

# MI VPE API

---

**Version 2.09**

© 2019 SigmaStar Technology Corp. All rights reserved.

SigmaStar Technology makes no representations or warranties including, for example but not limited to, warranties of merchantability, fitness for a particular purpose, non-infringement of any intellectual property right or the accuracy or completeness of this document, and reserves the right to make changes without further notice to any products herein to improve reliability, function or design. No responsibility is assumed by SigmaStar Technology arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights, nor the rights of others.

SigmaStar is a trademark of SigmaStar Technology Corp. Other trademarks or names herein are only for identification purposes only and owned by their respective owners.

## REVISION HISTORY

Revision No.	Description	Date
2.03	<ul style="list-style-type: none"> <li>Initial release</li> </ul>	04/12/2018
2.04	<ul style="list-style-type: none"> <li>Updated MI_VPE_SetPortMode</li> </ul>	01/02/2019
2.05	<ul style="list-style-type: none"> <li>Added MI_VPE_SetPortShowPosition</li> </ul>	03/20/2019
2.06	<ul style="list-style-type: none"> <li>Added MI_VPE_IspInitPara_t</li> </ul>	05/17/2019
2.07	<ul style="list-style-type: none"> <li>Added MI_VPE_AllocIspDataBuf</li> <li>Added MI_VPE_FreeIspDataBuf</li> </ul>	05/31/2019
2.08	<ul style="list-style-type: none"> <li>Added MI_VPE_LDCBegViewConfig</li> <li>Added MI_VPE_LDCEndViewConfig</li> <li>Added MI_VPE_LDCSetViewConfig</li> </ul>	09/16/2019
2.09	<ul style="list-style-type: none"> <li>Added shutter/gain short to MI_VPE_IspInitPara_t</li> </ul>	11/08/2019

## TABLE OF CONTENTS

<b>REVISION HISTORY .....</b>	<b>i</b>
<b>TABLE OF CONTENTS.....</b>	<b>ii</b>
<b>1. 概 述 .....</b>	<b>1</b>
1.1. 模块说明 .....	1
1.2. 流程框图 .....	1
1.2.1 328Q/329D/326D 框图 .....	1
1.2.2 325/325DE/327DE 框图 .....	1
1.2.3 336D/336Q/339G 框图 .....	1
1.2.4 335/337DE 框图 .....	2
1.3. 关键字说明 .....	2
<b>2. API 参考 .....</b>	<b>4</b>
2.1. MI_VPE_CreateChannel .....	6
2.2. MI_VPE_DestroyChannel .....	9
2.3. MI_VPE_GetChannelAttr .....	11
2.4. MI_VPE_SetChannelAttr .....	12
2.5. MI_VPE_StartChannel .....	13
2.6. MI_VPE_StopChannel .....	15
2.7. MI_VPE_EnablePort .....	16
2.8. MI_VPE_DisablePort .....	16
2.9. MI_VPE_SetChannelParam .....	18
2.10. MI_VPE_GetChannelParam .....	19
2.11. MI_VPE_SetChannelCrop .....	20
2.12. MI_VPE_GetChannelCrop .....	21
2.13. MI_VPE_GetChannelRegionLuma .....	23
2.14. MI_VPE_SetChannelRotation .....	24
2.15. MI_VPE_GetChannelRotation .....	26
2.16. MI_VPE_SetPortMode .....	27
2.17. MI_VPE_GetPortMode .....	30
2.18. MI_VPE_SetPortCrop .....	30
2.19. MI_VPE_GetPortCrop .....	33
2.20. MI_VPE_SetPortShowPosition .....	35
2.21. MI_VPE_GetPortShowPosition .....	36
2.22. MI_VPE_Alloc_IspDataBuf .....	36
2.23. MI_VPE_Free_IspDataBuf .....	38
2.24. MI_VPE_LDCEgViewConfig .....	39
2.25. MI_VPE_LDCEndViewConfig .....	40
2.26. MI_VPE_LDCSetViewConfig .....	41
2.27. MI_VPE_SkipFrame .....	42
<b>3. VPE 数据类型 .....</b>	<b>44</b>
3.1. MI_VPE_CHANNEL .....	45
3.2. MI_VPE_PORT .....	45
3.3. MI_VPE_RunningMode_e .....	45

3.4.	MI_VPE_SensorChannel_e.....	46
3.5.	MI_VPE_ChnPortMode_e.....	47
3.6.	MI_VPE_IspApiHeader_t .....	47
3.7.	MI_VPE_ChannelAttr_t.....	48
3.8.	MI_VPE_PqParam_t.....	51
3.9.	MI_VPE_HDRTYPE_e.....	52
3.10.	MI_VPE_3DNR_Level_e.....	52
3.11.	MI_VPE_ChannelPara_t.....	54
3.12.	MI_VPE_RegionInfo_t .....	55
3.13.	MI_VPE_PortMode_t .....	56
3.14.	MI_VPE_IspInitPara_t .....	58
<b>4.</b>	<b>错误码 .....</b>	<b>60</b>

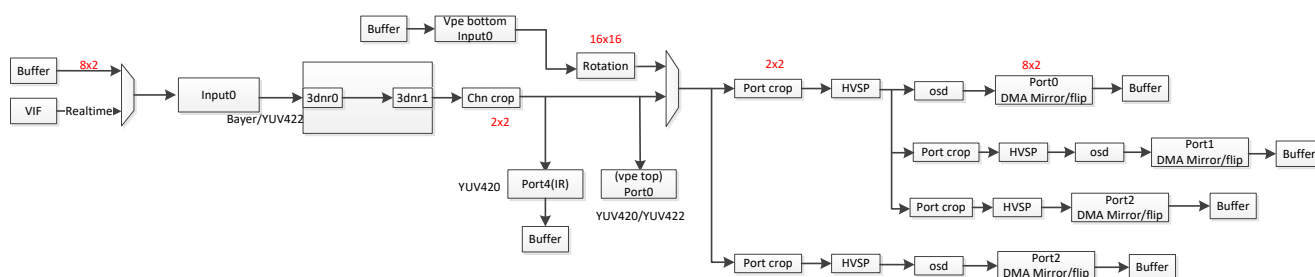
## 1. 概述

### 1.1. 模块说明

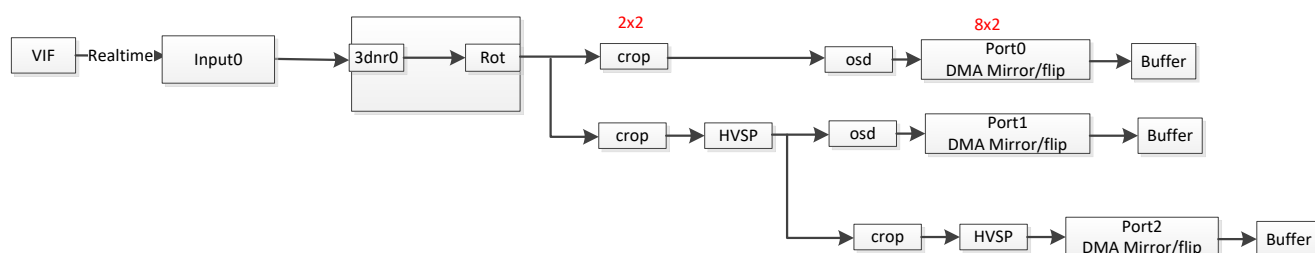
VPE(Video Process Engine)图像处理引擎，支持对一幅输入的图像首先进行图像质量调整，包括降噪，锐化，亮度调整等，然后再分别缩放到一定的分辨率通过各个 output port 口输出。该模块另外包含 HDR，旋转，裁剪等功能。

### 1.2. 流程框图

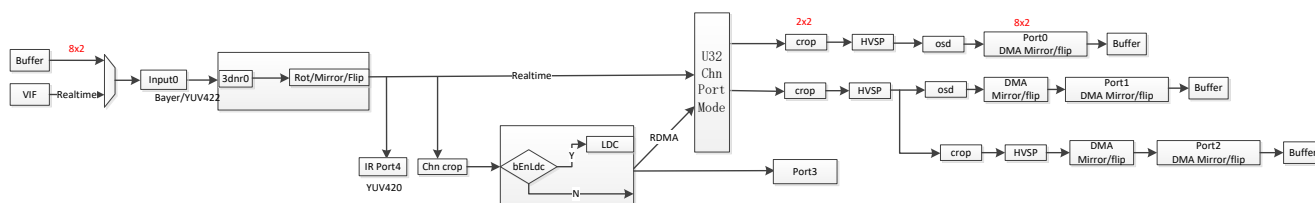
#### 1.2.1 328Q/329D/326D 框图



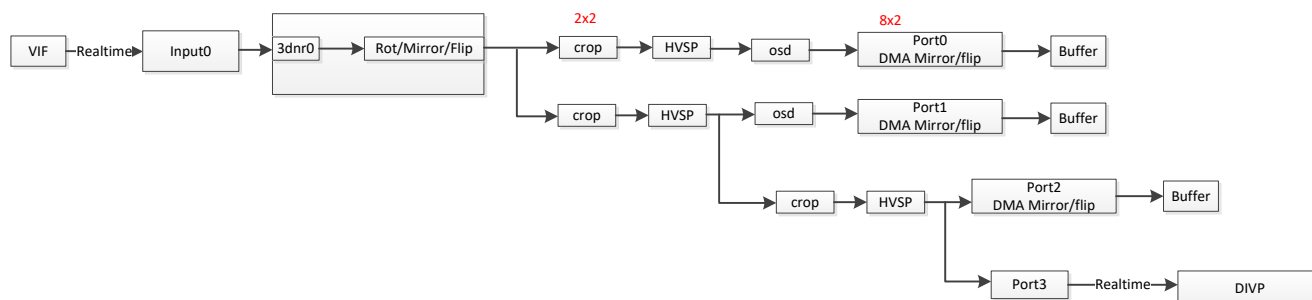
#### 1.2.2 325/325DE/327DE 框图



#### 1.2.3 336D/336Q/339G 框图



### 1.2.4 335/337DE 框图



※ 注意  
框图中 8x2/2x2/16x16 表示该位置宽/高 alignment 限制。

### 1.3. 关键字说明

- Channel  
VPE module 处理通道，各通道分时复用 VPE 硬件。
- ISP  
Image Signal Processing 图像信号处理单元，负责图像降噪/颜色渲染/亮度调整等功能。
- SCL  
Scaler(缩放)缩写。
- Port  
端口， VPE 包含一个 input port， output port 端口分布参考[流程框图](#)。
- 3DNR  
3D Denoising, 3D 降噪。  
2D 降噪：对一个像素将其与周围像素平均，平均后噪声降低，但缺点是会造成画面模糊；  
3D 降噪：增添了时域处理， 2d 降噪只考虑一帧图像，而 3d 降噪进一步考虑帧与帧之间的时域关系，对每个像素进行时域上的平均。
- HDR  
High-Dynamic Range, 高动态范围图像。
- Rotation  
将原始图像绕中心点做旋转 0/90/180/270°
- Crop  
裁剪， 对输入图像进行裁剪。
- HVSP  
H/V 方向 Scaling Process。

- **LDC**  
lens distortion correction 镜头畸变矫正。
- **ZOOM**  
快速放大/缩小， 电子变焦功能。
- **View**  
窗口，内部 LDC 功能中的一个窗口。
- **Realtime**  
实时连接， 指两个模块之间硬件直连。  
优点：模块之间不消耗 **Dram**；  
缺点：不能分时复用，只能创建一个通道。



## 2. API 参考

该功能模块提供以下 API:

API 名	功能
<a href="#">MI_VPE_CreateChannel</a>	创建一个 VPE channel
<a href="#">MI_VPE_DestroyChannel</a>	销毁一个 VPE channel
<a href="#">MI_VPE_GetChannelAttr</a>	获取一个 VPE channel 属性
<a href="#">MI_VPE_SetChannelAttr</a>	设定一个 VPE channel 属性
<a href="#">MI_VPE_StartChannel</a>	启用 VPE channel
<a href="#">MI_VPE_StopChannel</a>	禁用 VPE channel
<a href="#">MI_VPE_EnablePort</a>	启用 VPE 端口
<a href="#">MI_VPE_DisablePort</a>	禁用 VPE 端口
<a href="#">MI_VPE_SetChannelParam</a>	设定 VPE channel 参数
<a href="#">MI_VPE_GetChannelParam</a>	获取 VPE channel 参数
<a href="#">MI_VPE_SetChannelCrop</a>	设定 VPE channel crop window
<a href="#">MI_VPE_GetChannelCrop</a>	获取 VPE channel crop window
<a href="#">MI_VPE_GetChannelRegionLuma</a>	获取 VPE 通道 Luma 直方图统计
<a href="#">MI_VPE_SetChannelRotation</a>	设定 VPE 通道视频旋转类型
<a href="#">MI_VPE_GetChannelRotation</a>	获取 VPE 通道视频旋转类型
<a href="#">MI_VPE_SetPortMode</a>	设定 VPE 端口模式
<a href="#">MI_VPE_GetPortMode</a>	获取 VPE 端口模式
<a href="#">MI_VPE_SetPortCrop</a>	设定 VPE out port crop window
<a href="#">MI_VPE_GetPortCrop</a>	获取 VPE out port crop window 设置参数
<a href="#">MI_VPE_SetPortShowPosition</a>	设置 vpe output port 显示位置
<a href="#">MI_VPE_GetPortShowPosition</a>	获取 vpe output port 显示位置
<a href="#">MI_VPE_Alloc_IspDataBuf</a>	申请 MI_ISP API data Buffer
<a href="#">MI_VPE_Free_IspDataBuf</a>	释放 MI_ISP API data Buffer

API 名	功能
<a href="#">MI_VPE_LDCBegViewConfig</a>	view 开始配置
<a href="#">MI_VPE_LDCEndViewConfig</a>	View 结束配置
<a href="#">MI_VPE_LDCSetViewConfig</a>	View 配置 config bin buffer
<a href="#">MI_VPE_SkipFrame</a>	设置跳过 frame num

2.1. MI\_VPE\_CreateChannel

➤ 描述

创建一个 VPE channel。

➤ 语法

```
MI_S32 MI_VPE_CreateChannel(MI_VPE_CHANNEL VpeCh,  
MI_VPE_ChannelAttr_t *pstVpeChAttr);
```

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入
pstVpeChAttr	VPE channel 属性指针。	输入

➤ 返回值

返回值 { 0 成功。  
非 0 失败，参照[错误码](#)。

➤ 需求

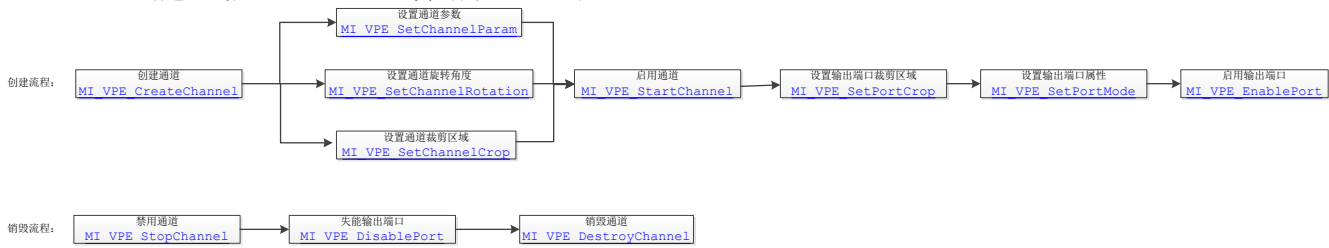
- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

- Realtime mode 下，不支持多 channel。

➤ 举例

创建一路主码流通道，并销毁退出流程：



```
MI_VPE_ChannelAttr_t stChannelVpeAttr;  
MI_SYS_WindowRect_t stChnCropWin;  
MI_VPE_CHANNEL VpeChannel=0;  
MI_VPE_PORT VpePort=0;  
MI_VPE_ChannelPara_t stChannelVpeParam;  
MI_SYS_Rotate_e eRot = E_MI_SYS_ROTATE_NONE;  
MI_SYS_WindowRect_t stPortCropWin;  
MI_S32 s32Ret = MI_SUCCESS;
```

```

MI_SNR_PAD_ID_e eSnrPadId = E_MI_SNR_PAD_ID_0;
MI_SNR_PlaneInfo_t stSnrPlane0Info;
MI_U32 u32CapWidth = 0, u32CapHeight = 0;
MI_SYS_PixelFormat_e ePixFormat;

memset(&stChannelVpeAttr, 0x0, sizeof(MI_VPE_ChannelAttr_t));
memset(&stChannelVpeParam, 0x0, sizeof(MI_VPE_ChannelPara_t));
memset(&stSnrPlane0Info, 0x0, sizeof(MI_SNR_PlaneInfo_t));
memset(&stPortCrop, 0x0, sizeof(MI_SYS_WindowRect_t));
memset(&stChnCropWin, 0x0, sizeof(MI_SYS_WindowRect_t));

MI_SNR_GetPlaneInfo(eSnrPadId, 0, &stSnrPlane0Info);

u32CapWidth = stSnrPlane0Info.stCapRect.u16Width;
u32CapHeight = stSnrPlane0Info.stCapRect.u16Height;
ePixFormat = (MI_SYS_PixelFormat_e)RGB_BAYER_PIXEL(stSnrPlane0Info.ePixPrecision,
stSnrPlane0Info.eBayerId);

stChannelVpeAttr.u32MaxW = u32CapWidth ;
stChannelVpeAttr.u32MaxH = u32CapHeight ;
stChannelVpeAttr.bNREn= FALSE;
stChannelVpeAttr.bEdgeEn= FALSE;
stChannelVpeAttr.bESEN= FALSE;
stChannelVpeAttr.bContrastEn= FALSE;
stChannelVpeAttr.bUVInvert= FALSE;
stChannelVpeAttr.ePixFmt = ePixFormat;
stChannelVpeAttr.eRunningMode = E_MI_VPE_RUN_REALTIME_MODE;
stChannelVpeAttr.eSensorBindId= E_MI_VPE_SENSOR0;

s32Ret = MI\_VPE\_CreateChannel(VpeChannel, &stChannelVpeAttr);
if(s32Ret != MI_SUCCESS)
{
    return s32Ret;
}

s32Ret = MI\_VPE\_GetChannelAttr(VpeChannel, & stChannelVpeAttr);
if(s32Ret != MI_SUCCESS)
{
    return s32Ret;
}

stChannelVpeParam.eHDRType = E_MI_VPE_HDR_TYPE_OFF;
stChannelVpeParam.e3DNRLevel = E_MI_VPE_3DNR_LEVEL2;
stChannelVpeParam.bMirror = FALSE;
stChannelVpeParam.bFlip = FALSE;
s32Ret = MI\_VPE\_SetChannelParam(VpeChannel, &stChannelVpeParam);
if(s32Ret != MI_SUCCESS)
{
    return s32Ret;
}

s32Ret = MI\_VPE\_SetChannelRotation(VpeChannel, eRot);
if(s32Ret != MI_SUCCESS)
{
    return s32Ret;
}

```

```

}

s32Ret = MI_VPE_GetChannelCrop(VpeChannel, &stCropWin) ;
if(s32Ret != MI_SUCCESS)
{
    return s32Ret;
}

stChnCropWin .u16X = 0;
stChnCropWin .u16Y = 0;
stChnCropWin .u16Width = 0;
stChnCropWin .u16Height = 0;
s32Ret = MI_VPE_SetChannelCrop(VpeChannel, & stChnCropWin );
if(s32Ret != MI_SUCCESS)
{
    return s32Ret;
}

s32Ret = MI_VPE_StartChannel(VpeChannel);
if(s32Ret != MI_SUCCESS)
{
    return s32Ret;
}

stPortCrop.u16X = 0;
stPortCrop.u16Y = 0;
stPortCrop.u16Width = 0;
stPortCrop.u16Height = 0;
s32Ret=MI_VPE_SetPortCrop(VpeChannel,VpePort,&stPortCrop);
if(s32Ret != MI_SUCCESS)
{
    return s32Ret;
}

stVpeMode.u16Width = u32CapWidth;
stVpeMode.u16Height = u32CapHeight;
stVpeMode.ePixelFormat = E_MI_SYS_PIXEL_FRAME_YUV_SEMIPLANAR_420;
stVpeMode.eCompressMode = E_MI_SYS_COMPRESS_MODE_NONE;
stVpeMode.bMirror = FALSE;
stVpeMode.bFlip = FALSE;
s32Ret = MI_VPE_SetPortMode(VpeChannel,VpePort, &stVpeMode);
if(s32Ret != MI_SUCCESS)
{
    return s32Ret;
}

s32Ret = MI_VPE_EnablePort(VpeChannel, VpePort);
if(s32Ret != MI_SUCCESS)
{
    return s32Ret;
}

/*****/

```

```
/* call sys bind interface */
/*****

/*****
/* Exit call sys unbind interface */
/*****
s32Ret = MI_VPE_StopChannel (VpeChannel);
if(s32Ret != MI_SUCCESS)
{
    return s32Ret;
}

s32Ret = MI_VPE_DisablePort(VpeChannel, VpePort);
if(s32Ret != MI_SUCCESS)
{
    return s32Ret;
}

s32Ret = MI_VPE_DestroyChannel(VpeChannel);
if(s32Ret != MI_SUCCESS)
{
    return s32Ret;
}
```

- 相关主题  
[MI\\_VPE\\_DestroyChannel](#)

## 2.2. MI\_VPE\_DestroyChannel

- 描述  
销毁一个 VPE channel.
- 语法  
MI\_S32 MI\_VPE\_DestroyChannel ([MI\\_VPE\\_CHANNEL](#) VpeCh);

- 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入

- 返回值  

返回值 { 0    成功。  
          非 0    失败，参照[错误码](#)。
- 需求

- 头文件 : mi\_vpe.h
- 库文件 : libmi\_vpe.a/libmi\_vpe.so

※ 注意

无

- 举例  
    请参见 [MI VPE CreateChannel](#)
- 相关主题  
    [MI VPE CreateChannel](#)

2.3. MI\_VPE\_GetChannelAttr

- 描述  
    获取一个 VPE channel 属性.
- 语法  
    MI\_S32 MI\_VPE\_GetChannelAttr([MI\\_VPE\\_CHANNEL](#) VpeCh, [MI\\_VPE\\_ChannelAttr\\_t](#) \*  
    pstVpeChAttr );

- 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入
pstVpeChAttr	VPE channel 属性指针。	输入

- 返回值  
    返回值 { 0   成功。  
            非 0   失败，参照[错误码](#)。
- 需求
  - 头文件：mi\_vpe.h
  - 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意  
    无

- 举例  
    请参见 [MI VPE CreateChannel](#)
- 相关主题  
    [MI VPE SetChannelAttr](#)



## 2.4. MI\_VPE\_SetChannelAttr

### ➤ 描述

设定一个 VPE channel 属性.

➤ 语法

MI\_S32 MI\_VPE\_SetChannelAttr ([MI\\_VPE\\_CHANNEL](#) VpeCh, [MI\\_VPE\\_ChannelAttr\\_t](#) \*pstVpeChAttr);

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入
pstVpeChAttr	VPE channel 属性指针。	输入

➤ 返回值

返回值 { 0 成功。  
非 0 失败，参照[错误码](#)。

➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

只有在 DVR mode 下 [MI\\_VPE\\_ChannelAttr\\_t](#) 中的 bNrEn,bEdgeEn,bEsEn,bContrastEn,bUvInvert 参数可以被更改。

➤ 举例

请参见 [MI\\_VPE\\_CreateChannel](#)

➤ 相关主题

[MI\\_VPE\\_GetChannelAttr](#)

## 2.5. MI\_VPE\_StartChannel

➤ 描述

启用 VPE channel.

➤ 语法

MI\_S32 MI\_VPE\_StartChannel([MI\\_VPE\\_CHANNEL](#) VpeCh);

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入

➤ 返回值

返回值 { 0 成功。  
非 0 失败，参照[错误码](#)。

➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

无

➤ 举例

请参见 [MI VPE CreateChannel](#)

➤ 相关主题

[MI VPE StopChannel](#)

## 2.6. MI\_VPE\_StopChannel

➤ 描述

禁用 VPE channel.

➤ 语法

```
MI_S32 MI_VPE_StopChannel(MI\_VPE\_CHANNEL VpeCh);
```

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入

➤ 返回值

返回值 { 0 成功。  
非 0 失败，参照[错误码](#)。

➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

当改变 channel 属性时需要先调用此 api，将 channel 禁用， channel 属性设置完后才重新开启。

➤ 举例

请参见 [MI VPE CreateChannel](#)

➤ 相关主题

[MI VPE StartChannel](#)

## 2.7. MI\_VPE\_EnablePort

➤ 描述

启用 VPE 端口。

➤ 语法

```
MI_S32 MI_VPE_EnablePort(MI VPE CHANNEL VpeCh, MI VPE PORT s32VpePort);
```

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入
s32VpePort	Vpe Port 号。	输入

➤ 返回值

返回值 { 0 成功。  
非 0 失败，参照[错误码](#)。

➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

无

➤ 举例

请参见 [MI VPE CreateChannel](#)

➤ 相关主题

[MI VPE DisablePort](#)

## 2.8. MI\_VPE\_DisablePort

➤ 描述

禁用 VPE 端口。

➤ 语法

MI\_S32 MI\_VPE\_DisablePort([MI\\_VPE\\_CHANNEL](#) VpeCh, [MI\\_VPE\\_PORT](#) VpePort);

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入
VpePort	Vpe Port 号。。	输入

➤ 返回值

返回值 { 0 成功。  
非 0 失败，参照[错误码](#)。

➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

当改变 port 属性时，需要先将 port 禁用掉，设置完 port 属性后再重新启用。

➤ 举例

请参见 [MI VPE CreateChannel](#)

➤ 相关主题

[MI VPE EnablePort](#)

## 2.9. MI\_VPE\_SetChannelParam

➤ 描述

设定 VPE channel 参数。

➤ 语法

```
MI_S32 MI_VPE_SetChannelParam (MI\_VPE\_CHANNEL VpeCh,MI_VPE_ChannelPara_t
*pstVpeParam);
```

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入
pstVpeParam	通道参数设置	输入

➤ 返回值

返回值 { 0 成功。  
非 0 失败，参照[错误码](#)。

➤ 需求

- 头文件：mi\_vpe.h

- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

- Channel 已经创建成功。pstVpeParam 详见 [MI\\_VPE\\_ChannelPara\\_t](#) 说明。
- 先通过 [MI\\_VPE\\_GetChannelParam](#) 获取当前参数，再进行设定。

➤ 举例

请参见 [MI\\_VPE\\_CreateChannel](#)

➤ 相关主题

[MI\\_VPE\\_GetChannelParam](#)

## 2.10. MI\_VPE\_GetChannelParam

➤ 描述

获取 VPE channel 参数。

➤ 语法

```
MI_S32 MI_VPE_GetChannelParam (MI\_VPE\_CHANNEL VpeCh, MI\_VPE\_ChannelPara\_t *pstVpeParam);
```

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入
pstVpeParam	通道参数设置	输入

➤ 返回值

返回值  $\left\{ \begin{array}{ll} 0 & \text{成功。} \\ \text{非 0} & \text{失败，参照[错误码](#)。} \end{array} \right.$

➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

- Channel 已经创建成功。pstVpeParam 详见 [MI\\_VPE\\_ChannelPara\\_t](#) 说明。

➤ 举例



无

➤ 相关主题

[MI\\_VPE\\_SetChannelParam](#)

## 2.11. MI\_VPE\_SetChannelCrop

➤ 描述

设定 VPE channel crop window。

➤ 语法

MI\_S32 MI\_VPE\_SetChannelCrop ([MI\\_VPE\\_CHANNEL](#) VpeCh, MI\_SYS\_WindowRect\_t \*pstCropInfo);

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入
pstCropInfo	通道 Crop Window 参数设置	输入

➤ 返回值

返回值 { 0    成功。  
          非 0    失败，参照[错误码](#)。

➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

芯片	是否支持
328Q/329D/326D	支持
325/325DE/327DE	不支持
336D/336Q/339G	LDC/ZOOM 场景下支持
335/337DE	不支持

- Channel 已经创建成功。Crop window 的设定都是基于原始画面大小。

➤ 举例

无

➤ 相关主题

[MI\\_VPE\\_GetChannelCrop](#)

## 2.12. MI\_VPE\_GetChannelCrop

➤ 描述

获取 VPE channel crop window。

➤ 语法

MI\_S32 MI\_VPE\_GetChannelCrop([MI\\_VPE\\_CHANNEL](#) VpeCh, MI\_SYS\_WindowRect\_t

```
*pstCropInfo);
```

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入
pstCropInfo	通道 Crop Window 参数设置	输出

➤ 返回值

返回值 { 0 成功。  
非 0 失败，参照[错误码](#)。

➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

- Channel 已经创建成功。

➤ 举例

无

➤ 相关主题

[MI\\_VPE\\_SetChannelCrop](#)

## 2.13. MI\_VPE\_GetChannelRegionLuma

➤ 描述

获取 VPE 通道 Luma 直方图统计。

➤ 语法

MI\_S32 MI\_VPE\_GetChannelRegionLuma ([MI\\_VPE\\_CHANNEL](#) VpeCh, [MI\\_VPE\\_RegionInfo\\_t](#) \*pstRegionInfo, MI\_U32 \*pu32LumaData, MI\_S32 s32MilliSec);

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。 取值范围：[0, <a href="#">MI_VPE_MAX_CHANNEL_NUM</a> )。	输入
pstRegionInfo	指定统计区域，及数量。	输入
pu32LumaData	输出统计数据	输出
s32MilliSec	API 等待的 timeout, 单位 ms	输入

➤ 返回值

返回值 { 0 成功。  
非 0 失败，参照[错误码](#)。

➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

- 仅 SAV538E/S， SAV638E/S， SAV838E/S 芯片支持该 API
- Channel 已经创建成功。

➤ 举例

无

➤ 相关主题

无

2.14. MI\_VPE\_SetChannelRotation

➤ 描述

设定 VPE 通道视频旋转类型。

➤ 语法

MI\_S32 MI\_VPE\_SetChannelRotation ([MI\\_VPE\\_CHANNEL](#) VpeCh, MI\_SYS\_Rotate\_e eType);

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入
eType	旋转角度设定	输入

➤ 返回值

返回值 { 0 成功。  
非 0 失败，参照[错误码](#)。

➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

芯片	使用方式
328Q/329D/326D	两个 channel RunningMode 分别用 Top/Bottom 配合，该 API 用在 Bottom channel 上。
325/325DE/327DE	需要 Sensor Mirror/flip 配合： Rot 90 + Sensor Mirror Rot 180 + Sensor Mirror/flip Rot 270 + Sensor Flip
336D/336Q/339G	单纯使用
335/337DE	单纯使用

➤ 举例

328Q/329D/326D Rotation 使用方法如下：

```
stVpeChannelInfo.eRunningMode = E_MI_VPE_RUN_REALTIME_TOP_MODE;
stVpeChannelInfo.eBindSensorId = E_MI_VPE_SENSOR0;
stVpeChannelInfo.bRotation = TRUE;
STCHECKRESULT(MI_VPE_CreateChannel(vpechn_top, &stVpeChannelInfo));
STCHECKRESULT(MI_VPE_StartChannel(vpechn_top));

VpePort=0 ; //only can use port0
STCHECKRESULT(MI_VPE_SetPortMode(vpechn_top, VpePort, &stVpeMode));
STCHECKRESULT(MI_VPE_EnablePort(vpechn_top, VpePort));

stVpeChannelInfo.eRunningMode = E_MI_VPE_RUN_REALTIME_BOTTOM_MODE;
stVpeChannelInfo.eBindSensorId = E_MI_VPE_SENSOR_INVALID;
stVpeChannelInfo.bRotation = TRUE;
STCHECKRESULT(MI_VPE_CreateChannel(vpechn_bot, &stVpeChannelInfo));
STCHECKRESULT(MI_VPE_StartChannel(vpechn_bot));

STCHECKRESULT(MI_VPE_SetChannelRotation(vpechn_bot, E_MI_SYS_ROTATE_90));

VpePort=2 ; //0~3
STCHECKRESULT(MI_VPE_SetPortMode(vpechn_bot, VpePort, &stVpeMode));
STCHECKRESULT(MI_VPE_EnablePort(vpechn_bot, VpePort));

/*****/
/* call sys bind interface
   Bind vpechn_top vpechn_bot*/
/*****/
```

➤ 相关主题

[MI\\_VPE\\_GetChannelRotation](#)

2.15. MI\_VPE\_GetChannelRotation

➤ 描述

获取 VPE 通道视频旋转类型.

➤ 语法

```
MI_S32 MI_VPE_GetChannelRotation (MI\_VPE\_CHANNEL VpeCh, MI_SYS_Rotate_e *pType);
```

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入
peType	旋转角度设定	输出

- 返回值

返回值 { 0 成功。  
非 0 失败，参照[错误码](#)。
- 需求
  - 头文件：mi\_vpe.h
  - 库文件：libmi\_vpe.a/libmi\_vpe.so
- ※ 注意
  - Channel 已经创建成功。
- 举例

无
- 相关主题

[MI VPE SetChannelRotation](#)

2.16. MI\_VPE\_SetPortMode

- 描述

设定 VPE 端口模式。
- 语法

```
MI_S32 MI_VPE_SetPortMode(MI\_VPE\_CHANNEL VpeCh, MI\_VPE\_PORT VpePort,
MI\_VPE\_PortMode\_t *pstVpeMode);
```
- 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入
VpePort	VPE port 号。	输入
pstVpeMode	VPE 端口模式	输出
- 返回值

返回值 { 0 成功。  
非 0 失败，参照[错误码](#)。
- 需求
  - 头文件：mi\_vpe.h



- 库文件 : libmi\_vpe.a/libmi\_vpe.so

※ 注意

● Max width/height

芯片	Port0	Port1	Port2	Port3	Port4(IR port)
328Q/329D/326D	MaxWidth = 3840	MaxWidth = 2688	MaxWidth = 2688	MaxWidth = 3840	No Scaling: Width= 1/2 SrcWidth Height= 1/2 SrcHeight
325/325DE/327DE	No Scaling: Width= SrcWidth Height=SrcHeight	MaxWidth = 2688	MaxWidth = 2688	No support	No support
336D/336Q/339G	MaxWidth = 3840	MaxWidth = 3840	MaxWidth = 3840	No Scaling: Width= SrcWidth Height=SrcHeight	No Scaling: Width= 1/2 SrcWidth Height= 1/2 SrcHeight
335/337DE	MaxWidth = 2688	MaxWidth = 2688	MaxWidth = 1920	MaxWidth = 1920	No support

● Pixel Format

芯片	Port0	Port1	Port2	Port3	Port4 (IR port)
328Q/329D/326D	YUV420/YUV422/ARGB8888/BGRA8888	同 Port0	同 Port0	同 Port0	YUV420 NV12
325/325DE/327DE	YUV420/YUV422	同 Port0	同 Port0	同 Port0	No support
336D/336Q/339G	YUV420/YUV422/ARGB8888/BGRA8888/ABGR8888	同 Port0	同 Port0	No Ldc/Zoom: YUV420/YUV422 Ldc/Zoom:	YUV420 NV12
335/337DE	YUV420/YUV422	同 Port0	同 Port0	同 Port0	No support

- 335/337DE 中 Port3 是 virtual port, 通过 Realtime Bind Divp 输出。

➤ 举例

无

➤ 相关主题

[MI VPE GetPortMode](#)

## 2.17. MI\_VPE\_GetPortMode

### ➤ 描述

获取 VPE 端口模式。

### ➤ 语法

```
MI_S32 MI_VPE_GetPortMode (MI_VPE_CHANNEL VpeCh, MI_VPE_PORT VpePort,
MI_VPE_PortMode_t *pstVpeMode);
```

### ➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入
VpePort	VPE port 号。	输入
pstVpeMode	VPE 端口模式	输出

### ➤ 返回值

返回值 { 0 成功。  
非 0 失败，参照[错误码](#)。

### ➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

### ※ 注意

无

### ➤ 举例

无

### ➤ 相关主题

[MI\\_VPE\\_SetPortMode](#)

## 2.18. MI\_VPE\_SetPortCrop

### ➤ 描述

设定 VPE output port crop window.

### ➤ 语法

```
MI_S32 MI_VPE_SetPortCrop (MI_VPE_CHANNEL VpeCh, MI_VPE_PORT VpePort,
MI_SYS_WindowRect_t *pstOutCropInfo);
```



➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入
VpePort	VPE port 号。	输入
*pstOutCropInfo	Output port crop window 设置	输入

➤ 返回值

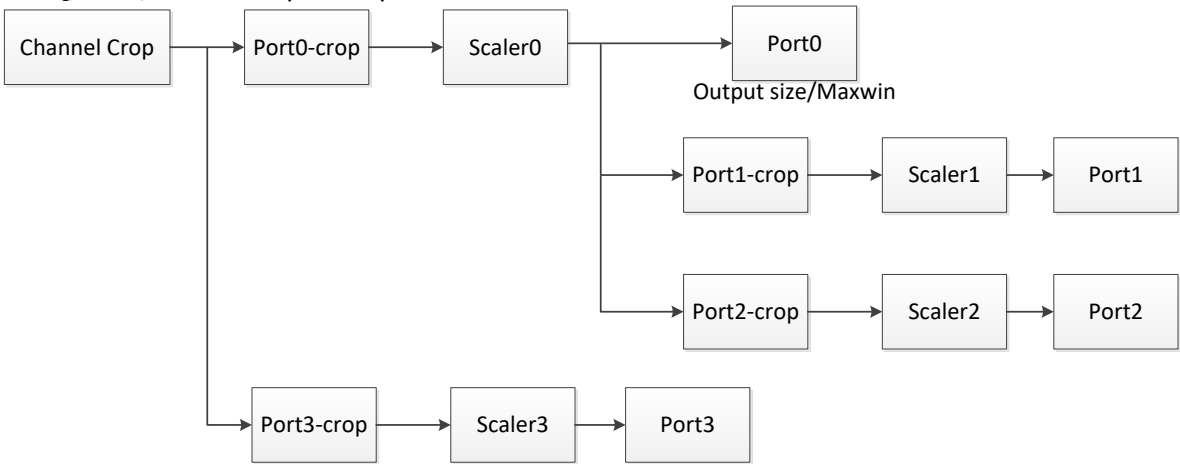
返回值 { 0    成功。  
          非 0    失败，参照[错误码](#)。

➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

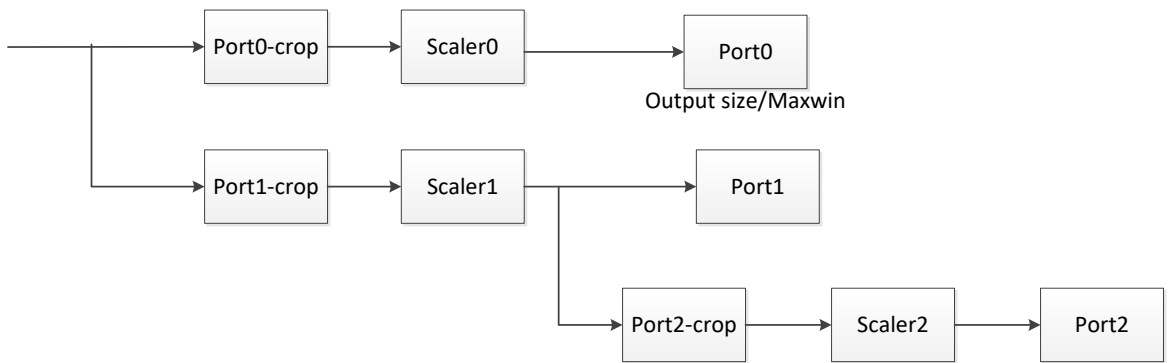
※ 注意

- 328Q/329D/326D 各个 port crop 如下：



由于 port1,2 的 Source 是 port0 的 output:

1. port0 enable, port1,2 的 crop win < port0 size,
  2. port0 disable, port1,2 的 crop win < Vpe input
- 336D/336Q/339G 335/337DE port crop 如下：



325/325DE/327DE 系列芯片没有上图 Scaler0,其它一致。

由于 port2 的 Source 是 port1 的 output:

- 1. port1 enable, port2 的 crop win < port1 size;
- 2. port1 disable, port2 的 crop win < Vpe input.

➤ 举例  
无

➤ 相关主题  
[MI\\_VPE\\_GetPortCrop](#)

2.19. MI\_VPE\_GetPortCrop

➤ 描述  
获取 VPE out port crop window 设置参数。

➤ 语法  
MI\_S32 MI\_VPE\_GetPortCrop ([MI\\_VPE\\_CHANNEL](#) VpeCh, [MI\\_VPE\\_PORT](#) VpePort, MI\_SYS\_WindowRect\_t \*pstOutCropInfo);

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入
VpePort	VPE port 号。	输入
*pstOutCropInfo	Output port crop window 参数	输出

➤ 返回值  
返回值 { 0 成功。  
非 0 失败，参照[错误码](#)。

➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

无

➤ 举例

无

➤ 相关主题

[MI\\_VPE\\_SetPortCrop](#)

2.20. MI\_VPE\_SetPortShowPosition

➤ 描述

设置 vpe output port 显示位置

➤ 语法

```
MI_S32 MI_VPE_SetPortShowPosition(MI\_VPE\_CHANNEL VpeCh, MI\_VPE\_PORT VpePort,
MI_SYS_WindowRect_t *pstPortPositionInfo);
```

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入
VpePort	VPE port 号。	输入
* pstPortPositionInfo	显示位置参数	输入

➤ 返回值

返回值 { 0      成功。  
          非 0    失败，参照[错误码](#)。

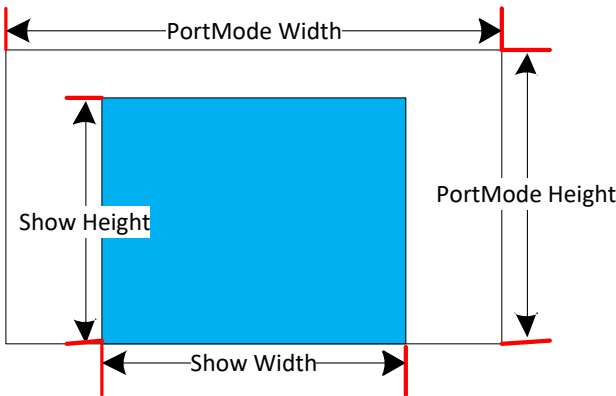
➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

1. 在没有调用此 API 的情况下，默认是 Show width/height = PortMode width/height 显示。
2. 需要在 [MI\\_VPE\\_SetPortMode](#) 后设置，没有 show 画面的区域将填黑。

PortMode width/height 和 Show width/height 关系如下图



➤ 举例

无



➤ 相关主题

[MI\\_VPE\\_GetPortShowPosition](#)

2.21. MI\_VPE\_GetPortShowPosition

➤ 描述

获取 vpe output port 显示位置

➤ 语法

```
MI_S32 MI_VPE_GetPortShowPosition(MI\_VPE\_CHANNEL VpeCh, MI\_VPE\_PORT VpePort,
MI_SYS_WindowRect_t *pstPortPositionInfo);
```

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入
VpePort	VPE port 号。	输入
* pstPortPositionInfo	显示位置参数	输出

➤ 返回值

返回值 { 1 成功。  
非 0 失败，参照[错误码](#)。

➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

无

➤ 举例

无

➤ 相关主题

[MI\\_VPE\\_SetPortShowPosition](#)

2.22. MI\_VPE\_Alloc\_IspDataBuf

➤ 描述

申请 MI\_ISP API Data Buffer

➤ 语法

```
MI_S32 MI_VPE_Alloc_IspDataBuf(MI_U32 u32Size,void **pUserVirAddr);
```

➤ 参数

参数名称	描述	输入/输出
u32Size	Alloc Buffer Size	输入
**pUserVirAddr	User Buffer 指针地址	输出

➤ 返回值

返回值 { 0      成功。  
          非 0    失败，参照[错误码](#)。

➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

一个线程申请一块 isp data buffer，避免线程间 buffer 共用，导致线程之间 buffer 互踩

➤ 举例

```
#define MI_ISP_MAX_DATA_SIZE (80*1024)
MI_VPE_Alloc_IspDataBuf(MI_ISP_MAX_DATA_SIZE, &pIspBuffer);
MI_ISP_IQ_COLORTOGRAY_TYPE_t *pstColorToGray = (MI_ISP_IQ_COLORTOGRAY_TYPE_t *)pIspBuffer;
MI_ISP_IQ_GetColorToGray( Channel, pstColorToGray);
if(pstColorToGray->bEnable == SS_TRUE)
    pstColorToGray->bEnable = SS_FALSE;
else
    pstColorToGray->bEnable = SS_TRUE;
MI_ISP_IQ_SetColorToGray( Channel, pstColorToGray);

MI_ISP_IQ_CONTRAST_TYPE_t *pstContrast = (MI_ISP_IQ_CONTRAST_TYPE_t *)pIspBuffer;
MI_ISP_IQ_GetContrast( Channel, pstContrast);
MI_ISP_IQ_SetContrast( Channel, pstContrast);

MI_VPE_Free_IspDataBuf(pIspBuffer)
```

➤ 相关主题

[MI\\_VPE\\_Free\\_IspDataBuf](#)

2.23. MI\_VPE\_Free\_IspDataBuf

➤ 描述

释放 MI\_ISP API Data 申请的 Buffer

➤ 语法

MI\_S32 MI\_VPE\_Free\_IspDataBuf(void \*pUserBuf);

➤ 参数

参数名称	描述	输入/输出
*pUserBuf	申请的 Isp Api Data Buffer 指针	输入

➤ 返回值

返回值 { 0 成功。  
非 0 失败，参照[错误码](#)。

➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

和 MI\_VPE\_Alloc\_IspDataBuf 成对出现

➤ 举例

见 [MI\\_VPE\\_Alloc\\_IspDataBuf](#) 举例

➤ 相关主题

[MI\\_VPE\\_Alloc\\_IspDataBuf](#)

## 2.24. MI\_VPE\_LDCBegViewConfig

➤ 描述

开始配置所有 Ldc 窗口的 bin 档

➤ 语法

```
MI_S32 MI_VPE_LDCBegViewConfig(MI_VPE_CHANNEL VpeCh);
```

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入

➤ 返回值

返回值 { 1 成功。  
非 0 失败，参照[错误码](#)。

➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

- 1. 仅 336D/336Q/339G 芯片支持 LDC 功能, [MI\\_VPE\\_ChannelAttr\\_t](#) 中 bEnLdc= TRUE,启用 LDC
- 2. 和 MI\_VPE\_LDCEndViewConfig 成对出现。
- 3. 在窗口数量发生变化时才需要使用此 API, 如果只是单独更换其中某一个窗口属性不需要调用此 api。

➤ 举例

```
MI_VPE_LDCBegViewConfig(vpechn);
for(i=0; i<viewnum;i++)
{
    MI_VPE_LDCSetViewConfig(vpechn, ldcBinBuffer[i], u32LdcBinSize[i]);
    free(ldcBinBuffer[i]);
}
MI_VPE_LDCEndViewConfig(vpechn);
```

➤ 相关主题

[MI\\_VPE\\_LDCEndViewConfig](#)

2.25. MI\_VPE\_LDCEndViewConfig

➤ 描述

结束配置所有 Ldc 窗口的 bin 档

➤ 语法

MI\_S32 MI\_VPE\_LDCEndViewConfig(MI\_VPE\_CHANNEL VpeCh)

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入

➤ 返回值

返回值 { 1, 成功。  
非 0 失败，参照[错误码](#)。

➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

- 1. 仅 336D/336Q/339G 芯片支持 LDC 功能, MI\_VPE\_ChannelAttr\_t 中 bEnLdc= TRUE,启用 LDC
- 2. 和 [MI\\_VPE\\_LDCBegViewConfig](#) 成对出现。

3. 在窗口数量发生变化时才需要使用此 API， 如果只是单独更换其中某一个窗口不需要调用此 api。

➤ 举例

见 [MI\\_VPE\\_LDCBegViewConfig](#) 举例。

➤ 相关主题

[MI\\_VPE\\_LDCBegViewConfig](#)

## 2.26. MI\_VPE\_LDCSetViewConfig

➤ 描述

配置 Ldc view 窗口 Config bin 档。

➤ 语法

```
MI_S32 MI_VPE_LDCSetViewConfig(MI_VPE_CHANNEL VpeCh, void *pConfigAddr, MI_U32 u32ConfigSize);
```

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入
pConfigAddr	Bin buffer 指针地址	输入
u32ConfigSize	Bin buffer size	输入

➤ 返回值

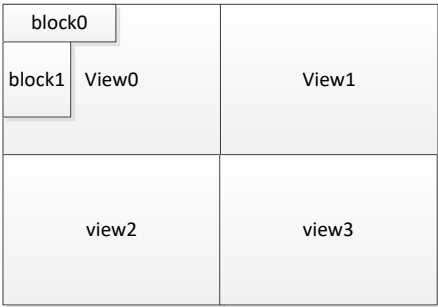
返回值 { 1, 成功。  
非 0 失败，参照[错误码](#)。

➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

1. 仅 336D/336Q/339G 芯片支持 LDC 功能, [MI\\_VPE\\_ChannelAttr\\_t](#) 中 bEnLdc= TRUE,启用 LDC
2. 如果没有发生窗口数量变化， 只是改变某几个窗口 bin buffer 时， 该 api 可以单独使用； 在窗口数量发生变化时， 需要配合 [MI\\_VPE\\_LDCBegViewConfig](#) 和 [MI\\_VPE\\_LDCEndViewConfig](#) 使用。



每一个 config bin buffer 对应一个 view 窗口的设置。

➤ 举例

见 [MI\\_VPE\\_LDCBegViewConfig](#) 举例。

➤ 相关主题

[MI\\_VPE\\_LDCBegViewConfig](#)

[MI\\_VPE\\_LDCEndViewConfig](#)

2.27. [MI\\_VPE\\_SkipFrame](#)

➤ 描述

设置跳过 frame num。

➤ 语法

MI\_S32 MI\_VPE\_SkipFrame(MI\_VPE\_CHANNEL VpeCh, MI\_U32 u32FrameNum);

➤ 参数

参数名称	描述	输入/输出
VpeCh	VPE channel 号。	输入
u32FrameNum	Frame 数量	输入

➤ 返回值

返回值 { 1, 成功。  
非 0 失败，参照[错误码](#)。

➤ 需求

- 头文件：mi\_vpe.h
- 库文件：libmi\_vpe.a/libmi\_vpe.so

※ 注意

无。

➤ 举例

无。

➤ 相关主题

无。



### 3. VPE 数据类型

---

VPE 模块相关数据类型定义如下：

<a href="#"><u>MI_VPE_CHANNEL</u></a>	定义 VPE channel 的类型
<a href="#"><u>MI_VPE_PORT</u></a>	定义 VPE port 的类型
<a href="#"><u>MI_VPE_RunningMode_e</u></a>	定义 VPE 运行模式
<a href="#"><u>MI_VPE_SensorChannel_e</u></a>	定义 VPE 绑定 sensor 的 ID
<a href="#"><u>MI_VPE_IspApiHeader_t</u></a>	定义 VPE 传给 ISP 数据头的信息
<a href="#"><u>MI_VPE_ChannelAttr_t</u></a>	定义 VPE channel 静态属性参数
<a href="#"><u>MI_VPE_PqParam_t</u></a>	定义 VPE PQ 种 NR, EdgeGain, Contrast 的设置
<a href="#"><u>MI_VPE_HDRType_e</u></a>	定义 VPE HDR 开关模式
<a href="#"><u>MI_VPE_3DNR_Level_e</u></a>	定义 3DNR 设置等级
<a href="#"><u>MI_VPE_ChannelPara_t</u></a>	定义 VPE channel 动态属性参数
<a href="#"><u>MI_VPE_RegionInfo_t</u></a>	定义 VPE 通道区域统计信息
<a href="#"><u>MI_VPE_PortMode_t</u></a>	定义 VPE 端口模式
<a href="#"><u>MI_VPE_ChnPortMode_e</u></a>	定义 Port 的输出模式

### 3.1. MI\_VPE\_CHANNEL

- 说明  
定义 MI\_VPE\_CHANNEL 类型。
- 定义  
typedef MI\_S32 MI\_VPE\_CHANNEL
- ※ 注意事项  
无。
- 相关数据类型及接口  
无。

### 3.2. MI\_VPE\_PORT

- 说明  
定义 MI\_VPE\_PORT 类型。
- 定义  
typedef MI\_S32 MI\_VPE\_PORT
- ※ 注意事项  
无。
- 相关数据类型及接口  
无。

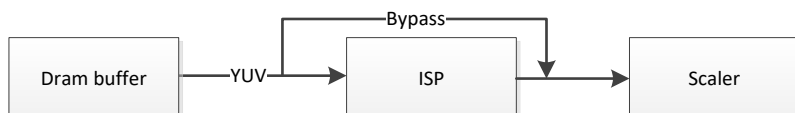
### 3.3. MI\_VPE\_RunningMode\_e

- 说明  
定义 VPE 工作在某种模式下。
- 定义  
typedef enum  
{  
    E\_MI\_VPE\_RUN\_INVALID                    = 0x00,  
    E\_MI\_VPE\_RUN\_DVR\_MODE                  = 0x01,  
    E\_MI\_VPE\_RUN\_CAM\_TOP\_MODE              = 0x02,  
    E\_MI\_VPE\_RUN\_CAM\_BOTTOM\_MODE          = 0x04,  
    E\_MI\_VPE\_RUN\_CAM\_MODE                  =  
        E\_MI\_VPE\_RUN\_CAM\_TOP\_MODE|E\_MI\_VPE\_RUN\_CAM\_BOTTOM\_MODE,  
    E\_MI\_VPE\_RUN\_REALTIME\_TOP\_MODE        = 0x08,

```
E_MI_VPE_RUN_REALTIME_BOTTOM_MODE = 0x10,
E_MI_VPE_RUN_REALTIME_MODE         =
    E_MI_VPE_RUN_REALTIME_TOP_MODE | E_MI_VPE_RUN_REALTIME_BOTTOM_MODE,
E_MI_VPE_RUNNING_MODE_MAX,
} MI_VPE_RunningMode_e;
```

※ 注意事项

E\_MI\_VPE\_RUN\_DVR\_MODE: 当输入是 YUV 格式时, ISP bypass, 不经过 ISP 处理。



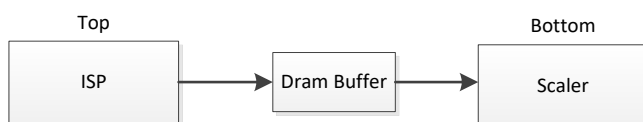
E\_MI\_VPE\_RUN\_CAM\_MODE: 当输入是 Bayer 格式时, 且数据来自 DRAM, ISP 支持分时复用, 例如多 sensor 场景。



E\_MI\_VPE\_RUN\_REALTIME\_MODE: 当输入是 Bayer 格式, VIF 和 ISP 之间硬件直连, ISP 不支持分时复用, 只支持一个通道, 例如单 sensor 场景。



仅在 328Q/329D/326D 芯片上支持 Top/Bottom mode, 用在 rotation 场景, Top/Bottom 是指 ISP 与 Scaler 之间的连接关系。



Top 通道没有 scaling 能力, Bottom 通道实现 Rotation, 使用方法参考 [MI VPE SetChannelRotation](#) 举例。

➤ 相关数据类型及接口

无。

### 3.4. MI\_VPE\_SensorChannel\_e

➤ 说明

定义 VPE 与哪个 sensor 有绑定关系。

➤ 定义

```
typedef enum
{
    E_MI_VPE_SENSOR_INVALID = 0,
    E_MI_VPE_SENSOR0,
    E_MI_VPE_SENSOR1,
```

```
E_MI_VPE_SENSOR2,  
E_MI_VPE_SENSOR3,  
E_MI_VPE_SENSOR_MAX  
}MI_VPE_SensorChannel_e;  
与硬件上 sensor Pad0/1/2/3 对应。
```

※ 注意事项

当 vpe 前端不接 sensor 的时候使用 E\_MI\_VPE\_SENSOR\_INVALID。

➤ 相关数据类型及接口

[MI\\_VPE\\_ChannelAttr\\_t](#)。

### 3.5. MI\_VPE\_ChnPortMode\_e

➤ 说明

定义 VPE channel 中每一个 port 的输出效果模式。

➤ 定义

```
typedef enum  
{  
    E_MI_VPE_ZOOM_LDC_NULL,  
    E_MI_VPE_ZOOM_LDC_PORT0 = 0X01,  
    E_MI_VPE_ZOOM_LDC_PORT1 = 0X02,  
    E_MI_VPE_ZOOM_LDC_PORT2 = 0X04,  
    E_MI_VPE_ZOOM_LDC_MAX = E_MI_VPE_ZOOM_LDC_PORT0|  
        E_MI_VPE_ZOOM_LDC_PORT1|E_MI_VPE_ZOOM_LDC_PORT2,  
}  
}MI_VPE_ChnPortMode_e;  
port0,1,2 输出包含 zoom/ldc 效果。
```

※ 注意事项

参考 [MI\\_VPE\\_ChannelAttr\\_t](#) 中注意说明

➤ 相关数据类型及接口

[MI\\_VPE\\_ChannelAttr\\_t](#)。

### 3.6. MI\_VPE\_IspApiHeader\_t

➤ 说明

定义 VPE 传给 ISP 的数据信息。

➤ 定义

```
typedef struct MI_VPE_IspApiHeader_s  
{  
    MI_U32 u32HeadSize;    //Size of MI_IspApiHeader_t
```

```
MI_U32 u32DataLen;    //Data length;
MI_U32 u32CtrlID;     //Function ID
MI_U32 u32Channel;    //Isp channel number
MI_S32 s32Ret;        //Isp api retuen value
} MI_VPE_IspApiHeader_t;
```

※ 注意事项  
无。

➤ 相关数据类型及接口  
[MI\\_VPE\\_IspApiData\\_t](#)。

### 3.7. MI\_VPE\_ChannelAttr\_t

➤ 说明  
定义 VPE channel 静态属性。

➤ 定义

```
typedef struct MI_VPE_ChannelAttr_s
{
    MI_U16 u16MaxW;
    MI_U16 u16MaxH;
    MI_SYS_PixelFormat_e ePixFmt;
    MI\_VPE\_SensorChannel\_e eSensorBindId;

    MI_BOOL bNrEn;
    MI_BOOL bEdgeEn;
    MI_BOOL bEsEn;
    MI_BOOL bContrastEn;
    MI_BOOL bUvInvert;
    MI_BOOL bRotation;
    MI\_VPE\_RunningMode\_e eRunningMode;
    MI\_VPE\_IspInitPara\_t tIspInitPara;
    MI_BOOL bEnLdc;
    MI_U32 u32ChnPortMode;
}MI_VPE_ChannelAttr_t;
```

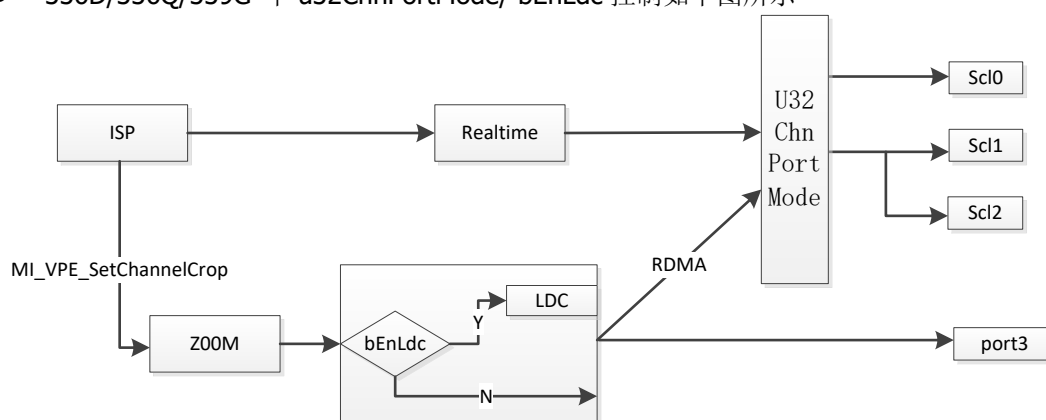
➤ 成员

成员名称	描述
u32MaxW	最大图像宽度。
u32MaxH	最大图像高度。
ePixFmt	输入像素格式。
eSensorBindId	前端接 sensor 时设置绑定 sensorid。
bNrEn	去噪使能。
bEdgeEn	锐化使能。
bEsEn	边缘平滑使能。

成员名称	描述
bContrastEn	对比度使能。
bUvInvert	UV 反转使能。
eRunningMode	VPE 运行模式。
bRotation	旋转功能使能。
tIspInitPara	初始 isp 参数。
bEnLdc	使能 ldc。
u32ChnPortMode	当前 channel 下 port 输出模式, 由 MI_VPE_ChnPortMode_e 成员赋值。

※ 注意事项

- 静态属性创建 channel 时设定，不可更改。
- ePixFmt:当 eRunningMode 为 E\_MI\_VPE\_RUN\_DVR\_MODE 时只支持 E\_MI\_SYS\_PIXEL\_FRAME\_YUV422\_YUYV,其它的 Running mode pixel 有 sensor 转换得到，如：  
ePixFormat = (MI\_SYS\_PixelFormat\_e)RGB\_BAYER\_PIXEL(stSnrPlane0Info.ePixPrecision, stSnrPlane0Info.eBayerId);
- eSensorBindId: vpe channel 前端 sensor 插的 sensor Pad 位置。当 eRunningMode 为 E\_MI\_VPE\_RUN\_DVR\_MODE 时， eSensorBindId = E\_MI\_VPE\_SENSOR\_INVALID。
- bNrEn/bEdgeEn/bEsEn/bContrastEn/bUvInvert 只有在 MSR930 芯片上设置有效。
- bRotation 只在 328Q/329D/326D 系列芯片 rotation 场景设置，参考 [MI\\_VPE\\_SetChannelRotation](#) 举例
- 336D/336Q/339G 中 u32ChnPortMode/ bEnLdc 控制如下图所示



1). 普通场景, u32ChnPortMode=0, ISP 与所有 SCL 之间 realtime 连接, 硬件直连, 不消耗 buffer.

2). scl1, 2 是同一个 source, scl0 和 scl1/2 可以通过 u32ChnPortMode 独立选择 source 是来自 Realtime 还是 RDMA, 一共是有 4 中组合, 例如:

u32ChnPortMode = E\_MI\_VPE\_ZOOM\_LDC\_PORT0; //port0 from RDMA

u32ChnPortMode = E\_MI\_VPE\_ZOOM\_LDC\_PORT1| E\_MI\_VPE\_ZOOM\_LDC\_PORT2;//port1,2 from RDMA

u32ChnPortMode = E\_MI\_VPE\_ZOOM\_LDC\_NULL; // all port from realtime

u32ChnPortMode = E\_MI\_VPE\_ZOOM\_LDC\_PORT0| E\_MI\_VPE\_ZOOM\_LDC\_PORT1| E\_MI\_VPE\_ZOOM\_LDC\_PORT2;//all scl from RDMA

3). bEnLdc==TRUE, RDMA buffer 来自 LDC out, port3 from LDC,  
bEnLdc==FALSE, RDMA buffer 来自 ZOOM out, port3 from ZOOM.

➤ 相关数据类型及接口

[MI\\_VPE\\_RunningMode\\_e](#)  
[MI\\_VPE\\_SensorChannel\\_e](#)  
[MI\\_VPE\\_IspInitPara\\_t](#)  
[MI\\_VPE\\_ChnPortMode\\_e](#)

### 3.8. MI\_VPE\_PqParam\_t

➤ 说明

定义 VPE channel 属性参数。

➤ 定义

```
typedef struct MI_VPE_ChannelPara_s
{
    MI_U8 u8NrcSfStr; //0 ~ 255;
    MI_U8 u8NrcTfStr; //0 ~ 255
    MI_U8 u8NrySfStr; //0 ~ 255
    MI_U8 u8NryTfStr; //0 ~ 255
    MI_U8 u8NryBlendMotionTh; //0 ~ 15
    MI_U8 u8NryBlendStillTh; //0 ~ 15
    MI_U8 u8NryBlendMotionWei; //0 ~ 31
    MI_U8 u8NryBlendOtherWei; //0 ~ 31
    MI_U8 u8NryBlendStillWei; //0 ~ 31
    MI_U8 u8EdgeGain[6]; //0~255
    MI_U8 u8Contrast; //0~255
} MI_VPE_ChannelPara_t;
```

➤ 成员

成员名称	描述
u8NrcSfStr	0~255, 空间域降彩噪强度
u8NrcTfStr	0~255, 时间域降彩噪强度
u8NrySfStr	0~255, 空间域降明度噪点强度
u8NryTfStr	0~255, 时间域降明度噪点强度
u8NryBlendMotionTh	0~15, 空间域明度降噪画面变动区侦测阈值
u8NryBlendStillTh	0~15, 空间域明度降噪画面静止区侦测阈值
u8NryBlendMotionWei	0~31, 画面变动区空间域降噪相对时域降噪的比重
u8NryBlendOtherWei	0~31, 静止与变动区间空间域降噪相对时域降噪的比重
u8NryBlendStillWei	0~31, 画面静止区空间域降噪相对时域降噪的比重
u8EdgeGain[6]	根據不同邊緣程度，做不同的銳化效果，index 0 代表代表发丝、草地等细小纹理，越往大邊緣程度越大。
u8Contrast	影响暗区和亮区调整幅度，值越大，暗区调整越明显，而亮度不會過曝

※ 注意事项

BLEND\_WEI 数值愈大空间域去噪点强度愈强，建议针对画面动态区设较强的值可达到较好的去噪点效果；针对静止区设定较弱的值，可保留较多细节。



- 相关数据类型及接口

[MI\\_VPE\\_ChannelPara\\_t](#)。

### 3.9. MI\_VPE\_HDRType\_e

- 说明

定义 VPE 是否开 HDR，开 HDR 是使用哪种方式。

- 定义

```
typedef enum
{
    E_MI_VPE_HDR_TYPE_OFF,
    E_MI_VPE_HDR_TYPE_VC,          //virtual channel mode HDR,vc0->long, vc1->short
    E_MI_VPE_HDR_TYPE_DOL,
    E_MI_VPE_HDR_TYPE_EMBEDDED, //compressed HDR mode
    E_MI_VPE_HDR_TYPE_LI,         //Line interlace HDR
    E_MI_VPE_HDR_TYPE_MAX
} MI_VPE_HDRType_e
```

- ※ 注意事项

具体使用哪一种 HDR Type 可以通过 MI\_SNR\_GetPadInfo 接口获取。

- 相关数据类型及接口

[MI\\_VPE\\_ChannelPara\\_t](#)。

### 3.10. MI\_VPE\_3DNR\_Level\_e

- 说明

定义 VPE 开启 3DNR 等级。

- 定义

```
typedef enu
{
    E_MI_VPE_3DNR_LEVEL_OFF,
    E_MI_VPE_3DNR_LEVEL1,
    E_MI_VPE_3DNR_LEVEL2,
    E_MI_VPE_3DNR_LEVEL3,
    E_MI_VPE_3DNR_LEVEL4,
    E_MI_VPE_3DNR_LEVEL5,
    E_MI_VPE_3DNR_LEVEL6,
    E_MI_VPE_3DNR_LEVEL7,
    E_MI_VPE_3DNR_TYPE_NUM
} MI_VPE_3DNR_Level_e;
```

- ※ 注意事项

Create channel 后设置，为静态属性只能设置一次。

有关 3DNR 等级，会导致部分 3DNR API 参数(NR3D\_PARAM\_t)无法使用，请参考下表。

【名称】

变量名称	LEVEL_OFF	Level 1 ~ 3	Level 4 ~ 7
u16MdThd	X	O	O
u16MdDiv	X	O	O
u8TfStr	X	O	O
u8TfStrEx	X	O	O
u16MdThdPre	X	X	O
u16MdGainPre	X	X	O
u8TfStrPre	X	X	O
u8TfStrExPre	X	X	O
u8MdThdByY[16]	X	O	O
u8MdDivByY[16]	X	O	O
u8M2SLut[16]	X	O	O
u8TfLut[16]	X	O	O
u8YSfStr	O	O	O
u8YSfBlendLut[16]	X	O	O
u8CSfStr	O	O	O
u8CSfExStr	O	O	O
u8CSfExBlendGain	X	O	O
u16CSfExBlendClip	X	O	O
u16ShpBlendLut[16]	X	O	O

➤ 相关数据类型及接口

[MI\\_VPE\\_ChannelPara\\_t](#)。

MI\_ISP\_IQ\_SetNR3D, MI\_ISP\_IQ\_GetNR3D。

MI\_ISP\_IQ\_NR3D\_TYPE\_t。

NR3D\_PARAM\_t。

### 3.11. MI\_VPE\_ChannelPara\_t

➤ 说明

定义 VPE 开启 3DNR 等级。

➤ 定义

```
typedef struct MI_VPE_ChannelPara_s
{
    MI_VPE_PqParam_t      stPqParam; // only dvr use
    MI_VPE_HDRType_e      eHDRType;
    MI_VPE_3DNR_Level_e   e3DNRLevel;
    MI_BOOL                bMirror;
    MI_BOOL                bFlip;
    MI_BOOL                bWdrEn;    //Wdr on/off
    MI_BOOL                bEnLdc;
} MI_VPE_ChannelPara_t;
```

➤ 成员

成员名称	描述
stPqParam	Pq 参数设置
eHDRType	HDR 开关参数
e3DNRLevel	3dnr 量级参数
bMirror	使能 Input Mirror
bFlip	使能 Input Flip
bWdrEn	使能 WDR
bEnLdc	使能 Ldc

※ 注意事项

芯片	MAX e3DNRLevel	是否支持 bMirror/ bFlip
328Q/329D/326D	E_MI_VPE_3DNR_LEVEL7	不支持
325/325DE/327DE	E_MI_VPE_3DNR_LEVEL2	不支持
336D/336Q/339G	E_MI_VPE_3DNR_LEVEL2	支持
335/337DE	E_MI_VPE_3DNR_LEVEL2	支持

- 不同 chip 支持的最大 3dnr level 如上表， 设置超过 MAX， 内部自动采用 MAX Level,Level 越高 3DNR 强度越强，消耗的 buffer 也会多一点。
- 支持的 eHDRType 可以通过 MI\_SNR\_GetPadInfo 查询。
- stPqParam 仅 DVR mode 可用。
- bMirror/ bFlip 支持芯片系列如上表， 预防某些 Sensor 不支持翻转。

➤ 相关数据类型及接口

```
MI_VPE_PqParam_t
MI_VPE_HDRType_e
MI_VPE_3DNR_Level_e
```

3.12. MI\_VPE\_RegionInfo\_t

➤ 说明

定义 VPE 通道区域统计信息。

➤ 定义

```
typedef struct MI_VPE_RegionInfo_s
{
    MI_VPE_Region_t *pstRegion;           // region attribute
    MI_U32 u32RegionNum;                  // count of the region
} MI_VPE_RegionInfo_t;
```

➤ 成员

成员名称	描述
pstRegion	统计区域信息
u32RegionNum	统计区域数量

※ 注意事项

无。

➤ 相关数据类型及接口

无。

### 3.13. MI\_VPE\_PortMode\_t

➤ 说明

定义 VPE 端口模式。

➤ 定义

```
typedef struct MI_VPE_PortMode_s
{
    MI_U32 u32Width;           // Width of target image
    MI_U32 u32Height;          // Height of target image
    MI_SYS_PixelFormat_e ePixelFormat; // Pixel format of target image
    MI_SYS_CompressMode_e eCompressMode; // Compression mode of the output
}MI_VPE_PortMode_t;
```

➤ 成员

成员名称	描述
u32Width	端口输出画面水平大小
u32Height	端口输出画面竖直大小
ePixelFormat	端口输出 pixel format
eCompressMode	端口输出压缩模式

※ 注意事项

相关限制说明参考 [MI\\_VPE\\_SetPortMode](#) 注意说明。



➤ 相关数据类型及接口

MI\_SYS\_PixelFormat\_e  
MI\_SYS\_CompressMode\_e  
[MI\\_VPE\\_SetPortMode](#)

3.14. MI\_VPE\_IspInitPara\_t

➤ 说明

定义 VPE ISP 初始参数 Buffer。

➤ 定义

```
typedef struct MI_VPE_IspInitPara_s
{
    MI_U32 u32Revision;
    MI_U32 u32Size;
    MI_U8  u8Data[64];
}MI_VPE_IspInitPara_t;
```

➤ 成员

成员名称	描述
u32Revision	版本
u32Size	Buffer size
u8Data	数据

※ 注意事项

无。

➤ 举例

```
#include <CusEarlyInit_para.h>

MasterEarlyInitParam_t *pstEarlyInitParam = (MasterEarlyInitParam_t*) &stVpeChnAttr.tIspInitPara.u8Data[0];

pstEarlyInitParam->u16SnrEarlyFps = pCameraBootSetting->u8SensorFrameRate;
pstEarlyInitParam->u16SnrEarlyFlicker = pCameraBootSetting->u8AntiFlicker;
pstEarlyInitParam->u32SnrEarlyShutter = pCameraBootSetting->u32shutter;
pstEarlyInitParam->u32SnrEarlyGainX1024 = pCameraBootSetting->u32SensorGain;
pstEarlyInitParam->u32SnrEarlyDGain = pCameraBootSetting->u32DigitalGain;
pstEarlyInitParam->u16SnrEarlyAwbRGain = pCameraBootSetting->u16AWBRGain;
pstEarlyInitParam->u16SnrEarlyAwbGGain = pCameraBootSetting->u16AWBGGain;
pstEarlyInitParam->u16SnrEarlyAwbBGain = pCameraBootSetting->u16AWBBGain;

stVpeChnAttr.tIspInitPara.u32Revision = EARLYINIT_PARAM_TYPE_MASTER;//branch or customer abbreviation
```

```
stVpeChnAttr.tIspInitPara.u32Size = sizeof(MasterEarlyInitParam_t);
```

➤ 相关数据类型及接口

[MI\\_VPE\\_ChannelAttr\\_t](#)



## 4. 错误码

VPE API 错误码如表 3-1 所示：

表 3-1 VPE API 错误码

错误代码	宏定义	描述
0xA0078001	MI_ERR_VPE_INVALID_DEVID	Chanel port 号无效
0xA0078002	MI_ERR_VPE_INVALID_CHNID	Chanel 通道号无效
0xA0078003	MI_ERR_VPE_ILLEGAL_PARAM	Chanel 参数设置无效
0xA0078004	MI_ERR_VPE_EXIST	Chanel port 已创建
0xA0078005	MI_ERR_VPE_UNEXIST	Chanel port 未创建
0xA0078006	MI_ERR_VPE_NULL_PTR	输入参数空指针错误
0xA0078008	MI_ERR_VPE_NOT_SUPPORT	操作不支持
0xA0078009	MI_ERR_VPE_NOT_PERM	操作不允许
0xA007800C	MI_ERR_VPE_NOMEM	分配内存失败
0xA007800D	MI_ERR_VPE_NOBUF	分配 BUF 池失败
0xA007800E	MI_ERR_VPE_BUF_EMPTY	图像队列为空
0xA0078010	MI_ERR_VPE_NOTREADY	Chanel 系统未初始化
0xA0078012	MI_ERR_VPE_BUSY	Chanel 系统忙