MI VENC API

Version 2.12

REVISION HISTORY

Revision No.	Description	Date
2.03	• Initial release	04/12/2018
2.04	• Added Mjpeg cbr mode, InterxxPredEn, and VUI info	01/18/2019
2.05	• Added E_MI_VENC_BASE_P_REFTOIDR	02/25/2019
2.06	• Added query framerate and bitrate info	03/14/2019
2.07	• Added u32RestartMakerPerRowCnt	05/05/2019
2.08	• Added API for smart encoding	09/01/2019
2.09	• Added MI_VENC_SetInputSourceConfig for SSC33X ring mode	10/29/2019
2.10	• Modified the definitions of MI_VENC_AllocCustomMap, MI_VENC_RoiCfg_t, and MI_VENC_AdvCustRcAttr_t for SSC33X	11/01/2019
2.11	• Added data structure for AVBR	12/30/2019
2.12	• Added MI_VENC_SetSmartDetInfo for Smart Encoding and the associated data structure, modified MI_VENC_AllocCustomMap	01/02/2020

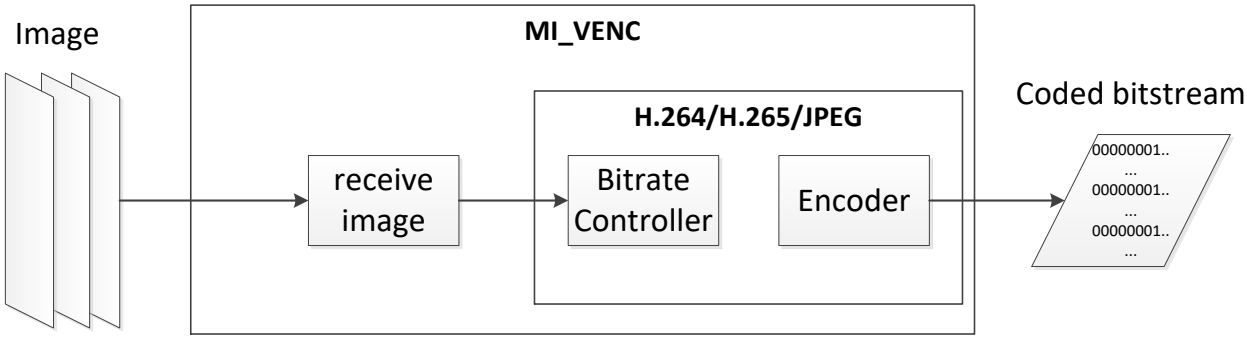
TABLE OF CONTENTS

1. API REFERENCE

1.1. Overview

The video coding module mainly provides the functions of creating and destroying a video coding channel, resetting a video coding channel, starting and stopping receiving images, setting and acquiring coding channel attributes, and acquiring and releasing a code stream. This module supports multi-channel real-time coding, each channel is independent, and different coding protocols and profiles can be set.

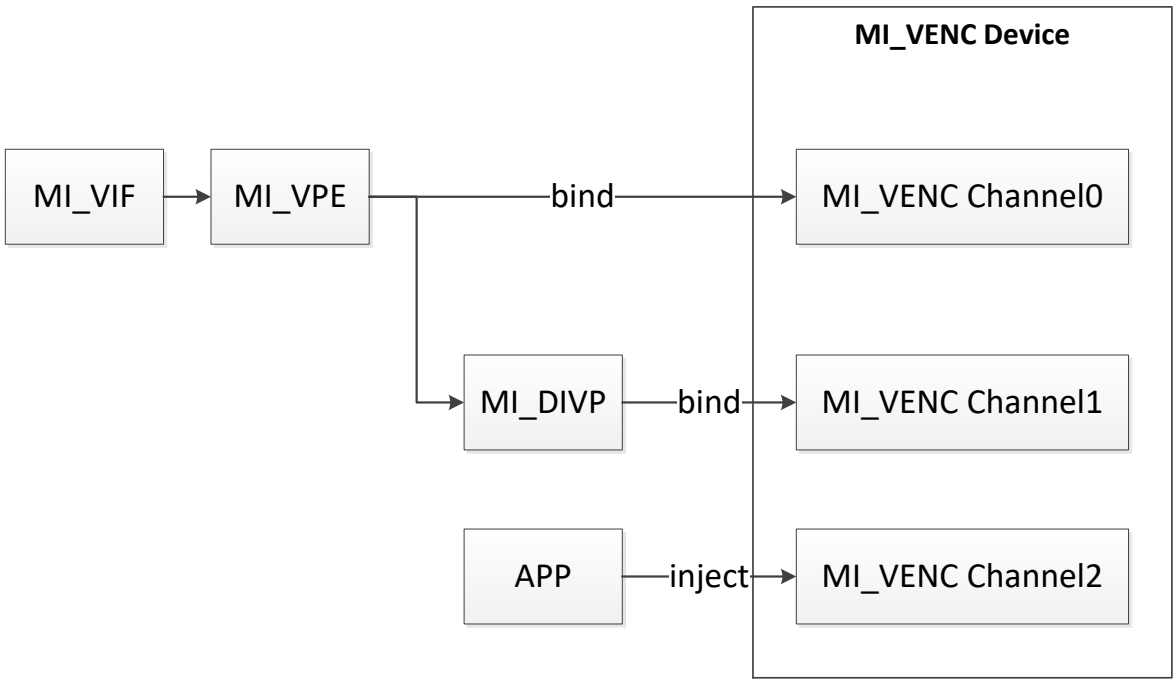
1.1.1 Encoding Process



The coding process includes receiving the input image, coding the input image, and outputting the encoded stream. The format of input source can be NV12 or YUYV422.

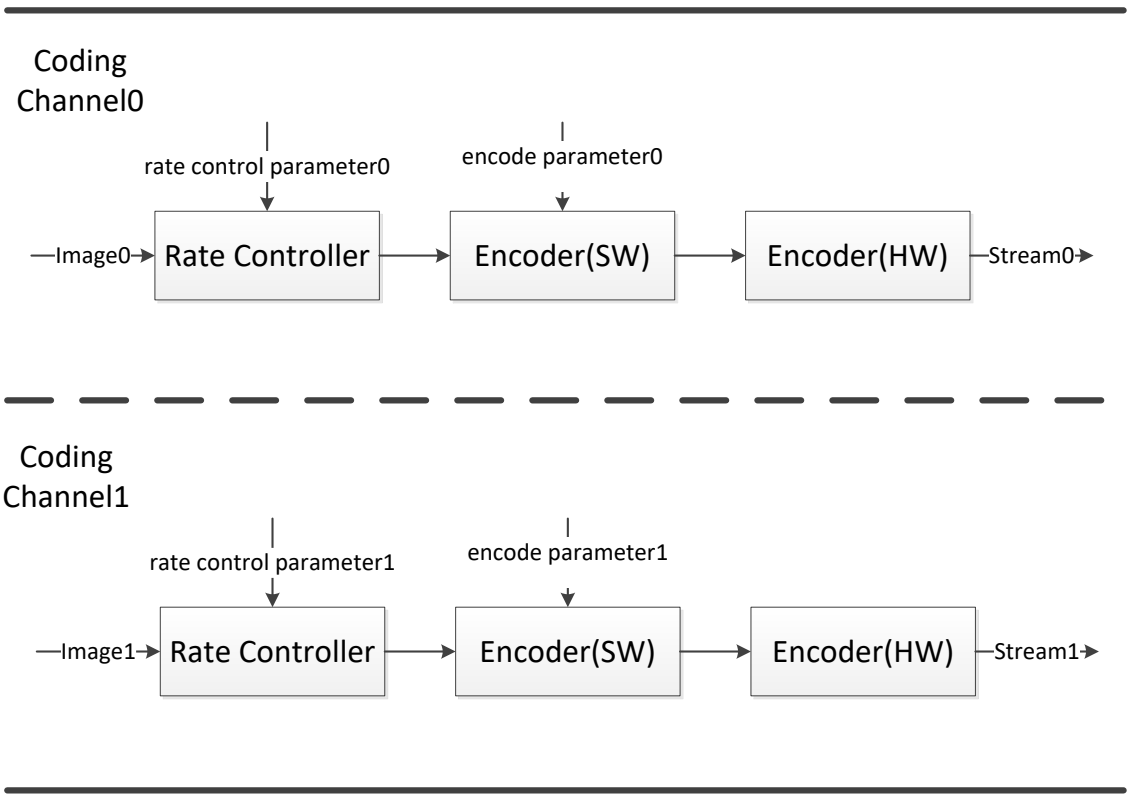
The input sources of this module include the following two types:

1. The user's app directly sends image data to the encoding module
2. The pre-bound module such as VPE or DIVP sends the image data to the encoding module, as shown in the figure below:



1.1.2 Coding Channel

As a basic control unit, coding channels are completely independent of each other. Encoder (HW) completes the function of converting the original image into a code stream, which is completed by the control of Rate Controller and the Encoder (SW) in cooperation. The Rate Controller and Encoder (SW) of each channel stores all the resources set and managed by all users of the current encoding channel, such as rate control, buffer demand and allocation, etc. The rate control ensures the balance between the output rate and the image quality, while the encoder focuses on the realization of the syntax elements of each codec. Software and hardware control, at the same time, a codec hardware time-sharing multiplexing. Taking two coding channels for example, the basic block diagram of coding channel is as follows:



1.2. Keyword Description

1.2.1 QP

Quantization Parameter: QP value corresponds to the sequence number of quantization step. The smaller the value is, the smaller the quantization step is, the higher the accuracy of quantization, the better the image quality and the larger the encoded size are.

1.2.2 GOP

Group Of Pictures: The interval between two I frames. The video image sequence consists of one or more GOPs which are independent.

1.2.3 MB

Macroblock: Coding macroblock, basic unit of H.264 encoding.

1.2.4 CU

Coding Unit: Coding unit, basic unit of H.265 encoding which is similar to H.264 encoding macroblock.

1.2.5 SPS

Sequence Parameter Set：It contains the public information of all images in a GOP.

1.2.6 PPS

Picture Parameter Set：It contains the parameters used to encode an image.

1.2.7 SEI

Supplemental Enhancement information

1.2.8 ECS

Entropy-coded segment: Compressed image strip after entropy encoding of JPEG

1.2.9 MCU

Minimum code unit: The basic unit of JPEG encoding

1.2.10 Rate control

From the perspective of Informatics, the image compression ratio is inversely proportional to the quality: the higher the compression ratio, the lower the quality; the lower the compression ratio, the higher the quality. Taking H.264 as an example, the lower the general image QP, the higher the image quality and the higher the bit rate; the higher the image QP, the lower the image quality and the lower the bit rate. The rate control algorithms supported by different coding protocols are as follows:

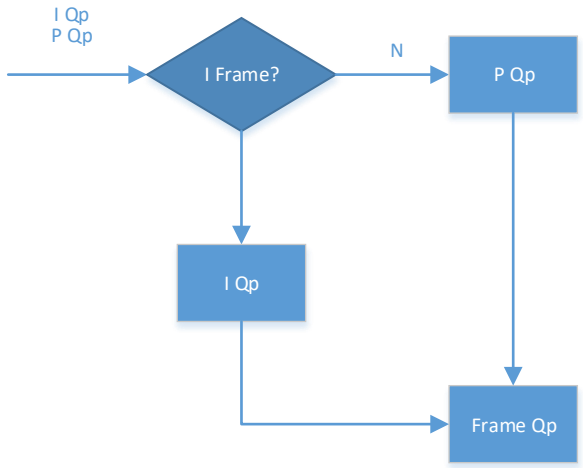
Chip	Rate Control Algorithm			
	FIXQP	CBR	VBR	AVBR
328Q/329D/326D	H.264/H.265/JPEG	H.264/H.265/JPEG	H.264/H.265	NONE
325/325DE/327DE	H.264/H.265/JPEG	H.264/H.265/JPEG	H.264/H.265	NONE
336D/336Q/339G	H.264/H.265/JPEG	H.264/H.265/JPEG	H.264/H.265	H.264/H.265
335/337DE	H.264/H.265/JPEG	H.264/H.265/JPEG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

1.2.10.1. FIXQP

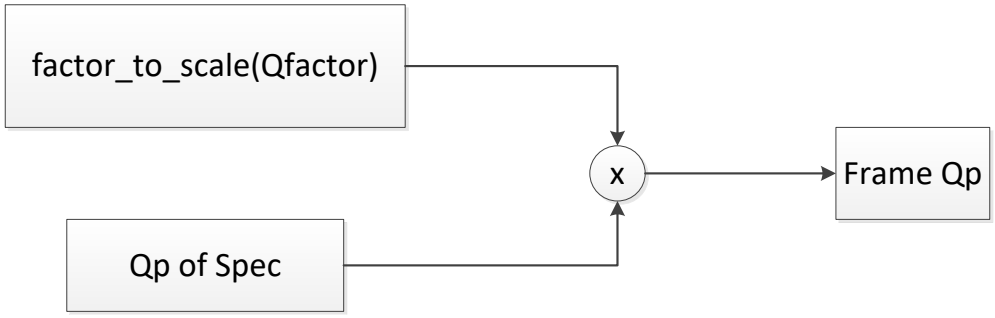
FIXQP: Fixed Quantization Parameter. At any time point, Qp of all MB/CU of the encoded image can directly be set by the user, and the Qp of I frame and P frame can be set respectively in H.264/H.265. However, some chips cannot set P frame Qp. The behavior of different chips is shown in the following table:

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

The control flow chart of H.264 and H.265 is as follows:



The control flow chart of JPEG is as follows:



It adopts the Qp table recommended by ITU-t81 K.1. The Qfactor set by the user is converted into a proportional factor according to a certain formula, and the multiplication of the two is the final Qp.

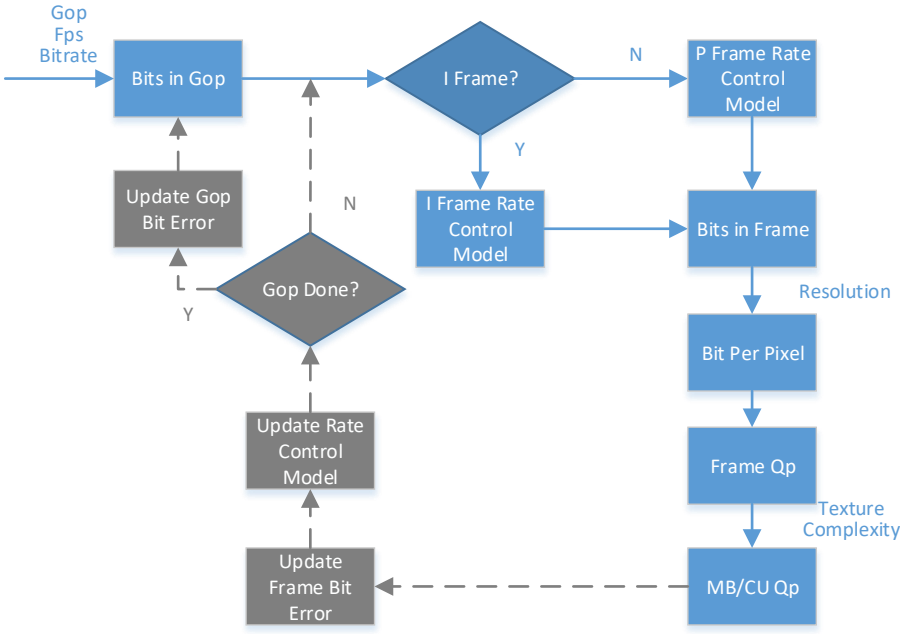
1.2.10.2. CBR

CBR: Constant Bit Rate. It can guarantee the stability of coding rate during code rate statistics.Because of different prediction methods, the size of I frame and P frame will be significantly different after coding. The bottom statistical time is based on Gop, and the bit accumulation and compensation will be realized between GOPs.

The main steps are as follows:

1. Convert Fps/Gop/bitrate set by user to bits of each GOP
2. I/P take different process, and BPP(bitperpixel) is calculated according to GOP bits and resolution
3. Map BPP to frame QP by rate control model
4. HW further adjusts MB/Cu QP based on frame QP by image texture complexity and other information
5. After coding, the rate control model is updated to achieve the continuous stability of the whole sequence, and the bit error is accumulated for the fine-tuning of bit allocation of subsequent frames
6. The bit error of the whole GOP is accumulated after the completion of the whole GOP, which is used for fine tuning of the next GOP bit allocation.

The control flow chart is as follows:

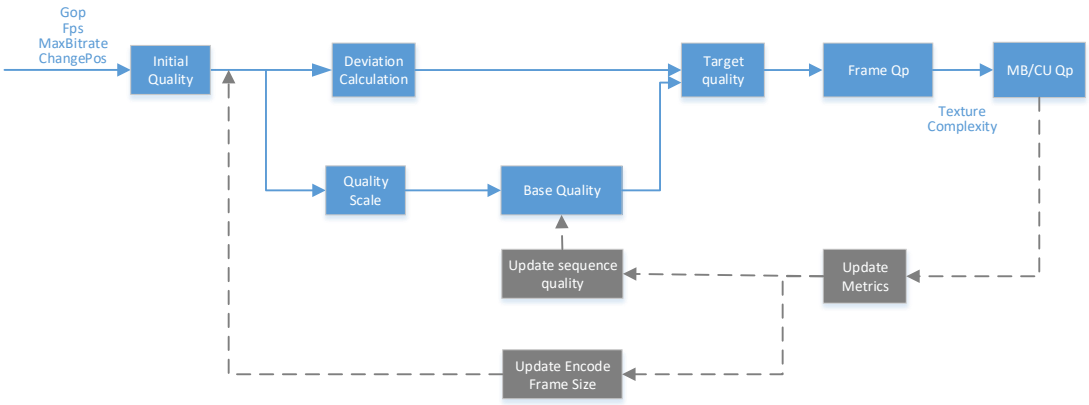


1.2.10.3. VBR

VBR: Variable Bit Rate. It allows the code rate to fluctuate within the code rate statistics time, so as to ensure the stable quality of the coding image. Take H.264 for example, users can set MaxQp, MinQp, MaxBitRate and ChangePos, details of which can be found in MI_VENC_ParamH264Vbr_t. MaxQp and MinQp are used to control the quality range of images. MaxBitRate is used for limiting maximum code rate within the statistics time. ChangePos is a percentage of the maximum bit rate, which is used to control the code rate reference line of starting to adjust QP, that is when the code rate is greater than MaxBitRate*ChangePos, the image QP will gradually adjust to MaxQp, if the image QP has reached MaxQp, the image QP will be clamped to the maximum value, the clamping effect of MaxBitRate will be lost, and the code rate may exceed MaxBitRate; if the code rate is less than MaxBitRate*ChangePos, the image QP will gradually adjust to MinQp. If the image QP has reached MinQp, the code rate has reached the maximum value and the image quality is best.
The specific process is as follows:

1. Convert the Fps/Gop/MaxBitRate/ChangePos and resolution set by user into the initial quality benchmark in order to set starting QP of the sequence.
2. HW further adjusts MB/CU QP based on frame QP by image texture complexity and other information.
3. Update rate control model after coding
4. Update the quality of the overall sequence
5. Calculate the deviation coefficient according to the difference between the current expected bit and the actual encoded bit, and decide whether the quality can be improved or reduced
6. Calculate the target quality of the current frame by the deviation coefficient and two quality coefficients
7. Map the target quality to frame QP, looping from 2.

The control flow chart is as follows:



1.2.10.4. AVBR

AVBR: Adaptive Variable Bit Rate. It allows the code rate to fluctuate within the code rate statistics time, so as to ensure the stable quality of the coding image. The rate control will detect the static state of the current scene, adopt higher rate coding when moving, and actively reduce the rate when still. Take H.264 for example, users can set MaxBitRate, ChangePos and MinStillPercent, details of which can be found in MI_VENC_ParamH264Avbr_t. MaxBitRate represents the maximum code rate in the motion scene, Minstitlpercent is the percentage of the minimum bit rate relative to the adjusted threshold bit rate in the static scene, MaxBitRate*ChangePos*MinStillPercent represents the minimum code rate in the static scene, and the target code rate will be adjusted between the minimum code rate and the maximum code rate according to the different degree of motion. MaxQp and MinQp are used to control the quality range of the image. The highest priority of the rate control is QP clamp. The rate control beyond MinQp and MaxQp will fail.

1.2.11 QPMAP

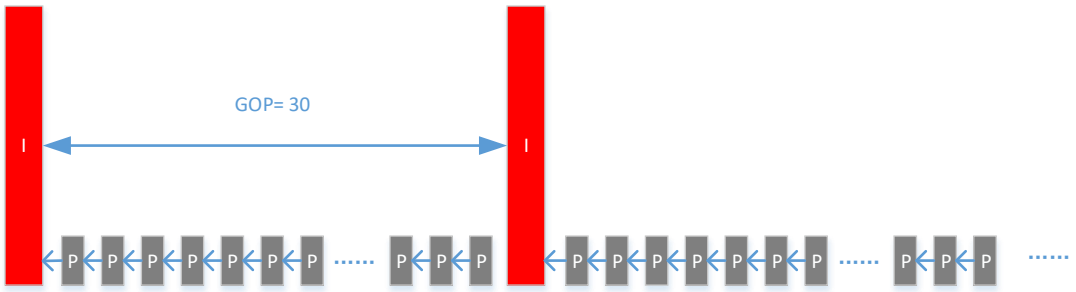
In qpmap mode, users are allowed to decide the rate control strategy freely, and absolute qpmap and relative qpmap are supported.Please refer to MI_VENC_AllocCustomMap.

1.2.12 Reference frame structure

H. 264/H.265 only supports one reference frame in a single frame, but the whole stream supports multiple reference frame buffers. For example: in LTR/TSVC3 mode, each P can only refer to one, but at most two reference frames will be saved for different P frames.
The reference frame supports 5 modes: NormalP, LTR (VI Ref IDR), LTR (VI Ref VI), TSVC-2 and TSVC-3. All reference frame structures are controlled by three parameters: u32Base, u32Enhance and bEnablePred. Please refer to MI_VENC_ParamRef_t for specific meanings. When u32Enhance is set to 0, it will be converted to NormalP reference frame structure. For other structures, please refer to the corresponding structure chart. The default structure is NormalP reference frame structure. The following details each reference diagram and parameter setting.

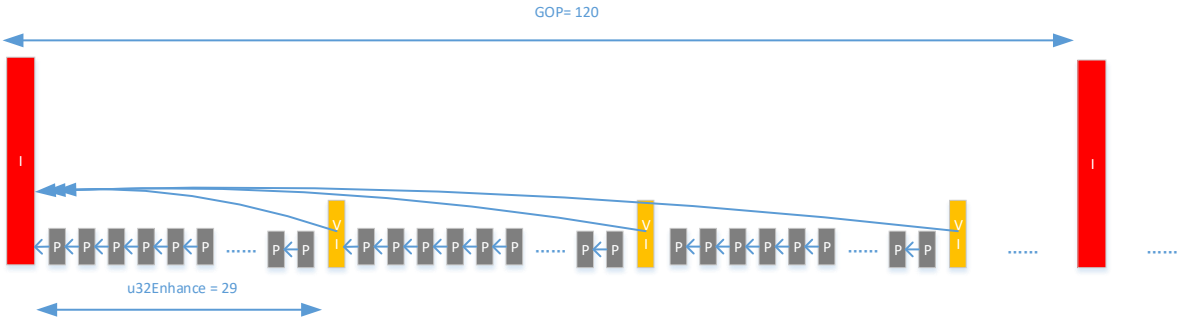
1.2.12.1. NormalP

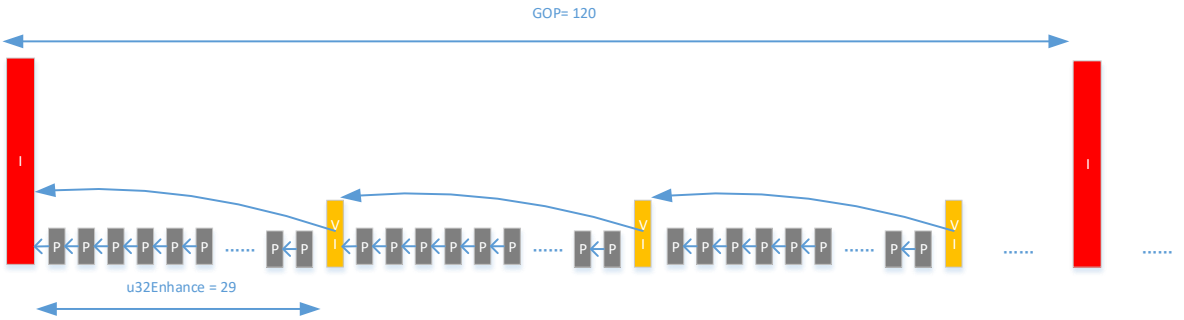
NormalP is the most basic and common reference relationship. P frame directly refers to the previous one, and its structure diagram is shown in the following figure:



1.2.12.2. LTR mode

LTR (Long Term Reference) uses long-term reference frame and short-term reference frame to realize the same frame can be referred to multiple frames. This mode defines one special P as virtual I frame (VI). Compared with all IDR, using VI can reduce the code rate while maintaining a certain error recovery power. At present, LTR supports two modes: all VI refers to the latest IDR and VI refers to the previous VI or IDR (in the case of the first VI). The structures of the two reference modes are as follows:

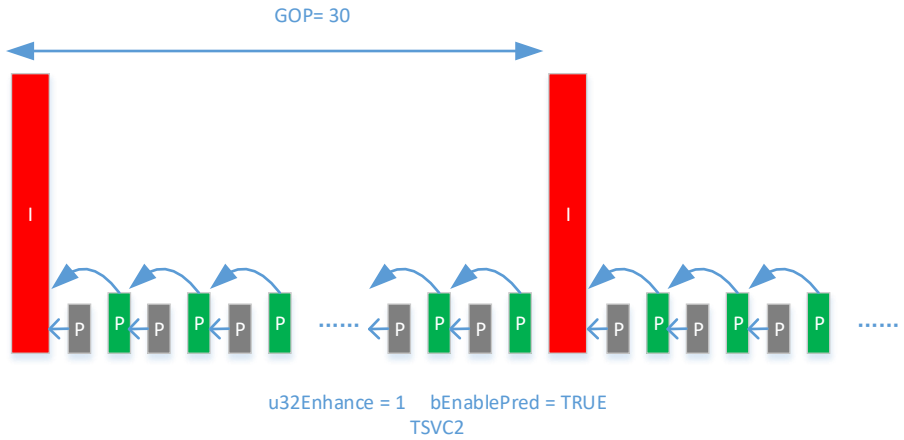




Note: an extra buffer is required to enable LTR mode.

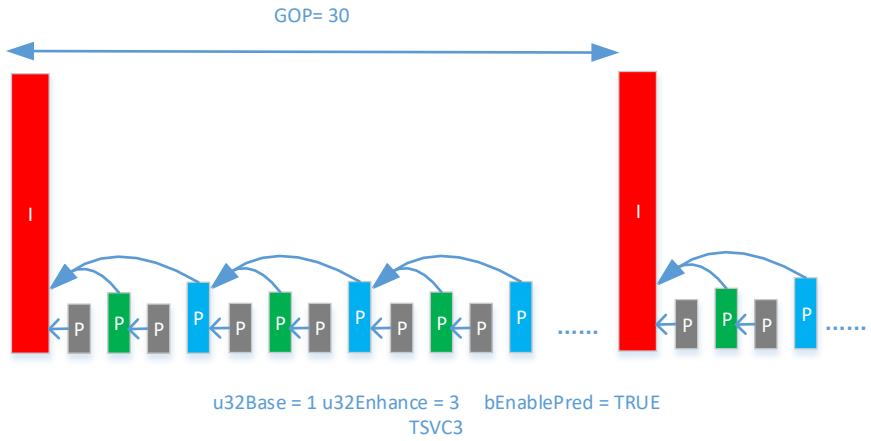
1.2.12.3. TSVC-2

TSVC-2 provides two-layers coding, and the frame rate can vary between 1/2 and full. The structure diagram is as follows:



1.2.12.4. TSVC-3

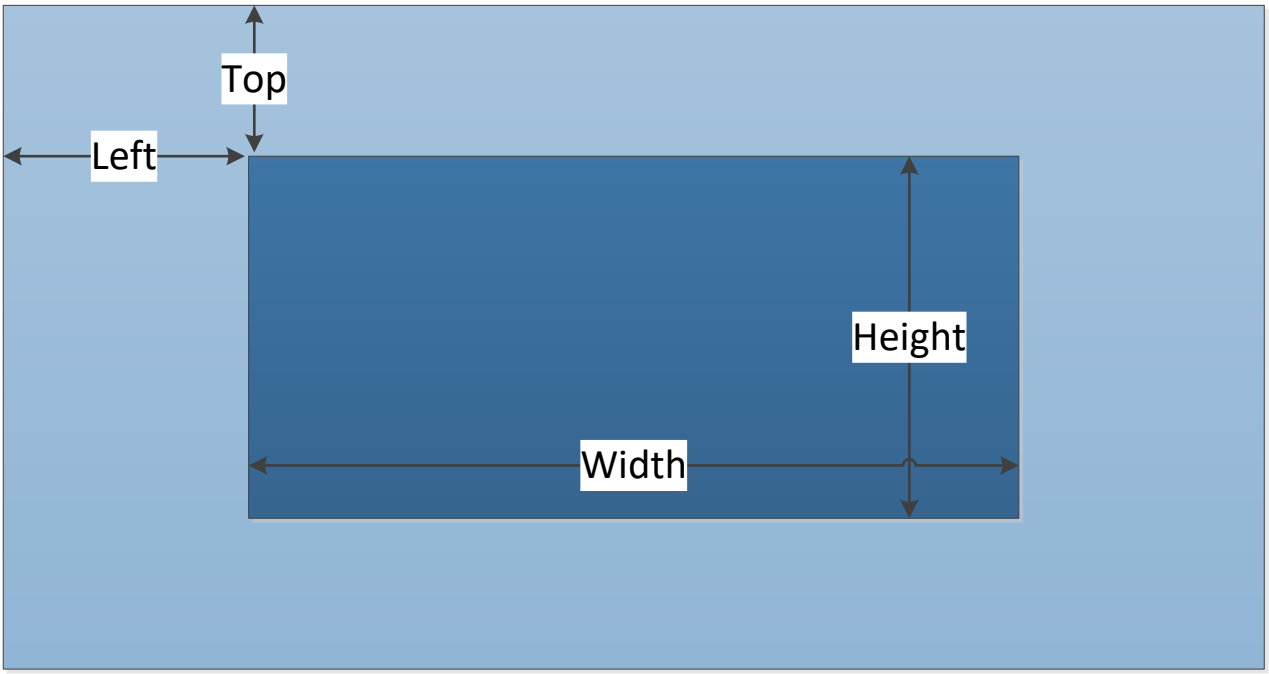
TSVC-3 provides three-layer coding, and the frame rate can vary between 1/4, 1/2, 3/4 and full. The structure diagram is as follows:



Note: an extra buffer is required to enable TSVC-3 mode.

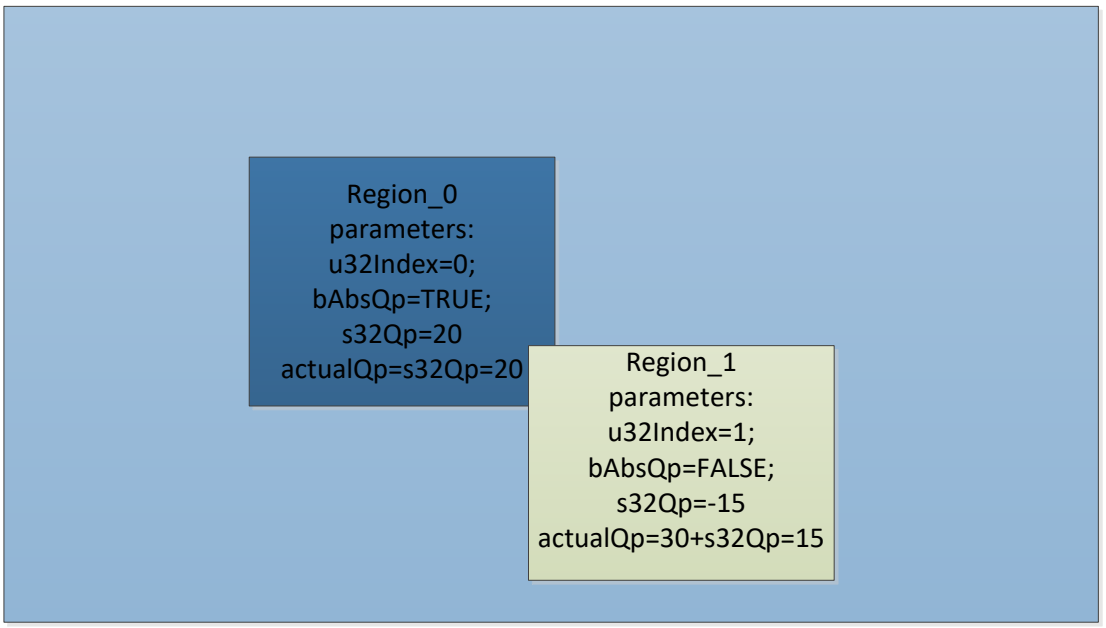
1.2.13 Crop

Crop, that is, cutting out a part of the image for encoding. User can set the starting point of clipping left/top, width and height, please refer to relevant API:MI_VENC_SetCrop. The schematic diagram is as follows:



1.2.14 ROI

ROI:Region Of Interest. Region of interest coding, users can configure ROI region to limit the image QP of this region, so as to realize the difference between the QP of this region and other image regions. The system supports H264 and h265 to set ROI, and provides 8 ROI areas for users to use at the same time.
Eight ROI regions can be overlapped with each other, and the priority of the overlapped region is raised in sequence according to the index number of 0-7, that is to say, the QP of the overlapped region is finally determined to be processed only according to the region with the highest priority. The ROI area can be configured with absolute QP and relative QP:
Absolute QP: QP in ROI area is the QP value set by the user
Relative QP: the QP in ROI area is the sum of the QP generated by rate control and the QP offset value set by the user.
In the following example, the encoded image adopts the fixqp mode, setting the image QP to 30, that is, the QP value of all macroblocks of the image is 30. ROI region 0 is set to absolute QP mode, QP value is 20, index is 0; ROI region 1 is set to relative QP mode, QP is -15, index is 1. Because the index of ROI region 0 is smaller than the index of ROI region 1, the QP of ROI region 1 with high priority is set in the overlapped image region. The QP value of region 1 is 30-15 = 15, and the QP value of region 0 except the overlapped image region is 20.



1.2.15 Low frame rate encode in non ROI area

Low frame rate encode in non ROI area, that is to say, the ROI area is normally encoded after the ROI is turned on, while the non ROI area can reduce the frame rate by setting the proportional relationship between the source and the target. According to the scale relationship, the non ROI area of some frames in a GOP will directly use the data of the corresponding area of the previous frame.The user can set the relative frame rate of non ROI area according to the actual situation, and only when the ROI is turned on can it take effect.Please refer to API:MI_VENC_SetRoiBgFrameRate,s32srcFrmRate and s32dstFrmRate only represent a proportional relationship, independent of the actual frame rate.

1.2.16 Bind Type

Generally speaking, the bind type between front level and VENC is E_MI_SYS_BIND_TYPE_FRAME_BASE, that is, the front level completes a frame buffer completely, and then outputs it to the Venc. In this mode, at least three frame buffers need to be allocated. if the front level is VPE, there are two bind types which can save memory: E_MI_SYS_BIND_TYPE_HW_RING and E_MI_SYS_BIND_TYPE_REALTIME:
E_MI_SYS_BIND_TYPE_HW_RING means that VPE and VENC can read and write on the same one frame buffer in same time. In addition, 335/337DE also supports reading and writing on half frame buffer; please refer to MI_VENC_SetInputSourceConfig for details.
E_MI_SYS_BIND_TYPE_REALTIME means that there is no need to allocate additional DRAM between VPE and VENC, because VPE and VENC are directly hardware connected.

Different chip and encode protocols support different bind type as follows:

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

1.2.17 Output buffer configuration of coded bitstream

Different chips support different configurations due to different SW architectures. The default configureations are as follows:

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

The above is the default output buffer configuration of each chip. User can specify the output buffer configuration by MI_VENC_CreateChn. Take H.264 as an example, user can specify the output buffer configuration by setting stVeAttr.stAttrH264e.u32BufSize,if the value is 0, the default configuration is used. In 328Q/329D/326D, frame mode was used, that is to say, for each stream of coding, a specific size buffer will be allocated, generally the size used in the actual encoding will be smaller than the allocated size, which will cause memory fragmentation; while in the ring pool mode, each channel will separately allocate an output ring pool, and each coding will obtain the maximum free buffer from the pool, and then record the size of the actual coding used, and recycle output ring pool, which effectively improves the utilization of the buffer and reduces memory fragmentation.

1.3. Module list

API name	Features
MI_VENC_GetChnDevid	Get the device id of the encoding channel
MI_VENC_SetModParam	Set the module parameters related to the encoding
MI_VENC_GetModParam	Get code related module parameters
MI_VENC_CreateChn	Create a code channel
MI_VENC_DestroyChn	Destroy code channel
MI_VENC_ResetChn	Reset code channel
MI_VENC_StartRecvPic	Turn on the encoding channel to receive the input image
MI_VENC_StartRecvPicEx	Turn on the encoding channel to receive the input image, and automatically stop receiving images after the specified number of frames is exceeded.
MI_VENC_StopRecvPic	Stop encoding channel to receive input image
MI_VENC_Query	Query code channel status
MI_VENC_SetChnAttr	Set the encoding properties of the encoding channel
MI_VENC_GetChnAttr	Get the encoding attribute of the encoding channel
MI_VENC_GetStream	Get the encoded code stream.
MI_VENC_ReleaseStream	Release stream buffer
MI_VENC_GetStreamBufInfo	Get the physical address and size of the stream buffer
MI_VENC_InsertUserData	Insert user data
MI_VENC_SetMaxStreamCnt	Set the maximum stream cache frame number
MI_VENC_GetMaxStreamCnt	Get the maximum number of stream cache frames
MI_VENC_RequestIdr	Request IDR frame
MI_VENC_EnableIdr	Enable IDR frame
MI_VENC_SetH264IdrPicId	Set the idr_pic_id of the IDR frame
MI_VENC_GetH264IdrPicId	Get the idr_pic_id of the IDR frame
MI_VENC_GetFd	Obtain the device file handle corresponding to the encoding channel
MI_VENC_CloseFd	Close the device file handle corresponding to the encoding channel
MI_VENC_SetRoiCfg	Set the region of interest encoding configuration for the encoding channel
MI_VENC_GetRoiCfg	Get the region of interest encoding configuration for the encoding channel
MI_VENC_SetRoiBgFrameRate	Set the frame rate configuration of the non-interest area of the encoding channel
MI_VENC_GetRoiBgFrameRate	Get the frame rate configuration of the non-interest area of the encoding channel
MI_VENC_SetH264SliceSplit	Set the slice split configuration of H.264 encoding
MI_VENC_GetH264SliceSplit	Get the slice split configuration of H.264 encoding
MI_VENC_SetH264InterPred	Set the interframe prediction configuration of H.264 encoding

API name	Features
MI_VENC_GetH264InterPred	Get the interframe prediction configuration of H.264 encoding
MI_VENC_SetH264IntraPred	Set the intra prediction configuration of H.264 encoding
MI_VENC_GetH264IntraPred	Get the intra prediction configuration of H.264 encoding
MI_VENC_SetH264Trans	Set the transform and quantization configuration of H.264 encoding
MI_VENC_GetH264Trans	Get the transform and quantization configuration of H.264 encoding
MI_VENC_SetH264Entropy	Set the entropy encoding configuration of H.264 encoding
MI_VENC_GetH264Entropy	Get the entropy encoding configuration of H.264 encoding
MI_VENC_SetH265InterPred	Set the interframe prediction configuration of H.265 encoding
MI_VENC_GetH265InterPred	Get the interframe prediction configuration of H.265 encoding
MI_VENC_SetH265IntraPred	Set H.265 encoded intra prediction configuration
MI_VENC_GetH265IntraPred	Get H.265 encoded intra prediction configuration
MI_VENC_SetH265Trans	Set H.265 coded transform, quantization configuration
MI_VENC_GetH265Trans	Get H.265 encoded transform, quantization configuration
MI_VENC_SetH264Dblk	Set the deblocking configuration of H.264 encoding
MI_VENC_GetH264Dblk	Get the deblocking configuration of H.264 encoding
MI_VENC_SetH265Dblk	Set the deblocking configuration of H.265 encoding
MI_VENC_GetH265Dblk	Get the H.264 5 coded deblocking configuration
MI_VENC_SetH264Vui	Set H.264 encoded VUI configuration
MI_VENC_GetH264Vui	Get H.264 encoded VUI configuration
MI_VENC_SetH265Vui	Set the H.265 encoded VUI configuration
MI_VENC_GetH265Vui	Get H.265 encoded VUI configuration
MI_VENC_SetH265SliceSplit	Set the slice split configuration of H.265 encoding
MI_VENC_GetH265SliceSplit	Get the slice split configuration of H.265 encoding
MI_VENC_SetJpegParam	Set the JPEG encoded parameter set
MI_VENC_GetJpegParam	Get the JPEG encoded parameter set
MI_VENC_SetRcParam	Set channel rate control advanced parameters
MI_VENC_GetRcParam	Get channel rate control advanced parameters
MI_VENC_SetRefParam	Set H.264/H.265 encoding channel advanced frame skipping reference parameters
MI_VENC_GetRefParam	Get advanced frame skipping reference parameters for H.264/H.265 encoding channel
MI_VENC_SetCrop	Set the cropping properties of VENC
MI_VENC_GetCrop	Get the cropping properties of VENC
MI_VENC_SetFrameLostStrategy	Set the configuration of the drop frame policy when the instantaneous bit rate exceeds the threshold.

API name	Features
MI_VENC_GetFrameLostStrategy	Obtain the configuration of the frame loss policy when the instantaneous bit rate exceeds the threshold.
MI_VENC_SetSuperFrameCfg	Set oversized frame processing configuration
MI_VENC_GetSuperFrameCfg	Get large frame processing configuration
MI_VENC_SetRcPriority	Set the priority type of rate control
MI_VENC_GetRcPriority	Get the priority type of rate control
MI_VENC_AllocCustomMap	Allocate the memory for Custom Map used by smart encoding
MI_VENC_ApplyCustomMap	Apply the configured Custom Map
MI_VENC_GetLastHistoStaticInfo	Get the output information when the latest frame is encoded
MI_VENC_ReleaseHistoStaticInfo	Release the memory used for the output information when the latest frame is encoded
MI_VENC_SetAdvCustRcAttr	Set the customer defined advanced RC configuration parameters
MI_VENC_SetInputSourceConfig	Set H.264/H.265 input source info

1.3.1 MI_VENC_GetChnDevid

- Features

  Get the device id of the encoding channel.

- Syntax

  MI_S32 MI_VENC_GetChnDevid(MI_VENC_CHN VeChn, MI_U32 *pu32Devid);

- Form parameter

parameter name	description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
pu32Devid	Encoding device id pointer.	Output

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Chip Difference

The device ID corresponding to different coding protocols are different under different chips, as shown in the following table:

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

- Note

  This interface is called after the channel is created, and returns if the channel is not created. Because the device ID corresponding to different coding protocols may be different, if the parameter of interface to be called contains the device ID, User need to call this interface first to obtain the corresponding device ID of the current channel.

- Example

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

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

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

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

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

    return MI_SUCCESS;
}
```

- Related APIs

  No.

1.3.2 MI_VENC_SetModParam

- Features

  Set the module parameters related to the encoding.

- Syntax

  MI_S32 MI_VENC_SetModParam(MI_VENC_ModParam_t*pstModParam)

- Form parameter

parameter name	description	Input/Output
pstModParam	Encoding module parameter pointer.	Input

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - This interface is called before the channel is created, and returns if the channel has been created.
  - MI_ERR_VENC_NOT_PERM.
  - You can set the parameters of the mi_venc.ko, mi_h264e.ko, mi_h265e.ko, mi_jpge.ko modules.
  - Not support yet.

- Example

  No.

- Related APIs

  No.

1.3.3 MI_VENC_GetModParam

- Features

  Get the module parameters related to the encoding.

- Syntax

  MI_S32 MI_VENC_GetModParam(MI_VENC_ModParam_t*pstModParam)

- Parameter

Parameter name	Description	Input/Output
pstModParam	Encoding module parameter pointer.	Output

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a
- Note

  You can get the parameters of the mi_venc.ko, mi_h264e.ko, mi_h265e.ko, mi_jpge.ko modules.
  Not support yet.
- Example

  No.
- Related APIs

  No.

1.3.4 MI_VENC_CreateChn

- Features

  Create a code channel.
- Syntax

  MI_S32 MI_VENC_CreateChn(MI_VENC_CHN Vehn,MI_VENC_ChnAttr_t*pstAttr);
- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
pstAttr	Coding channel property pointer.	Input

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details
- Requirement
  - Header files：mi_common.h, mi_comm_rc.h, mi_venc.h
  - Library file: libmi.a
- Chip Difference

The supported coding specifications of different chip are shown in the following table:

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

The width and height of coding channels supported by different chips are shown in the table below:

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

➢ Note

- The encoding channel attribute consists of two parts, the encoder attribute and the rate controller attribute.
- The encoder attribute first needs to select the encoding protocol, and then assign values to the attributes corresponding to the various protocols.
- The maximum width and height of the encoder attributes must be as follows:
  MaxPicWidth∈[MIN_WIDTH,MAX_WIDTH]
  MaxPicHeight∈[MIN_HEIGHT,MAX_HEIGHT]
  PicWidth∈[MIN_WIDTH,MaxPicwidth]
  PicHeight∈[MIN_HEIGHT,MaxPicHeight]
- The maximum width and height, the channel width must be an integer multiple of MIN_ALIGN.
- Where MIN_WIDTH, MAX_WIDTH, MIN_HEIGHT, MAX_HEIGHT, MIN_ALIGN represent the minimum width, maximum width, minimum height, maximum height, and minimum alignment unit (pixels) supported by the encoding channel, respectively.

- The encoding channel can be encoded when the input image size is not greater than the maximum width and height of the channel encoded image.
- The recommended code widths are: 3840x2160 (4k*2k), 1920x1080(1080P)、1280x720 (720P), 960x540, 640x360, 704x576, 704x480, 352x288, 352x240.
- In addition to the channel width and height and profile, the encoder attribute is a static attribute. Once the code channel is created successfully, the static attribute does not support modification unless the channel is destroyed and recreated. Please refer to the MI_VENC_SetChnAttr interface description for the matters needing attention during setup.
- The rate controller attribute first needs to configure the RC mode. The JPEG capture channel does not need to configure the rate controller attribute. Other protocol type channels (H.264/H.265/MJPEG) must be configured. The rate controller attribute RC mode must match the encoder attribute protocol type.
- Three encoding protocols (H.264/H.265 /MJPEG) support three modes for rate control: CBR, VBR and FIXQP. And for different protocols, the attribute variables of the same RC mode are basically the same. Table 6-6 describes the common properties of the three RC modes.
- u32srcFrmRateNum refers to the molecular part of the frame rate of the encoding module, and u32srcFrmRateDen refers to the denominator part of the frame rate of the encoding module. u32srcFrmRateNum/u32srcFrmRateDen should be set to the actual frame rate generated by TimeRef.RC needs to count the actual frame rate and control the code rate according to u32srcFrmRateNum/u32srcFrmRateDen.If the source of the encoded image is VI,u32srcFrmRateNum/u32srcFrmRateDen should be set to the actual output frame rate of VI, because TimeRef is generated when VI output. If the actual output frame rate of VI is 30,u32srcFrmRateNum should be set to 30 and u32srcFrmRateDen should be set to 1.
- In addition to the above attributes, CBR also needs to set the average bit rate and fluctuation level.
  1) The unit of the average bit rate is kbps. The setting of the average bit rate is related to the encoding channel width and the image frame rate. The typical average bit rate settings are shown in Table 6-7. Note that the setting of the average bit rate in the table below is the setting when the channel encoding frame rate is the full frame rate (30 fps). When the user sets the code output frame rate not to the full frame rate, the ratio of the user-set frame rate to the full frame rate (30 fps) can be converted to the bit rate in the table below.

Image W/H/bitrate level	**D1 (720x576)**	**720p (1280x720)**	**1080p (1920x1080)**
Low bit rate	＜400kbps	＜800kbps	＜2000kbps
Medium code rate	400～1000Kbps	800kbps～4000Kpbs	2000～8000Kbps
High code rate	1000Kbps	4000Kbps	8000Kbps

2) The fluctuation level setting is divided into 5 levels. The larger the fluctuation level, the larger the fluctuation range of the system allowed code rate. If the fluctuation level is set high, the image quality may be smoother for some scenes with complex images and dramatic changes. It is suitable for scenes with rich network bandwidth. If the fluctuation level is set low, the code rate of the code will be relatively stable. For some images, In scenes with dramatic changes, the image quality may not be as high as the volatility level. It is suitable for scenes with less bandwidth, reserved, and temporarily unused.

- In addition to the above attributes, VBR needs to set MaxBitRate, MaxQp, MinQp.
  1) MaxBitRate: The maximum code rate allowed for the encoding channel during the rate statistics period.
  2) MaxQp: The maximum QP allowed for the image.
  3) MinQp: The minimum QP allowed for the image.
- FIXQP addition to the above properties, also need to set IQp, PQp.
  1) IQp: QP value used for fixed images in I frames.
  2) PQp: The QP value used for fixed images in P frames.

  When the I frame QP and the P frame QP are set, the I frame QP and the P frame QP can be adjusted upward or downward simultaneously according to the current bandwidth limitation. In order to reduce the breathing effect, it is recommended that the I frame QP is always 2 to 3 smaller than the QP of the P frame.

➢ Example

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

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

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

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

    return MI_SUCCESS;
}
```

- Related APIs

  No.

1.3.5 MI_VENC_DestroyChn

- Features

  Destroy the encoding channel.

- Syntax

  MI_S32 MI_VENC_DestroyChn(MI_VENC_CHN VeChn);

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - Destroy a channel that does not exist and return failure.
  - You must stop receiving images before destroying, otherwise the return fails.
  - If the OSD is enabled, the interface MI_RGN_DetachFrmChn is called to remove the OSD area from the current channel before the encoding channel is destroyed.

- Example

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

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

    return MI_SUCCESS;
}
```

- Related APIs

  No.

1.3.6 MI_VENC_ResetChn

- Features

  Reset the channel. It will clear the cached image and bitstream before calling the interface.

- Syntax

  MI_S32 MI_VENC_ResetChn(MI_VENC_CHN VeChn);

- Parameter

Parameter name	Description	Input/Output
VeChn	Channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
  - RequirementHeader files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - Reset does not exist in the channel, returning failure MI_ERR_VENC_UNEXIST.
  - If a channel does not stop receiving images and resets the channel, it returns a failure.
  - Not support yet.

- Example

  No.

- Related APIs

  No.

1.3.7 MI_VENC_StartRecvPic

- Features

  Turn on the encoding channel to receive the input image.

- Syntax

  MI_S32 MI_VENC_StartRecvPic(MI_VENC_CHN VeChn);

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - If the channel is not created, the failure MI_ERR_VENC_UNEXIST is returned.
  - This interface does not determine whether the reception is currently enabled, that is, allows repeated activation without returning an error.
  - The encoder starts receiving image coding only after receiving the reception.

- Example

  Please refer to MI_VENC_CreateChn Example.

- Related APIs

  No.

1.3.8 MI_VENC_StartRecvPicEx

- Features

  Turn on the encoding channel to receive the input image, and automatically stop receiving images after the specified number of frames.

- Syntax

  MI_S32 MI_VENC_StartRecvPicEx(MI_VENC_CHN VeChn, MI_VENC_RecvPicParam_t *pstRecvParam);

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
pstRecvParam	Receives an image parameter structure pointer that specifies the number of image frames that need to be received.	Input

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - If the channel is not created, the failure MI_ERR_VENC_UNEXIST is returned.
  - If the channel has already called MI_VENC_StartRecvPic to start receiving images without stopping receiving images, or has not received enough images since the last call to MI_VENC_StartRecvPicEx, calling this interface again is not allowed.
  - This interface is used to continuously receive N frames and encode scenes. When N=0, the interface is equivalent to MI_VENC_StartRecvPic.
  - If the channel has already called MI_VENC_StartRecvPic to start receiving images, stop receiving images, and call MI_VENC_StartRecvPicEx to start encoding again, it is recommended that the user call MI_VENC_ResetChn to clear the image and code stream buffered by the encoding module before calling the interface.
  - If you create a jpeg channel capture, it is recommended that you call MI_VENC_StartRecvPicEx because you need to receive an integer image and stop receiving it automatically.

- Example

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

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

    //…omit other video encode assignments here.

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

    //…wait until all pictures have been encoded.

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

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

        return MI_SUCCESS;
    }
```

- Related APIs

  No.

1.3.9 MI_VENC_StopRecvPic

- Features

  Stop the encoding channel to receive the input image.

- Syntax

  MI_S32 MI_VENC_StopRecvPic(MI_VENC_CHN VeChn);

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - If the channel is not created, it returns a failure.
  - This interface does not determine whether to stop receiving currently, that is, to allow repeated stop reception without returning an error.
  - This interface is used to encode the channel to stop receiving images for encoding, and must stop receiving images before the encoding channel is destroyed or reset.
  - Calling this interface only stops receiving the original data encoding, and the stream buffer is not cleared.
  - Call the MI_VENC_StartRecvPic and MI_VENC_StartRecvPicEx interfaces to start receiving images, you can call this interface to stop receiving.

- Example

  Please refer to MI_VENC_DestroyChn Example.

- Related APIs

  No.

1.3.10 MI_VENC_Query

- Features

  Query the status of the encoded channel.

- Syntax

  MI_S32 MI_VENC_Query(MI_VENC_CHN VeChn,MI_VENC_ChnStat_t*pstStat);

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
pstStat	The status pointer of the encoding channel.	Output

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - If the channel is not created, it returns a failure.
    This interface is used to query the encoder state at the time of this function call. pstStat contains four main pieces of information:
    1) u32LeftPics represents the number of frames to be encoded. Before resetting the channel, you can determine the reset timing by querying whether there are still images

to be encoded, and prevent the frames that may need to be cleaned out during reset.

2) u32LeftStreamBytes represents the number of bytes remaining in the code stream buffer. Before resetting the channel, you can determine the reset timing by querying whether the code stream is still not processed, and prevent the code stream that may be needed from being cleared when resetting.
3) u32LeftStreamFrames represents the number of frames remaining in the code stream buffer. Before resetting the channel, you can determine the reset timing by querying whether the code stream of the image is not taken, and prevent the code stream that may be needed from being cleared when resetting.
4) u32CurPacks represents the number of code stream packets of the current frame. Make sure u32CurPacks is greater than 0 before calling MI_VENC_GetStream; otherwise, it will return an error that no buffer is available.

- The current frame may not be a complete frame (taken a part) when acquired by packet, and the number of packets representing the current complete frame when acquired by frame (or 0 if there is no frame data). When the user needs to obtain the code stream by frame, the number of packets of a complete frame needs to be queried. In this case, the query operation can usually be performed after the select succeeds. At this time, u32CurPacks is the number of packets in the current complete frame.
- In the code channel state structure, u32LeftRecvPics indicates the number of frames remaining waiting to be received after calling the MI_VENC_StartRecvPicEx interface.
- In the code channel state structure, u32LeftEncPics indicates the number of frames waiting to be encoded after calling the MI_VENC_StartRecvPicEx interface.
- If MI_VENC_StartRecvPicEx is not called, the number of u32LeftRecvPics and u32LeftEncPics is always 0.

➢ Example

Please refer to MI_VENC_GetStream Example.

➢ Related APIs

No.

1.3.11 MI_VENC_SetChnAttr

➢ Features

Set the encoding channel dynamic properties, including coded width and height, profile, and rate control parameters.

➢ Syntax

MI_S32 MI_VENC_SetChnAttr(MI_VENC_CHN VeChn,MI_VENC_ChnAttr_t*pstAttr);

➢ Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
pstAttr	Coding channel property pointer.	Input

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - The maximum width and maximum height of the encoding channel cannot be dynamically set.
  - The encoding channel attribute includes two parts: the encoder attribute and the rate controller attribute.
  - Sets the properties of an uncreated channel and returns a failure.
  - If pstAttr is empty, it returns a failure.
  - As the channel limitations and restrictions created when the coded picture size setting
  - The coding channel attributes are divided into dynamic attributes and static attributes. The attribute value of the dynamic attribute is configured when the channel is created, and can be modified before the channel is destroyed. The attribute value of the static attribute is configured when the channel is created, and cannot be modified after the channel is created.
  - This interface can only set dynamic properties in the encoding channel properties, or return failure if the static properties are set. The encoding protocol of the encoding channel, the way of acquiring the code stream (acquiring the code stream by frame or by packet), and the maximum width and height of the encoded image attribute are static attributes. In addition, the static attributes of each encoding protocol are specified by each protocol module. For details, see MI_VENC_Chn_Attr_t.
  - Set the code rate controller attribute in the code channel attribute. The rate control mode is VBR. When the rate control advanced parameter is u32MinIQp is less than u32MinQp, where u32MinQp is the rate controller attribute, this interface changes the value of u32MinIQp to u32MinQp.
  - When setting the channel width and height attributes in the encoder properties, the priority of the channel will not be restored to the default. All other parameters of the encoding channel are restored to the default values, the OSD is closed, and the stream buffer and cached image queue are cleared.

- Example

  No.

- Related APIs

  No.

1.3.12 MI_VENC_GetChnAttr

- Features

  Get the encoding channel properties.

- Syntax

  MI_S32 MI_VENC_GetChnAttr(MI_VENC_CHN VeChn,MI_VENC_ChnAttr_t*pstAttr);

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
pstAttr	Coding channel property pointer.	Output

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - Get the properties of the channel that was not created, returning the failure MI_ERR_VENC_UNEXIST.
  - If pstAttr is empty, it returns a failure.

- Example

  No.

- Related APIs

  No.

1.3.13 MI_VENC_GetStream

- Features

  Get the encoded code stream.

- Syntax

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

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
pstStream	Code stream structure pointer. The upper layer needs to point the pointer to an allocated block of memory, and also needs to allocate the memory of PackCount*sizeof(MI_VENC_Pack_t) for the member MI_VENC_Pack_t *pstPack.	Input/Output
s32MilliSec	The waiting time to call the API until the code stream is obtained. Value range: greater than or equal to 1. -1: Blocked. 0: Non-blocking. Greater than 0: Timeout, which is relative time, in ms.	Input

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

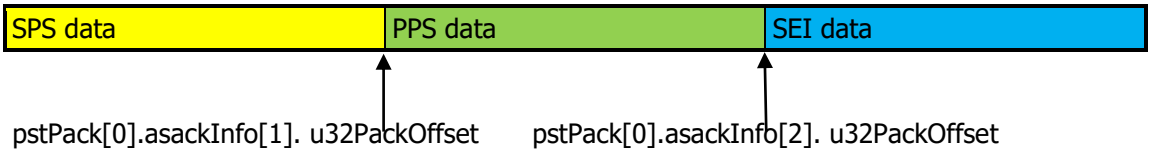
- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - If the channel is not created, the return fails.
  - Returns MI_ERR_VENC_NULL_PTR if pstStream is empty.
  - If s32MilliSec is less than -1, return MI_ERR_VENC_ILLEGAL_PARAM.
  - Support timeout mode acquisition. Support for select/poll system calls.
  - When s32MilliSec=0, it is non-blocking acquisition, that is, if there is no data in the buffer, it returns the failure MI_ERR_VENC_BUF_EMPTY. When s32MilliSec=-1, it is blocked, that is, if there is no data in the buffer, it will return to the success if it waits for data. When s32MilliSec>0, it is timeout, that is, if there is no data in the buffer, it will wait for the timeout time set by the user. If there is data within the set time, the return is successful, otherwise the return timeout fails.
  - The code stream structure MI_VENC_Stream_t contains four parts:
    1) the code stream packet information pointer pstPack points to a memory space of a group of MI_VENC_Pack_t, which space is allocated by the caller. The size is u32PackCount x sizeof(MI_VENC_Pack_t), which can be obtained by calling MI_VENC_Query after selection.
    2) U32PackCount specifies the number of MI_VENC_Pack_t in the pstPack, which is the number of code streams in a complete frame.
    3) U32Seq represents the sequence of a frame.

4) The stH264Info/stJpegInfo/stMpeg4Info/stH265Info combinations are the stream feature, which contains the characteristic information of code stream corresponding to different coding protocols. The output of the characteristic information of code stream is used to support the upper application of users.

- This interface should be paired with MI_Venc_ReleaseStream for use. The system will not actively release the bitstream cache after the user obtains the bitstream. The user needs to release the acquired bitstream cache in time; otherwise, it may cause the bitstream buffer to be full, thereby affecting the encoder coding.
- It is recommended that the user use the select method to obtain the code stream, and follow the following process:
  1) call the MI_VENC_Query function to query the encoding channel state.
  2) ensure that u32CurPacks and u32LeftStreamFrames are greater than 0 at the same time.
  3) call malloc to allocate u32CurPacks packet information structures.
  4) call MI_VENC_GetStream to obtain the encoded code stream.
  5) Call MI_VENC_ReleaseStream to release the code stream buffer.
- Taking H.264 as an example to introduce the code stream structure stored in pstPack[0], the corresponding information of pstPack[0] is as follows:

  pstPack[0].u32DataNum=3,
  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 ;
  The pstPack[0] contains three NALU packages, including SPS/PPS/SEI, as shown in the figure below:



➢ Example

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

    pFile = fopen("stream.h264","wb");
    if (pFile == NULL)
    {
        return E_MI_ERR_FAILED;
    }
    s32VencFd = MI_VENC_GetFd(VeChn);
    do{
        FD_ZERO(&read_fds);
        FD_SET(s32VencFd, &read_fds);
        s32Ret = select(s32VencFd+1, &read_fds, NULL, NULL, NULL);
        if (s32Ret < 0)
        {
            printf("select err\n");
            return E_MI_ERR_FAILED;
        }
        else if (0 == s32Ret)
        {
            printf("timeout\n");
            return E_MI_ERR_FAILED;
        }
        else
        {
            if (FD_ISSET(s32VencFd, &read_fds))
            {
                s32Ret = MI_VENC_Query(VeChn, &stStat);
                if (s32Ret != MI_SUCCESS)
                {
                    return E_MI_ERR_FAILED;
                }
    /**************************************************
    suggest to check both u32CurPacks and u32LeftStreamFrames at
    the same time,for example:
                if (0 == stStat.u32CurPacks || 0 == stStat.u32LeftStreamFrames)
                {
```

```
                    continue;
                }
    ******************************************************/
                if (0 == stStat.u32CurPacks)
                {
                    continue;
                }
                stStream.pstPack =
(MI_VENC_Pack_t*)malloc(sizeof(MI_VENC_Pack_t)*stStat.u32CurPacks);
                if (NULL == stStream.pstPack)
                {
                    return E_MI_ERR_FAILED;
                }
                stStream.u32PackCount = stStat.u32CurPacks;
                s32Ret = MI_VENC_GetStream(VeChn, &stStream, -1);
                if (MI_SUCCESS != s32Ret)
                {
                    free(stStream.pstPack);
                    stStream.pstPack = NULL;
                    return E_MI_ERR_FAILED;
                }
                for (i=0; i<stStream.u32PackCount; i++)
                {

fwrite(stStream.pstPack[i].pu8Addr+stStream.pstPack[i].u32Offset, 1,
                            stStream.pstPack[i].u32Len-stStream.pstPack[i].u32Offset,
pFile);
                }
                s32Ret = MI_VENC_ReleaseStream(VeChn, &stStream);
                if (MI_SUCCESS != s32Ret)
                {
                    free(stStream.pstPack);
                    stStream.pstPack=NULL;
                    return E_MI_ERR_FAILED;
                }
                free(stStream.pstPack);
                stStream.pstPack = NULL;
            }
        }
        u32FrameIdx++;
    }while(u32FrameIdx<0xff);
    fclose(pFile);

    return s32Ret;
}
```

- Related APIs

  No.

1.3.14 MI_VENC_ReleaseStream

- Features

  Release stream buffer.

- Syntax

  MI_S32 MI_VENC_ReleaseStream(MI_VENC_CHN VeChn,MI_VENC_Stream_t *pstStream);

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
pstStream	Code stream structure pointer.	Input

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - If the channel is not created, an error code is returned MI_ERR_VENC_UNEXIST.
  - in case pstStream Empty, return error code MI_ERR_VENC_NULL_PTR.
  - This interface should be MI_VENC_GetStream Paired and used, the user must release the acquired code stream cache in time after obtaining the code stream, otherwise the code stream may be caused. Buffer Full, affecting the encoder encoding, and the user must release the already acquired stream buffer in the order in which they were first released first.
  - After the code channel is reset, all unreleased stream packets are invalid and cannot be used or released.
  - Invalid stream cache.
  - Releasing an invalid stream will return a failure MI_ERR_VENC_ILLEGAL_PARAM.

- Example

  Please refer to MI_VENC_GetStream Example.

- Related APIs

  No.

1.3.15 MI_VENC_InsertUserData

- Features

  Insert user data.

- Syntax

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

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
pu8Data	User data pointer.	Input
u32Len	User data length. Value range: (0,1024), in bytes.	Input

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - If the channel is not created, it returns a failure.
  - In case of pu8Data, if it is empty, it will return failure.
  - Insertion of user data only supports H.264/H.265.
  - H.264 Protocol channels are allocated at the same time. The block memory space is used to cache user data, and the data size of each segment of the user does not exceed 1kbyte. If the user inserts 4 extra data blocks, or inserts piece of user data greater than 1kbyte, this interface will return an error. Each piece of user data is inserted as an SEI package before the next I frame. After a certain user data packet is encoded and sent, H.264 The memory space of the user data buffered in the channel is cleared to store new user data.

- Example

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

    return s32Ret;
}
```

- Related APIs

  No.

1.3.16 MI_VENC_SetMaxStreamCnt

- Features

  Set the maximum number of cache frames for the stream.

- Syntax

  MI_S32 MI_VENC_SetMaxStreamCnt(MI_VENC_CHN VeChn,MI_U32 u32MaxStrmCnt);

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
u32MaxStrmCnt	Maximum number of stream buffer frames.	Input

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - This interface is used to set the code stream of the encoding channel. Buffer The maximum number of streams that can be cached.
  - If the number of cached stream frames has reached the maximum number of stream frames, the current image to be encoded is due to the code stream. Buffer Full without being encoded.
  - The maximum number of streams is specified by the system when the channel is created. The default value is 3.
  - This interface can be called after the channel is created and before the code channel is destroyed. Take effect before the next frame starts encoding. This interface allows multiple calls. It is recommended to set the encoding before starting the encoding. It is not recommended to dynamically adjust during the encoding process.

- Example

  No.

- Related APIs

  No.

1.3.17 MI_VENC_GetMaxStreamCnt

- Features

  Get the maximum number of cache frames in the stream.

- Syntax

  MI_S32 MI_VENC_GetMaxStreamCnt(MI_VENC_CHN VeChn,MI_U32 *pu32MaxStrmCnt);

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
pu32MaxStrmCnt	A pointer to the maximum number of stream buffer frames.	Output

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h

- Library file: libmi.a

➢ Note

No.

➢ Example

No.

➢ Related APIs

No.

1.3.18 MI_VENC_RequestIdr

➢ Features

Request an IDR frame.

➢ Syntax

MI_S32 MI_VENC_RequestIdr(MI_VENC_CHN VeChn,MI_BOOL bInstant);

➢ Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
bInstant	Whether to enable encoding of IDR frames immediately. 0: IDR frames are coded in the shortest possible time. 1: The next frame is an IDR frame.	Input

➢ Return Value

- MI_OK: Successful
- Not MI_OK: Failed, see error code for details

➢ Requirement

- Header files：mi_common.h, mi_venc.h
- Library file: libmi.a

➢ Note

- If the channel is not created, it returns a failure.
- I Frame request, only support H.264/H.265Coding protocol.
- Since this interface is not affected by frame rate control, an IDR will be encoded when call this interface.Frequent calls will affect the stability of frame rate and code rate.

➢ Example

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

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

    return s32Ret;
}
```

- Related APIs

  No.

1.3.19 MI_VENC_EnableIdr

- Features

  Whether to enable IDR frames.

- Syntax

  MI_S32 MI_VENC_EnableIdr(MI_VENC_CHN VeChn,MI_BOOL bEnableIdr);

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
bEnableIDR	Whether the flag is enabled.	Input

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
- Note
  - If the channel is not created, it returns a failure.
  - If not enabled IDR Frame, it will not be edited after the next frame IDR Frame or I Frame until it is enabled again.
  - This interface only supports H.264/H.265 Coding protocol.
- Example

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

    return s32Ret;
}
```

- Related APIs

  No.

1.3.20 MI_VENC_SetH264IdrPicId

- Features

  Set the idr_pic_id attribute of the IDR frame.

- Syntax

  MI_S32 MI_VENC_SetH264IdrPicId(MI_VENC_CHN VeChn,MI_VENC_H264IdrPicIdCfg_t*pstH264eIdrPicIdCfg);

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
pstH264IdrPicIdCfg	The parameter pointer to idr_pic_id.	Input

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - If the channel is not created, it returns a failure.
  - Idr_pic_id The parameters are mainly determined by two parameters:
    1) enH264eIdrPicIdMode: Settings IDR frame Idr_pic_id Mode, enH264eIdrPicIdMode = E_MI_VENC_H264E_IDR_PIC_ID_MODE_AUTO Representation generated automatically by the internal process of the code Idr_pic_id, enH264eIdrPicIdMode=E_MI_VENC_H264E_IDR_PIC_ID_MODE_USR Indicates that the user is set Idr_pic_id Value.
    2) u32H264eIdrPicId: Settings IDR frame Idr_pic_id value. This value is valid only when enH264eIdrPicIdMode = E_MI_VENC_H264E_IDR_PIC_ID_MODE_USR.
  - Setting IDR Framed Idr_pic_id, only supported H.264 Coding protocol.
  - This interface can be called after the channel is created and before the code channel is destroyed. Take effect before the next frame starts encoding. This interface allows multiple calls. It is recommended to set the encoding before starting the encoding. It is not recommended to dynamically adjust during the encoding process.
  - Not support yet.

- Example

  No.

- Related APIs

  No.

1.3.21 MI_VENC_GetH264IdrPicId

- Features

  Get the configuration attribute of the idr_pic_id of the IDR frame.

- Syntax

  MI_S32 MI_VENC_GetH264IdrPicId(MI_VENC_CHN VeChn,MI_VENC_H264IdrPicIdCfg_t*pstH264eIdrPicIdCfg);

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
pstH264eIdrPicIdCfg	The parameter pointer to idr_pic_id.	Output

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - If the channel is not created, it returns a failure.
  - This interface can be called after the channel is created and before the code channel is destroyed.
  - Not support yet.

- Example

  No.

- Related APIs

  No.

1.3.22 MI_VENC_GetFd

- Features

  Get the device file handle corresponding to the encoding channel. It will be called before calling select/poll to listen for encoded data, thereby reducing CPU usage compared to polling.

- Syntax

  MI_S32 MI_VENC_GetFd(MI_VENC_CHN VeChn);

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details
- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a
- Note

  No.
- Example

  See the example in MI_VENC_GetStream.
- Related APIs

  No.

1.3.23 MI_VENC_CloseFd

- Features

  Close the device file handle corresponding to the encoding channel.
- Syntax

  MI_S32 MI_VENC_CloseFd(MI_VENC_CHN VeChn);
- Parameter

Parameter name	Description	Input/Output
VeChn	Video encoding channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details
- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a
- Note

  No.
- Example

  No.
- Related APIs

  No.

1.3.24 MI_VENC_SetRoiCfg

- Features

  Set the ROI attribute of the H.264/H.265 channel.

- Syntax

  MI_S32 MI_VENC_SetRoiCfg(MI_VENC_CHN VeChn,MI_VENC_RoiCfg_t *pstVencRoiCfg);

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
pstVencRoiCfg	ROI area parameters.	Input

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Chip Difference

Different chip and coding protocols have different restrictions on ROI parameters, as shown in the following table:

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

- Note
  - This interface is used to set H.264/H.265 Protocol coding channel ROI Regional parameters, ROI Mainly by parameters 5 Parameter decision.
    1) u32Index: System supports each channel to be set 8 One ROI Area, inside the system 0 ~ 7 Index number pair ROI Regional management, u32Index User settings indicated ROI Index number. ROI Areas can be superimposed on each other, and when overlays occur, ROI Priority between regions according to index number 0 ~ 7 Increase in turn.

2) bEnable: specify the current ROI Whether the area is enabled.
3) bAbsQp: specify the current ROI absolute use of the area QP Way or relative QP.
4) s32Qp: when bAbsQp for True Time, s32Qp Express ROI All macroblocks within the region are used QP Value, when bAbsQp for False Time, s32Qp Express ROI Relative to all macroblocks within the region QP value.
5) tRect: specify the current ROI The position coordinates of the area and size of the area. ROI starting point coordinates of the area must be within the image range.

- This interface belongs to the advanced interface. The system does not have a default. ROI The zone is enabled and the user must call this interface to start ROI.
- This interface can be set before the code channel is destroyed after the code channel is created. This interface takes effect when the next frame is called when it is called during encoding.
- It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.
- It is recommended that the user call before calling this interface. MI_VENC_GetRoiCfg Interface, get the current channel ROI Configure and then set it up.

➢ Example

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

    return s32Ret;
}
```

- Related APIs

No.

1.3.25 MI_VENC_GetRoiCfg

- Features

  Get the Roi configuration properties of the H.264/H.265 channel.

- Syntax

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

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
u32Index	H.264 protocol encoding channel ROI region index.	Input
pstVencRoiCfg	Corresponds to the configuration of the ROI area.	Output

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - This interface is used to obtain H.264 Protocol coding channel Index of ROI Configuration.
  - This interface can be called after the encoding channel is destroyed after the encoding channel is created.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.

- Example

  Please refer to MI_VENC_GetRoiCfg Example.

- Related APIs

  No.

1.3.26 MI_VENC_SetRoiBgFrameRate

- Features

  Set the non-ROI area frame rate attribute of the H.264/H.265 channel.

- Syntax

  MI_S32 MI_VENC_SetRoiBgFrameRate(MI_VENC_CHN VeChn,MI_VENC_RoiBgFrameRate_t*pstRoiBgFrmRate);

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
pstRoiBgFrmRate	Non-ROI area frame rate control parameters.	Input

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - This interface is used to set H.264 Protocol coding channel ROI The parameters of the area frame rate control.
  - This interface must be enabled before it can be called. ROI region.
  - This interface belongs to the advanced interface. The system is not enabled by default. ROI The area frame rate attribute, the user must call this interface to enable.
  - This interface can be set before the code channel is destroyed after the code channel is created. This interface takes effect when the next frame is called when it is called during encoding.
  - Input frame rate when setting channel frame rate control attribute SrcFrmRate Greater than the output frame rate DstFrmRate And DstFrmRate greater or equal to 0 Or at the same time -1.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.
  - It is recommended that the user call MI_VENC_GetRoiBgFrameRate interface before calling this interface.
  - Get the current channel's non ROI Regional frame rate configuration, and then set.
  - For non ROI, the frame rate of the area is lower than ROI Frame skipping or Svc-t coding. Frame skip reference or Svc-t Encoding, if set is not ROI The frame rate of the area is lower than ROI Area, the actual coded non ROI The regional target frame rate may be greater than the user configured target frame rate because Base Non-layer ROI Area cannot be encoded as Pskip Piece. For example, before calling this interface to set the frame rate ratio of non ROI area to 2:1, MI_VENC_SetRefParam is also called to set the long reference frame, which will result in the actual frame rate ratio of non ROI area being less than 2:1.

- Example

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

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

    return s32Ret;
}
```

- Related APIs

No.

1.3.27 MI_VENC_GetRoiBgFrameRate

- Features

Get the non-Roi area frame rate configuration attribute of the H.264/H.265 channel.

- Syntax

MI_S32 MI_VENC_GetRoiBgFrameRate(MI_VENC_CHN VeChn,MI_VENC_RoiBgFrameRate_t*pstRoiBgFrmRate);

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
pstRoiBgFrmRate	Configuration of non-ROI area frame rate.	Output

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - This interface can be called after the encoding channel is destroyed after the encoding channel is created.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.

- Example

  Please refer to MI_VENC_SetRoiBgFrameRate Example.

- Related APIs

  No.

1.3.28 MI_VENC_SetH264SliceSplit

- Features

  Set the slice split attribute of the H.264 channel.

- Syntax

  MI_S32 MI_VENC_SetH264SliceSplit(MI_VENC_CHN VeChn,MI_VENC_ParamH264SliceSplit_t*pstSliceSplit);

- Parameter

Parameter name	Description	Input/Output
VeChn	Code channel number. Value range: [0, VENC_MAX_CHN_NUM).	Input
pstSliceSplit	H.264 code stream slice segmentation parameters.	Input

- Return Value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Requirement
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - This interface is used to set H.264 Protocol encoding channel code stream segmentation
  - Slice Split attribute is mainly Two Parameter decision:
    1) bSplitEnable: Whether the current frame is in progress Slice segmentation.
    2) u32SliceRowCount: Slice Split according to the macro block line. u32SliceRowCount represents the number of lines of each splice in the image macro block. And when encoding to the last few lines of the image, insufficient u32SliceRowCount will occur when the remaining macroblock lines are divided into one Slice.
  - The default value of bSplitEnable is false. When bSplitEnable is set to true, the u32SliceRowCount macroblock row is used as a slice to encode separately. After each slice is encoded, the user can get the stream data of the slice. It is better not to set u32SliceRowCount too small. The smaller the u32SliceRowCount is, the more frequent the user receives the notification, the greater the system overhead, and the easier it is to cause bit stream output memory fragmentation,and reduce the use efficiency of the bit stream output memory.
  - It is recommended to call this interface after creating the channel and before starting the channel, to reduce the number of calls during the encoding process. It takes effect at the next frame.

  - It is recommended that the user call before calling this interface. MI_VENC_GetH264SliceSplit Interface, get the current channel SliceSplit Configure and then set it up.

- Example

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

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

    return s32Ret;
}
```

- Related APIs

  No.

1.3.29 MI_VENC_GetH264SliceSplit

- description

  Get the slice split attribute of the H.264 channel

- prototype

  MI_S32 MI_VENC_GetH264SliceSplit(MI_VENC_CHN VeChn,MI_VENC_ParamH264SliceSplit_t*pstSliceSplit);

- parameters

parameter name	description	input/output
VeChn	Enocode channel number.. Value range：[0,VENC_MAX_CHN_NUM).	input
pstSliceSplit	H.264 stream slice splite parameters.	output

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to obtain the slice segmentation attribute of the H.264 protocol encoding channel.
  - This interface can be called before the encoding channel is destroyed and after the encoding channel is created.
  - It is recommended that the user call this interface before starting the encoding and after creating the channel, reducing the number of calls during the encoding process

- Example

  See the example of MI_VENC_SetH264SliceSplit

- related topic

  no.

1.3.30 MI_VENC_SetH264InterPred

- description

  Set the inter prediction property of the H.264 protocol encoding channel.

- prototype

  MI_S32 MI_VENC_SetH264InterPred(MI_VENC_CHN VeChn,MI_VENC_ParamH264InterPred_t*pstH264InterPred);

- parameters

Parameter Name	Description	Input/Output
VeChn	Encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH264InterPred	Inter-prediction configuration of the H.264 protocol encoding channel.	input

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to set the interframe prediction configuration of the H.264 protocol encoding channel.
  - The properties of inter prediction are mainly determined by seven parameters:
    1) u32HWSize: the size of the horizontal search window when inter prediction.
    2) u32VWSize: The size of the vertical search window for inter prediction.
    3) bInter16x16PredEn: 16x16 block inter prediction enable flag.
    4) bInter16x8PredEn: 16x8 block inter prediction enable flag.
    5) bInter8x16PredEn: 8x16 block inter prediction enable flag.
    6) bInter8x8PredEn: 8x8 block inter prediction enable flag.
    7) bExtedgeEn: Interframe prediction expansion search enable. When inter-prediction is performed on a macroblock on an image boundary, the range of the search window may exceed the boundary of the image. At this time, the expansion search enables the image to be expanded and inter-frame predicted. If the extended search enable is turned off, no prediction will be made for search windows that are outside the range of the image.
  - This interface belongs to the advanced interface, which can be called by the user. It is recommended not to call, and the system will have default values. The default search window size will vary depending on the resolution of the image, and the other five predictive enable switches are all enabled by default.
  - The predictive switch of bInter16x16PredEn can only be turned on.
  - This interface can be set before the code channel is destroyed after the code channel is created. When this interface is called during the encoding process, it takes effect until the next frame.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process
  - Users are advised before calling this interface, first call MI_VENC_GetH264InterPred interface configured to obtain InterPred current channel coding, and then set it.

- Example

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

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

    return s32Ret;
}
```

- related topic

  no.

1.3.31 MI_VENC_GetH264InterPred

- description

  Obtain the inter prediction property of the H.264 protocol coding channel.

- prototype

  MI_S32 MI_VENC_GetH264InterPred(MI_VENC_CHN VeChn,MI_VENC_ParamH264InterPred_t*pstH264InterPred);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH264InterPred	Inter-prediction parameters of the H.264 protocol coding channel.	output

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to obtain the interframe prediction mode of the H.264 protocol coding channel.
  - This interface can be called after the encoding channel is destroyed after the encoding channel is created.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.

- Example

  See the example of MI_VENC_SetH264InterPred.

- related topic

  no.

1.3.32 MI_VENC_SetH264IntraPred

- description

  Set the intra prediction property of the H.264 protocol encoding channel.

- prototype

  MI_S32 MI_VENC_SetH264IntraPred(MI_VENC_CHN VeChn,MI_VENC_ParamH264IntraPred_t*pstH264IntraPred);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH264IntraPred	Intra prediction configuration of the H.264 protocol coding channel.	input

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to set the intra prediction configuration of the H.264 protocol encoding channel.
  - The inter-prediction attribute is mainly determined by four parameters:

    1) bIntra16x16PredEn: 16x16 block intra prediction enable flag.
    2) bIntraNxNPredEn: NxN block intra prediction enable flag. Among them, NxN means 4x4 and 8x8.
    3) bIpcmEn: IPCM block intra prediction enable flag.
    4) constrained_intra_pred_flag: For the specific meaning, please refer to the H.264 protocol.
  - This interface is a high-level interface that can be called by the user. It is not recommended to call. The system has default values. bIntra16x16PredEn and bIntraNxNPredEn are enabled by default. bIpcmEn has different default values for different chips. Constrained_intra_pred_flag defaults to 0
  - There must be one of the two intra prediction enable switches, bIntra16x16PredEn and bIntraNxNPredEn, and the system does not support that all two switches are turned off.
  - This interface can be set before the code channel is destroyed after the code channel is created. When this interface is called during the encoding process, it takes effect until the next I frame.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.
  - It is recommended that the user call the MI_VENC_GetH264IntraPred interface to obtain the IntraPred configuration of the current encoding channel before calling this interface, and then set it.

- Example

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

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

    return s32Ret;
}
```

- related topic

  no.

1.3.33 MI_VENC_GetH264IntraPred

- description

  Get the intra prediction attribute of the H.264 protocol encoding channel.

- prototype

  MI_S32 MI_VENC_GetH264IntraPred(MI_VENC_CHN VeChn,MI_VENC_ParamH264IntraPred_t*pstH264IntraPred);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH264IntraPred	Intra prediction parameters of the H.264 protocol coding channel.	output

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to obtain the intra prediction mode of the H.264 protocol coding channel.
  - This interface can be called after the encoding channel is destroyed after the encoding channel is created.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.

- Example

  See the example of MI_VENC_SetH264IntraPred.

- related topic

  no.

1.3.34 MI_VENC_SetH264Trans

- description

  Set the transform and quantized attributes of the H.264 protocol encoding channel.

- prototype

  MI_S32 MI_VENC_SetH264Trans(MI_VENC_CHN VeChn,MI_VENC_ParamH264Trans_t*pstH264Trans);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH264Trans	The transform and quantization attributes of the H.264 protocol coding channel.	input

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: mi_venc.ko, libmi_venc.so

- note
  - This interface is used to set the transformation and quantization configuration of the H.264 protocol encoding channel.
  - Transform, quantization property mainly by three parameters.
    1) u32IntraTransMode: The transform attribute of the intra prediction macroblock. u32IntraTransMode=0 indicates that 4x4 transform and 8x8 transform are supported for intra prediction macroblocks ; u32IntraTransMode=1indicates that only 4x4 transform is supported for intra prediction macroblocks ; u32IntraTransMode=2 indicates that only 8x8 transform is supported for intra prediction macroblocks. The 8x8 transform can be selected only when the encoding channel protocol is highprofile. That is, under the baselineprofile and mainprofile, the system only supports the configuration of u32IntraTransMode=1.
    2) u32InterTransMode: The transform attribute of the inter prediction macroblock. u32InterTransMode=0 indicates that 4x4 transform and 8x8 transform are supported for inter-predicted macroblocks ; u32InterTransMode=1indicates that only 4x4 transform is supported for inter-predicted macroblocks ; u32InterTransMode=2 indicates that only 8x8 transform is supported for inter-predicted macroblocks. The 8x8 transform can be selected only when the encoding channel protocol is highprofile. That is, under the baselineprofile and mainprofile, the system only supports the configuration of u32InterTransMode=1.
    3) s32ChromaQpIndexOffset: For details, see the chroma_qp_index_offset of H.264 protocol.
  - This interface is a high-level interface that can be called by the user. It is not recommended to call. The system has default values. The default parameters are set according to different profiles. The default trans parameters of the system under different profiles are shown in the table below.

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

- This interface can be set before the code channel is destroyed after the code channel is created. When this interface is called during the encoding process, it takes effect until the next I frame.
- It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.
- Users are advised before calling this interface, first call MI_VENC_GetH264Trans interfaces, access to pre-coded channel when the trans configuration, and then set it.

➢ Example

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

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

    return s32Ret;
}
```

- related topic

  no.

1.3.35 MI_VENC_GetH264Trans

- description

  Obtain the transform and quantization attributes of the H.264 protocol coding channel.

- prototype

  MI_S32 MI_VENC_GetH264Trans(MI_VENC_CHN VeChn,MI_VENC_ParamH264Trans_t *pstH264Trans);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH264Trans	The transform and quantization parameters of the H.264 protocol coding channel.	output

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to obtain the transform and quantization configuration of the H.264 protocol coding channel.
  - This interface can be called after the encoding channel is destroyed after the encoding channel is created.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.

- Example

  See the example of MI_VENC_SetH264Trans.

- related topic

  no.

1.3.36 MI_VENC_SetH264Entropy

- description

  Set the entropy coding mode of the H.264 protocol coding channel.

- prototype

  MI_S32 MI_VENC_SetH264Entropy(MI_VENC_CHN VeChn,MI_VENC_ParamH264Entropy_t*pstH264EntropyEnc);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH264EntropyEnc	Entropy coding mode of the H.264 protocol coding channel.	input

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to set the entropy coding configuration of the H.264 protocol coding channel.
  - Transform quantization property mainly by two parameters:
    1) u32EntropyEncModeI: entropy coding mode I frame, u32EntropyEncModeI = 0 indicates that the I-frame coding using cavlc, u32EntropyEncModeI = 1 indicates an I frame encoding cabac used.
    2) u32EntropyEncModeP: entropy encoding of P frames, u32EntropyEncModeP = 0 indicates coding P frames use cavlc, u32EntropyEncModeP = 1 indicates P-frame coding using cabac.
  - The entropy coding mode of the I frame and the entropy coding mode of the P frame can be set separately.
  - Baselineprofile does not support cabac encoding, supports cavlc encoding, mainprofile and highprofile support cabac encoding and cavlc encoding.
  - The Cabac encoding method requires a larger amount of computation than the cavlc encoding method, but generates less code streams. If the system performance is not rich enough, it is recommended that users can take I frame cavlcencoding, P frame cabac encoding.
  - This interface is a high-level interface that can be called by the user. It is not recommended to call. The system has default values. The default parameters are set according to different profiles. The default entropy encoding parameters of the system under different profiles are shown in the table below.

Profile	u32EntropyEncModeI	u32EntropyEncModeP
Baseline	0	0
Mainprofile/Highprofile	1	1

  - This interface can be set before the code channel is destroyed after the code channel is created. When this interface is called during the encoding process, it takes effect until the next I frame.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.
  - It is recommended that the user call the MI_VENC_GetH264Entropy interface to obtain the Entropy configuration of the current encoding channel before calling this interface, and then set it.

- Example

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

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

    return s32Ret;
}
```

- related topic

no.

1.3.37 MI_VENC_GetH264Entropy

- description

Obtain the entropy encoding attribute of the H.264 protocol encoding channel.

- prototype

  MI_S32 MI_VENC_GetH264Entropy(MI_VENC_CHN VeChn,MI_VENC_ParamH264Entropy_t*pstH264EntropyEnc);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH264EntropyEnc	The entropy coding attribute of the H.264 protocol coding channel.	output

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to obtain the entropy coding configuration of the H.264 protocol coding channel.
  - This interface can be called after the encoding channel is destroyed after the encoding channel is created.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.

- Example

  See the example of MI_VENC_SetH264Entropy

- related topic

  no.

1.3.38 MI_VENC_SetH265InterPred

- description

  Set the inter prediction property of the H.265 protocol encoding channel.

- prototype

  MI_S32 MI_VENC_SetH265InterPred(MI_VENC_CHN VeChn,MI_VENC_ParamH265InterPred_t*pstH265InterPred);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH265InterPred	Inter-prediction configuration of the H.265 protocol coding channel	input

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to set the interframe prediction configuration of the H.265 protocol encoding channel.
  - The properties of inter prediction are mainly determined by seven parameters:
    1) u32HWSize: the size of the horizontal search window when inter prediction.
    2) u32VWSize: The size of the vertical search window for inter prediction.
    3) bInter16x16PredEn: 16x16 block inter prediction enable flag.
    4) bInter16x8PredEn: 16x8 block inter prediction enable flag.
    5) bInter8x16PredEn: 8x16 block inter prediction enable flag.
    6) bInter8x8PredEn: 8x8 block inter prediction enable flag.
    7) bExtedgeEn: Interframe prediction expansion search enable. When inter-prediction is performed on a macroblock on an image boundary, the range of the search window may exceed the boundary of the image. At this time, the expansion search enables the image to be expanded and inter-frame predicted. If the extended search enable is turned off, no prediction will be made for search windows that are outside the range of the image.
  - This interface belongs to the advanced interface, which can be called by the user. It is recommended not to call, and the system will have default values. The default search window size will vary depending on the resolution of the image, and the other five predictive enable switches are all enabled by default.
  - The predictive switch of bInter16x16PredEn can only be turned on.
  - This interface can be set before the code channel is destroyed after the code channel is created. When this interface is called during the encoding process, it takes effect until the next frame.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.
    Before calling this interface, users are advised to call MI_VENC_GetH265InterPred interface first to obtain InterPred current channel coding, and then set it.

- Example

Please refer to the example of MI_VENC_SetH264InterPred

- related topic

  no.

1.3.39 MI_VENC_GetH265InterPred

- description

  Obtain the inter prediction property of the H.265 protocol coding channel.

- prototype

  MI_S32 MI_VENC_GetH265InterPred(MI_VENC_CHN VeChn,MI_VENC_ParamH265InterPred_t*pstH265InterPred);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH265InterPred	Inter-prediction parameters of the H.265 protocol coding channel.	output

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to obtain the inter-frame prediction mode of the H.265 protocol coding channel.
  - This interface can be called after the encoding channel is destroyed after the encoding channel is created.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.

- Example

  See the example of MI_VENC_SetH265InterPred

- related topic

  no.

1.3.40 MI_VENC_SetH265IntraPred

- description

  Set the intra prediction property of the H.265 protocol encoding channel.

- prototype

  MI_S32 MI_VENC_SetH265IntraPred(MI_VENC_CHN VeChn,MI_VENC_ParamH265IntraPred_t*pstH265IntraPred);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH265IntraPred	Intra prediction configuration of the H.265 protocol coding channel.	input

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

※ note
  - This interface is used to set the intra prediction configuration of the H.265 protocol encoding channel.
  - The intra-frame prediction is mainly determined by the properties of three parameters.
    1) u32Intra32x32Penalty: The probability that 32 x 32 block intra prediction is selected. The larger the value, the lower the probability.
    2) u32Intra16x16Penalty: The probability that 16 x 16 block intra prediction is selected. The larger the value, the lower the probability.
    3) u32Intra8x8Penalty: 8 x 8 block intra prediction is selected probability, the larger the value, the lower the probability.
  - This interface is a high-level interface that can be called by the user. It is not recommended to call. The system has default values.
  - This interface can be set before the code channel is destroyed after the code channel is created. When this interface is called during the encoding process, it takes effect until the next I frame.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.
  - It is recommended that the user call the MI_VENC_GetH265IntraPred interface to obtain the IntraPred configuration of the current encoding channel before calling this interface, and then set it.

- Example

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

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

    return s32Ret;
}
```

- related topic

  no.

1.3.41 MI_VENC_GetH265IntraPred

- description

  Obtain the intra prediction attribute of the H.265 protocol coding channel.

- prototype

  MI_S32 MI_VENC_GetH265IntraPred(MI_VENC_CHN VeChn,MI_VENC_ParamH265IntraPred_t*pstH265IntraPred);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH265IntraPred	The intra prediction parameter of the H.265 protocol coding channel.	output

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to obtain the intra prediction mode of the H.265 protocol coding channel.
  - This interface can be called after the encoding channel is destroyed after the encoding channel is created.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.

- Example

  See the example of MI_VENC_SetH265IntraPred

- related topic

  no.

1.3.42 MI_VENC_SetH265Trans

- description

  Set the transform and quantized attributes of the H.265 protocol encoding channel.

- prototype

  MI_S32 MI_VENC_SetH265Trans(MI_VENC_CHN VeChn,MI_VENC_ParamH265Trans_t*pstH265Trans);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH265Trans	Transform and quantization attributes of the H.265 protocol coding channel.	input

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to set the conversion and quantization configuration of the H.265 protocol encoding channel.
  - Transform, quantization property mainly by three parameters.
    1) u32IntraTransMode: The transform attribute of the intra prediction macroblock. u32IntraTransMode=0 indicates that 4x4 transform and 8x8 transform are supported for intra prediction macroblocks ; u32IntraTransMode=1indicates that only 4x4 transform is supported for intra prediction macroblocks ; u32IntraTransMode=2 indicates that only 8x8 transform is supported for intra prediction macroblocks. The 8x8 transform can be selected only when the encoding channel protocol is highprofile. That is, under the baselineprofile and mainprofile, the system only supports the configuration of u32IntraTransMode=1.
    2) u32InterTransMode: The transform attribute of the inter prediction macroblock. u32InterTransMode=0 indicates that 4x4 transform and 8x8 transform are supported for inter-predicted macroblocks ; u32InterTransMode=1indicates that only 4x4 transform is supported for inter-predicted macroblocks ; u32InterTransMode=2 indicates that only 8x8 transform is supported for inter-predicted macroblocks. The 8x8 transform can be selected only when the encoding channel protocol is highprofile. That is, under the baselineprofile and mainprofile, the system only supports the configuration of u32InterTransMode=1.
    3) s32ChromaQpIndexOffset: For details, see the slice_cb_qp_offset of H.265 protocol.
  - This interface is a high-level interface that can be called by the user. It is not recommended to call. The system has default values. The default parameters are set according to different profiles. The default trans parameters of the system under different profiles are shown in the table below.

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

  - This interface can be set before the code channel is destroyed after the code channel is created. When this interface is called during the encoding process, it takes effect until the next I frame.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.
  - It is recommended that the user call the MI_VENC_GetH265Trans interface before calling this interface.

- Example

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

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

    return s32Ret;
}
```

- related topic

no.

1.3.43 MI_VENC_GetH265Trans

- description

Obtain the transform and quantization attributes of the H.265 protocol coding channel.

- prototype

MI_S32 MI_VENC_GetH265Trans(MI_VENC_CHN VeChn,MI_VENC_ParamH265Trans_t *pstH265Trans);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH265Trans	The transform and quantization parameters of the H.265 protocol coding channel.	output

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to obtain the transform and quantization configuration of the H.265 protocol coding channel.
  - This interface can be called after the encoding channel is destroyed after the encoding channel is created.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.

- Example

  See the example of MI_VENC_SetH265Trans

- related topic

  no.

1.3.44 MI_VENC_SetH264Dblk

- description

  Set the Deblocking type of the H.264 protocol encoding channel.

- prototype

  MI_S32 MI_VENC_SetH264Dblk(MI_VENC_CHN VeChn,MI_VENC_ParamH264Dblk_t*pstH264Dblk);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH264Dblk	Deblocking parameter of the H.264 protocol encoding channel.	input

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to set the Deblocking configuration of the H.264 protocol encoding channel.
  - The transform and quantization attributes are mainly composed of three parameters:
    1) disable_deblocking_filter_idc: For details, see the H.264 protocol.
    2) Slice_alpha_c0_offset_div2: For details, see the H.264 protocol.
    3) Slice_beta_offset_div2: See the H.264 protocol for the specific meaning.
  - The system disables the deblocking function by default. The default disable_deblocking_filter_idc=0, slice_alpha_c0_offset_div2=0, slice_beta_offset_div2=0.
  - If the user wants to turn off the deblocking function, set disable_deblocking_filter_idc to 1.
  - This interface can be set before the code channel is destroyed after the code channel is created. When this interface is called during the encoding process, it takes effect until the next frame.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.
  - It is recommended that the user call the MI_VENC_GetH264Dblk interface before calling this interface to get the Dblk configuration and then set.
- Example

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

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

    return s32Ret;
}
```

- related topic

  no.

1.3.45 MI_VENC_GetH264Dblk

- description

  Get the dblk type of the H.264 protocol encoding channel.

- prototype

  MI_S32 MI_VENC_GetH264Dblk(MI_VENC_CHN VeChn,MI_VENC_ParamH264Dblk_t *pstH264Dblk);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH264Dblk	The dblk attribute of the H.264 protocol encoding channel.	output

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to obtain the dblk configuration of the H.264 protocol encoding channel.
  - This interface can be called after the encoding channel is destroyed after the encoding channel is created.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.

- Example

  See the example of MI_VENC_SetH264Dblk

- related topic

  no.

1.3.46 MI_VENC_SetH265Dblk

- description

  Set the Deblocking type of the H.265 protocol encoding channel.

- prototype

  MI_S32 MI_VENC_SetH265Dblk(MI_VENC_CHN VeChn,MI_VENC_ParamH265Dblk_t*pstH265Dblk);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH265Dblk	H.265 Deblocking parameters of the protocol coding channel.	input

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files: mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to set the channel coding H.265 protocol configuration Deblocking
  - The transform and quantization attributes are mainly composed of three parameters:
    1) disable_deblocking_filter_idc: For details, see the slice_deblocking_filter_disabled_flag of H.265 Protocol.
    2) slice_tc_offset_div2: For details, see H.265 Protocol.
    3) slice_beta_offset_div2: For details, see H.265 protocol.
  - The system disables the deblocking function by default. The default disable_deblocking_filter_idc=0, slice_tc_offset_div2=0, slice_beta_offset_div2=0.
  - If the user wants to turn off the deblocking function, set disable_deblocking_filter_idc to 1.
  - This interface can be set before the code channel is destroyed after the code channel is created. When this interface is called during the encoding process, it takes effect until the next frame.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.
  - It is recommended that the user call the MI_VENC_GetH265_Dblk interface to get the Dblk configuration before calling this interface.

- Example

  See MI_VENC_SetH264Dblk Example.

- related topic

  no.

1.3.47 MI_VENC_GetH265Dblk

- description

  Get the dblk type of the H.265 protocol encoding channel.

- prototype

  MI_S32 MI_VENC_GetH265Dblk(MI_VENC_CHN VeChn,MI_VENC_ParamH265Dblk_t *pstH265Dblk);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH265Dblk	H.265 dblk channel coding protocol attributes.	output

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to obtain the dblk configuration of the H.265 protocol encoding channel.
  - This interface can be called after the encoding channel is destroyed after the encoding channel is created.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.

- Example

  See the example of MI_VENC_SetH264Dblk

- related topic

  no.

1.3.48 MI_VENC_SetH264Vui

- description

  Set the vui parameter of the H.264 protocol encoding channel.

- prototype

  MI_S32 MI_VENC_SetH264Vui(MI_VENC_CHN VeChn,MI_VENC_ParamH264Vui_t*pstH264Vui);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH264Vui	The Vui parameter of the H.264 protocol encoding channel.	input

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to set the VUI configuration of the H.264 protocol encoding channel.
  - This interface can be set before the code channel is destroyed after the code channel is created. When this interface is called during the encoding process, it takes effect until the next I frame.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.
  - Users are advised before calling this interface, first call MI_VENC_GetH264Vui interface to get the current channel coding Vui configuration, and then set it.

- Example

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

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

    return s32Ret;
}
```

- related topic

  no.

1.3.49 MI_VENC_GetH264Vui

- description

  Obtain the Vui configuration of the H.264 protocol encoding channel.

- prototype

  MI_S32 MI_VENC_GetH264Vui(MI_VENC_CHN VeChn,MI_VENC_ParamH264Vui_t *pstH264Vui);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH264Vui	The Vui attribute of the H.264 protocol encoding channel.	output

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to obtain the Vui configuration of the H.264 protocol encoding channel.
  - This interface can be called after the encoding channel is destroyed after the encoding channel is created.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.

- Example

  See the example of MI_VENC_SetH264Vui

- related topic

  no.

1.3.50 MI_VENC_SetH265Vui

- description

  Set the vui parameter of the H.265 protocol encoding channel.

- prototype

  MI_S32 MI_VENC_SetH265Vui(MI_VENC_CHN VeChn, MI_VENC_ParamH265Vui_t *pstH265Vui);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH265Vui	H.265 Vui parameters of the protocol coding channel.	input

- return value
  - MI_OK: Successful

- Not MI_OK: Failed, see error code for details

➢ dependence
- Header files：mi_common.h, mi_venc.h
- Library file: libmi.a

➢ note
- This interface is used to set the VUI configuration of the H.265 protocol encoding channel.
- This interface can be set before the code channel is destroyed after the code channel is created. When this interface is called during the encoding process, it takes effect until the next I frame.
- It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.
- It is recommended that the user call the MI_VENC_GetH265Vui interface before calling this interface to obtain the Vui configuration of the current encoding channel, and then set it.

➢ Example

See the example of MI_VENC_SetH264Vui Example

➢ related topic

no.

1.3.51 MI_VENC_GetH265Vui

➢ description

Obtain the Vui configuration of the H.265 protocol encoding channel.

➢ prototype

MI_S32 MI_VENC_GetH265Vui(MI_VENC_CHN VeChn,MI_VENC_ParamH265Vui_t *pstH265Vui);

➢ parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstH265Vui	H.265 Vui attribute of the protocol encoding channel.	output

➢ return value
- MI_OK: Successful
- Not MI_OK: Failed, see error code for details

➢ dependence
- Header files：mi_common.h, mi_venc.h
- Library file: libmi.a

➢ note

- This interface is used to obtain the Vui configuration of the H.264 5 protocol encoding channel.
- This interface can be called after the encoding channel is destroyed after the encoding channel is created.
- It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.

➢ Example

See the example of MI_VENC_SetH264Vui

➢ related topic

no.

1.3.52 MI_VENC_SetH265SliceSplit

➢ description

Set the slice split attribute of H.265 channels.

➢ prototype

MI_S32 MI_VENC_SetH265SliceSplit(MI_VENC_CHN VeChn,MI_VENC_ParamH265SliceSplit_t*pstSliceSplit);

➢ parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstSliceSplit	H.265 code stream slice segmentation parameters.	input

➢ return value

- MI_OK: Successful
- Not MI_OK: Failed, see error code for details

➢ dependence

- Header files：mi_common.h, mi_venc.h
- Library file: libmi.a

➢ note

- This interface is used to set the split mode of the H.265 protocol coded channel code stream.
- Slice dividing property consists of two decision parameters.
  1）bSplitEnable: whether the current frame slice division.
  2）u32SliceRowCount: The slice is split according to the macro block line. u32SliceRowCount indicates that each slice occupies the number of image macroblock lines. And when encoding to the last few lines of the image, less than u32SliceRowCount, the remaining macroblock lines are divided into a slice.
- This interface belongs to the advanced interface. The user can call it selectively. It is

recommended not to call. The default bSplitEnable is false. This interface can be set before the code channel is destroyed after the code channel is created. This interface takes effect when the next frame is called when it is called during encoding.

- It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.
- It is recommended that the user call the MI_VENC_GetH265 SliceSplit interface to obtain the sliceplit configuration of the current channel before calling this interface, and then set it.

➢ Example

See the example of MI_VENC_SetH264SliceSplit

➢ related topic

no.

1.3.53 MI_VENC_GetH265SliceSplit

➢ description

Get the slice split attribute of H.265 channel.

➢ prototype

MI_S32 MI_VENC_GetH265SliceSplit(MI_VENC_CHN VeChn,MI_VENC_ParamH265SliceSplit_t*pstSliceSplit);

➢ parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstSliceSplit	H.265 code stream slice segmentation parameters.	output

➢ return value

- MI_OK: Successful
- Not MI_OK: Failed, see error code for details

➢ dependence

- Header files：mi_common.h, mi_venc.h
- Library file: libmi.a

➢ note

- This interface is used to obtain the slice split attribute of the H.265 protocol code channel.
- This interface can be called after the encoding channel is destroyed after the encoding channel is created.
- It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.

➢ Example

See the example of MI_VENC_SetH264SliceSplit

- related topic

  no.

1.3.54 MI_VENC_SetJpegParam

- description

  Set advanced parameters for the JPEG protocol encoding channel.

- prototype

  MI_S32 MI_VENC_SetJpegParam(MI_VENC_CHN VeChn,MI_VENC_ParamJpeg_t *pstJpegParam);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstJpegParam	A collection of advanced parameters for the JPEG protocol encoding channel.	input

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to set advanced parameters of the JPEG protocol encoding channel.
  - The advanced parameters are mainly composed of 4 parameters:
    1) u32Qfactor: The quantization table factor range is [1,99]. The larger the u32Qfactor is, the smaller the quantization coefficient in the quantization table is, the better the image quality will be, and the lower the compression ratio. Similarly the smaller the u32Qfactor is, the greater the quantization coefficients in the quantization table, the image quality will be obtained even worse, while a higher code compression rate. See the RFC2435 standard for the relationship between the specific u32Qfactor and the quantization table.
    2) au8YQt [64], au8CbCrQt [64]: corresponding to two quantization table space, by which the user can two sets a quantization table of user parameters.
    3) u32McuPerEcs: Each Ecs contains much Mcu. The system mode u32MCUPerECS=0 indicates that all MCUs of the current frame are encoded as one ECS. The minimum value of u32MCUPerECS is 0, and the maximum value does not exceed (picwidth+15)>>4x(picheight+15)>>4x2.
  - If you want to use your own quantization table, set the Qfactor to 50 while setting the quantization table.
  - This interface can be set before the code channel is destroyed after the code channel is

created. When this interface is called during the encoding process, it will take effect when the next frame is encoded.

- It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.
- Users are advised before calling this interface, first call MI_VENC_GetJpegParam interface to obtain when the front channel coding JpegParam configuration, and then set it.

➢ Example

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

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

    return s32Ret;
}
```

- related topic

  no.

1.3.55 MI_VENC_GetJpegParam

- description

  Get the advanced parameter configuration of the JPEG protocol encoding channel.

- prototype

  MI_S32 MI_VENC_GetJpegParam(MI_VENC_CHN VeChn,MI_VENC_ParamJpeg_t *pstJpegParam);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstJpegParam	Advanced parameter configuration of the Jpeg protocol encoding channel.	output

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to obtain the advanced parameter configuration of the JPEG protocol encoding channel.
  - This interface can be called after the encoding channel is destroyed after the encoding channel is created.
  - It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.

- Example

  See the example of MI_VENC_SetJpegParam

- related topic

  no.

1.3.56 MI_VENC_SetRcParam

- description

  Set advanced parameters for the code channel rate controller.

- prototype

  MI_S32 MI_VENC_SetRcParam(MI_VENC_CHN VeChn,MI_VENC_RcParam_t *pstRcParam);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstRcParam	Advanced parameters for encoding channel rate controllers.	input

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- Chip Difference

The rate control algorithms supported by different chips and different coding protocols are different, as shown in the following table:

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

- note
  - The advanced parameters of the coded channel rate controller have default values, rather than having to call this interface to start the code channel.
  - It is recommended that the user first call the MI_VENC_GetRcParam interface to obtain the RC advanced parameters, then modify the corresponding parameters, and then call this interface to set the advanced parameters.
  - The RC advanced parameters now only support H.264/H.265/Mjpeg rate control modes.
  - The advanced parameters of the rate controller consist of the following parameters: u32ThrdI[RC_TEXTURE_THR_SIZE], u32ThrdP[RC_TEXTURE_THR_SIZE]: A set of thresholds for measuring the macroblock complexity of I frames and P frames, respectively. The thresholds are arranged in order from small to large, and each threshold has a value range of [0, 255]. This set of thresholds is used to appropriately adjust the Qp of each macroblock according to image complexity when performing macroblock level rate control. For H.264, the macroblock level rate control only adds direction (maximum plus 12), that is, if the image complexity of the current macroblock is between some two thresholds, the Qp value of the current macroblock starts at the macroblock line. Qp based on adding value X. The value of X is shown in the following table (C represents image complexity):

C Range	x
C<=u32Thrd[0]	0
u32Thrd[0]<C<=u32Thrd[1]	1
u32Thrd[1]<C<=u32Thrd[2]	2
u32Thrd[2]<C<=u32Thrd[3]	3
u32Thrd[3]<C<=u32Thrd[4]	4
u32Thrd[4]<C<=u32Thrd[5]	5
u32Thrd[5]<C<=u32Thrd[6]	6
u32Thrd[6]<C<=u32Thrd[7]	7
u32Thrd[7]<C<=u32Thrd[8]	8
u32Thrd[8]<C<=u32Thrd[9]	9
u32Thrd[9]<C<=u32Thrd[10]	10
u32Thrd[10]<C<=u32Thrd[11]	11
u32Thrd[11]<C	12

For H.265, macroblock The code rate control has both the plus direction (maximum plus 8) and the minus direction (maximum minus 4), that is, if the image complexity of the current macroblock is less than or equal to the u32Thr[3] threshold, the Qp value of the current macroblock is in the macro. block rows starting Qp subtracting the value on the basis of X ; If the current macroblock image complexity is greater than u32Thr [3] the threshold value, the current macroblock Qp value is a macroblock row starting Qp based on adding value y. The values of X and Y are shown in the following table (C represents image complexity):

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

- u32RowQpDelta: The amplitude of the fluctuation of the starting QP of each row of macroblocks relative to the starting QP of the frame at the time of macroblock level rate control. For the next rate fluctuations more stringent scenario, this parameter can try to transfer large to achieve a more accurate rate control, but may cause image quality to the interior of the frame images have some difference different. At high bit rates, this value is recommended to be 0; the medium code rate is recommended to be 0 or 1 ; for low bit rates it is recommended to be 2 to 5.

- The CBR parameters are as follows:
  1) u32MinIprop&u32MaxIprop: CBR advanced parameters, which represent the minimum IP ratio and the maximum IP ratio. The IP ratio represents the ratio of the number of Bits of the I frame and the P frame. These two values are used to clamp the range of IP ratios. When u32MinIprop is adjusted to be large, the I frame is clear and the P frame is blurred. When u32MaxIprop is adjusted to be small, the I frame is blurred and the P frame is clear. Under normal circumstances, it is not recommended to constrain the IP size ratio to avoid breathing effects and code rate fluctuations. The default u32MinIprop is set to 1 and u32MaxIprop is set to 20. In a scenario where the I frame size is constrained, u32MinIprop can be set according to the dependence on the I frame size fluctuation. And the value of u32MaxIprop.
  2) u32MaxQp&u32MinQp&u32MaxStartQp: CBR advanced parameters, u32MaxQp, u32MinQp represent the maximum QP and minimum QP of the current frame. This clamp has the strongest effect, and all other adjustments to the imageQP, such as macroblock rate control, are ultimately constrained to this maximum QP and minimum QP. u32MaxStartQp represents the maximum QP of the frame-level rate control output, which is in the range [u32MinQp, u32MaxQp]. Macroblock level rate control may cause QPs of some macroblocks to be larger or smaller than u32MaxStartQp, but will be clamped to [u32MinQp, u32MaxQp]. The default value u32MinQp is 10, u32MaxQp is 51, andu32MaxStartQp is 51. It is recommended not to change this group of parameters without special requirements for quality.
  3) u32MaxPPDeltaQp&u32MaxIPDeltaQp: The CBR advanced parameter indicates the maximum difference between the maximum difference between the starting Qp of two consecutive P frames and the starting QP of consecutive IP frames. When there is a large motion, scene switching, etc., the guaranteed code rate is stable, which may cause the QP difference between consecutive P frames to be too large, resulting in image mosaic, etc., and the QP changes can be clamped by adjusting these two parameters. Avoid image quality fluctuations. The default value of u32MaxPPDelta is 3 ; the default value of u32MaxIPdeltaQP is 5, which can be modified according to the difference between the quality and the code rate requirements.
  4) s32IPQPDelta: CBR advanced parameter, which indicates the difference between the average QP value and the current I frame QP. This parameter can be negative. Can be used to adjust I frame oversize and breathing effects. The system default value is 2, increase this value, and the I frame becomes clear. u32RQRatio[8]: The CBR advanced parameter indicates the weight of the rate constant and the quality stability in the rate control. u32RQRatio[i]/100 represents the quality stability weight, 1-u32RQRatio[i]/100 represents the code rate stability weight, for example: u32RQRatio[i]=75, indicating that the quality is stable and accounts for 75% of the weight, code rate Stability accounts for 25% of the weight. CBR is provided. 8 scene modes, namely: Normal (normal scenario), the Move (moving scene), Still (static scenario), StillToMove (static scene into motion), MoveToStill (sports scene brought to a standstill), SceneSwitch (scenes Switch), SharpMove (violent motion scene), Init (program initialization), respectively correspond to the index number of 0 to 7 in u32RQRatio. The system default now is u32RQRatio[]={75,75,75,50,50,20,30,0}. Users can set the weight of the rate control in different scenarios to meet specific dependencies. For example, in a large sports scenario, the quality is stable at 30%, and the code rate is stable at 70%. Users can continue the quality ratio. Smaller, the rate fluctuations become smaller, reserved, not used.

- The VBR parameters are as follows.
  1) s32DeltaQP: The VBR advanced parameter indicates the maximum QP change between frames and frames during VBR quality fluctuations. This parameter can be used to prevent mosaics from appearing. The system defaults to 2.
  2) s32ChangePos: VBR advanced parameter, which indicates the ratio of the code rate to the maximum code rate when VBR starts to adjust QP. The system defaults to 90. If the content changes drastically and the maximum bit rate is not exceeded, it is recommended to reduce this value and increase s32DeltaQP, but the code rate is small and the quality deviation is stable when the rate control is stable.
- The AVBR parameters are as follows.
  1) s32ChangePos: AVBR advanced parameter, which indicates the ratio of code rate to maximum code rate when AVBR starts to adjust QP. The default value of the system is 80. If the content changes drastically and the maximum bit rate is not exceeded, it is recommended to reduce this value and increase s32DeltaQP, but the code rate is small and the quality deviation is stable when the rate control is stable.
  2) u32MaxQp&u32MinQp: AVBR advanced parameters, u32MaxQp and u32MinQp represent the maximum and minimum QP of the current frame. The clamping effect is the most powerful, and all other adjustments to image QP such as macroblock level rate control, will eventually be constrained to this maximum and minimum QP. The default value of u32MinQp is 12, and u32MaxQp is 48. If there is no special requirement for quality, it is not recommended to change this group of parameters.
  3) u32MaxIQp&u32MinIQp: AVBR advanced parameters, u32MaxIQp and u32MinIQp represent the maximum and minimum QP of IDR frame in the current sequence. The clamping effect is the most powerful, and all other adjustment of image QP such as macroblock level rate control, will eventually be constrained to the maximum and minimum QP. If there is no special requirement for quality, it is not recommended to change this group of parameters.
- pRcParam: RC advanced parameters formulated by the user, reserved, not used at the moment.
- It is recommended that the user call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process.

➢ Example

➢ related topic

no.

1.3.57 MI_VENC_GetRcParam

➢ description

Get the channel rate control advanced parameters.

➢ prototype

MI_S32 MI_VENC_GetRcParam(MI_VENC_CHN VeChn,MI_VENC_RcParam_t *pstRcParam);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstRcParam	Channel rate control parameter pointer.	output

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

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

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

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

    return s32Ret;
}
```

- note
  - This interface is used to obtain advanced parameters of the H.264 encoding channel rate control. The specific meaning of each parameter, please refer to MI_VENC_RcParam_t.
  - If pstRcParam is empty, it returns a failure.

- Example

no.

- related topic

  no.

1.3.58 MI_VENC_SetRefParam

- description

  Set the H.264/H.265 encoding channel advanced frame skipping reference parameters.

- prototype

  MI_S32 MI_VENC_SetRefParam(MI_VENC_CHN VeChn,MI_VENC_ParamRef_t*pstRefParam)

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstRefParam	H.264/H.265 coding channel advanced frame skipping reference parameters.	input

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - If pstRefParam is empty, it returns a failure.
  - Create H.264 / H.265 when channel coding protocol, the default mode is the reference frame skipping 1 times the reference frame skip mode. If the user needs to modify the frame skipping reference of the encoding channel, it is recommended to call this interface before starting the encoding after creating the channel, reducing the number of calls during the encoding process
  - When this interface is called during the encoding process, it takes effect until the next I frame.
  - If the user sets a time reference frame skip mode, the corresponding configuration is: bEnablePred = TRUE, u32Enhance = 0, u32Base = 1;
    if set 2 times the reference frame skip mode, the corresponding configuration is:bEnablePred = TRUE, u32Enhance = 1, u32Base = 1;
    if set 4 times the reference frame skip mode, the corresponding configuration is: bEnablePred = TRUE, u32Enhance = 1, u32Base = 2

- Example

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

    //...omit other thing
```

- related topic

  no.

1.3.59 MI_VENC_GetRefParam

- description

  Obtain the advanced frame skipping reference parameters of the H.264/H.265 coding channel.

- prototype

  MI_S32 MI_VENC_GetRefParam(MI_VENC_CHN VeChn,MI_VENC_ParamRef_t*pstRefParam)

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstRefParam	H.264/H.265 coding channel advanced frame skipping reference parameters.	output

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note

  If pstRefParam is empty, it returns a failure.

- Example

  no.

- related topic

  no.

1.3.60 MI_VENC_SetCrop

- description

  Set the crop properties of the channel.

- prototype

  MI_S32 MI_VENC_SetCrop (VENC_CHNVeChn, MI_VENC_CropCfg_t *pstCropCfg)

- parameters

parameter name	description	input/output
VeChn	Channel number	input
pstCropCfg	Channel cropping properties.	input

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to set the cropping properties of the channel.
  - This interface must be called after the channel is created and before the channel is destroyed.
  - The crop property consists of two parts:
    1) bEnable: Whether the channel cropping function is enabled.
    2) stRect: Crop area properties, including the coordinates of the starting point of the crop area, and the size of the crop area.

- Example

  no.

- related topic

  no.

1.3.61 MI_VENC_GetCrop

- description

  Get the crop properties of the channel.

- prototype

  MI_S32 MI_VENC_GetCrop (MI_VENC_CHN VeChn, MI_VENC_CropCfg_t*pstCropCfg)

- parameters

parameter name	description	input/output
VeChn	Channel number	input
pstCropCfg	Channel cropping properties	output

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  - This interface is used to obtain the cropping properties of the channel.
  - This interface must be called after the channel is created and before the channel is destroyed.

- Example

  no.

- related topic

  no.

1.3.62 MI_VENC_SetFrameLostStrategy

- description

  Set the frame loss policy when the code channel instantaneous code rate exceeds the threshold.

- prototype

  MI_S32 MI_VENC_SetFrameLostStrategy (MI_VENC_CHN VeChn,
  MI_VENC_ParamFrameLost_t*pstFrmLostParam)

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstFrmLostParam	The parameter of the code channel drop frame strategy.	input

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

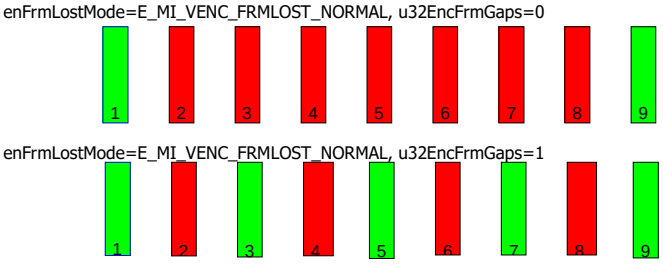
- note
  - If pstFrmLostParam is empty, it returns a failure.
  - pstFrmLostParam is mainly determined by four parameters.
    1) enFrmLostMode: frame loss strategy mode.
    2) u32EncFrmGaps: Drop frame interval.
    3) bFrmLostOpen: Drop frame switch.
    4) u32FrmLostBpsThr: Drop frame threshold.
  - This interface belongs to the advanced interface. The user can select it selectively. The system has default values. By default, the frame is dropped when the instantaneous code rate exceeds the threshold.
  - This interface provides two processing modes when the instantaneous bit rate exceeds the threshold: frame loss and encoding pskip frames. u32EncFrmGaps controls whether frames are evenly dropped or uniformly encoded pskip frames. As the figure below:

T

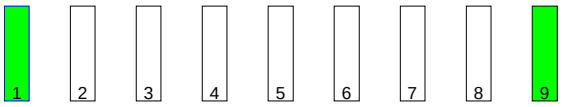
->The instantaneous encoding frame rate does not exceed the threshold value, the frame and retained

-> The frame where the instantaneous bit rate exceeds the threshold when the frame is encoded, and the dropped frame

-> When the frame is encoded, the instantaneous code rate exceeds the threshold and is encoded as a pskip frame

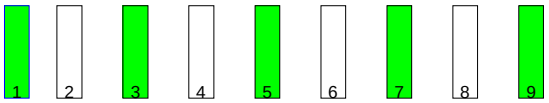


enFrmLostMode=E_MI_VENC_FRMLOST_PSKIP, u32EncFrmGaps=0



enFrmLostMode=E_MI_VENC_FRMLOST_PSKIP, u32EncFrmGaps=1

- This interface can be set before the code channel is destroyed after the code channel is



created. This interface takes effect when the next frame is called when it is called during encoding.

➢ Example

no.

➢ related topic

no.

1.3.63 MI_VENC_GetFrameLostStrategy

➢ description

Obtain the frame loss policy when the instantaneous code rate of the coding channel exceeds the threshold.

➢ prototype

MI_S32 MI_VENC_GetFrameLostStrategy(MI_VENC_CHN VeChn,MI_VENC_ParamFrameLost_t*pstFrmLostParam);

➢ parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstFrmLostParam	The parameter of the code channel drop frame strategy	output

➢ return value

- MI_OK: Successful
- Not MI_OK: Failed, see error code for details

➢ dependence

- Header files：mi_common.h, mi_venc.h
- Library file: libmi.a

➢ note

If pstFrmLostParam is empty, it returns a failure.

- Example

  no.

- related topic

  no.

1.3.64 MI_VENC_SetSuperFrameCfg

- description

  Set the encoding jumbo frame configuration.

- prototype

  MI_S32 MI_VENC_SetSuperFrameCfg(MI_VENC_CHN VeChn,MI_VENC_SuperFrameCfg_t*pstSuperFrmParam);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstSuperFrmParam	Encode jumbo frame configuration parameters.	input

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_comm_rc.h, mi_venc.h
  - Library file: libmi.a

- note
  - If the channel is not created, it returns a failure.
  - This interface is a high-level interface that users can optionally call, the system defaults. The system default is eSuperFrmMode = E_MI_VENC_SUPERFRM_NONE.
  - This interface can be set before the code channel is destroyed after the code channel is created.
  - When super frame mode is E_MI_VENC_SUPERFRM_NONE , MI do nothing.
  - When super frame mode is E_MI_VENC_SUPERFRM_DISCARD, MI will discard the super frame and continue encoding the next frame.If the chip version is SSC325,SSC327,SSC335,SSC337,MI will force to request I frame after discarding the super frame.
  - When super frame mode is E_MI_VENC_SUPERFRM_REENCODE,MI will increase the QP of the frame by 4 and reencode the super frame.If it still exceeds the threshold after repeating up to 4 times, MI will discard the frame.
  - Super frame processing is mainly to prevent the picture from getting stuck due to a certain frame being too large.The unit of threshold is bit.It is recommended to set it according to the actual scene.

- Example

  no.

- related topic

  no.

1.3.65 MI_VENC_GetSuperFrameCfg

- description

  Get the encoded jumbo frame configuration.

- prototype

  MI_S32 MI_VENC_GetSuperFrameCfg(MI_VENC_CHN VeChn,MI_VENC_SuperFrameCfg_t*pstSuperFrmParam);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstSuperFrmParam	Encode jumbo frame configuration parameters.	output

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note

  If pstSuperFrmParam is empty, it returns a failure.

- Example

  no.

- related topic

  no.

1.3.66 MI_VENC_SetRcPriority

- Description

  Set the priority type for rate control.

- Syntax

  MI_S32 MI_VENC_SetRcPriority(MI_VENC_CHN VeChn,MI_VENC_RcPriority_e eRcPriority);

- Parameters

Parameter	Description	Input/Output
VeChn	Channel number	Input
enRcPriority	The enumeration of the priority type.	Input

- Return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Dependency
  - Header file: mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - This interface is used to set the rate control to the target bit rate or to the high frame threshold with high priority.
  - When the rate control is high priority with the target code rate, when the code rate is insufficient, a large frame may be encoded to compensate the code rate, and the oversized frame will not be reprogrammed. When the rate control is based on the jumbo frame threshold being high priority, the jumbo frame will reprogram the number of bits, which may result in insufficient code rate.
  - This interface must be called after the code channel is created and before the code channel is destroyed.
  - This interface only supports the rate control mode of both H.264 and H.265 protocols.

- Example

  no.

- Related topic

  no.

1.3.67 MI_VENC_GetRcPriority

- Description

  Get the priority type of rate control.

- Syntax

  MI_S32 MI_VENC_GetRcPriority(MI_VENC_CHN VeChn,MI_VENC_RcPriority_e *peRcPriority);

- Parameters

Parameter	Description	Input/Output
VeChn	Channel number	Input
penRcPriority	The pointer to the priority type	Output

- Return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Dependency
  - Header file: mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note

  This interface must be called after the code channel is created and before the code channel is destroyed.

- Example

  no.

- Related topic

  no.

1.3.68 MI_VENC_AllocCustomMap

- Description

  Allocate the memory for Custom Map used by smart encoding. The Custom Map includes QP Map and Mode Map.

- Syntax

  MI_S32 MI_VENC_ AllocCustomMap (MI_VENC_CHN VeChn, MI_PHY *pstPhyAddr, MI_VOID **ppCpuAddr, MI_U32 *pu32Width, MI_U32 *pu32Height);

- Parameters

Parameter	Description	Input/Output
VeChn	Encode channel number. Value range: [0,VENC_MAX_CHN_NUM).	Input
pstPhyAddr	Pointer to the allocated virtual physical address.	Output
ppCpuAddr	Pointer to the allocated CPU address.	Output
pu32Width	Width of the frame.	Output
pu32Height	Height of the frame.	Output

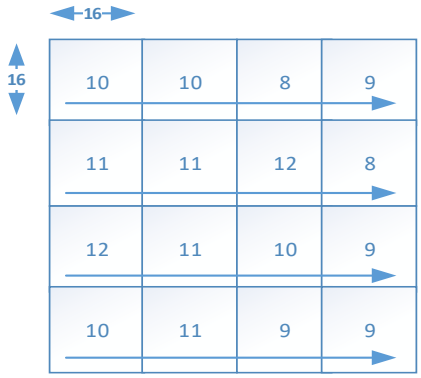
- Return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Dependency
  - Header file: mi_common.h, mi_venc.h
  - Library file: libmi.a

➢ Note

- This interface is used for allocating the memory needed by Custom Map. It returns the address of the allocated memory including the physical address, CPU address and size of the frame.
- If the memory allocation fails, the pstPhyAddr or ppCpuAddr is null and a failure will be returned.
- This interface must be called after the code channel is created and before the code channel is destroyed. The allocated memory will be free when the code channel is destroyed.
- The Custom Map includes QP Map and Mode Map. The two types of Custom Map could be applied simultaneously.
- In H.264, the QP Map contains QPs for every 16x16 block. The QP value for each 16x16 block could be set with the customer defined value. The QP Map is as illustrated below. The QP values should be successively set to the memory which is assigned for QP Map. The QP value could be set with relative or absolute value (refer to MI_VENC_SetAdvCustRcAttr). The value range is [12, 48] if absolute QP is configured, while the value range should be [0, 63] if relative QP is configured. (The efficient QP value used is the relative QP value minus 32 and the corresponding efficient value range is [-32, 31].)
  The Mode Map contains Mode value for every 16x16 block. The Mode value for each 16x16 block could be set with the customer defined value. The Mode Map is as illustrated below. The Mode values should be successively set to the memory which is assigned for Mode Map. One byte with Mode/QP value could be set every 16x16 blocks. The configuration data struct is as shown below.

```
struct {
        uint8   qp      :   6;
        uint8   mode    :   2;
} H264CustomMapConf;
```

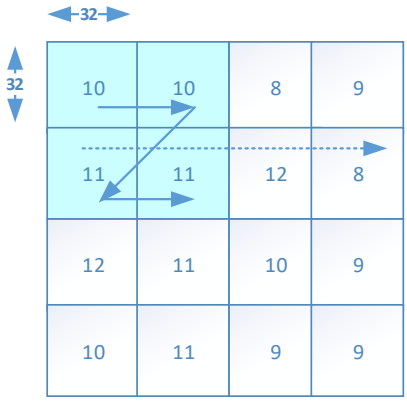


AVC

- In H.265, the QP Map contains QPs for every 4x32x32 block. The QP value for each 4x32x32 block could be set with the customer defined value. The QP Map is as illustrated below. The QP values should be successively set to the memory which is assigned for QP Map. The QP value could be set with relative or absolute value (refer to MI_VENC_SetAdvCustRcAttr). The Mode Map contains Mode value for every 4x32x32 block. The Mode value for each 4x32x32 block could be set with the customer defined value. The Mode Map is just the same as the QP Map. The Mode values should be successively set to the memory which is assigned for Mode Map.
  Four bytes with QP/Mode value could be set every 4x32x32 blocks. The configuration data

structure is as shown below. The Mode value uses 2 bits. The QP value for every 32x32 block uses 6 bits.

```
struct {
        uint32  sub_ctu_qp_0    :  6;
        uint32  mode            :  2;
        uint32  sub_ctu_qp_1    :  6;
        uint32  reserved_0      :  2;
        uint32  sub_ctu_qp_2    :  6;
        uint32  reserved_1      :  2;
        uint32  sub_ctu_qp_3    :  6;
        uint32  reserved_2      :  2;
} H265CustomMapConf;
```



HEVC

- Example

  no.

- Related topic

  no.

1.3.69 MI_VENC_ApplyCustomMap

- Description

  Apply the configured Custom Map.

- Syntax

  MI_S32 MI_VENC_ApplyCustomMap (MI_VENC_CHN VeChn, MI_PHY PhyAddr);

- Parameters

Parameter	Description	Input/Output
VeChn	Encode channel number. Value range: [0,VENC_MAX_CHN_NUM).	Input
PhyAddr	The allocated virtual physical address.	Input

- Return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Dependency
  - Header file: mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - This interface is used for applying the configuration after the customer defined Custom Map is configured into the corresponding address.
  - This interface must be called after the Custom Map is initialized or modified.

- Example

  no.

- Related topic

  no.

1.3.70 MI_VENC_GetLastHistoStaticInfo

- Description

  Get the output information when the latest frame is encoded.

- Syntax

  MI_S32 MI_VENC_GetLastHistoStaticInfo (MI_VENC_CHN VeChn,
  MI_VENC_FrameHistoStaticInfo_t *ppFrmHistoStaticInfo);

- Parameters

Parameter	Description	Input/Output
VeChn	Encode channel number. Value range: [0,VENC_MAX_CHN_NUM).	Input
ppFrmHistoStaticInfo	Pointer to the address of the saved output information for the latest encoded frame.	Output

- Return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Dependency
  - Header file: mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - This interface is used for getting the output information when the latest frame is encoded. It includes the encoded frame type, the number of blocks and the frame size, etc.
  - If the getting of the output content fails, it returns a failure.

- This interface must be called after the code channel is created and before the code channel is destroyed. The allocated memory would be free when the code channel is destroyed or when the memory is released by using the interface below.

➢ Example

no.

➢ Related topic

no.

1.3.71 MI_VENC_ReleaseHistoStaticInfo

➢ Description

Release the memory used for the output information when the latest frame is encoded.

➢ Syntax

MI_S32 MI_VENC_ReleaseHistoStaticInfo (MI_VENC_CHN VeChn);

➢ Parameters

Parameter	Description	Input/Output
VeChn	Encode channel number. Value range: [0,VENC_MAX_CHN_NUM).	Input

➢ Return value

- MI_OK: Successful
- Not MI_OK: Failed, see error code for details

➢ Dependency

- Header file: mi_common.h, mi_venc.h
- Library file: libmi.a

➢ Note

- This interface is used for releasing memory used for saving the output information when the latest frame is encoded.

➢ Example

no.

➢ Related topic

no.

1.3.72 MI_VENC_SetAdvCustRcAttr

➢ Description

Set the customer defined advanced RC configuration parameters.

- Syntax

  MI_S32 MI_VENC_ SetAdvCustRcAttr (MI_VENC_CHN Vehn, MI_VENC_AdvCustRcAttr_t sAdvCustRcAttr);

- Parameters

Parameter	Description	Input/Output
VeChn	Encode channel number. Value range: [0,VENC_MAX_CHN_NUM).	Input
sAdvCustRcAttr	Customer defined advanced RC configuration parameters.	Input

- Return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Dependency
  - Header file: mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - This interface is used for setting the customer defined advanced RC configuration parameters.
  - This interface must be called after the code channel is created and before the picture starts to be received.
  - After the switch of customer defined advanced RC configuration is set, the related function could be started by calling the functions MI_VENC_AllocCustomMap and MI_VENC_GetLastHistoStaticInfo.

- Example

  no.

- Related topic

  no.

1.3.73 MI_VENC_SetInputSourceConfig

- description

  Set H.264/H.265 input source info.

- prototype

  MI_S32 MI_VENC_SetInputSourceConfig(MI_VENC_CHN VeChn, MI_VENC_InputSourceConfig_t *pstInputSourceConfig);

- parameters

parameter name	description	input/output
VeChn	encode channel number. Value range：[0,VENC_MAX_CHN_NUM).	input
pstInputSourceConfig	input source info	input

- return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- dependence
  - Header files：mi_common.h, mi_venc.h
  - Library file: libmi.a

- note
  When using input ring mode, different chips have different restrictions on whether to call this interface, as shown in the following table：

Chip	Whether to call this interface when using input ring？
328Q/329D/326D	N
325/325DE/327DE	N
336D/336Q/339G	N
335/337DE	Y

  - If the channel is not created, it returns a failure.
  - This interface must be called after the code channel is created and before the picture starts to be received.
  - It allows to be called for only one channel to set input ring info.
  - If set E_MI_VENC_INPUT_MODE_RING_ONE_FRM or E_MI_VENC_INPUT_MODE_RING_HALF_FRM, then APP should set E_MI_SYS_BIND_TYPE_HW_RING and the corresponding ring buffer height when call MI_SYS_BindChnPort2. For example, if the resolution is 1920x1080 and set E_MI_VENC_INPUT_MODE_RING_ONE_FRM, then the ring buffer height should set 1080; if set E_MI_VENC_INPUT_MODE_RING_HALF_FRM, then the ring buffer height should set 540.

- Example

  no.

- related topic

  no.

1.3.74 MI_VENC_SetSmartDetInfo

- Description

  For third-party intelligent detection algorithms to provide VENC with statistics needed for intelligent coding.

- Syntax

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

- Parameters

Parameter	Description	Input/Output
VeChn	Encode channel number. Value range: [0,VENC_MAX_CHN_NUM).	Input
pstSmartDetInfo	related statistics of intelligent detection algorithm	Input

- Return value
  - MI_OK: Successful
  - Not MI_OK: Failed, see error code for details

- Dependency
  - Header file: mi_common.h, mi_venc.h
  - Library file: libmi.a

- Note
  - It will return failure if the channel is not created
  - This interface must be called after the code channel is created and before the picture starts to be received.
  - This interface parameter setting only takes effect on SSC33X.

- Example

  no.

- Related topic

  no.

2. VENC DATA TYPE

The relevant data types and data structures are defined as follows:

VENC_MAX_CHN_NUM	Define the maximum number of channels
RC_TEXTURE_T HR_SIZE	Define the threshold number of texture level code control
MI_VENC_H264eNaluType_e	Define H.264 code stream NALU type
MI_VENC_H264eRefSliceType_e	Defining which reference frame in the frame skip reference mode the acquired H.264 code stream belongs to
M I_VENC_H264eRefType_e	Define the frame type and reference attribute of the H.264 frame skip reference stream
MI_VENC_JpegePackType_e	Define the PACK type of the JPEG stream
MI_VENC_H265eNaluType_e	Define H.265 code stream NALU type
MI_VENC_DataType_t	Defining code stream result union
MI_VENC_PackInfo_t	A structure that defines other types of codestream data contained in the current stream packet data.
MI_VENC_Pack_t	Define a frame code stream packet structure
MI_VENC_StreamInfoH264_t	Define H.264 protocol code stream feature information
MI_VENC_StreamInfoJpeg_t	Define JPEG protocol stream feature information
MI_VENC_StreamInfoH265_t	Define H.265 protocol code stream feature information
MI_VENC_Stream_t	Define the frame code stream type structure
MI_VENC_StreamBufInfo_t	Structure for defining code stream buffer information
MI_VENC_AttrH264_t	Define the H.264 encoder attribute structure
MI_VENC_AttrJpeg_t	Define JPEG capture encoder attribute structure
MI_VENC_AttrH265_t	Define the H.265 encoder attribute structure
MI_VENC_Attr_t	Define the encoder attribute structure
MI_VENC_Chn A ttr_t	Define the encoding channel attribute structure
MI_VENC_ChnStat_t	Define the state structure of the code channel
MI_VENC_ParamH264SliceSplit_t	Define the H.264 encoding channel slice segmentation attribute
MI_VENC_ParamH264InterPred_t	Defining H.264 encoding channel inter prediction properties
MI_VENC_ParamH264Trans_t	Define H.264 encoding channel transform, quantization properties
MI_VENC_ParamH264Entropy_t	Define H.264 encoding channel entropy encoding properties
MI_VENC_ParamH265 InterPred_t	Defining H.264 encoding channel inter prediction properties

MI_VENC_ParamH265_IntraPred_t	Define H.264 encoding channel intra prediction properties
MI_VENC_ParamH265_Trans_t	Define H.264 encoding channel transform, quantization properties
MI_VENC_ParamH264Dblk_t	Define the H.264 encoding channel Deblocking property
MI_VENC_ParamH264Vui_t	Define H.264 encoding channel VUI properties
MI_VENC_ParamH264VuiAspectRatio_t	A structure defining the AspectRatio information in the H.264 protocol encoding channel Vui
MI_VENC_ParamH264Vui TimeInfo_t	A structure defining TIME_INFO information in the H.264 protocol encoding channel Vui
MI_VENC_Param H264 VuiVideoSignal_t	A structure defining the VIDEO_SIGNAL information in the H.264 protocol encoding channel Vui
MI_VENC_ParamJpeg_t	Define JPEG encoding parameter sets
MI_VENC_RoiCfg_t	Define the coding channel interest area coding attribute
MI_VENC_RoiBgFrameRate_t	Define the frame rate attribute of a non-Roi area
MI_VENC_RcAttr_t	Define the encoding channel rate controller attribute
MI_VEN C_RcMode_e	Define the code channel rate controller mode
MI_VENC_AttrH264Cbr_t	Define the HBR encoding channel CBR attribute structure
MI_VENC_AttrH264Vbr_t	Define the HBR encoding channel VBR attribute structure
MI_VENC_AttrH264Fi xQp_t	Define the H.264 encoding channel Fixqp attribute structure
MI_V ENC_A ttr H264A br__t	Define the HBR encoding channel ABR attribute structure
MI_VENC_SuperFrmMode_e	Define the jumbo frame processing mode in rate control
MI_V ENC_A ttr H265C br__t	Define the H.265 encoding channel CBR attribute structure
MI_V ENC_A ttr H265V br__t	Define the HBR attribute channel VBR attribute structure
MI_VENC_AttrH265FixQp_t	Define the H.265 encoding channel Fixqp attribute structure
MI_VENC_ParamH264 Vbr_t	Define H264 protocol encoding channel VBR rate control mode advanced parameter configuration
MI_VENC_ParamH264Cbr_t	Define H264 protocol encoding channel CBR new version rate control mode advanced parameter configuration
MI_VENC_ParamH265Vbr_t	Define H265 protocol encoding channel VBR rate control mode advanced parameter configuration
MI_VENC_ParamH265Cbr_t	Define H265 protocol encoding channel CBR new version rate control mode advanced parameter configuration
MI_VENC_RcParam_t	Define the rate control advanced parameters of the coding channel
MI_VENC_CropCfg_t	Define channel intercept parameters
MI_VENC_RecvPicParam_t	Receive the specified frame number image encoding
MI_VENC_H264eIdrPicIdMode_e	Set the mode of IDR frame or Id_id_id of I frame
MI_VENC_H264IdrPicIdCfg_t	Idr_pic_id parameter of IDR frame or I frame

MI_VENC_FrameLostMode_e	Defining the frame loss mode when the code channel instantaneous code rate exceeds the threshold
MI_VENC_ParamFrameLost_t	Defining the frame loss strategy when the code channel instantaneous code rate exceeds the threshold
MI_VENC_SuperFrameCfg_t	Large frame processing strategy parameters
MI_VENC_RcPriority_e	Rate control priority enumeration
MI_VENC_ModParam_t	Define encoding module parameters
MI_VENC_ModType_e	Define module parameter types
MI_VENC_ParamModVenc_t	Define mi_venc.ko module parameters
MI_VENC_ParamModH264e_t	Define mi_h264e.ko module parameters
MI_VENC_ParamModH265e_t	Define mi_h265.ko module parameters
MI_VENC_ParamModJpege_t	Define mi_jpege.ko module parameters
MI_VENC_AdvCustRcAttr_t	Define customer defined advanced RC configuration parameters
MI_VENC_FrameHistoStaticInfo_t	Define attributes of the encoded frame
MI_VENC_InputSourceConfig_t	Define H.264/H.265 input buffer config info structure
MI_VENC_InputSrcBufferMode_e	Define H.264/H.265 input buffer mode
MI_VENC_AttrH264Avbr_t	Define the H.264 encoding channel AVBR attribute structure
MI_VENC_AttrH265Avbr_t	Define the H.265 encoding channel AVBR attribute structure
MI_VENC_ParamH264Avbr_t	Define H264 protocol encoding channel AVBR rate control mode advanced parameter configuration
MI_VENC_ParamH265Avbr_t	Define H265 protocol encoding channel AVBR rate control mode advanced parameter configuration

2.1. VENC_MAX_CHN_NUM

- Description

  Define the maximum number of channels.

- Definition

  #define VENC_MAX_CHN_NUM 16

- Note

  no.

- Related data types and interfaces

  no.

2.2. RC_TEXTURE_THR_SIZE

- Description

  Defines the number of thresholds for RC macroblock complexity.

- Definition

  #define RC_TEXTURE_THR_SIZE 1

- Note

  no.

- Related data types and interfaces

  no.

2.3. MI_VENC_H264eNaluType_e

- Description

  Define the H.264 code stream NALU type.

- Definition

```
typedef enum
{
    E_MI_VENC_H264E_NALU_PSLICE = 1,
    E_MI_VENC_H264E_NALU_ISLICE = 5,
    E_MI_VENC_H264E_NALU_SEI    = 6,
    E_MI_VENC_H264E_NALU_SPS    = 7,
    E_MI_VENC_H264E_NALU_PPS    = 8,
    E_MI_VENC_H264E_NALU_IPSLICE  = 9,
    E_MI_VENC_H264E_NALU_PREFIX = 14,
    E_MI_VENC_H264E_NALU_MAX
}MI_VENC_H264eNaluType_e;
```

- Members

Member	Description
E_MI_VENC_H264E_NALU_PSLICE	PSLICE type
E_MI_VENC_H264E_NALU_ISLICE	ISLICE type
E_MI_VENC_H264E_NALU_SEI	SEI type
E_MI_VENC_H264E_NALU_SPS	SPS type
E_MI_VENC_H264E_NALU_PPS	PPS type
E_MI_VENC_H264E_NALU_PREFIX	PREFIX type
E_MI_VENC_H264E_NALU_IPSLICE	P frame bruch ISLICE type (not supported at this time)

- Note

no.

- Related data types and interfaces

no.

2.4. MI_VENC_H264eRefSliceType_e

- Description

Defines the reference frame in which the acquired H.264 code stream belongs in the frame skip reference mode.

- Definition

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

- Members

Member name	Description
E_MI_VENC_H264E_REFSLICE_FOR_1X	Reference frame at 1x frame skip reference.
E_MI_VENC_H264E_REFSLICE_FOR_2X	2x frame skip reference frame or 4x frame skip reference for 2 The reference frame of the double-jump frame reference.
E_MI_VENC_H264E_REFSLICE_FOR_4X	Reference frame for 4x frame skip reference.
E_MI_VENC_H264E_REFSLICE_ MAX	Non-reference frame.

- Note

no.

- Related data types and interfaces

  no.

2.5. MI_VENC_H264eRefType_e

- Description

  Defines the frame type and reference attribute of the H.264 frame skip reference stream.

- Definition

```
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;
```

- Members

Member name	Description
E_MI_VENC_BASE_IDR	IDR frame in the base layer.
E_MI_VENC_BASE_P_REFTOIDR	A P frame in the base layer for reference to IDR frame.
E_MI_VENC_BASE_P_REFBYBASE	A P frame in the base layer for reference by other frames in the base layer.
E_MI_VENC_BASE_P_REFBYENHANCE	P frame in the base layer, used for reference of frames in the enhancement layer.
E_MI_VENC_ENHANCE_P_REFBYENHANCE	P frame in the enhance layer, used in the enhance layer The reference of his frame.
E_MI_VENC_ENHANCE_P_NOTFORREF	P frame in the enhance layer, not used for reference

- Note

  no.

- Related data types and interfaces

  no.

2.6. MI_VENC_JpegePackType_e

- Description

  Define the PACK type of the JPEG stream.

- Definition

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

- Members

Member	Description
E_MI_VENC_JPEGE_PACK_ECS	ECS type
E_MI_VENC_JPEGE_PACK_APP	APP type
E_MI_VENC_JPEGE_PACK_VDO	VDO type
E_MI_VENC_JPEGE_PACK_PIC	PIC type

- Note

  no.

- Related data types and interfaces

  no.

2.7. MI_VENC_H265eNaluType_e

- Description

  Define the H.265 code stream NALU type.

- Definition

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

- Members

Member	Description
E_MI_VENC_H265E_NALU_PSLICE	PSLICE type
E_MI_VENC_H265E_NALU_ISLICE	ISLICE type
E_MI_VENC_H265E_NALU_VPS	VPS type
E_MI_VENC_H265E_NALU_SPS	SPS type
E_MI_VENC_H265E_NALU_PPS	PPS type
E_MI_VENC_H265E_NALU_SEI	SEI type

- Note

  no.

- Related data types and interfaces

  no.

2.8. MI_VENC_Rect_t

- Description

  A rectangular box that defines the encoding description.

- Definition

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

- Members

Member name	Description
u32Left	The distance between the left side of the rectangle and the left side of the actual screen, in pixel.
u32Top	The distance between the top edge of the rectangle and the top edge of the actual screen, in pixel
u32Width	The width of the rectangle in pixel
u32Height	The height of the rectangle in pixel

- Note

  The rectangular frame cannot exceed the range of the actual image being encoded.

- Related data types and interfaces

  MI_VENC_SetRoiCfg
  MI_VENC_SetCrop
  MI_VENC_SetRoiBgFrameRate

2.9. MI_VENC_DataType_t

- Description

  Define the stream result type.

- Definition

```
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;
```

- Members

Member	Description
enH264EType	H.264 code stream packet type
enJPEGEType	JPEG code stream packet type
enMPEG4EType	MPEG-4 code stream packet type
enH265EType	H.265 code stream packet type

- Note

  no.

- Related data types and interfaces

  MI_VENC_H264eNaluType_e
  MI_VENC_JpegePackType_e

2.10. MI_VENC_PackInfo_t

- Description

  A structure that defines other types of codestream data contained in current stream packet data.

- Definition

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

- Members

Member	Description
u32PackType	The current stream packet data contains the types of other stream packets.
u32PackOffset	The current stream packet data contains offsets of other

Member	Description
	stream packet data.
u32PackLength	The current stream packet data contains the size of other stream packet data.

2.11. MI_VENC_Pack_t

- Description

  Define the frame code stream packet structure.

- Definition

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

- Members

Member name	Description
pu8Addr	The first address of the stream packet.
phyAddr	The code stream packet physical address.
u32Len	The length of the stream packet.
DataType	The stream type supports data packets of the H.264/JPEG/MPEG-4 protocol type.
u64PTS	Timestamp. Unit: us.
bFrameEnd	End of frame identification.Ranges: MI_TRUE: This stream packet is the last packet of the frame. MI_FALSE: The stream packet is not the last packet of the frame.
u32Offset	The offset of the valid data in the stream packet from the first address of the stream packet, pu8Addr.
u32DataNum	The current stream packet (the type of the current packet is specified by the DataType) is included in the data. The number of other types of stream packets.
stPackInfo[8]	The current stream packet data contains other types of stream packet data information.

- Note

  no.

- Related data types and interfaces

  MI_VENC_DataType_t

2.12. MI_VENC_StreamInfoH264_t

- Description

  Define H.264 protocol code stream feature information.

- Definition

```
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;
```

- Members

Member name	Description
u32PicBytesNum	Encode the number of bytes (BYTE) of the current frame
u32PSkipMbNum	Encoding the number of macroblocks in the current frame using the skip (SKIP) encoding mode
u32IpcmMbNum	Encoding the number of macroblocks in the current frame using the IPCM encoding mode
u32Inter16x8MbNum	Encoding the number of macroblocks in the current frame using the Inter16x8 prediction mode
u32Inter16x16MbNum	Encoding the number of macroblocks in the current frame using the Inter16x16 prediction mode
u32Inter8x16MbNum	Encoding the number of macroblocks in the current frame using the Inter8x16 prediction mode
u32Inter8x8MbNum	Encoding the number of macroblocks in the current frame using the Inter8x8 prediction mode

Member name	Description
u32Intra16MbNum	Encoding the number of macroblocks in the current frame using the Intra16 prediction mode
u32Intra8MbNum	Encoding the number of macroblocks in the current frame using the Intra8 prediction mode
u32Intra4MbNum	Encoding the number of macroblocks in the current frame using the Intra4 prediction mode
enRefSliceType	Coding with the reference frame in the frame skip reference mode of the current frame
enRefType	Coded frame type under advanced frame skip reference
u32UpdateAttrCnt	The number of times the channel attribute or parameter (including the RC parameter) is set
u32StartQp	Initial Qp value used for encoding

- Note
  - When saving the H.264 code stream, only the reference frame in the corresponding frame skip reference mode can be selected:
  - When the frame skip reference mode is the 1x frame skip reference mode, only the code stream whose enRefSliceType is equal to E_MI_VENC_H264E_REFSLICE_FOR_1X can be saved.
  - When the frame skip reference mode is the 2x frame skip reference mode, only the code stream whose enRefSliceType is equal to E_MI_VENC_H264E_REFSLICE_FOR_2X can be saved.
  - When the frame skip reference mode is the 4x frame skip reference mode, only the code stream whose enRefSliceType is equal to _MI_VENC_H264E_REFSLICE_FOR_4X can be saved.
  - Or save the code stream with enRefSliceType equal to E_MI_VENC_H264E_REFSLICE_FOR_2X and E_MI_VENC_H264E_REFSLICE_FOR_4X.

- Related data types and interfaces

  MI_VENC_DataType_t
  MI_VENC_H264eRefSliceType_e

2.13. MI_VENC_StreamInfoJpeg_t

- Description

  Define JPEG protocol code stream feature information.

- Definition

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

- Members

Member name	Description
u32PicBytesNum	The size of a frame of jpeg code stream, in bytes.
u32Updat eAttrCnt	The number of times the channel attribute or parameter (including the RC parameter) is set.
u32Qfactor	Encodes the Qfactor of the current frame.

- Note

  no.

- Related data types and interfaces

  MI_VENC_DataType_t

2.14. MI_VENC_StreamInfoH265_t

- Description

  Define the H.265 protocol code stream feature information.

- Definition

```
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;
```

- Members

Member name	Description
u32PicBytesNum	Encode the number of bytes (BYTE) of the current frame
u32Inter64x64CuNum	Encode the number of CU blocks in the current frame using Inter64x64 prediction mode
u32Inter32x32CuNum	Encoding the number of CU blocks in the current frame using the Inter32x32 prediction mode
u32Inter16x16CuNum	Encoding the number of CU blocks in the current frame using Inter16x16 prediction mode

Member name	Description
u32Inter8x8CuNum	Encoding the number of CU blocks in the current frame using the Inter8x8 prediction mode
u32Intra32x32CuNum	Encoding the number of CU blocks in the current frame using the Intra32x32 prediction mode
u32Intra16x16CuNum	Encoding the number of CU blocks in the current frame using the Intra16x16 prediction mode
u32Intra8x8CuNum	Encoding the number of CU blocks in the current frame using the Intra8x8 prediction mode
u32Intra4x4CuNum	Encoding the number of CU blocks in the current frame using the Intra4x4 prediction mode
e RefType	Coded frame type under advanced frame skip reference
u32UpdateAttrCnt	The number of times the channel attribute or parameter (including the RC parameter) is set
u32StartQp	Initial Qp value used for encoding

- Note

  See the description of the enRefType variable on MI_VENC_StreamInfoH264_t.

- Related data types and interfaces

  MI_VENC_DataType_t
  MI_VENC_H264eRefSliceType_e
  MI_VENC_H264eRefType_e

2.15. MI_VENC_Stream_t

- Description

  Define the framestream type structure.

- Definition

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

- Members

Member name	Description
pstPack	Frame code stream packet structure.
u32PackCount	The number of all packets of a frame stream.
u32Seq	Code stream serial number.The frame number is obtained by frame; the packet sequence number is obtained by packet.
hMiSys	Buffer handle.
stH624Info/stJpegInfo/ stMpeg4Info/stH265Info	Code stream feature information.

- Note

no.

- Related data types and interfaces

MI_VENC_Pack_t
MI_VENC_GetStream

2.16. MI_VENC_StreamBufInfo_t

- Description

A structure that defines the stream buffer information.

- Definition

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

- Members

Member name	Description
u32PhyAddr	The starting physical address of the code stream buffer.
pUserAddr	The virtual address of the stream buffer.
u32BufSize	The size of the stream buffer.

- Note

no.

- Related data types and interfaces

MI_VENC_GetStreamBufInfo

2.17. MI_VENC_AttrH264_t

- Description

  Define the H.264 encoding attribute structure.

- Definition

```
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;
```

- Members

Member name	Description
u32MaxPicWidth	The maximum width of the encoded image. Value range: [MIN_WIDTH, MAX_WIDTH], in pixels. Must be an integer multiple of MIN_ALIGN. Static attribute.
u32PicWidth	Encode image width. Value range: [MIN_WIDTH, u32MaxPicWidth], in pixels. Must be an integer multiple of MIN_ALIGN. Dynamic properties.
u32MaxPicHeight	The maximum height of the encoded image. Value range: [MIN_HEIGHT, MAX_HEIGHT], in pixels. Must be an integer multiple of MIN_ALIGN Static attribute.
u32PicHeight	Encode image height. Value range: [MIN_HEIGHT, u32MaxPicHeight], in pixels Bit. Must be an integer multiple of MIN_ALIGN Dynamic properties.
u32BufSize	Code stream buffer size. Value range: [Min, Max], in bytes. Must be an integer multiple of 64. Recommended value: One maximum encoded image size. The recommended value is u32MaxPicWidth u32MaxPicHeight is 1.5 bytes. Min: 1/2 of the maximum encoded image size. Max: No limit, but it will consume more memory. Static attribute.

Member name	Description
bByFrame	The frame/packet mode acquires the code stream. Value range: {MI_TRUE, MI_FALSE}. • MI_TRUE: Get by frame. • MI_FALSE: Get by package. Static attribute.
u32Profile	The level of coding. Value range: [0, 3].0: Baseline. 1: MainProfile. 2: HighProfile. 3: Svc-T
u32BFrameNum	Encoding supports the number of B frames. Value range: [0, Max]. Max: Unlimited. Static attribute, reserved field, not supported at this time.
u32RefNum	Encoding supports the number of reference frames. Value range: [1, 2]. Static attribute, reserved field, not supported at this time.

- Note

  no.

- Related data types and interfaces

  no.

2.18. MI_VENC_AttrJpeg_t

- Description

  Define the JPEG capture attribute structure.

- Definition

```
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;
```

- Members

Member name	Description
u32MaxPicWidth	The maximum width of the encoded image. Value range: [MIN_WIDTH, MAX_WIDTH], in pixels. Must be an integer multiple of MIN_ALIGN. Static attribute.
u32PicWidth	Encode image width.

Member name	Description
	Value range: [MIN_WIDTH, u32MaxPicWidth], in pixels. Must be an integer multiple of MI_ALIGN. Dynamic properties.
u32MaxPicHeight	The maximum height of the encoded image. Value range: [MIN_HEIGHT, MAX_HEIGHT], in pixels. Must be an integer multiple of MIN_ALIGN Static attribute.
u32PicHeigh	Encode image height. Value range: [MIN_HEIGHT, u32MaxPicHeight], in pixels bit. Must be an integer multiple of MIN_ALIGN Dynamic properties.
u32BufSize	Configure the buffer size. Value range: not less than the product of the maximum width and height of the image 16 aligned. Must be an integer multiple of 64. Static attribute.
bByFrame	Get the stream mode, frame or package. Value range: {MI_TRUE, MI_FALSE}. • MI_TRUE: Get by frame. • MI_FALSE: Get by package. Static attribute.
bSupportDCF	Whether to support Jpeg thumbnails. Static attribute.
u32RestartMakerPerRowCnt	Restart marker after designated rows

- Note

  no.

- Related data types and interfaces

  no.

2.19. MI_VENC_AttrH265_t

- Description

  Define the H.265 encoding attribute structure.

- Definition

```
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;
```

- Members

Member name	Description
u32MaxPicWidth	The maximum width of the encoded image. Value range: [MIN_WIDTH, MAX_WIDTH], in pixels. Must be an integer multiple of MIN_ALIGN. Static attribute.
u32PicWidth	Encode image width. Value range: [MIN_WIDTH, u32MaxPicWidth], in pixels. Must be an integer multiple of MIN_ALIGN. Dynamic properties.
u32MaxPicHeight	The maximum height of the encoded image. Value range: [MIN_HEIGHT, MAX_HEIGHT], in pixels. Must be an integer multiple of MIN_ALIGN Static attribute.
u32PicHeight	Encode image height. Value range: [MIN_HEIGHT, u32MaxPicHeight], in pixels bit. Must be an integer multiple of MIN_ALIGN Dynamic properties.
u32BufSize	Code stream buffer size. Value range: [Min, Max], in bytes. Must be an integer multiple of 64. Recommended value: One maximum encoded image size. The recommended value is u32MaxPicWidth u32MaxPicHeight is 1.5 bytes. Min: 1/2 of the maximum encoded image size. Max: No limit, but it will consume more memory. Static attribute.
bByFrame	The frame/packet mode acquires the code stream. Value range: {MI_TRUE, MI_FALSE}. MI_TRUE: Get by frame. MI_FALSE: Get by package. Static attribute.

Member name	Description
u32Profile	The level of coding. Value range: 0. 0: MainProfile. Static attribute.
u32BFrameNum	Encoding supports the number of B frames. Value range: [0, Max]. Max: Unlimited. Static attribute, reserved field, not supported at this time.
u32RefNum	Encoding supports the number of reference frames. Value range: [1, 2]. Static attribute, reserved field, not supported at this time.

- Note

no.

- Related data types and interfaces

no.

2.20. MI_VENC_Attr_t

- Description

Define the encoder attribute structure.

- Definition

```
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;
```

- Members

Member name	Description
e Type	The encoding protocol type.
stAttrH264e/stAttrMjpeg/stAttrJpeg/ stAttrMpeg4/stAttrH265e	The encoder property of a protocol.

- Note

no.

- Related data types and interfaces
  MI_VENC_CreateChn

2.21. MI_VENC_ChnAttr_t

- Description
  Define the encoding channel attribute structure.

- Definition

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

- Members

Member name	Description
stVeAttr	Encoder properties.
stRcAttr	Rate controller properties.

- Note
  no.

- Related data types and interfaces
  MI_VENC_CreateChn

2.22. MI_VENC_ChnStat_t

- Description
  Define the state structure of the code channel.

- Definition

```
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;
```

- Members

Member name	Description
u32LeftPics	The number of images to be encoded.
u32LeftStreamBytes	The number of bytes remaining in the code stream buffer.
u32LeftStreamFrames	The number of frames remaining in the stream buffer.
u32LeftStreamMillisec	The number of time remaining in the stream, in ms
u32CurPacks	The number of streams of the current frame.
u32LeftRecvPics	The number of remaining frames to be received, set by the user Valid after MI_VENC_StartRecvPicEx.
u32LeftEncPics	The number of frames remaining to be encoded, set in the user Valid after MI_VENC_StartRecvPicEx.
u32FrmRateNum	Encoder frame rate molecule, in integer units
u32FrmRateDen	Encoder frame rate denominator, in integer units.
u32Bitrate	Stream bitrate, in kbps.

- Note

no.

- Related data types and interfaces

MI_VENC_Query

2.23. MI_VENC_ParamH264SliceSplit_t

- Description

Define the H.264 protocol encoding channel SLICE partition structure.

- Definition

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

- Members

Member name	Description
bSplitEnable	Whether slice split is enabled.
u32SliceRowCount	Represents the number of macroblock lines occupied by each slice. The minimum value is 1: and the maximum value is: (image height +15)/16.

- Note

  no.

- Related data types and interfaces

  MI_VENC_SetH264SliceSplit
  MI_VENC_GetH264SliceSplit

2.24. MI_VENC_ParamH265SliceSplit_t

- Description

  Define the H.265 protocol coding channel SLICE partition structure.

- Definition

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

- Members

Member name	Description
bSplitEnable	Whether slice split is enabled.
u32SliceRowCount	Represents the number of macroblock lines occupied by each slice.The minimum value is 1: and the maximum value is: (image height +15)/16.

- Note

  no.

- Related data types and interfaces

  MI_VENC_SetH265SliceSplit
  MI_VENC_GetH265SliceSplit

2.25. MI_VENC_ParamH264InterPred_t

- Description

  Define the H.264 protocol coding channel inter-frame prediction structure.

- Definition

```
typedef struct MI_VENC_ParamH264InterPred_s
{
    /* search window */
    MI_U32 u32HWSize;
    MI_U32 u32VWSize;
    MI_BOOL bInter16x16PredEn;
```

```
    MI_BOOL bInter16x8PredEn;
    MI_BOOL bInter8x16PredEn;
    MI_BOOL bInter8x8PredEn;
    MI_BOOL bInter8x4PredEn;
    MI_BOOL bInter4x8PredEn;
    MI_BOOL bInter4x4PredEn;
    MI_BOOL bExtedgeEn;
} MI_VENC_ParamH264InterPred_t;
```

- Members

Member name	Description
u32HWSize	Horizontal search window size.
u32VWSize	Vertical search window size.
bInter16x16PredEn	16x16 interframe prediction enable switch, enabled by default.
bInter16x8PredEn	16x8 interframe prediction enable switch, enabled by default.
bInter8x16PredEn	8x16 interframe prediction enable switch, enabled by default.
bInter8x8PredEn	8x8 interframe prediction enable switch, enabled by default.
bInter8x4PredEn	8x4 interframe prediction enable switch, enabled by default.
bInter4x8PredEn	4x8 interframe prediction enable switch, enabled by default.
bInter4x4PredEn	4x4 interframe prediction enable switch, enabled by default.
bExtedgeEn	When the search encounters the boundary of the image, it is beyond the scope of the image, and whether the enable switch for the search is enabled is enabled by default.

- Note

  no.

- Related data types and interfaces

  MI_VENC_SetH264InterPred
  MI_VENC_GetH264InterPred

2.26. MI_VENC_ParamH264IntraPred_t

- Description

  Define the H.264 protocol encoding channel intra prediction structure.

- Definition

```
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;
```

- Members

Member name	Description
bIntra16x16PredEn	16x16 intra prediction enable, enabled by default. Value range: 0 or 1. 0: not enabled; 1: Enable.
bIntraNxNPredEn	NxN intra prediction is enabled, enabled by default. Value range: 0 or 1. 0: Not enabled; 1: Enable.
bConstrainedIntraPredFlag	The default is 0. Value range: 0 or 1.For details, please refer to the constrained_intra_pred_flag of H.264 protocal.
bIpcmEn	IPCM prediction is enabled, and the default values vary according to different chips. Value range: 0 or 1.
u32Intra16x16Penalty	The default is 0.
u32Intra4x4Penalty	The default is 0.
bIntraPlanarPenalty	The default is 0.

- Note

no.

- Related data types and interfaces

MI_VENC_SetH264IntraPred
MI_VENC_GetH264IntraPred

2.27. MI_VENC_ParamH264Trans_t

- Description

Define the H.264 protocol encoding channel transform and quantize the structure.

- Definition

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

- Members

Member name	Description
u32IntraTransMode	Intra prediction conversion mode: • 0: Supports 4x4, 16x16 conversion, highprofile support. • 1: 4x4 conversion, baseline, main, highprofile are supported. • 2: 8x8 conversion, highprofile support.The system defaults to the selection of TransMode according to the channel protocol type.
u32InterTransMode	Transformation mode of inter prediction: • 0: Supports 4x4, 8x8 conversion, highprofile support. • 1: 4x4 conversion, baseline, main, highprofile are supported. • 2: 8x8 conversion, highprofile support. The system defaults to the selection of TransMode according to the channel protocol type.
s32ChromaQpIndexOffset	For details, see the slice_qp_offset_index of H.264 protocol. The system default is 0. Value range: [-12, 12].

- Note

no.

- Related data types and interfaces

MI_VENC_SetH264Trans
MI_VENC_GetH264Trans

2.28. MI_VENC_ParamH264Entropy_t

- Description

Define the H.264 protocol coding channel entropy coding structure.

- Definition

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

- Members

Member name	Description
u32EntropyEncModeI	I frame entropy coding mode. • 0: cavlc • 1: cabac >=2 has no meaning.

Member name	Description
	Baseline does not support cabac.
u32EntropyEncModeP	P frame entropy coding mode. • 0: cavlc • 1: cabac >=2 has no meaning. Baseline does not support cabac.

➢ Note

no.

➢ Related data types and interfaces

MI_VENC_SetH264Entropy
MI_VENC_GetH264Entropy

2.29. MI_VENC_ParamH265InterPred_t

➢ Description

Define the H.265 protocol coding channel inter-frame prediction structure.

➢ Definition

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

➢ Members

Member name	Description
u32HWSize	Horizontal search window size.
u32VWSize	Vertical search window size.
bInter16x16PredEn	16x16 interframe prediction enable switch, enabled by default.

Member name	Description
bInter16x8PredEn	16x8 interframe prediction enable switch, enabled by default.
bInter8x16PredEn	8x16 interframe prediction enable switch, enabled by default.
bInter8x8PredEn	8x8 interframe prediction enable switch, enabled by default.
bInter8x4PredEn	8x4 interframe prediction enable switch, enabled by default.
bInter4x8PredEn	4x8 interframe prediction enable switch, enabled by default.
bInter4x4PredEn	4x4 interframe prediction enable switch, enabled by default.
u32Inter32x32Penalty	32x32 block inter prediction is selected probability, the larger the value, the lower the probability. The default is 0
u32Inter16x16Penalty	16x16 block inter prediction is selected probability, the larger the value, the lower the probability. The default is 0
u32Inter8x8Penalty	8x8 block inter prediction is selected probability, the larger the value, the lower the probability. The default is 0
bExtedgeEn	When the search encounters the boundary of the image, it is beyond the scope of the image, and whether the enable switch for the search is enabled is enabled by default.

- Note

  no.

- Related data types and interfaces

  MI_VENC_SetH265InterPred
  MI_VENC_GetH265InterPred

2.30. MI_VENC_ParamH265IntraPred_t

- Description

  Define the H.265 protocol coding channel intra prediction structure.

- Definition

```
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;
```

- Members

Member name	Description
bIntra32x32PredEn	32x32 intra prediction enable, enabled by default. Value range: 0 or 1. 0: not enabled; 1: Enable.
bIntra16x16PredEn	16x16 intra prediction enable, enabled by default. Value range: 0 or 1. 0: not enabled; 1: Enable.
bIntra8x8PredEn	8x8 intra prediction enable, enabled by default. Value range: 0 or 1. 0: not enabled; 1: Enable.
bConstrainedIntraPredFlag	The default is 0. Value range: 0 or 1.For details, please refer to the constrained_intra_pred_flag of H.265 protocol.
u32Intra32x32Penalty	32x32 block intra prediction is selected probability, the larger the value, the lower the probability. The default is 0.
u32Intra16x16Penalty	16 x 16 block intra prediction is selected probability, the larger the value, the lower the probability The default is 0.
u32Intra 8 x 8 Penalty	8 x 8 block intra prediction is selected probability, the larger the value, the lower the probability The default is 0.

- Note

no.

- Related data types and interfaces

MI_VENC_SetH265IntraPred
MI_VENC_GetH265IntraPred

2.31. MI_VENC_ParamH265Trans_t

- Description

Define the H.265 protocol coding channel transform and quantize the structure.

- Definition

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

- Members

Member name	Description
u32IntraTransMode	Intra prediction conversion mode: • 0: Supports 4x4, 16x16, 32x32 conversion, highprofile support. • 1: 4x4 conversion, baseline, main, highprofile are supported. • 2: 8x8 conversion, highprofile support. The system defaults to the selection of TransMode according to the channel protocol type.
u32InterTransMode	Transformation mode of inter prediction: • 0: Supports 4x4, 8x8 conversion, highprofile support. • 1: 4x4 conversion, baseline, main, highprofile are supported. • 2: 8x8 conversion, highprofile support. The system defaults to the selection of TransMode according to the channel protocol type.
s32ChromaQpIndexOffset	For details, please refer to the slice_cb_qp_offset of H.265 protocol. The system default is 0. Value range: [-12, 12].

- Note

no.

- Related data types and interfaces

MI_VENC_SetH265Trans
MI_VENC_GetH265Trans

2.32. MI_VENC_ParamH264Dblk_t

- Description

Define the H.264 protocol encoding channel Dblk structure.

- Definition

```
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;
```

- Members

Member name	Description
Disable_deblocking_filter_idc	The value range is [0, 2], and the default value is 0. For details, see the H.264 protocol.
Slice_alpha_c0_offset_div2	The value range [-6,6], the default value is 0. For details, see the H.264 protocol.
Slice_beta_offset_div2	The value range [-6,6], the default value is 0. For details, see the H.264 protocol.

- Note

no.

- Related data types and interfaces

MI_VENC_SetH264Dblk
MI_VENC_GetH264Dblk

2.33. MI_VENC_ParamH265Dblk_t

- Description

Define the H.265 protocol encoding channel Dblk structure.

- Definition

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

- Members

Member name	Description
disable_deblocking_filter_idc	The value range is [0, 2], and the default value is 0. For details, see the slice_deblocking_filter_disabled_flag of H.265 Protocol.
slice_tc_offset_div2	The value range is [-6,6], and the default value is 0. For details, see H.265 Protocol.
slice_beta_offset_div2	The value range is [-6,6], and the default value is 0. For details, see H.265 Protocol.

- Note

no.

- Related data types and interfaces
  MI_VENC_SetH265Dblk
  MI_VENC_GetH265Dblk

2.34. MI_VENC_ParamH264Vui_t

- Description
  Define the Vui structure of the H.264 protocol encoding channel.

- definition

```
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;
```

- member

member name	description
stVuiAspectRatio	For details, see the Annex E Video usability information of H.264 protocol.
stVuiTimeInfo	For details, see the Annex E Video usability information of H.264 protocol.
stVuiVideoSignal	For details, see the Annex E Video usability information of H.264 protocol.

- precaution
  no.

- Related data types and interfaces
  MI_VENC_SetH264Vui
  MI_VENC_GetH264Vui

2.35. MI_VENC_ParamH264VuiAspectRatio_t

- Description
  A structure that defines the AspectRatio information in the H.264 protocol encoding channel Vui.

- definition

```
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;
```

- member

member name	description
u8AspectRatioInfoPresentFlag	For details, see the H.264 protocol. The system defaults to 0. Value range: 0 or 1.
u8AspectRatioIdc	For details, see the Annex E Video usability information of H.264 Protocol. The system defaults to 1. Value range: [0, 255], 17~254 reserved.
u8OverscanInfoPresentFlag	For details, see the Annex E Video usability information of H.264 protocol. The system defaults to 0. Value range: 0 or 1.
u8OverscanAppropriateFlag	For details, see the Annex E Video usability information of H.264 protocol. The system defaults to 0. Value range: 0 or 1.
u16SarWidth	For details, see the Annex E Video usability information of H.264 Protocol. The system defaults to 1. The value range is (0,65535) and is relatively prime to u16SarHeight.
u16SarHeight	For details, see the Annex E Video usability information of H.264 Protocol. The system defaults to 1. The value range is (0,65535) and is mutually prime with u16SarWidth.

- Precaution

no.

- Related data types and interfaces

MI_VENC_SetH264Vui

2.36. MI_VENC_ParamH264VuiTimeInfo_t

- Description

A structure that defines TIME_INFO information in the H.264 protocol encoding channel Vui.

- definition

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

- member

member name	description
u8TimingInfoPresentFlag	For details, see the Annex E Video usability information of H.264 protocol. The system defaults to 0. Value range: 0 or 1.
u32NumUnitsInTick	For details, see the Annex E Video usability information of H.264 Protocol. The system defaults to 1. Value range: greater than 0.
u32TimeScale	For details, see the Annex E Video usability information of H.264 Protocol. The system defaults to 60. Value range: greater than 0.
u8FixedFrameRateFlag;	For details, see the Annex E Video usability information of H.264 Protocol. The system defaults to 1. Value range: 0 or 1.

- precaution

no.

- Related data types and interfaces

MI_VENC_SetH264Vui
MI_VENC_GetH264Vui

2.37. MI_VENC_ParamH264VuiVideoSignal_t

- Description

A structure that defines VIDEO_SIGNAL information in the H.264 protocol encoding channel Vui.

- definition

```
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;
```

- member

member name	description
u8VideoSignalTypePresentFlag	For details, see the Annex E Video usability information of H.264 Protocol. The system defaults to 1. Value range: 0 or 1.
u8VideoFormat	For details, see the Annex E Video usability information of H.264 protocol. The system defaults to 5.Value range: [0,7].
u8VideoFullRangeFlag	For details, see the Annex E Video usability information of H.264 Protocol. The system defaults to 1. Value range: 0 or 1.
u8ColourDescriptionPresentFlag;	For details, see the Annex E Video usability information of H.264 Protocol. The system defaults to 1. Value range: 0 or 1.
u8ColourPrimaries	For details, see the Annex E Video usability information of H.264 Protocol. The system defaults to 1. Value range: [0, 255].
u8TransferCharacteristics	For details, see the Annex E Video usability information of H.264 Protocol. The system defaults to 1. Value range: [0, 255].
u8MatrixCoefficients	For details, see the Annex E Video usability information of H.264 Protocol. The system defaults to 1. Value range: [0, 255].

- precaution

  no.

- Related data types and interfaces

  MI_VENC_SetH264Vui

2.38. MI_VENC_ParamH265Vui_t

- Description

  Define the H.265 protocol encoding channel Vui structure.

- definition

```
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;
```

- member

member name	description
stVuiAspectRatio	For details, please refer to the Annex E Video usability information of H.265 protocol.
stVuiTimeInfo	For details, please refer to the Annex E Video usability information of H.265 protocol.
stVuiVideoSignal	For details, please refer to the Annex E Video usability information of H.265 protocol.

- precaution

no.

- Related data types and interfaces

MI_VENC_SetH265Vui
MI_VENC_GetH265Vui

2.39. MI_VENC_ParamH265VuiAspectRatio_t

- Description

A structure that defines the AspectRatio information in the H.265 protocol encoding channel Vui.

- definition

```
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;
```

- member

member name	description
u8AspectRatioInfoPresentFlag	For details, see the Annex E Video usability information of H.265 Protocol. The system defaults to 0. Value range: 0 or 1.

member name	description
u8AspectRatioIdc	For details, see the Annex E Video usability information of H.265 Protocol. The system defaults to 1. Value range: [0, 255], 17~254 reserved.
u8OverscanInfoPresentFlag	For details, see the Annex E Video usability information of H.265 Protocol. The system defaults to 0. Value range: 0 or 1.
u8OverscanAppropriateFlag	For details, see the Annex E Video usability information of H.265 Protocol. The system defaults to 0. Value range: 0 or 1.
u16SarWidth	For details, see the Annex E Video usability information of H.265 Protocol. The system defaults to 1. The value range is (0,65535) and is relatively prime to u16SarHeight.
u16SarHeight	For details, see the Annex E Video usability information of H.265 Protocol. The system defaults to 1. The value range is (0,65535) and is mutually prime with u16SarWidth.

- precaution

  no.

- Related data types and interfaces

  MI_VENC_SetH265Vui

2.40. MI_VENC_ParamH265VuiTimeInfo_t

- Description

  TIME_INFO definition structure information encoding channel Vui H.265 of protocol

- definition

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

- member

member name	description
u8TimingInfoPresentFlag	For details, see the Annex E Video usability information of H.265 Protocol. The system defaults to 0. Value range: 0 or 1.
u32NumUnitsInTick	For details, see the Annex E Video usability information of H.265 Protocol. The system defaults to 1. Value range: greater than 0.
u32TimeScale	For details, see the Annex E Video usability

member name	description
	information of H.265 Protocol. The system default is 60. Value range: greater than 0.
u8FixedFrameRateFlag;	For details, see the Annex E Video usability information of H.265 Protocol. The system defaults to 1. Value range: 0 or 1.

- precaution

  no.

- Related data types and interfaces

  MI_VENC_SetH265Vui
  MI_VENC_GetH265Vui

2.41. MI_VENC_ParamH265VuiVideoSignal_t

- Description

  H.265 protocol defined in VIDEO_SIGNAL Vui channel coding structure information.

- definition

```
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;
```

- member

member name	description
u8VideoSignalTypePresentFlag	For details, see the Annex E Video usability information of H.265 Protocol. The system defaults to 1. Value range: 0 or 1.
u8VideoFormat	For details, see the Annex E Video usability information of H.265 Protocol. The system defaults to 5. Value range: [0,7].
u8VideoFullRangeFlag	For details, see the Annex E Video usability information of H.265 Protocol. The system defaults to 1. Value range: 0 or 1.
u8ColourDescriptionPresentFlag;	For details, see the Annex E Video usability information of H.265 Protocol. The system

member name	description
	defaults to 1. Value range: 0 or 1.
u8ColourPrimaries	For details, see the Annex E Video usability information of H.265 Protocol. The system defaults to 1. Value range: [0, 255].
u8TransferCharacteristics	For details, see the Annex E Video usability information of H.265 Protocol. The system defaults to 1. Value range: [0, 255].
u8MatrixCoefficients	For details, see the Annex E Video usability information of H.265 Protocol. The system defaults to 1. Value range: [0, 255].

- precaution

  no.

- Related data types and interfaces

  MI_VENC_SetH265Vui

2.42. MI_VENC_ParamJpeg_t

- Description

  Define the JPEG protocol encoding channel advanced parameter structure.

- definition

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

- member

member name	description
u32Qfactor	For details, see RFC2435. The system defaults to 90. Value range: [1,99].
au8YQt	Y quantization table. Value range: [1,255].
au8CbCrQt	Cb Cr quantization table. Value range: [1,255].
u32MCUPerEcs	How many MCUs are included in each ECS, the system defaults to 0, indicating no Divide Ecs.

member name	description
	u32MCUPerECS:[0,(picwidth+15)>>4x(pi cheight+15)>>4x2]

- precaution

  no.

- Related data types and interfaces

  MI_VENC_SetJpegParam
  MI_VENC_GetJpegParam

2.43. MI_VENC_RoiCfg_t

- Description

  Define the information of the region of interest of the code.

- definition

```
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;
```

- member

member name	description
u32Index	The index of the ROI area, the index range supported by the system is [0,7], not supported. An index beyond this range.
bEnable	Whether to enable this ROI area.
bAbsQp	ROI area QP mode. • MI_FALSE: Relative QP • MI_TURE: Absolute QP
s32Qp	QP value, when QP mode is MI_FALSE, s32Qp is QP offset, s32Qp range [-51, 51], when QP mode is MI_TRUE, s32Qp is macroblock QP value, s32Qp range [0, 51].
stRect	ROI area.

- precaution

  The QP mode of the ROI area could only support the relative QP (bAbsQp = MI_FALSE) from SSC33X. As a result, the value range of s32Qp is modified to [-32, 31].

- Related data types and interfaces
  MI_VENC_SetRoiCfg
  MI_VENC_GetRoiCfg

2.44. MI_VENC_RoiBgFrameRate_t

- Description
  Define the frame rate of the non-coded region of interest.

- definition

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

- member

member name	description
s32SrcFrmRate	Source frame rate for non-Roi regions.
s32DstFrmRate	The target frame rate of the non-Roi area.

- precaution
  no.

- Related data types and interfaces
  MI_VENC_SetRoiBgFrameRate
  MI_VENC_GetRoiBgFrameRate

2.45. MI_VENC_ParamRef_t

- Description
  Define advanced frame skipping reference parameters for H.264/H.265 encoding.

- definition

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

- member

member name	description
u32Base	The period of the base layer.Value range: (0, +∞).
u32Enhance	The period of the enhance layer.Value range: [0, 255].

member name	description
bEnablePred	Whether the frame representing the base layer is used as a reference by other frames in the base layer. When When MI_FALSE, all frames of the base layer refer to the IDR frame.

- precaution

  no.

- Related data types and interfaces

  no.

2.46. MI_VENC_RcAttr_t

- Description

  Define the encoding channel rate controller attribute.

- definition

```
typedef struct MI_VENC_RcAttr_s
{
    MI_VENC_RcMode_e      eRcMode;
    union
    {
     MI_VENC_AttrH264Cbr_t          stAttrH264Cbr;
     MI_VENC_AttrH264Vbr_t          stAttrH264Vbr;
     MI_VENC_AttrH264FixQp_t   stAttrH264FixQp;
     MI_VENC_AttrH264Abr_t          stAttrH264Abr;
     MI_VENC_AttrH264Avbr_t stAttrH264Avbr;
     MI_VENC_AttrMjpegCbr_t     stAttrMjpegCbr;
     MI_VENC_AttrMjpegFixQp_t  stAttrMjpegFixQp;
     MI_VENC_AttrH265Cbr_t          stAttrH265Cbr;
     MI_VENC_AttrH265Vbr_t          stAttrH265Vbr;
     MI_VENC_AttrH265FixQp_t   stAttrH265FixQp;
     MI_VENC_AttrH265Avbr_t stAttrH265Avbr;
    };
    MI_VOID* pRcAttr;
}MI_VENC_RcAttr_t;
```

- member

member name	description
enRcMode	RC mode.
stAttrH264Cbr	H.264 protocol encoding channel Cbr mode attribute.
stAttrH264Vbr	H.264 protocol encoding channel Vbr mode attribute.
stAttrH264FixQp	H.264 protocol encoding channel Fixqp mode attribute.
stAttrH264Abr	H.264 protocol encoding channel Abr mode attribute

member name	description
	(not supported at this time).
stAttrH264Avbr	H.264 protocol encoding channel Avbr mode attribute.
stAttrMjpegCbr	MJPEG protocol encoding channel Cbr mode attribute.
stAttrMjpegFixQp	MJPEG protocol encoding channel FixQp mode attribute.
stAttrH265Cbr	H.265 protocol encoding channel Cbr mode attribute.
stAttrH265Vbr	H.265 protocol encoding channel Vbr mode attribute.
stAttrH265FixQp	H.265 protocol encoding channel Fixqp mode attribute.
stAttrH265Avbr	H.265 protocol encoding channel Avbr mode attribute.

- precaution

  no.

- Related data types and interfaces

  MI_VENC_CreateChn

2.47. MI_VENC_RcMode_e

- Description

  Define the code channel rate controller mode.

- definition

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

- member

member name	description
E_MI_VENC_RC_MODE_H264CBR	H.264 CBR mode.
E_MI_VENC_RC_MODE_H264VBR	H.264 VBR mode.
E_MI_VENC_RC_MODE_H264ABR	H.264 ABR mode (not available at the moment)
E_MI_VENC_RC_MODE_H264FIXQP	H.264 FixQp mode.
E_MI_VENC_RC_MODE_H264AVBR	H.264 AVBR mode
E_MI_VENC_RC_MODE_MJPEGCBR	MJPEG CBR mode
E_MI_VENC_RC_MODE_H265CBR	H.265 CBR mode.

member name	description
E_MI_VENC_RC_MODE_H265VBR	H.265 VBR mode.
E_MI_VENC_RC_MODE_H265FIXQP	H.265 FixQp mode
E_MI_VENC_RC_MODE_H265AVBR	H.265 AVBR mode

- precaution

  no.

- Related data types and interfaces

  MI_VENC_CreateChn

2.48. MI_VENC_AttrH264Cbr_t

- Description

  Define the CBR attribute structure of the H.264 encoding channel.

- definition

```
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;
```

- member

member name	description
u32Gop	H.264gop value. Value range: [1,65536].
u32StatTime	CBR code rate statistics time in seconds. Value range: [1, 60].
u32SrcFrmRateNum	The encoder frame rate molecule, in integer units.
u32SrcFrmRateDen	Encoder frame rate denominator, in integer units.
u32BitRate	Average bitrate, in kbps.Value range: [2,102400].
u32FluctuateLevel	The maximum code rate relative to the average bit rate fluctuation level, reserved, temporarily Do not use. Value range: [0, 5]. A volatility level of 0 is recommended.

- precaution

  Src F rmRate should be set to the actual frame rate of the input encoder, and RC needs to be rate controlled according to SrcFrmRate.u32SrcFrmRateNum / u32SrcFrmRateDen range between (0,30].

- Related data types and interfaces
  MI_VENC_CreateChn

2.49. MI_VENC_AttrH264Vbr_t

- Description
  Define the VBR attribute structure of the H.264 encoding channel.

- definition

```
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;
```

- member

member name	description
u32Gop	H.264gop value. Value range: [1,65536].
u32StatTime	VBR code rate statistics time in seconds. Value range: [1, 60].
u32SrcFrmRateNum	Encoder frame rate numerator, in integer units.
u32SrcFrmRateDen	Encoder denominator framerate, in units of integers.
u32MaxBitRate	The encoder outputs the maximum code rate in kbps. Value range: [2,102400].
u32MaxQp	The encoder supports image maximum QP. Value range: (u32MinQp, 51].
u32MinQp	The encoder supports image minimum QP. Value range: [0, 51].

- precaution
  See MI_VENC_AttrH264Cbr_t for a description of u32SrcFrmRateNum and u32SrcFrmRateDen.

- Related data types and interfaces
  MI_VENC_CreateChn

2.50. MI_VENC_AttrH264FixQp_t

- Description
  Define the Fixqp attribute structure of the H.264 encoding channel.

- definition

```
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;
```

- member

member name	description
u32Gop	H.264gop value. Value range: [1,65536].
u32SrcFrmRateNum	Encoder frame rate numerator, in integer units.
u32SrcFrmRateDen	Encoder denominator framerate, in units of integers.
u32IQp	I frame all macroblock Qp values. Value range: [0, 51].
u32PQp	P frame all macroblock Qp values. Value range: [0, 51].

- precaution

See MI_VENC_AttrH264Cbr_t for a description of u32SrcFrmRateNum and u32SrcFrmRateDen.

- Related data types and interfaces

MI_VENC_CreateChn

2.51. MI_VENC_AttrH264Abr_t

- Description

Define the ABR attribute structure of the H.264 encoding channel.

- definition

```
typedef struct MI_VENC_AttrH264Abr_s
{
  MI_U32      u32Gop;
  MI_U32      u32StatTime;
  MI_U32      u32SrcFrmRateNum;
  MI_U32      u32SrcFrmRateDen;
  MI_U32      u32AvgBitRate;
  MI_U32      u32MaxBitRate;
}MI_VENC_AttrH264Abr_t;
```

- member

member name	description
u32Gop	H.264gop value. Value range: [1,65536]
u32StatTime	ABR code rate statistics time in seconds. Value range: [1, 60].
u32SrcFrmRateNum	Encoder frame rate numerator, in integer units.
u32SrcFrmRateDen	Encoder denominator framerate, in units of integers.
u32AvgBitRate	Average code rate. Value range [2000, u32MaxBitRate)
u32MaxBitRate	Maximum code rate. Value range [2000,102400000]

- precaution

See MI_VENC_AttrH264Cbr_t for a description of u32SrcFrmRateNum and u32SrcFrmRateDen.

- Related data types and interfaces

MI_VENC_CreateChn

2.52. MI_VENC_AttrH264Avbr_t

- Description

Define the AVBR attribute structure of the H.264 encoding channel.

- definition

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

- member

member name	description
u32Gop	gop value. Value range: [1,65536].
u32StatTime	Code rate statistics time in seconds. Value range: [1, 60].
u32SrcFrmRateNum	Encoder frame rate numerator, in integer units.
u32SrcFrmRateDen	Encoder denominator framerate, in units of integers.
u32MaxBitRate	The encoder outputs the maximum code rate in kbps. Value range: [2,102400].

member name	description
u32MaxQp	The encoder supports image maximum QP. Value range: (u32MinQp, 48].
u32MinQp	The encoder supports image minimum QP. Value range: [12, 48].

- precaution

  See MI_VENC_AttrH264Cbr_t for a description of u32SrcFrmRateNum and u32SrcFrmRateDen.

- Related data types and interfaces

  MI_VENC_CreateChn

2.53. MI_VENC_AttrMjpegCbr_t

- Description

  Define the CBR attribute structure of the MJPEG encoding channel.

- definition

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

- member

member name	description
u32BitRate	Stream bit rate. Value range: [2000,102400000]
u32SrcFrmRateNum	Encoder frame rate numerator, in integer units.
u32SrcFrmRateDen	Encoder denominator framerate, in units of integers.

- precaution

  See MI_VENC_AttrH264Cbr_t for a description of u32SrcFrmRateNum and u32SrcFrmRateDen.

- Related data types and interfaces

  MI_VENC_CreateChn

2.54. MI_VENC_AttrMjpegFixQp_t

- Description

  Define the CBR attribute structure of the MJPEG encoding channel.

- definition

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

- member

member name	description
u32Qfactor	Qfactor. Value range: [10,90]
u32SrcFrmRateNum	Encoder frame rate numerator, in integer units.
u32SrcFrmRateDen	Encoder denominator framerate, in units of integers.

- precaution

See MI_VENC_AttrH264Cbr_t for a description of u32SrcFrmRateNum and u32SrcFrmRateDen.

- Related data types and interfaces

MI_VENC_CreateChn

2.55. MI_VENC_AttrH265Cbr_t

- Description

Define the HBR 265 code channel CBR attribute structure.

- definition

```
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;
```

- member

member name	description
u32Gop	H.265gop value. Value range: [1,65536].
u32StatTime	CBR code rate statistics time in seconds. Value range: [1, 60].
u32SrcFrmRateNum	Encoder frame rate numerator, in integer units.
u32SrcFrmRateDen	Encoder denominator framerate, in units of integers.
u32BitRate	Average bitrate, in kbps.Value range: [2,102400].

member name	description
u32FluctuateLevel	The maximum code rate relative to the average bit rate fluctuation level reserved, temporarily Do not use. Value range: [0, 5]. A volatility level of 0 is recommended.

- precaution

  See MI_VENC_AttrH264Cbr_t for a description of u32SrcFrmRateNum and u32SrcFrmRateDen.

- Related data types and interfaces

  MI_VENC_CreateChn

2.56. MI_VENC_AttrH265Vbr_t

- Description

  Define the HBR attribute channel VBR attribute structure.

- definition

```
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;
```

- member

member name	description
u32Gop	H.265gop value. Value range: [1,65536]
u32StatTime	VBR code rate statistics time in seconds. Value range: [1, 60].
u32SrcFrmRateNum	Encoder frame rate numerator, in integer units.
u32SrcFrmRateDen	Encoder denominator frame rate, in units of integers.
u32MaxBitRate	The encoder outputs the maximum code rate in kbps. Value range: [2,102400].
u32MaxQp	The encoder supports image maximum QP. Value range: (u32MinQp, 51].
u32MinQp	The encoder supports image minimum QP. Value range: [0, 51].

- precaution

  See MI_VENC_AttrH264Cbr_t for a description of u32SrcFrmRateNum and u32SrcFrmRateDen.

- Related data types and interfaces
  MI_VENC_CreateChn

2.57. MI_VENC_AttrH265FixQp_t

- Description
  Define the H.265 encoding channel Fixqp attribute structure.

- definition

```
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;
```

- member

member name	description
u32Gop	H.265gop value. Value range: [1,65536].
u32SrcFrmRateNum	Encoder frame rate numerator, in integer units.
u32SrcFrmRateDen	Encoder frame rate denominator, in integer units.
u32IQp	I frame all macroblock Qp values. Value range: [0, 51].
u32PQp	P frame all macroblock Qp values. Value range: [0, 51].

- precaution
  See MI_VENC_AttrH264Cbr_t for a description of u32SrcFrmRateNum and u32SrcFrmRateDen.

- Related data types and interfaces
  MI_VENC_CreateChn

2.58. MI_VENC_AttrH265Avbr_t

- Description
  Define the AVBR attribute structure of the H.265 encoding channel.

- definition

```
typedef struct MI_VENC_AttrH265Avbr_s
{
    MI_U32 u32Gop;
    MI_U32 u32StatTime;
```

```
    MI_U32 u32SrcFrmRateNum;
    MI_U32 u32SrcFrmRateDen;
    MI_U32 u32MaxBitRate;
    MI_U32 u32MaxQp;
    MI_U32 u32MinQp;
} MI_VENC_AttrH265Avbr_t;
```

- member

member name	description
u32Gop	gop value. Value range: [1,65536].
u32StatTime	Code rate statistics time in seconds. Value range: [1, 60].
u32SrcFrmRateNum	Encoder frame rate numerator, in integer units.
u32SrcFrmRateDen	Encoder denominator framerate, in units of integers.
u32MaxBitRate	The encoder outputs the maximum code rate in kbps. Value range: [2,102400].
u32MaxQp	The encoder supports image maximum QP. Value range: (u32MinQp, 48].
u32MinQp	The encoder supports image minimum QP. Value range: [12, 48].

- precaution

  See MI_VENC_AttrH264Cbr_t for a description of u32SrcFrmRateNum and u32SrcFrmRateDen.

- Related data types and interfaces

  MI_VENC_CreateChn

2.59. MI_VENC_SuperFrmMode_e

- Description

  Define the superframe processing mode in rate control.

- definition

```
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;
```

- member

member name	description
E_MI_VENC_SUPERFRM_NONE	No special strategy.
E_MI_VENC_SUPERFRM_DISCARD	Discard jumbo frames.
E_MI_VENC_SUPERFRM_REENCODE	Reprogram oversized frames.

- precaution

  no.

- Related data types and interfaces

  MI_VENC_GetRcParam

2.60. MI_VENC_ParamH264Vbr_t

- Description

  Define the H264 protocol encoding channel VBR rate control mode advanced parameter configuration.

- definition

```
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;
```

- member

member name	description
s32IPQPDelta	IPQP change value.Value range: [-12, 12].
s32ChangePos	The ratio of the code rate at the time when VBR starts to adjust Qp with respect to the maximum code rate. Value range: [50,100].
u32MaxIQp	The maxmum QP of the I frame. The minimum number of bits used to control the I frame.Value range: (MinQp, MaxQp].
u32MinIQP	The minimum QP of the I frame. The maximum number of bits used to control the I frame.Value range: [MinQp, MaxQp).
u32MaxIPProp	Maximum ratio of I/P frame size, range of value [5,100].

- precaution

  no.

- Related data types and interfaces
  MI_VENC_GetRcParam
  MI_VENC_SetRcParam

2.61. MI_VENC_ParamH264Cbr_t

- Description
  Define the H264 protocol encoding channel CBR new version rate control mode advanced parameter configuration.

- definition

```
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;
```

- member

member name	description
u32MaxQp	Frame maximum QP for clamp quality. Value range: (u32MinQp, 51].
u32MinQp	Frame minimum QP for clamp rate fluctuations. Value range: [0, u32MaxQp).
s32IPQPDelta	IPQP change value. Value range: [-12,12].
u32MaxIQp	The maximum QP of the I frame. The minimum number of bits used to control the I frame. Value range: [u32MinIQp, 51].
u32MinIQp	The minimum QP of the I frame. The maximum number of bits used to control the I frame. Range of values: [0, u32MaxIQp).
u32MaxIPProp	Maximum ratio of I/P frame size, range of value [5,100].

- precaution
  no.

- Related data types and interfaces
  MI_VENC_GetRcParam
  MI_VENC_SetRcParam

2.62. MI_VENC_ParamH264Avbr_t

- Description

  Define the H264 protocol encoding channel AVBR rate control mode advanced parameter configuration.

- definition

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

- member

member name	description
s32IPQPDelta	IPQP change value.Value range: [-12, 12]. default value:0
s32ChangePos	The ratio of the code rate at the time when VBR starts to adjust Qp with respect to the maximum code rate. Value range: [50,100]. default value:80
u32MinIQP	The minimum QP of the I frame. The maximum number of bits used to control the I frame.Value range: [MinQp, MaxQp). default value:12
u32MaxIPProp	Maximum ratio of I/P frame size, range of value [5,100].
u32MaxIQp	The maximum QP of the I frame. The minimum number of bits used to control the I frame.Value range: (MinQp, MaxQp]. default value:48
u32MaxISize	The maximum size of the I frame.Rate control will try to let the target size not exceed the max size.If value is 0, it means that rate control will not consider the parameter. Value range:[0, 800*1024] default value:0
u32MaxPSize	The maximum size of the P frame.Rate control will try

member name	description
	to let the target size not exceed the max size. If value is 0, it means that rate control will not consider the parameter. Value range:[0, 800*1024] default value:0
u32MinStillPercent	The minimum percent of target bitrate in still state.If the value set to 100,then AVBR will not actively decrease target rate in still state,so the behavior of AVBR will be same with VBR. Value range:[5,100]
u32MaxStillQp	The maximum QP of the I frame in still state. Value range:[MinIQp,MaxIQp] Not support yet.
u32MotionSensitivity	The sensitivity of adjusting bitrate depending on the motion of picture. Not support yet.

- precaution

  no.

- Related data types and interfaces

  MI_VENC_GetRcParam
  MI_VENC_SetRcParam

2.63. MI_VENC_ParamMjpegCbr_t

- Description

  Define the MJPEG protocol encoding channel CBR new version rate control mode advanced parameter configuration.

- definition

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

- member

member name	description
u32MaxQfactor	Maximum Qfactor for clamp quality. Value range: (u32MinQfactor, 90].
u32MinQfactor	Minimum Qfactor for clamp quality. Value range: [10, u32MaxQfactor).

- precaution

  no.

- Related data types and interfaces

  MI_VENC_GetRcParam
  MI_VENC_SetRcParam

2.64. MI_VENC_ParamH265Vbr_t

- Description

  Define the advanced parameter configuration of the VBR rate control mode of the H265 protocol code channel.

- definition

```
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;
```

- member

member name	description
s32IPQPDelta	IPQP change value.Value range: [-12, 12].
s32ChangePos	The ratio of the code rate at the time when VBR starts to adjust Qp with respect to the maximum code rate. Value range: [50,100].
u32MaxIQp	The maximum QP of the I frame. The minimum number of bits used to control the I frame.Value range: (MinQp, MaxQp].
u32MinIQP	The minimum QP of the I frame. The maximum number of bits used to control the I frame.Value range: [MinQp, MaxQp).
u32MaxIPProp	Minimum Qfactor for clamp quality. Value range: [10, u32MaxQfactor).

- precaution

  no.

- Related data types and interfaces

  MI_VENC_GetRcParam
  MI_VENC_SetRcParam

2.65. MI_VENC_ParamH265Cbr_t

- Description

  Define the H265 protocol encoding channel CBR new version rate control mode advanced parameter configuration.

- definition

```
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;
```

- member

member name	description
u32MaxQp	Frame maximum QP for clamp quality. Value range: (u32MinQp, 51].
u32MinQp	Frame minimum QP for clamp rate fluctuations. Value range: [0, u32MaxQp).
s32IPQPDelta	IPQP change value.Value range: [-12, 12].
u32MaxIQp	The maximum QP of the I frame. The minimum number of bits used to control the I frame.Value range: [u32MinIQp, u32MaxQp).
u32MinIQp	The minimum QP of the I frame. The maximum number of bits used to control the I frame.Value range: [u32MinQp, u32MaxIQp).
u32MaxIPProp	Minimum Qfactor for clamp quality. Value range: [10, u32MaxQfactor).

- precaution

  no.

- Related data types and interfaces

  MI_VENC_GetRcParam
  MI_VENC_SetRcParam

2.66. MI_VENC_ParamH265Avbr_t

- Description

  Define the H265 protocol encoding channel AVBR rate control mode advanced parameter configuration.

- definition

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

- member

member name	description
s32IPQPDelta	IPQP change value.Value range: [-12, 12]. default value:0
s32ChangePos	The ratio of the code rate at the time when VBR starts to adjust Qp with respect to the maximum code rate. Value range: [50,100]. default value:80
u32MinIQP	The minimum QP of the I frame. The maximum number of bits used to control the I frame.Value range: [MinQp, MaxQp). default value:12
u32MaxIPProp	Maximum ratio of I/P frame size, range of value [5,100].
u32MaxIQp	The maxmum QP of the I frame. The minimum number of bits used to control the I frame.Value range: (MinQp, MaxQp]. default value:48
u32MaxISize	The maximum size of the I frame.Rate control will try to let the target size not exceed the max size. If value is 0, it means that rate control will not consider the parameter. Value range:[0, 800*1024] default value:0
u32MaxPSize	The maximum size of the P frame.Rate control will try to let the target size not exceed the max size. If value is 0, it means that rate control will not consider the parameter. Value range:[0, 800*1024] default value:0

member name	description
u32MinStillPercent	The minimum percent of target bitrate in still state.If the value set to 100,then AVBR will not actively decrease target rate in still state,so the behavior of AVBR will be same with VBR. Value range:[5,100]
u32MaxStillQp	The maximum QP of the I frame in still state. Value range:[MinIQp,MaxIQp] Not support yet.
u32MotionSensitivity	The sensitivity of adjusting bitrate depending on the motion of picture. Not support yet.

- precaution

  no.

- Related data types and interfaces

  MI_VENC_GetRcParam
  MI_VENC_SetRcParam

2.67. MI_VENC_RcParam_t

- Description

  Define the rate control advanced parameters of the code channel.

- definition

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

- member

member name	description
u32ThrdI	The mad threshold of the I frame macroblock level code rate control. Value range: [0, 255].
u32ThrdP	The mad threshold of the P frame macroblock level rate control. Value range: [0, 255].
u32RowQpDelta	Row level rate control. Value range: [0,10].
stParamH264Cbr	H.264 channel CBR (ConstantBitRate) rate control mode advanced parameters.
stParamH264Vbr	H.264 channel VBR (VariableBitRate) rate control mode advanced parameters.
stParamH264Avbr	H.264 channel AVBR (AdaptiveVariableBitRate) rate control mode advanced parameters.
stParamH265Cbr	H.265 channel CBR (ConstantBitRate) rate control mode advanced parameters.
stParamH265Vbr	H.265 channel VBR (VariableBitRate) rate control mode advanced parameters.
stParamH265Avbr	H.265 channel AVBR (AdaptiveVariableBitRate) rate control mode advanced parameters.
pRcParam	The parameters are reserved and are not currently used.

- precaution

  no.

- Related data types and interfaces

  MI_VENC_SetRcParam
  MI_VENC_GetRcParam

2.68. MI_VENC_CropCfg_t

- Description

  Define the channel intercept parameter.

- definition

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

- member

member name	description
bEnable	Whether to cut. MI_TRUE: Enable cropping. MI_FALSE: Cropping is not enabled.
stRect	Cropped area.

member name	description
	stRect.s32X: H.264 must be 16 pixels aligned and H.265 must be 32 pixels aligned. stRect.s32Y: H.264/H.265 must be 2 pixels aligned. stRect.u32Width, s32Rect.u32Height: H.264 must be 16 pixels aligned, H.265 must be 32 pixels aligned.

- precaution

  no.

- Related data types and interfaces

  MI_VENC_SetCrop
  MI_VENC_GetCrop

2.69. MI_VENC_RecvPicParam_t

- Description

  Receives the specified frame number image encoding.

- definition

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

- member

member name	description
s32RecvPicNum	The number of frames that the encoding channel continuously receives and encodes.

- precaution

  no.

- Related data types and interfaces

  MI_VENC_StartRecvPicEx

2.70. MI_VENC_H264eIdrPicIdMode_e

- Description

  Set the mode of the IDR frame or the idr_pic_id of the I frame.

- definition

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

- member

member name	description
E_MI_VENC_H264E_IDR_PIC_ID_MODE_USR	User mode; ie idr_pic_id is set by the user.

- precaution

  no.

- Related data types and interfaces

  MI_VENC_SetH264IdrPicId

2.71. MI_VENC_H264IdrPicIdCfg_t

- Description

  The idr_pic_id parameter of the IDR frame or I frame.

- definition

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

- member

member name	description
enH264eIdrPicIdMode	Set the mode of idr_pic_id.
u32H264eIdrPicId	The value of idr_pic_id. The value range is: [0,65535].

- precaution

  no.

- Related data types and interfaces

  MI_VENC_SetH264IdrPicId

2.72. MI_VENC_FrameLostMode_e

- Description

  Drop frame mode when the instantaneous code rate exceeds the threshold.

- definition

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

- member

member name	description
E_MI_VENC_FRMLOST_NORMAL	Normal frame loss when the instantaneous code rate exceeds the threshold.
E_MI_VENC_FRMLOST_PSKIP	The pskip frame is encoded when the instantaneous code rate exceeds the threshold.

- precaution

no.

- Related data types and interfaces

MI_VENC_SetFrameLostStrategy
MI_VENC_GetFrameLostStrategy

2.73. MI_VENC_ParamFrameLost_t

- Description

The frame loss policy parameter when the instantaneous code rate exceeds the threshold.

- definition

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

- member

member name	description
bFrmLostOpen	The frame loss switch when the instantaneous code rate exceeds the threshold.
u32FrmLostBpsThr	Drop frame threshold. (in bits/s)
enFrmLostMode	Frame loss mode when the instantaneous code rate exceeds the threshold.
u32EncFrmGaps	Frame loss interval, the default is 0.Value range: [0,65535]

- precaution

  no.

- Related data types and interfaces

  MI_VENC_SetFrameLostStrategy

2.74. MI_VENC_SuperFrameCfg_t

- Description

  Very large frame processing strategy parameters.

- definition

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

- member

member name	description
eSuperFrmMode	Large frame processing mode.
u32SuperIFrmBitsThr	The I frame has a large threshold and defaults to 500000. Value range: greater than or equal to 0
u32SuperPFrmBitsThr	The P frame has a large threshold and defaults to 500000. Value range: greater than or equal to 0
u32SuperBFrmBitsThr	The B frame has a large threshold and defaults to 500000. Value range: greater than or equal to 0

- precaution

  no.

- Related data types and interfaces

  MI_VENC_SetSuperFrameCfg

2.75. MI_VENC_RcPriority_e

- Description

  Rate control priority enumeration.

- definition

```
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;
```

- member

member name	description
E_MI_VENC_RC_PRIORITY_BITRATE_FIRST	The target bit rate is prioritized.
E_MI_VENC_RC_PRIORITY_FRAMEBITS_FIRST	The oversized frame threshold is prioritized.

- precaution

no.

- Related data types and interfaces

no.

2.76. MI_VENC_ModParam_t

- Description

Encode related module parameters.

- definition

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

- member

member name	description
enVencModType	Set or get the type of module parameter.
stH265eModParam/ stJpegeModParam	mi_h265.ko/ Mi_jpege.ko module parameter structure.

- precaution

  no.

- Related data types and interfaces

  No

2.77. MI_VENC_ModType_e

- Description

  Encoding related module parameter types.

- definition

```
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;
```

- member

member name	description
E_MI_VENC_MODTYPE_VENC	Mi_venc.ko module parameter type
E_MI_VENC_MODTYPE_H264E	Mi_h264e.ko module parameter type
E_MI_VENC_MODTYPE_H265E	Mi_h265e.ko module parameter type
E_MI_VENC_MODTYPE_JPEGE	Mi_jpege.ko module parameter type

- precaution

  no.

- Related data types and interfaces

  MI_VENC_ModParam_t

2.78. MI_VENC_ParamModH265e_t

- Description

  mi_h265.ko module parameters.

- definition

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

- member

member name	description
u32OneStreamBuffer	The u32OneStreamBuffer parameter in the mi_h265.ko module. 0: Multi-package mode 1: single package mode
u32H265eMiniBufMode	u32H265eMiniBufMode parameter in the mi_h265.ko module. 0: code stream buffer is allocated according to resolution 1: The minimum buffer of the stream buffer is 32k, which is guaranteed by the user.

- precaution

  no.

- Related data types and interfaces

  MI_VENC_ModParam_t

2.79. MI_VENC_ParamModJpege_t

- Description

  Parameters in the mi_jpeg.ko module.

- definition

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

- member

member name	description
u32OneStreamBuffer	The u32OneStreamBuffer parameter in the mi_jpeg.ko module. 0: Multi-package mode 1: single package mode
u32JpegeMiniBufMode	u32JpegeMiniBufMode parameter in the mi_jpeg.ko module. 0: code stream buffer is allocated according to resolution 1: The minimum buffer of the stream buffer is 32k, which is guaranteed by the user.

- precaution

  no.

- Related data types and interfaces

  MI_VENC_ModParam_t

2.80. MI_VENC_AdvCustRcAttr_t

- Description

  Customer defined advanced RC configuration parameters.

- Definition

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

- Member

Member Name	Description
bEnableQPMap	Whether to enable QPMap function. MI_TRUE: ON. MI_FALSE: OFF.
bAbsQP	Whether to use the absolute QP when QP Map is configured. MI_TRUE: YES. MI_FALSE: NO. The relative QP value is used.
bEnableModeMap	Whether to enable ModeMap function. MI_TRUE: ON. MI_FALSE: OFF.
bEnabelHistoStaticInfo	Whether to enable the lastest information acquisition function of the encoded frame. MI_TRUE: ON. MI_FALSE: OFF.

- precaution

  no.

- Related data types and interfaces

  MI_VENC_SetAdvCustRcAttr

2.81. MI_VENC_FrameHistoStaticInfo_t

- Description

  Attributes of the encoded frame.

- definition

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

- member

Member Name	Description
u8PicSkip	The frame skip flag.
u16PicType	Encoded frame type: 0: I 1: P 2: B
u32PicPoc	A POC value of the currently encoded picture.
u32PicSliNum	The number of slice segments.
u32PicNumIntra	The number of intra blocks in the encoded frame (8x8 unit).
u32PicNumMerge	The number of merge blocks in the encoded frame (8x8 unit).
u32PicNumSkip	The number of skip blocks in the encoded frame (8x8 unit).
u32PicAvgCtuQp	An average value of CTU QPs.
u32PicByte	The size of encoded picture in byte.
u32GopPicIdx	A picture index in GOP.
u32PicNum	The encoded picture number.
u32PicDistLow	Low 32-bit SSD between source Y picture and reconstructed Y picture.
u32PicDistHigh	Low 32-bit SSD between source Y picture and reconstructed Y picture.

- precaution

no.

- Related data types and interfaces

MI_VENC_GetLastHistoStaticInfo

2.82. MI_VENC_InputSourceConfig_t

- Description

  Input source config parameters.

- definition

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

- member

member name	description
eInputSrcBufferMode	Input buffer mode

- precaution

  no.

- Related data types and interfaces

  MI_VENC_InputSrcBufferMode_e, MI_VENC_SetInputSourceConfig

2.83. MI_VENC_InputSrcBufferMode_e

- Description

  Define H.264/H.265 input buffer mode enumeration.

- definition

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

- member

member name	description
E_MI_VENC_INPUT_MODE_NORMAL_FRMBASE	Handshake with input by about 3 buffers in frame mode.
E_MI_VENC_INPUT_MODE_RING_ONE_FRM	Handshake with input by one buffer in ring mode.
E_MI_VENC_INPUT_MODE_RING_HALF_FRM	Handshake with input by half buffer in ring mode.

- precaution

  no.

- Related data types and interfaces

  MI_VENC_InputSourceConfig_t, MI_VENC_SetInputSourceConfig

2.84. VENC_MAX_SAD_RANGE_NUM

- Description

  define the maximum number of statistical SAD values that fall within the range of the intelligent detection.

- definition

  ```
  #define  VENC_MAX_SAD_RANGE_NUM        16
  ```

- precaution

  no.

- Related data types and interfaces

  no.

2.85. MI_VENC_SmartDetType_e

- Description

  define the intelligent detection type.

- definition

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

- member

member name	description
E_MI_VENC_MD_DET	type of motion detection
E_MI_VENC_ROI_DET	type of ROI detection

- precaution

  no.

- Related data types and interfaces

  no.

2.86. MI_VENC_MdInfo_t

- Description

  define the statistics of motion detection.

- definition

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

- member

member name	description
u8SadRangeRatio	The percentage of SAD of all MB in each frame that falls into each interval.

- precaution

  The value range of SAD is [0, 255]. All SAD values are divided into 16 ranges:
  [0,10)，[10,15)，[15,20)，[20,25)，[25,30)，[30,40)，[40,50)，[50,60)，[60,70)，[70,80)，[80,90)，[90,100)，[100,120)，[120,140)，[140,160)，[160, 255]。
  Here is the percentage of all MB(8*8) in a frame that falls into the above range.

- Related data types and interfaces

  VENC_MAX_SAD_RANGE_NUM

2.87. MI_VENC_SmartDetInfo_t

- Description

  define the statistics required by the intelligent detection algorithm.

- definition

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

- member

member name	description
eSmartDetType	type of intelligent detection
stMdInfo	statistics of motion detection
bRoiExist	information in the ROI region

member name	description
	MI_TRUE: the current frame has ROI MI_FALSE: no ROI exists for the current frame
u8ProtectFrmNum	number of protected frames

- precaution

no.

3. ERROR CODE

The video coding API error codes are shown in the table below:

Table 1: Video Encoding API Error Codes

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