MI_S32 VencGetH264StreamBufInfo(MI_VOID)
{
    MI_S32 i;
    MI_S32
    s32Ret;MI_S32
    s32VencFd;
    MI_U32
    u32FrameIdx=0;MI_VEN
    C_CHN VeChn=0;
    MI_VENC_CHN
    Stat_tstStat;MI_VENC
    _Stream_tstStream;fd
    _setread_fds;
    MI_VENC_StreamBufInfo_tstStreamBufI
    nfo;MI_U32  u32Left;
    MI_U32  u32SrcPhyAddr,u32DestPhyAddr;
    s32Ret=MI_VENC_GetStreamBufInfo(VeChn,&stStreamBufInfo);
    if(MI_SUCCESS!=s32Ret)
    {
                                                                    }
                                                                    do{
    returnMI_FAILURE;
s32Ret=MI_VENC_Query(VeChn,&stStat);if(s32Ret!=MI_SUCCESS)
{
                returnMI_FAILURE;
            }
            stStream.pstPack=(MI_VENC_Pack_t*)
                        malloc(sizeof(MI_VENC_Pack_t)*stStat.u32Cu
            rPacks);if(NULL==stStream.pstPack)
            {
                returnMI_FAILURE;
```

```
}
stStream.u32PackCount=stStat.u32CurPacks;
s32Ret=MI_VENC_GetStream(VeChn,&stStream,-1);if
(MI_SUCCESS!=s32Ret)
{
    free(stStream.pstPack
    );stStream.pstPack=NU
    LL;returnMI_FAILURE;
}
//DMATransfer,onlyusephysicaladdress
for(i=0;i<stStream.u32PackCount;i++)
{
    if(stStream.pstPack[i].u32PhyA
    ddr+stStream.pstPack[i].u32Len
    >=stStreamBufInfo.u32PhyAddr+s
    tStreamBufInfo.u32BufSize)
    {
        if(stStream.pstPack[i].u32PhyA
        ddr+stStream.pstPack[i].u32Off
        set>=stStreamBufInfo.u32PhyAdd
        r+stStreamBufInfo.u32BufSize)
        {
            //(a)branchinpicture
            u32SrcPhyAddr=stStreamBufInfo.u32PhyAd
            dr+((stStream.pstPack[i].u32PhyAddr+st
            Stream.pstPack[i].u32Offset)-(stStream
            BufInfo.u32PhyAddr+stStreamBufInfo.u32
            BufSize));
            DMA_TransFer(u32SrcPhyAddr,u32DestPhyA
            ddr,stStream.pstPack[i].u32Len-stStrea
            m.pstPack[i].u32Offset);
        }
                                                    else
                                                    {
//(b)branchinpicture
u32Left=(stStreamBufInfo.u32PhyAddr+stStreamBufInfo.u32BufSize)-stSt
      ream.pstPack[i].u32PhyAddr;
DMA_TransFer(stStream.pstPack[i].u32PhyAddr+stStream.pstPack[i].u32Offset
```

```
   ,u32DestPhyAddr,u32Left-stStream.pstPack[i].u32Offset);
                                                                                    }
                                                                                }else
{
      DMA_TransFer(stStreamBufInfo.u32PhyAddr,u32DestPhyAddr,stStream.pstPa
      ck[i].u32Len-u32Left);
                                 //(c)branchinpicture
                                 DMA_TransFer(stStream.pstPack[i].u32PhyAd
                                 dr+stStream.pstPack[i].u32Offset,stStream
                                 .pstPack[i].u32Len-stStream.pstPack[i].u3
                                 2Offset);
                             }
                         }
                         s32Ret=MI_VENC_ReleaseStream(VeChn,&stStream);i
                         f(MI_SUCCESS!=s32Ret)
                         {
                             free(stStream.pstPack
                             );stStream.pstPack=NU
                             LL;returnMI_FAILURE;
                         }
                         free(stStream.pstPack
                         );stStream.pstPack=NU
                         LL;

                         u32FrameIdx++;
                     }while(u32FrameIdx<0xff);
                     returnMI_SUCCESS;
                 }
```

➢ **相关主题**

无。

1.2.16 MI_VENC_InsertUserData

➢ **描述**

插入用户数据。

➢ **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 和 MJPEG/JPEG 编码协议，不支持 MPEG-4
- 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 包的形式被插入到最新的图像码流包之前。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。
- JPEG/MJPEG 协议通道最多分配 1 块内存空间 ,用于缓存 1Kbyte 的用户数据。用户数据以 APPsegment（0xFFEF）形式添加到图像码流中。在用户数据被编码发送之后，JPEG/MJPEG 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。

➢ 举例

```
MI_U8 au8UserData[]="hisilicon2011";
s32Ret=MI_VENC_InsertUserData(VeChn,au8UserData,
          sizeof(au8UserData)
   );if(MI_SUCCESS!=s32Ret)
   {
       printf("MI_VENC_InsertUserDataerr0x%xn",s32Ret);
   }
```

➢ 相关主题

无。

1.2.17 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。	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 满而不被编码。
- 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 200。
- 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。

1.2.18 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。

1.2.19 MI_VENC_RequestIdr

➢ **描述**

请求 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。	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。
- 接受 IDR 帧或 I 帧请求后，当 bInstant=0 时，则在尽可能短的时间内编出 IDR 帧或 I 帧，当 bInstant=1 时，则在下帧立即编码 IDR 帧或 I 帧。
- I 帧请求，只支持 H.264/H.265 和 MPEG-4 编码协议。

➢ 举例

```
MI_S32
s32Ret;MI_VENC_CH
N VeChn=0;
s32Ret=MI_VENC_RequestIDR(VeChn);if
(MI_SUCCESS!=s32Ret)
{
    printf("MI_VENC_RequestIDRerr0x%xn",s32Ret);
}
```

➢ 相关主题

无。

1.2.20 MI_VENC_EnableIdr

➢ 描述

是否使能 IDR 帧。

➢ **语法**

MI_S32 MI_VENC_EnableIdrVENC_CHNVeChn,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 s32Ret;
MI_VENC_CHN VeChn=0;
s32Ret=MI_VENC_EnableIDR(VeChn);if
(MI_SUCCESS!=s32Ret)
{
    printf("MI_VENC_EnableIDRerr0x%xn",s32Ret);
}
```

➢ **相关主题**

无。

1.2.21 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：
- 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。
- 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。

➢ 举例

无。

➢ 相关主题

无。

1.2.22 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。

➢ 举例

无。

➢ 相关主题

无。

1.2.23 MI_VENC_GetFd

➢ 描述

获取编码通道对应的设备文件句柄。

➢ 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

无。

➢ **相关主题**

无。

1.2.24 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。

1.2.25 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。

➢ 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 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。bEnable :指定当前的 ROI 区域是否使能。bAbsQp：指定当前的 ROI 区域采用绝对 QP 方式或是相对 QP。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。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，且必须 16 对齐；ROI 区域的长宽必须是 16 对齐；ROI 区域必须在图像范围内。
- 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。

➢ 举例

```
MI_S32 SetRoi(MI_VOID)
{
    MI_S32 s32Ret=MI_FAILURE;
    MI_VENC_RoiCfg_t
                    stRoiCfg;
    MI_VENC_CHN Attr_t
```

```
                stChnAttr
;MI_S32 index=0;
MI_VENC_CHN VeChnId=0;
//...omitotherthing
s32Ret=MI_VENC_GetChnAttr(VeChnId,&stChnAttr);i
f(MI_SUCCESS!=s32Ret)
{
    printf("MI_VENC_GetChnAttrerr0x%x\n",s32Ret);r
    eturnMI_FAILURE;
}
s32Ret=MI_VENC_GetRoiCfg(VeChnId,index,&stRoiCfg);if
(MI_SUCCESS!=s32Ret)
{
    printf("MI_VENC_GetRoiCfgerr0x%x\n",s32Ret);r
    eturnMI_FAILURE;
}
stSrcVencRoiCfg.bEnable     =MI_TRUE;
stSrcVencRoiCfg.bAbsQp      =MI_TRUE;
stSrcVencRoiCfg.s32Qp       =10;
stSrcVencRoiCfg.stRect.s32X  =16;
stSrcVencRoiCfg.stRect.s32Y  =16;
stSrcVencRoiCfg.stRect.u32Width=16;
stSrcVencRoiCfg.stRect.u32Height=16;
stSrcVencRoiCfg.u32Index     =0;
s32Ret=MI_VENC_SetRoiCfg(VeChnId,&stSrcVencRoiCfg);
if(MI_SUCCESS!=s32Ret)
{
    printf("MI_VENC_SetRoiCfgerr0x%x\n",s32Ret);
    returnMI_FAILURE;
}
returnMI_SUCCESS;
}
```

- **相关主题**

  无。

1.2.26 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。

1.2.27 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，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。

➢ 举例

```
MI_S32 SetRoiFrameRate(MI_VOID)
{
    MI_S32 s32Ret=MI_FAILURE;
    MI_VENC_RoiCfg_t  stRoiCfg;MI_VENC_CHN
    Attr_t
                    stChnAttr;MI_VENC_RoiB
    gFrameRate_tstRoiBgFrameRate；MI_S32
    index=0;
```

```
MI_VENC_CHN VeChnId=0;
//...omitotherthing
s32Ret=MI_VENC_GetChnAttr(VeChnId,&stChnAttr);i
f(MI_SUCCESS!=s32Ret)
{
    printf("MI_VENC_GetChnAttrerr0x%x\n",s32Ret);r
    eturnMI_FAILURE;
}
s32Ret=MI_VENC_GetRoiCfg(VeChnId,index,&stRoiCfg);if
(MI_SUCCESS!=s32Ret)
{
    printf("MI_VENC_GetRoiCfgerr0x%x\n",s32Ret);r
    eturnMI_FAILURE;
}
stSrcVencRoiCfg.bEnable     =MI_TRUE;
stSrcVencRoiCfg.bAbsQp      =MI_TRUE;
stSrcVencRoiCfg.s32Qp       =10;
stSrcVencRoiCfg.stRect.s32X  =16;
stSrcVencRoiCfg.stRect.s32Y  =16;
stSrcVencRoiCfg.stRect.u32Width=16;
stSrcVencRoiCfg.stRect.u32Height=16;
stSrcVencRoiCfg.u32Index     =0;
s32Ret=MI_VENC_GetRoiCfg(VeChnId,&stSrcVencRoiCfg);if
(MI_SUCCESS!=s32Ret)
{
       printf("MI_VENC_GetRoiCfgerr0x%x\n",s32Ret);r
    eturnMI_FAILURE;
}
stRoiBgFrameRate.s32SrcFrmRate=30;
stRoiBgFrameRate.s32DstFrmRate=15;
s32Ret=MI_VENC_SetRoiBgFrameRate(VeChnId,&stRoiBgFrameRate);i
f(MI_SUCCESS!=s32Ret)
{
       printf("MI_VENC_SetRoiBgFrameRateerr0x%x\n",s32Ret);
    returnMI_FAILURE;
}
returnMI_SUCCESS;
```

```
}
```

➢ **相关主题**

无。

1.2.28 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。

1.2.29 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
- bSplitEnable：当前帧是否进行 slice 分割。
- 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_GetH264SliceSplit 接口，获取当前通道的 slicesplit 配置，然后再进行设置。

➢ 举例

```
MI_S32 SetSliceSplit(MI_VOID)
```

```
{
    MI_S32
    s32Ret=MI_FAILURE;MI_VENC_ParamH26
    4SliceSplit_tstSlice;MI_VENC_CHN
    VeChnId=0;

    //...omitotherthing
    s32Ret=MI_VENC_GetH264SliceSplit(VeChnId,&stSlice);
    if(MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_GetH264SliceSpliterr0x%x\n",s32Ret);
        returnMI_FAILURE;
    }
    stSlice.bSplitEnable=MI_TRUE
    ;
    stSlice. u32SliceRowCount=8;
    s32Ret=MI_VENC_SetH264SliceSplit(VeChnId,&stSlice);
    if(MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_SetH264SliceSpliterr0x%x\n",s32Ret);
        return MI_FAILURE;
    }
    returnMI_SUCCESS;
}
```

- **相关主题**

  无。

1.2.30 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。

1.2.31 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**

- 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 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 u32HWSize：帧间预测时的水平搜索窗的大小。u32VWSize：帧间预测时的垂直搜索窗的大小。bInter16x16PredEn：16x16 块帧间预测使能标识。bInter16x8PredEn：16x8 块帧间预测使能标识。bInter8x16PredEn：8x16 块帧间预测使能标识。bInter8x8PredEn：8x8 块帧间预测使能标识。
- 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。
- 水平搜索窗的大小存在以下约束：
- 水平搜索窗必须大于等于 MIN_HW，且小于等于 MAX_HW。
- 图像宽度（以宏块为单位）必须比搜索窗实际宽度至少大 2 个宏块。（以宏块为单位，实际搜索窗宽度=(1+u32HWSize)x2+1。如搜索窗的宽度设置为 10，则实际的图像宽度至少大于：((1+10)x2+1+1）x16=384，如 386。
- 垂直搜索窗的大小存在以下约束：
- 垂直搜索窗必须大于等于 MIN_VW，且小于等于 MAX_VW。
- 图像高度（以宏块为单位）必须比搜索窗实际高度至少大 1 个宏块。（以宏块为单位，实际搜索窗高度=(1+u32VWSize)x2+1。如搜索窗的高度设置为 4 ,则实际的图像高度至少大于 :((1+4)x2+1) x16=176，如 178。
- 当图像宽度小于等于 720 时，垂直搜索窗必须小于等于 2；当图像
- 宽度大于 720，且小于等于 1280 时，垂直搜索窗必须小于等于 2，当图像宽度大
- 于 1920，且小于等于 1688 时，垂直搜索窗必须小于等于 1，当图像宽度大于 2688，垂直搜索窗只能等于 0。

- 表 6-9、表 6-10 和表 6-11 中给出典型的搜索窗配置，且该配置需符合上述水平搜索窗大小和垂直搜索窗大小的限制，当默认配置在某些小分辨率图像不符合水平搜索窗大小和垂直搜索窗大小的限制时，水平搜索窗和垂直搜索窗默认大小以各个芯片支持的最小分辨率(160x64)根据约束计算出来的大小为准，例如 CIF 图像（352x288），默认水平搜索窗大小为 11，按照上述约束，实际的图像宽度至少大于((1+11)x2+1+1)x16=416，故不符合该限制，此时将水平搜索窗大小默认设置修改为 2，垂直搜索窗的大小依次类推。

➢ 举例

```
MI_S32 SetInterPred(MI_VOID)
{
    MI_S32 s32Ret=MI_FAILURE;
    MI_VENC_ParamH264InterPred_tstInterPred
    ;MI_VENC_CHN VeChnId=0;
    //...omitotherthing
    s32Ret=MI_VENC_GetH264InterPred(VeChnId,&stInterPred);i
    f(MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_GetH264InterPrederr0x%x\n",s32Ret);
        returnMI_FAILURE;
    }
    stInterPred.bExtedgeEn=MI_TRUE;stInt
    erPred.bInter16x16PredEn=MI_TRUE;stI
    nterPred.bInter16x8PredEn=MI_FALSE;s
    tInterPred.bInter8x16PredEn=MI_FALSE
    ;stInterPred.bInter8x8PredEn=MI_FALS
    E;stInterPred.u32HWSize=4;
    stInterPred.u32VWSize=2;
    s32Ret=MI_VENC_SetH264InterPred(VeChnId,&stInterPred);if
    (MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_SetH264InterPrederr0x%x\n",s32Ret);r
        eturnMI_FAILURE;
    }
    returnMI_SUCCESS;
}
```

➢ 相关主题

无。

1.2.32 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。

1.2.33 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
- 芯片差异不支持 IPCM 预测功能，用户可根据依赖，设定 IPCM 的开关，VENC 默认使能 IPCM 功能。
- 帧间预测的属性主要由四个参数决定 bIntra16x16PredEn：16x16 块帧内预测使能标识。bIntraNxNPredEn：NxN 块帧内预测使能标识。其中，NxN 表示 4x4 和 8x8。bIpcmEn：IPCM 块帧内预测使能标识。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 SetIntraPred(MI_VOID)
{
    MI_S32 s32Ret=MI_FAILURE;
    MI_VENC_ParamH264IntraPred_tstIntraPred
    ;MI_VENC_CHN VeChnId=0;

    //...omitotherthing
    s32Ret=MI_VENC_GetH264IntraPred(VeChnId,&stIntraPred);i
    f(MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_GetH264IntraPrederr0x%x\n",s32Ret);
        returnMI_FAILURE;
    }
    stIntraPred.bIntra16x16PredEn=MI_TRUE;s
    tIntraPred.bIntraNxNPredEn
                            =MI_FALSE;stInt
    raPred.bIpcmEn
                            =MI_FALSE;stInt
    raPred.constrained_intra_pred_flag=0;
    s32Ret=MI_VENC_SetH264IntraPred(VeChnId,&stIntraPred);if
    (MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_SetH264IntraPrederr0x%x\n",s32Ret);
        returnMI_FAILURE;
    }
    returnMI_SUCCESS;
}
```

- **相关主题**

  无。

1.2.34 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。

1.2.35 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：
  - 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。
  - 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。
  - s32ChromaQpIndexOffset：具体含义请参见 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。系统会根据不同的 profile 设置默认参数。表 6-12 不同 profile 下系统默认 trans 参数

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 SetTrans(MI_VOID)
{
    MI_S32 s32Ret=MI_FAILURE;
    MI_VENC_ParamH264Trans_tstTra
    ns;MI_VENC_CHN VeChnId=0;

    //...omitotherthing
    s32Ret=MI_VENC_GetH264Trans(VeChnId,&stTrans);if
    (MI_SUCCESS!=s32Ret)
    {
        printf("stTranserr0x%x\n",s32Ret
        );returnMI_FAILURE;
    }
    stTrans.u32IntraTransMode =2;
    stTrans.u32InterTransMode
                              =2;stTrans
    .bScalingListValid        =MI_FALSE;

    stTrans.s32ChromaQpIndexOffset=2;
    s32Ret=MI_VENC_SetH264Trans(VeChnId,&stTrans);if(
    MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_SetH264Transerr0x%x\n",s32Ret);
        returnMI_FAILURE;
    }
    returnMI_SUCCESS;
}
```

- **相关主题**

  无。

1.2.36 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。

1.2.37 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：
  - 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。
  - 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。
- Baselineprofile 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。

  表 6-13 不同 profile 下系统默认熵编码参数

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 SetEntropy(MI_VOID)
{
    MI_S32 s32Ret=MI_FAILURE;
    MI_VENC_ParamH264Entropy_tstEntr
    opy;MI_VENC_CHN VeChnId=0;

    //...omitotherthing
    s32Ret=MI_VENC_GetH264Entropy(VeChnId,&stTrans);
    if(MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_GetH264Entropyerr0x%x\n",s32Ret);
        returnMI_FAILURE;
    }
    stEntropy.u32EntropyEncModeI=0;
    stEntropy.u32EntropyEncModeP=0;
    stEntropy.cabac_stuff_en  =0;
    stEntropy.Cabac_init_idc  =0;
    s32Ret=MI_VENC_SetH264Entropy(VeChnId,&stEntropy);if
    (MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_SetH264Entropyerr0x%x\n",s32Ret);
        returnMI_FAILURE;
    }
    returnMI_SUCCESS;
}
```

- **相关主题**

  无。

1.2.38 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。

1.2.39 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 u32HWSize：帧间预测时的水平搜索窗的大小。u32VWSize：帧间预测时的垂直搜索窗的大小。bInter16x16PredEn：16x16 块帧间预测使能标识。bInter16x8PredEn：16x8 块帧间预测使能标识。bInter8x16PredEn：8x16 块帧间预测使能标识。bInter8x8PredEn：8x8 块帧间预测使能标识。
- 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。
- 水平搜索窗的大小存在以下约束：水平搜索窗必须大于等于 MIN_HW，且小于等于 MAX_HW。图像宽度（以宏块为单位）必须比搜索窗实际宽度至少大 2 个宏块。（以宏块为单位，实际搜索窗宽度=(1+u32HWSize)x2+1。如搜索窗的宽度设置为 10，则实际的图像宽度至少大于：((1+10)x2+1+1)x16=384，如 386。
- 垂直搜索窗的大小存在以下约束：垂直搜索窗必须大于等于 MIN_VW，且小于等于 MAX_VW。图像高度（以宏块为单位）必须比搜索窗实际高度至少大 1 个宏块。（以宏块为单位，实际搜索窗高度=(1+u32VWSize)x2+1。如搜索窗的高度设置为 4，则实际的图像高度至少大于：（(1+4)x2+1）x16=176，如 178。
- 当图像宽度小于等于 720 时，垂直搜索窗必须小于等于 2；当图像宽度大于 720，且小于等于 1280 时，垂直搜索窗必须小于等于 2，当图像宽度大于 1920，且小于等于 1688 时，垂直搜索窗必须小于等于 1，当图像宽度大于 2688，垂直搜索窗只能等于 0。

- 表 6-9、表 6-10 和表 6-11 中给出典型的搜索窗配置，且该配置需符合上述水平搜索窗大小和垂直搜索窗大小的限制，当默认配置在某些小分辨率图像不符合水平搜索窗大小和垂直搜索窗大小的限制时，水平搜索窗和垂直搜索窗默认大小以各个芯片支持的最小分辨率(160x64)根据约束计算出来的大小为准，例如 CIF 图像（352x288），默认水平搜索窗大小为 11，按照上述约束，实际的图像宽度至少大于((1+11)x2+1+1）x16=416，故不符合该限制，此时将水平搜索窗大小默认设置修改为 2，垂直搜索窗的大小依次类推。

- **举例**

```
MI_S32 SetInterPred(MI_VOID)
{
    MI_S32 s32Ret=MI_FAILURE;
    MI_VENC_ParamH265InterPred_tstInterPred
    ;MI_VENC_CHN VeChnId=0;
    //...omitotherthing
    s32Ret=MI_VENC_GetH265InterPred(VeChnId,&stInterPred);i
    f(MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_GetH265InterPrederr0x%x\n",s32Ret);
        returnMI_FAILURE;
    }
    stInterPred.bExtedgeEn=MI_TRUE;stInt
    erPred.bInter16x16PredEn=MI_TRUE;stI
    nterPred.bInter16x8PredEn=MI_FALSE;s
    tInterPred.bInter8x16PredEn=MI_FALSE
    ;stInterPred.bInter8x8PredEn=MI_FALS
    E;stInterPred.u32HWSize=4;
    stInterPred.u32VWSize=2;
    s32Ret=MI_VENC_SetH265InterPred(VeChnId,&stInterPred);if
    (MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_SetH265InterPrederr0x%x\n",s32Ret);r
        eturnMI_FAILURE;
    }
    returnMI_SUCCESS;
}
```

- **相关主题**

  无。

1.2.40 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。

1.2.41 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
  - u32Intra32x32Penalty：32x32 块帧内预测被选中概率，值越大概率越低。
  - u32Intra16x16Penalty：16x16 块帧内预测被选中概率，值越大概率越低。
  - 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 SetIntraPred(MI_VOID)
{
    MI_S32 s32Ret=MI_FAILURE;
    MI_VENC_ParamH265IntraPred_tstIntraPred
    ;
    MI_VENC_CHN VeChnId=0;

    //...omitotherthing
    s32Ret=MI_VENC_GetH265IntraPred(VeChnId,&stIntraPred);i
    f(MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_GetH265IntraPrederr0x%x\n",s32Ret);
        returnMI_FAILURE;
    }
    stIntraPred. u32Intra32x32Penalty=0;
    stIntraPred. u32Intra16x16Penalty=0;
```

```
    stIntraPred. u32Intra8x8Penalty=10;
    s32Ret=MI_VENC_SetH265IntraPred(VeChnId,&stIntraPred);if
    (MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_SetH265IntraPrederr0x%x\n",s32Ret);
        returnMI_FAILURE;
    }
    returnMI_SUCCESS;
}
```

➢ **相关主题**

无。

1.2.42 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**

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。

1.2.43 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：
  - 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。

- 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。
- s32ChromaQpIndexOffset：具体含义请参见 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。系统会根据不同的 profile 设置默认参数。

  表 6-12 不同 profile 下系统默认 trans 参数

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 SetTrans(MI_VOID)
{
    MI_S32 s32Ret=MI_FAILURE;
    MI_VENC_ParamH265Trans_tstTra
    ns;MI_VENC_CHN VeChnId=0;
    //...omitotherthing
    s32Ret=MI_VENC_GetH265Trans(VeChnId,&stTrans);if
    (MI_SUCCESS!=s32Ret)
    {
        printf("stTranserr0x%x\n",s32Ret
        );returnMI_FAILURE;
    }
    stTrans.u32IntraTransMode =2;
    stTrans.u32InterTransMode
                              =2;stTrans
    .bScalingListValid        =MI_FALSE;
    stTrans.s32ChromaQpIndexOffset=2;
    s32Ret=MI_VENC_SetH265Trans(VeChnId,&stTrans);if(
    MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_SetH265Transerr0x%x\n",s32Ret);
```

```
            returnMI_FAILURE;
        }
        returnMI_SUCCESS;
    }
```

- **相关主题**

  无。

1.2.44 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。

1.2.45 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：disable_deblocking_filter_idc：具体含义请参见 H.264 协议。slice_alpha_c0_offset_div2：具体含义请参见 H.264 协议。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 SetDblk(MI_VOID)
{
    MI_S32 s32Ret=MI_FAILURE;
    MI_VENC_ParamH264Dblk_t
                                stDblk
    ;MI_VENC_CHN VeChnId=0;

    //...omitotherthing
    s32Ret=MI_VENC_GetH264Dblk(VeChnId,&stDblk);if(
    MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_GetH264Dblkerr0x%x\n",s32Ret);r
        eturnMI_FAILURE;
    }
    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_SetH264Dblkerr0x%x\n",s32Ret);r
        eturnMI_FAILURE;
    }
    returnMI_SUCCESS;
}
```

- **相关主题**

  无。

1.2.46 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。

1.2.47 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：disable_deblocking_filter_idc：具体含义请参见 H.265 协议。slice_alpha_c0_offset_div2：具体含义请参见 H.265 协议。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_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_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。

1.2.48 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。

1.2.49 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_VOID)
{
    MI_S32 s32Ret=MI_FAILURE;
    MI_VENC_ParamH264Vui_t stVui;
    MI_VENC_CHN VeChnId=0;
    //...omitotherthing
    s32Ret=MI_VENC_GetH264Vui(VeChnId,&stVui);
    if(MI_SUCCESS!=s32Ret)
    {
```

```
        printf("MI_VENC_GetH264Vuierr0x%x\n",s32Ret);
        returnMI_FAILURE;
    }
    stVui.u8TimingInfoPresentFlag=1;
    s32Ret=MI_VENC_SetH264Vui(VeChnId,&stVui);
    if(MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_SetH264Vuierr0x%x\n",s32Ret);
        returnMI_FAILURE;
    }
    returnMI_SUCCESS;
}
```

- **相关主题**

  无。

1.2.50 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。

1.2.51 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。

1.2.52 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。

1.2.53 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
- bSplitEnable：当前帧是否进行 slice 分割。
- 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。

1.2.54 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。

1.2.55 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
  - 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。
  - au8YQt[64]，au8CbCrQt[64]：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。
  - 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_VOID)
{
    MI_S32 s32Ret=MI_FAILURE;
    MI_VENC_ParamJpeg_tstParamJ
    peg;MI_VENC_CHN VeChnId=0;
    //...omitotherthing
    s32Ret=MI_VENC_GetJpegParam(VeChnId,&stParamJpeg);if
    (MI_SUCCESS!=s32Ret)
    {
        printf("MI_VENC_GetJpegParamerr0x%x\n",s32Ret);
        returnMI_FAILURE;
    }
    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_SetJpegParamerr0x%x\n",s32Ret);
    returnMI_FAILURE;
}
returnMI_SUCCESS;
}
```

- **相关主题**

  无。

1.2.56 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。

1.2.57 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

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

- 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/Mpeg4CBR、H.264/H.265/Mjpeg/Mpeg4VBR 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 :C 表示图像复杂度 ,若 Cu32Thrd[0] ,x=0 ;u32Thrd[0]<C u32Thr[1]，x=1；u32Thrd[1]<C u32Thr[2]，x=2；u32Thrd[2]<C u32Thr[3]，x=3；u32Thrd[3]<C u32Thr[4] ,x=4 ;u32Thrd[4]<C u32Thr[5] ,x=5 ;u32Thrd[5]<C u32Thr[6]，x=6；u32Thrd[6]<C u32Thr[7]，x=7；u32Thrd[7]<C u32Thr[8]，x=8；u32Thrd[8]<C u32Thr[9]，x=9；u32Thrd[9]<C u32Thr[10]，x=10；u32Thrd[10]<C u32Thr[11]，x=11；u32Thrd[11]<C，x=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：C 表示图像复杂度，C<u32Thrd[0]，x=4；u32Thrd[0] C<u32Thr[1]，x=3；u32Thrd[1] C<u32Thr[2]，x=2；u32Thrd[2] C<u32Thr[3]，x=1；u32Thrd[3] C u32Thr[4]，x=y=0；u32Thrd[4]<C u32Thr[5]，y=1；u32Thrd[5]<C u32Thr[6] ,y=2 ;u32Thrd[6]<C u32Thr[7] ,y=3 ;u32Thrd[7]<C u32Thr[8]，y=4；u32Thrd[8]<C u32Thr[9]，y=5；u32Thrd[9]<C u32Thr[10]，y=6；u32Thrd[10]<C u32Thr[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：

  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。
- 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。

- 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。
- 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：

  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。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。
- 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。

- **举例**

```
MI_S32 SetRcParam(MI_VOID)
{
   MI_S32 s32Ret=MI_FAILURE;
   MI_VENC_RcParam_tstVencRcP
   ara;MI_VENC_CHN VeChnId=0;
   //...omitotherthing
   s32Ret=MI_VENC_GetRcParam(VeChnId,&stVencRcPara);i
   f(MI_SUCCESS!=s32Ret)
   {
   printf("MI_VENC_GetRcParamerr0x%x\n",s32Ret);re
      turnMI_FAILURE;
   }

   stVencRcPara.stParamH264Cbr.enSuperFrmMode=E_MI_VENC_SUP

   ERFRM_DISCARD;s32Ret=MI_VENC_SetRcParam(VeChnId,&stVencR

   cPara);
   if(MI_SUCCESS!=s32Ret)
   {
       printf("MI_VENC_SetRcParamerr0x%x\n",s32Ret);re
   turnMI_FAILURE;
   }
   //...omitotherthing
   returnMI_SUCCESS;
}
```

- **相关主题**

  无。

1.2.58 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。

1.2.59 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。	输入

	取值范围：[0,VENC_MAX_CHN_NUM)。	
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。

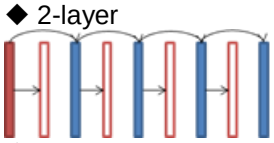
➢ 依赖

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

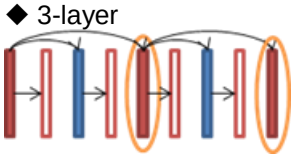
※ 注意

- 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 H.264 协议编码通道的 profile 为 svc-t，调用该接口则返回操作不允许错误。
- 如果用户设置 1 倍跳帧参考模式 对应的配置为 bEnablePred=MI_TRUE ,u32Enhance=0 ,u32Base=1；如果设置 2 倍跳帧参考模式，对应的配置为：bEnablePred=MI_TRUE，u32Enhance=1，u32Base=1；如果设置 4 倍跳帧参考模式，对应的配置为：bEnablePred=MI_TRUE，u32Enhance=1，u32Base=2。
- MFE/MHE 支援的 referencestructure 有 Singlelayer:IPPPencode 以及下列三種：

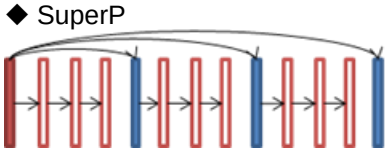
◆ 2-layer



◆ 3-layer



◆ SuperP



➢ 举例

无。

➢ 相关主题

无。

1.2.60 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。

1.2.61 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。
- 通道会优先进行图像裁剪，然后基于裁剪之后的图像尺寸，与编码通道的尺寸进行比较，决定是否进行缩小。
- 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：bEnable：是否使能通道裁剪功能。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。

1.2.62 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。

1.2.63 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。

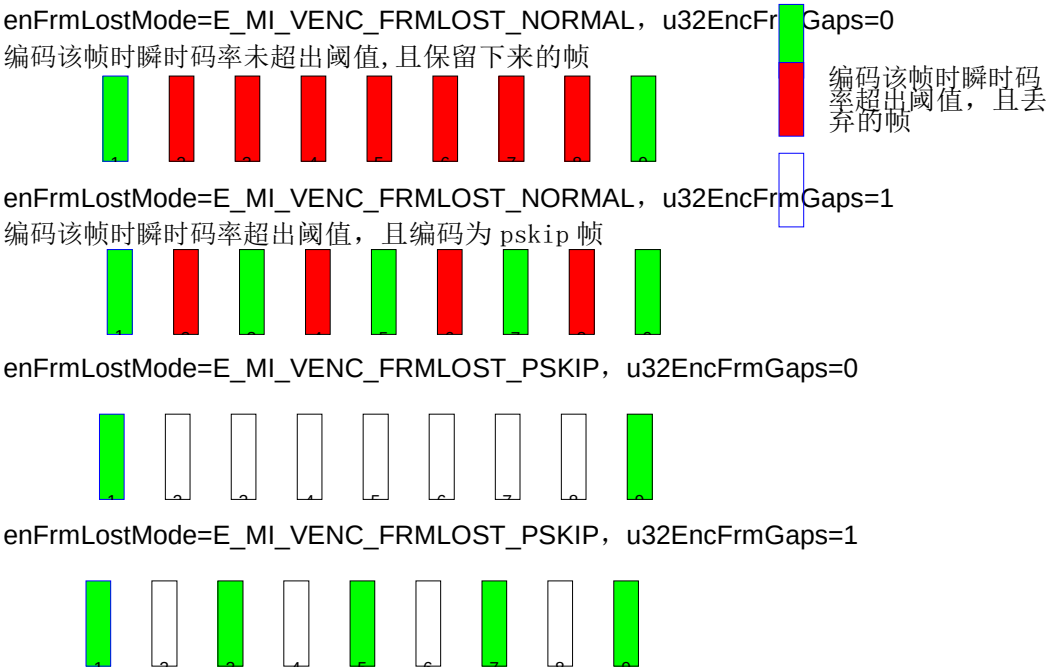
➢ **依赖**

- 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。
- p-stFrmLostParam 主要由四个参数决定 enFrmLostMode：丢帧策略模式。u32EncFrmGaps：丢帧间隔。bFrmLostOpen：丢帧开关。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。如图 6-9 所示：

图 6-9 瞬时码率超过阈值时的丢帧策略



- 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。

1.2.64 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。

1.2.65 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 eSuperFrmMode=E_MI_VENC_SUPERFRM_NONE，u32SuperIFrmBitsThr、u32SuperPFrmBitsThr、u32SuperBFrmBitsThr 为 500000。
- 本接口可在编码通道创建之后，编码通道销毁之前设置。

➢ 举例

无。

➢ 相关主题

无。

1.2.66 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。

1.2.67 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

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

- 本接口用于设置码率控制以目标码率还是以超大帧阈值高优先级。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，超大帧则会重编降比特数，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。

1.2.68 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。

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

2.1. VENC_MAX_CHN_NUM

➢ 说明

定义最大通道个数。

➢ 定义

```
#define   VENC_MAX_CHN_NUM      64
```

※ 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     12
```

※ 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_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_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,
    E_MI_VENC_H264E_REFSLICE_FOR_2X,
    E_MI_VENC_H264E_REFSLICE_FOR_4X,
    E_MI_VENC_H264E_REFSLICE_FOR_MAX
}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
    E_MI_VENC_JPEGE_PACK_APP
    E_MI_VENC_JPEGE_PACK_VDO
    E_MI_VENC_JPEGE_PACK_PIC
    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
    E_MI_VENC_H265E_NALU_ISLICE
    E_MI_VENC_H265E_NALU_VPS
    E_MI_VENC_H265E_NALU_SPS
    E_MI_VENC_H265E_NALU_PPS
    E_MI_VENC_H265E_NALU_SEI
    E_MI_VENC_H265E_NALU_MAX
}MI_VENC_H265eNaluType_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。
enMPEG4EType	MPEG-4 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_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。

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_U32       u32PhyAddr;
  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。
u32PhyAddr	码流包物理地址。
u32Len	码流包长度。
DataType	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： MI_TRUE：该码流包是该帧的最后一个包。 MI_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。
stPackInfo[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
u32Intra4MbNum	编码当前帧中采用 Intra4 预测模式的宏块数

成员名称	描述
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;
  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。按帧获取帧序号；按包获取包序号。
stH624Info/stJpegInfo/stMpeg4Info/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 的起始物理地址。
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 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：{MI_TRUE,MI_FALSE}。 MI_TRUE：按帧获取。 MI_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
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。
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： {MI_TRUE,MI_FALSE}。

成员名称	描述
	MI_TRUE：按帧获取。 MI_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

成员名称	描述
	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： {MI_TRUE,MI_FALSE}。 MI_TRUE：按帧获取。 MI_FALSE：按包获取。 静态属性。
u32Profile	编码的等级。 取值范围：0。 0：MainProfile。 静态属性。
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/stAttrMpeg4/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。
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。

成员名称	描述
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。
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

以上参数具体含义请参见 H.264 协议。

➢ 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。系统默认值为 0。取值范围：[-12,12]。

※ 注意事项

以上参数具体含义请参见 H.264 协议。

➢ 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。 Baseline 不支持 cabac。

成员名称	描述
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
{
  /*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_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。
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。

成员名称	描述
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。

成员名称	描述
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。
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

以上参数具体含义请参见 H.265 协议。

➢ 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。 • 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。

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。
s32ChromaQpIndexOffset	具体含义请参见 H.265 协议。系统默认值为 0。取值范围：[-12,12]。

※ 注意事项

以上参数具体含义请参见 H.265 协议。

➢ 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;
  MI_S32 slice_alpha_c0_offset_div2;
  MI_S32 slice_beta_offset_div2;
}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。
slice_alpha_c0_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。
stVuiTimeInfo	具体含义请参见 H.264 协议。
stVuiVideoSignal	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_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，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，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，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，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，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，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，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，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，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，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，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，系统默认为 5。取值范围：[0,7]。
u8VideoFullRangeFlag	具体含义请参见 H.264 协议，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，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，系统默认为 1。取值范围：[0,255]。
u8TransferCharacteristics	具体含义请参见 H.264 协议，系统默认为 1。取值范围：[0,255]。
u8MatrixCoefficients	具体含义请参见 H.264 协议，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。
stVuiTimeInfo	具体含义请参见 H.265 协议。
stVuiVideoSignal	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_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，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，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，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，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，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，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，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，系统默认为 1。取值范围：大于 0。

成员名称	描述
u32TimeScale	具体含义请参见 H.265 协议，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，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，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，系统默认为 5。取值范围：[0,7]。
u8VideoFullRangeFlag	具体含义请参见 H.265 协议，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，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，系统默认为 1。取值范围：[0,255]。

成员名称	描述
u8TransferCharacteristics	具体含义请参见 H.265 协议，系统默认为 1。取值范围：[0,255]。
u8MatrixCoefficients	具体含义请参见 H.265 协议，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。 - MI_FALSE：相对 QP - MI_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 MI_TRUE 时，s32Qp 为宏块 QP 值，s32Qp 范围[0,51]。
stRect	ROI 区域。

※ 注意事项

无。

➢ 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_AttrMjpegCbr_t      stAttrMjpegCbr;
     MI_VENC_AttrMjpegFixQp_t    stAttrMjpegFixQp;
     MI_VENC_AttrH265Cbr_t       stAttrH265Cbr;
     MI_VENC_AttrH265Vbr_t       stAttrH265Vbr;
     MI_VENC_AttrH265FixQp_t     stAttrH265FixQp;
    };
    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）。
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。

※ 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_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_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_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。

※ 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）
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_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

2.53. 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      u32Qfactor;
  MI_U32      u32SrcFrmRateNum;
  MI_U32      u32SrcFrmRateDen;
} 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

2.54. 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

2.55. 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

成员名称	描述
	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

2.56. 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。
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

2.57. 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

2.58. 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

2.59. 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

2.60. 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

2.61. 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

2.62. 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

2.63. 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_ParamMjpegCbr_t stParamMjpegCbr;
    MI_VENC_ParamH265Cbr_t  stParamH265Cbr;
    MI_VENC_ParamH265Vbr_t stParamH265Vbr;
  };
  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。
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。
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

2.64. 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;/*Cropregionenable*/
  MI_VENC_Rect_t stRect;       /*Cropregion,note:s32Xmustbemultiof16*/
}MI_VENC_CropCfg_t;
```

➢ 成员

成员名称	描述
bEnable	是否进行裁剪。 MI_TRUE：使能裁剪。 MI_FALSE：不使能裁剪。
stRect	裁剪的区域。 stRect.s32X：H.264 必须 16 像素对齐，H.265 必须 32 像素对齐。 stRect.s32Y：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

2.65. 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

2.66. 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

2.67. 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

2.68. 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

2.69. 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_U32  u32MinFrmRateNum;
  MI_U32  u32MinFrmRateDen;
}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]
u32MinFrmRateNum	帧率低于 MinFrmRate 时，就不再丢帧。此参数表示分子。
u32MinFrmRateDen	帧率低于 MinFrmRate 时，就不再丢帧。此参数表示分母。

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_SetFrameLostStrategy

2.70. 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

2.71. 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。

2.72. 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;
      MI_VENC_ParamModH264e_t   stH264eModParam;
      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。
stVencModParam/ stH264eModParam/ stH265eModParam/ stJpegeModParam	mi_venc.ko/mi_h264e.ko/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

2.73. 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

2.74. MI_VENC_ParamModVenc_t

➢ 说明

mi_venc.ko 模块参数。

➢ 定义

```
typedef struct MI_VENC_ParamModVenc_s
{
    MI_U32   u32VencBufferCache;
}MI_VENC_ParamModVenc_t;
```

➢ 成员

成员名称	描述
U32VencBufferCache	mi_venc.ko 模块中 u32VencBufferCache 参数。0：关闭码流 BufferCache 1：打开码流 BufferCache

※ 注意事项

无。

➢ 相关数据类型及接口

MI_VENC_ModParam_t

2.75. MI_VENC_ParamModH264e_t

➢ 说明

mi_h264e.ko 模块参数。

➢ 定义

```
typed efstruct MI_VENC_ParamModH264e_s
{
    MI_U32   u32OneStreamBuffer;
    MI_U32   u32H264eVBSource;
    MI_U32   u32H264eRcnEqualRef;
    MI_U32   u32H264eMiniBufMode;
}MI_VENC_ParamModH264e_t;
```

➢ 成员

成员名称	描述
u32OneStreamBuffer	mi_h264e.ko 模块中 u32OneStreamBuffer 参数。 0：多包模式 1：单包模式
u32H264eVBSource	mi_h264e.ko 模块中 u32H264eVBSource 参数。1：私有 VB 2：用户 VB
u32H264eRcnEqualRef	mi_h264e.ko 模块中 u32H264eRcnEqualRef 参数。 0：重构帧不复用参考帧亮度空间 1：重构帧复用参考帧亮度空间
u32H264eMiniBufMode	mi_h264e.ko 模块中 u32H264eMiniBufMode 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

2.76. 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

2.77. 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

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
0x0	MI_VENC_OK	success
0x1	MI_ERR_VENC_INVALID_DEVID	invalid device ID
0x2	MI_ERR_VENC_INVALID_CHNID	invalid channel ID
0x3	MI_ERR_VENC_ILLEGAL_PARAM	at lease one parameter is illegal
0x4	MI_ERR_VENC_EXIST	channel exists
0x5	MI_ERR_VENC_UNEXIST	channel unexist
0x6	MI_ERR_VENC_NULL_PTR	using a NULL point
0x7	MI_ERR_VENC_NOT_CONFIG	try to enable or initialize device or channel, before configuring attribute
0x8	MI_ERR_VENC_NOT_SUPPORT	operation is not supported by NOW
0x9	MI_ERR_VENC_NOT_PERM	operation is not permitted
0xC	MI_ERR_VENC_NOMEM	failure caused by malloc memory
0xD	MI_ERR_VENC_NOBUF	failure caused by malloc buffer
0xE	MI_ERR_VENC_BUF_EMPTY	no data in buffer
0xF	MI_ERR_VENC_BUF_FULL	no buffer for new data
0x10	MI_ERR_VENC_NOTREADY	System is not ready
0x11	MI_ERR_VENC_BADADDR	bad address
0x12	MI_ERR_VENC_BUSY	resource is busy
0x13	MI_ERR_VENC_CHN_NOT_STARTED	channel not start
0x14	MI_ERR_VENC_CHN_NOT_STOPPED	channel not stop
	MI_ERR_VENC_PRIVATE_START	
	MI_ERR_VENC_UNDEFINED	