

# MI VIF API

---

**Version 2.03**

© 2018 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>	07/25/2018

## TABLE OF CONTENTS

<b>REVISION HISTORY .....</b>	<b>i</b>
<b>TABLE OF CONTENTS.....</b>	<b>ii</b>
<b>1. API 参考 .....</b>	<b>1</b>
1.1. 概述.....	1
1.2. 功能模块 API .....	1
1.2.1 MI_VIF_SetDevAttr .....	2
1.2.2 MI_VIF_GetDevAttr .....	5
1.2.3 MI_VIF_EnableDev .....	6
1.2.4 MI_VIF_DisableDev .....	8
1.2.5 MI_VIF_SetChnPortAttr .....	9
1.2.6 MI_VIF_GetChnPortAttr .....	10
1.2.7 MI_VIF_EnableChnPort.....	11
1.2.8 MI_VIF_DisableChnPort .....	12
1.2.9 MI_VIF_Query .....	13
1.2.10 MI_VIF_SetDev2SnrPadMux.....	14
<b>2. VIF 数据类型 .....</b>	<b>16</b>
2.1. MI_VIF_MAX_DEV_NUM.....	17
2.2. MI_VIF_MAX_WAY_NUM_PER_DEV .....	17
2.3. MI_VIF_MAX_PHYCHN_NUM .....	17
2.4. MI_VIF_EXT_CHN_START .....	18
2.5. MI_VIF_MAX_EXT_CHN_NUM.....	18
2.6. MI_VIF_MAX_EXTCHN_BIND_PER_CHN .....	19
2.7. MI_VIF_MAX_CHN_NUM .....	20
2.8. MI_VIF_SUBCHN .....	20
2.9. MI_VIF_IntfMode_e .....	21
2.10. MI_VIF_WorkMode_e.....	22
2.11. MI_VIF_FrameRate_e .....	24
2.12. MI_VIF_DataYuvSeq_e .....	24
2.13. MI_VIF_ClkEdge_e.....	25
2.14. MI_VIF_BitOrder_e .....	26
2.15. MI_VIF_HDRTYPE_e .....	27
2.16. MI_VIF_Polar_e .....	28
2.17. MI_VIF_SyncAttr_t.....	29
2.18. MI_VIF_DevAttr_t.....	30
2.19. MI_VIF_ChnPortAttr_t.....	31
2.20. MI_VIF_ChnPortStat_t .....	32
2.21. MI_VIF_SNRPad_e.....	33
2.22. MI_VIF_Dev2SnrPadMuxCfg_t .....	33
<b>3. VIF 错误码 .....</b>	<b>36</b>
<b>4. 典型应用场景 .....</b>	<b>37</b>



## 1. API 参考

---

### 1.1. 概述

视频输入 ( VIF ) 实现启用视频输入设备、视频输入通道、绑定视频输入通道等功能。

### 1.2. 功能模块 API

API名	功能
<a href="#">MI_VIF_SetDevAttr</a>	设置 VIF 设备属性
<a href="#">MI_VIF_GetDevAttr</a>	获取 VIF 设备属性
<a href="#">MI_VIF_EnableDev</a>	启用 VIF 设备
<a href="#">MI_VIF_DisableDev</a>	禁用 VIF 设备
<a href="#">MI_VIF_SetChnPortAttr</a>	设置 VIF 通道Port属性
<a href="#">MI_VIF_GetChnPortAttr</a>	获取 VIF 通道Port属性
<a href="#">MI_VIF_EnableChnPort</a>	启用 VIF 通道Port
<a href="#">MI_VIF_DisableChnPort</a>	禁用 VIF 通道Port
<a href="#">MI_VIF_Query</a>	查询 VIF 通道的Port中断计数、平均帧率等信息
<a href="#">MI_VIF_SetDev2SnrPadMux</a>	设置VIF 设备和sensor 之间绑定关系

1.2.1 MI\_VIF\_SetDevAttr

➤ 功能

设置 VIF 设备属性。基本设备属性默认了部分芯片配置，满足绝大部分的 AD 芯片对接要求。

➤ 语法

```
MI_S32 MI_VIF_SetDevAttr (MI_VIF_DEV u32VifDev, MI_VIF_DevAttr_t *pstDevAttr)
```

➤ 形参

参数名称	描述	输入/输出
u32VifDev	VIF 设备号。 取值范围：[0, MI_VIF_MAX_DEV_NUM)。	输入
pstDevAttr	VIF 设备属性指针。 静态属性。	输入

➤ 返回值

返回值

MI\_OK 成功。

非 0 失败，详情参照[错误码](#)。

➤ 依赖

- 头文件：mi\_vif\_datatype.h、mi\_vif.h
- 库文件：libmi\_vif.a

※ 注意

- 在调用前要保证 VIF 设备处于禁用状态。如果 VIF 设备已处于使能状态，可以使用 [MI\\_VIF\\_DisableDev](#) 来禁用设备。
- 参数 pstDevAttr 主要用来配置指定 VIF 设备的视频接口模式，用于与外围 camera、sensor 或 codec 对接，支持的接口模式包括 BT.656、digital camera、BT.1120 逐行以及 BT.1120 隔行模式。用户需要配置以下几类信息，具体属性意义参见 [3.5 数据类型](#) 部分的说明：
  - 接口模式信息 enIntfMode：接口模式为 BT656、digital camera、BT.1120 等模式
  - 工作模式信息 enWorkMode：1 路、2 路、4 路复合模式
  - 时钟信息 enClkEdge：上升沿或下降沿采样，当前端送来双沿数据时，VIF 还可以

进行双沿采样。

- 数据布局信息：有效数据的布局
- 数据信息：隔行或逐行输入、YUV 数据输入顺序
- 同步时序信息：垂直、水平同步信号的属性

#### ➤ 举例

```
MI_S32 s32Ret;
MI_U32 VifDev = 0;
MI_CHN VifChn = 0;
MI_VIF_ChnPortAttr_t stChnAttr;
MI_VIF_DevAttr_t stDevAttr;
MI_SNR_PADInfo_t stPad0Info;
MI_SNR_PlaneInfo_t stSnrPlane0Info;
memset(&stDevAttr, 0x0, sizeof(MI_VIF_DevAttr_t));
memset(&stPad0Info, 0x0, sizeof(MI_SNR_PADInfo_t));
memset(&stSnrPlane0Info, 0x0, sizeof(MI_SNR_PlaneInfo_t));

MI_SNR_GetPadInfo(E_MI_SNR_PAD_ID_0, &stPad0Info);
MI_SNR_GetPlaneInfo(E_MI_SNR_PAD_ID_0, 0, &stSnrPlane0Info);

stDevAttr.eIntfMode = stPad0Info.eIntfMode;
stDevAttr.eWorkMode = E_MI_VIF_WORK_MODE_RGB_REALTIME;
stDevAttr.eHDRType = eVifHdrType;
if(stDevAttr.eIntfMode == E_MI_VIF_MODE_BT656)
    stDevAttr.eClkEdge = stPad0Info.unIntfAttr.stBt656Attr.eClkEdge;
else
    stDevAttr.eClkEdge = E_MI_VIF_CLK_EDGE_DOUBLE;

if(stDevAttr.eIntfMode == E_MI_VIF_MODE_MIPI)
    stDevAttr.eDataSeq = stPad0Info.unIntfAttr.stMipiAttr.eDataYUVOrder;
else
    stDevAttr.eDataSeq = E_MI_VIF_INPUT_DATA_YUYV;
```



```

if(stDevAttr.eIntfMode == E_MI_VIF_MODE_BT656)

    memcpy(&stDevAttr.stSyncAttr, &stPad0Info.unIntfAttr.stBt656Attr.stSyncAttr,
sizeof(MI_VIF_SyncAttr_t));

stDevAttr.eBitOrder = E_MI_VIF_BITORDER_NORMAL;
s32Ret = MI\_VIF\_SetDevAttr (VifDev, &stDevAttr) ;
if (s32Ret != MI_SUCCESS)
{
    printf("Set dev attributes failed with error code %#x!\n", s32Ret) ;
    return MI_FAILURE;
}
s32Ret = MI\_VIF\_EnableDev (VifDev) ;
if (s32Ret != MI_SUCCESS)
{
    printf("Enable dev failed with error code %#x!\n", s32Ret) ;
    return MI_FAILURE;
}

stChnAttr.stCapRest.s32X = 0;
stChnAttr.stCapRect.s32Y = 0;
stChnAttr.stCapRect.u32Width = stSnrPlane0Info.stCapRect.u16Width;
stChnAttr.stCapRect.u32Height = stSnrPlane0Info.stCapRect.u16Height;
stChnAttr.stDestSize.u32Width = stSnrPlane0Info.stCapRect.u16Width;
stChnAttr.stDestSize.u32Height = stSnrPlane0Info.stCapRect.u16Height;
stChnAttr.enCapSel = E_MI_SYS_FIELDTYPE_BOTH;
stChnAttr.enScanMode = E_MI_VIF_SCAN_PROGRESSIVE;
stChnAttr.enPixFormat = E_MI_SYS_PIXEL_FRAME_YUV_SEMIPLANAR_420;
stChnAttr.eFrameRate = E_MI_VIF_FRAMERATE_FULL;
s32Ret = MI\_VIF\_SetChnPortAttr (VifChn, &stChnAttr) ;
if (s32Ret != MI_SUCCESS)
{
    printf("Set chn attributes failed with error code %#x!\n", s32Ret) ; return MI_FAILURE;
}

```

```

}

s32Ret = MI_VIF_EnableChnPort (VifChn, 0); //Main stream
if (s32Ret != MI_SUCCESS)
{
    printf("Enable chn failed with error code %#x!\n", s32Ret);
    return MI_FAILURE;
}

/* now, VIF is capturing images, you can do something else ... */
s32Ret = MI_VIF_DisableChnPort (VifChn, 0); //Disable
if (s32Ret != MI_SUCCESS)
{
    printf("Disable Vif chn failed with error code %#x!\n", s32Ret);
    return MI_FAILURE;
}

s32Ret = MI_VIF_DisableDev (VifDev);

.....

```

➤ 相关主题

[MI\\_VIF\\_GetDevAttr](#)

1.2.2 MI\_VIF\_GetDevAttr

➤ 功能

获取 VIF 设备属性。

➤ 语法

```
MI_S32 MI_VIF_GetDevAttr (MI_VIF_DEV u32VifDev, MI\_VIF\_DevAttr\_t *pstDevAttr)
```

➤ 形参

参数名称	描述	输入/输出
u32VifDev	VIF 设备号。 取值范围：[0, MI_VIF_MAX_DEV_NUM)。 注：在Bind的时候VIF的devid需设置为0.	输入
pstDevAttr	VIF 设备属性指针。	输出

➤ 返回值

返回值 { MI\_OK 成功。  
非 0 失败，详情参照[错误码](#)。

➤ 依赖

- 头文件：mi\_vif\_datatype.h、mi\_vif.h
- 库文件：libmi\_vif.a

➤ 注意

如果未设置 VIF 设备属性，该接口将返回失败。

➤ 举例

无。

➤ 相关主题

[MI\\_VIF\\_SetDevAttr](#)

1.2.3 MI\_VIF\_EnableDev

➤ 功能

启用 VIF 设备。

➤ 语法

MI\_S32 MI\_VIF\_EnableDev (MI\_VIF\_DEV u32VifDev) ;

➤ 形参

参数名称	描述	输入/输出
u32VifDev	VIF 设备号。 取值范围：[0, MI_VIF_MAX_DEV_NUM)。	输入

➤ 返回值

{ MI\_OK 成功。

## 返回值

非 0 失败，详情参照[错误码](#)。

### ➤ 依赖

- 头文件：mi\_vif\_datatype.h、mi\_vif.h
- 库文件：libmi\_vif.a

## ※ 注意

- 启用前必须已经设置设备属性，否则返回失败。
- 可重复启用，不返回失败。

## ➤ 举例

请参见 [MI\\_VIF\\_SetDevAttr](#) 的举例。

## ➤ 相关主题

[MI\\_VIF\\_DisableDev](#)

## 1.2.4 MI\_VIF\_DisableDev

## ➤ 功能

禁用 VIF 设备。

## ➤ 语法

```
MI_S32 MI_VIF_DisableDev (MI_VIF_DEV u32VifDev);
```

## ➤ 形参

参数名称	描述	输入/输出
u32VifDev	VIF 设备号。 取值范围：[0, MI_VIF_MAX_DEV_NUM)。	输入

## ➤ 返回值

返回值 {  
MI\_OK 成功。  
非 0 失败，详情参照[错误码](#)。

## ➤ 依赖

- 头文件：mi\_vif\_datatype.h、mi\_vif.h
- 库文件：libmi\_vif.a

## ※ 注意

- 必须先禁用所有与该 VIF 设备绑定的 VIF 通道后，才能禁用 VIF 设备。
- 可重复禁用，不返回失败。

➤ 举例

请参见 [MI\\_VIF\\_SetDevAttr](#) 的举例。

➤ 相关主题

[MI\\_VIF\\_EnableDev](#)

1.2.5 MI\_VIF\_SetChnPortAttr

➤ 功能

设置 VIF 通道次属性。

➤ 语法

MI\_S32 MI\_VIF\_SetChnPortAttr (MI\_VIF\_CHN u32VifChn, MI\_VIF\_PORT u32ChnPort, [MI\\_VIF\\_ChnPortAttr\\_t](#) \*pstAttr) ;

➤ 形参

参数名称	描述	输入/输出
u32VifChn	VIF 通道号。 取值范围：[0, MI_VIF_MAX_PHYCHN_NUM)。	输入
u32ChnPort	Port 号	输入
pstAttr	VIF 通道Port属性指针。	输入

➤ 返回值

返回值 {  
MI\_OK 成功。  
非 0 失败，详情参照[错误码](#)。

➤ 依赖

- 头文件：mi\_vif\_datatype.h、mi\_vif.h
- 库文件：libmi\_vif.a

➤ 注意

- 默认情况下，使用 MI\_VIF\_SetChnPortAttr 接口的目的是设置端口属性,如CapSize、DestSize、FrameRate等等。

➤ 举例

```

MI\_VIF\_ChnPortAttr\_t stAttr;

/* set attribute of VIF device*/
/* ... .. */
/* enable VIF device*/
/* ... .. */

StAttr.eFrameRate = E_MI_VIF_FRAMERATE_HALF;

/* set main attribute of VIF channel, size is D1 */
/* ... .. */
MI\_VIF\_GetChnPortAttr (VifChn, VifPort, &stAttr);

stAttr.eCapSel = E_MI_SYS_FIELDTYPE_BOTH;
stAttr.eScanMode = E_MI_SYS_FRAME_SCAN_MODE_PROGRESSIVE;

/* set sub attribute of VIF channel, size is CIF */
if (MI\_VIF\_SetChnPortAttr (VifChn, VifPort, &stAttr))
{
    BBIT_ERR("set chn attr ex fail\n");
    return -1;
}

/* enable VIF channel*/
MI\_VIF\_EnableChnPort (VifChn, VifPort);
/* ... .. */

```

➤ 相关主题

[MI\\_VIF\\_GetChnPortAttr](#)

## 1.2.6 MI\_VIF\_GetChnPortAttr

➤ 功能

获取 VIF 通道次属性。

➤ 语法

```
MI_S32 MI_VIF_GetChnPortAttr(MI_VIF_CHN u32VifChn, MI_VIF_PORT u32ChnPort,
```

[MI\\_VIF\\_ChnPortAttr\\_t](#) \*pstAttr) ;

➤ 形参

参数名称	描述	输入/输出
u32VifChn	VIF 通道号。 取值范围：[0, MI_VIF_MAX_PHYCHN_NUM)。	输入
u32ChnPort	Port 号	输入
pstAttr	VIF 通道Port属性指针。动态属性。	输出

➤ 返回值

返回值 {  
MI\_OK 成功。  
非 0 失败，详情参照[错误码](#)。

➤ 依赖

- 头文件：mi\_vif\_datatype.h、mi\_vif.h
- 库文件：libmi\_vif.a

※ 注意

必须先设置属性再获取属性，否则将返回 MI\_ERR\_VIF\_FAILED\_NOTCONFIG。

➤ 举例

请参见 [MI\\_VIF\\_SetChnPortAttr](#) 的举例。

➤ 相关主题

[MI\\_VIF\\_SetChnPortAttr](#)

1.2.7 MI\_VIF\_EnableChnPort

➤ 功能

启用 VIF 通道。

➤ 语法



MI\_S32 MI\_VIF\_EnableChnPort (MI\_VIF\_CHN u32VifChn , MI\_VIF\_PORT u32ChnPort) ;

➤ 形参

参数名称	描述	输入/输出
VifChn	VIF 通道号。	输入

➤ 返回值

返回值 { MI\_OK 成功。  
非 0 失败，详情参照[错误码](#)。

➤ 依赖

- 头文件：mi\_vif\_datatype.h、mi\_vif.h
- 库文件：libmi\_vif.a

※ 注意

- 必须先设置通道属性，且通道所绑定的 VIF 设备必须使能。
- 如果设置了 VIF 与 DISP/VENC 的绑定关系，成功调用该接口后即可得到视频数据。
- 可重复启用 VIF 通道，不返回失败。

➤ 举例

请参见 [MI\\_VIF\\_SetDevAttr](#) 的举例。

➤ 相关主题

[MI\\_VIF\\_DisableChnPort](#)

1.2.8 MI\_VIF\_DisableChnPort

➤ 功能

禁用 VIF 通道。

➤ 语法

MI\_S32 MI\_VIF\_DisableChnPort (MI\_VIF\_CHN u32VifChn, MI\_VIF\_PORT u32ChnPort) ;

➤ 形参

参数名称	描述	输入/输出
u32VifChn	VIF 通道号。	输入
u32ChnPort	Port号	输入

➤ 返回值

返回值 { MI\_OK 成功。  
非 0 失败，详情参照[错误码](#)。

➤ 依赖

- 头文件：mi\_vif\_datatype.h、mi\_vif.h
- 库文件：libmi\_vif.a

※ 注意

- 禁用 VIF 通道Port后，此 VIF 通道Port即停止采集视频输入数据，如果已经绑定 DISP 或 VENC，则 VO 或 VENC 将不会再接收到视频图像。
- 可重复禁用 VIF 通道Port，不返回失败。

➤ 举例

请参见 [MI\\_VIF\\_SetDevAttr](#) 的举例。

➤ 相关主题

[MI\\_VIF\\_EnableChnPort](#)

1.2.9 MI\_VIF\_Query

➤ 功能

查询 VIF 通道的中断计数、平均帧率等信息。

➤ 语法

MI\_S32 MI\_VIF\_Query (MI\_VIF\_CHN u32VifChn, MI\_VIF\_PORT u32ChnPort, [MI\\_VIF\\_ChnPortStat\\_t](#) \*pstStat) ;

➤ 形参

参数名称	描述	输入/输出
u32VifChn	VIF 通道号。	输入
u32ChnPort	Port口	
pstStat	通道信息结构体指针。	输出

➤ 返回值

返回值 { MI\_OK 成功。  
非 0 失败，详情参照[错误码](#)。

➤ 依赖

- 头文件：mi\_vif\_datatype.h、mi\_vif.h
- 库文件：libmi\_vif.a

※ 注意

- 该接口可查询中断计数、通道使能状态、平均帧率、中断丢失数、获取 VB 失败次数、图像宽高等信息。
- 通过该接口获取到的帧率是每 1 秒钟的平均帧率，即 VIF 会每隔一秒统计一次平均帧率，该值并不精确，会有些波动。
- 用户可通过该接口查询中断丢失数，如果该数值一直在增加，说明 VIF 工作出现异常。

➤ 举例

无。

➤ 相关主题

无。

1.2.10 MI\_VIF\_SetDev2SnrPadMux

➤ 功能

设置 Vif Dev 和 sensor Pad 之间的绑定关系

➤ 语法

MI\_S32 MI\_VIF\_SetDev2SnrPadMux([MI\\_VIF\\_Dev2SnrPadMuxCfg\\_t](#) stVifDevMap[], MI\_U8

u8Length);

➤ 形参

参数名称	描述	输入/输出
stVifDevMap	Dev 和SensorPad Mapping 关系	输入
u8Length	Dev Num	输入

➤ 返回值

返回值 { MI\_OK 成功。  
非 0 失败，详情参照[错误码](#)。

➤ 依赖

- 头文件：mi\_vif\_datatype.h、mi\_vif.h
- 库文件：libmi\_vif.a

※ 注意

在默认情况下 vif Dev 和 SensorPad 对应关系是 vif Dev0 -> SensorPad0, vif Dev1 -> SensorPad1.

➤ 举例

无。

➤ 相关主题

无。

## 2. VIF 数据类型

---

视频输入相关数据类型定义如下：

<a href="#">MI_VIF_MAX_DEV_NUM</a>	定义视频输入设备的最大个数。
<a href="#">MI_VIF_MAX_WAY_NUM_PER_DEV</a>	定义一个视频输入设备下的数据通路最大个数。
<a href="#">MI_VIF_MAX_PHYCHN_NUM</a>	定义 VIF 物理通道的最大个数
<a href="#">MI_VIF_EXT_CHN_START</a>	定义 VIF 扩展通道号的起始值
<a href="#">MI_VIF_MAX_EXT_CHN_NUM</a>	定义 VIF 扩展通道的最大个数
<a href="#">MI_VIF_MAX_EXTCHN_BIND_PER_CHN</a>	定义 VIF 单个通道可绑定的扩展通道最大个数。
<a href="#">MI_VIF_MAX_CHN_NUM</a>	定义 VIF 通道的最大个数
<a href="#">MI_VIF_SUBCHN</a>	定义 VIF 通道的次通道号
<a href="#">MI_VIF_IntfMode_e</a>	定义视频输入设备的接口模式
<a href="#">MI_VIF_WorkMode_e</a>	定义视频设备的复合工作模式
<a href="#">MI_VIF_DataYuvSeq_e</a>	定义视频设备接收的 YUV 数据的数据排列顺序
<a href="#">MI_VIF_ClkEdge_e</a>	定义视频设备接收的时钟类型
<a href="#">MI_VIF_BitOrder_e</a>	定义视频设备的data bit翻转顺序
<a href="#">MI_VIF_HDRTYPE_e</a>	定义视频设备HDR 类型
<a href="#">MI_VIF_Polar_e</a>	定义信号极性
<a href="#">MI_VIF_SyncAttr_t</a>	定义同步信号属性
<a href="#">MI_VIF_DevAttr_t</a>	定义视频输入设备的属性。
<a href="#">MI_VIF_ChnPortAttr_t</a>	定义 VIF 通道属性。
<a href="#">MI_VIF_ChnPortStat_t</a>	定义VIF 通道信息结构体
<a href="#">MI_VIF_SNRPad_e</a>	定义 SensorPad Id
<a href="#">MI_VIF_Dev2SnrPadMuxCfg_t</a>	定义VIF 设备和SensorPad 绑定关系

## 2.1. MI\_VIF\_MAX\_DEV\_NUM

➤ 说明

定义视频输入设备的最大个数。

➤ 定义

```
#define MI_VIF_MAX_DEV_NUM 4
```

※ 注意事项

无。

➤ 相关数据类型及接口

无。

## 2.2. MI\_VIF\_MAX\_WAY\_NUM\_PER\_DEV

➤ 说明

定义一个视频输入设备下的数据通路最大个数。

➤ 定义

```
#define MI_VIF_MAX_WAY_NUM_PER_DEV 4
```

※ 注意事项

无。

➤ 相关数据类型及接口

无。

## 2.3. MI\_VIF\_MAX\_PHYCHN\_NUM

➤ 说明

定义 VIF 物理通道的最大个数。

➤ 定义

```
#define MI_VIF_MAX_PHYCHN_NUM 16
```

※ 注意事项

具体见各款芯片“视频物理通道”小节的描述。

➤ 相关数据类型及接口

无。

## 2.4. MI\_VIF\_EXT\_CHN\_START

➤ 说明

定义 VIF 扩展通道号的起始值。

➤ 定义

```
#define MI_VIF_EXT_CHN_START (MI\_VIF\_MAX\_PHYCHN\_NUM)
```

※ 注意事项

无。

➤ 相关数据类型及接口

无。

## 2.5. MI\_VIF\_MAX\_EXT\_CHN\_NUM

➤ 说明

定义 VIF 扩展通道的最大个数。

➤ 定义

```
#define MI_VIF_MAX_EXT_CHN_NUM 0
```

※ 注意事项

无。

➤ 相关数据类型及接口

无。

## 2.6. MI\_VIF\_MAX\_EXTCHN\_BIND\_PER\_CHN

### ➤ 说明

定义 VIF 单个通道可绑定的扩展通道最大个数。



➤ 定义

```
#define MI_VIF_MAX_EXTCHN_BIND_PER_CHN 0
```

※ 注意事项

无。

➤ 相关数据类型及接口

无。

## 2.7. MI\_VIF\_MAX\_CHN\_NUM

➤ 说明

定义 VIF 通道的最大个数。

➤ 定义

```
#define MI_VIF_MAX_CHN_NUM 16
```

➤ 相关数据类型及接口

无。

## 2.8. MI\_VIF\_SUBCHN

➤ 说明

定义 VIF 通道的次通道号。

➤ 定义

```
#define MI_VIF_SUBCHN (VifChn) (VifChn + 16)
```

※ 注意事项

VifChn 取值为{0}。

➤ 相关数据类型及接口

无。

## 2.9. MI\_VIF\_IntfMode\_e

### ➤ 说明

定义视频设备的接口模式。

### ➤ 定义

```
typedef enum
{
    E_MI_VIF_MODE_BT656 = 0,
    E_MI_VIF_MODE_DIGITAL_CAMERA,
    E_MI_VIF_MODE_BT1120_STANDARD,
    E_MI_VIF_MODE_BT1120_INTERLEAVED,
    E_MI_VIF_MODE_MIPI
    E_MI_VIF_MODE_NUM
} MI_VIF_IntfMode_e;
```

### ➤ 成员

成员名称	描述
E_MI_VIF_MODE_BT656	输入数据的协议符合标准 BT.656 协议，端口数据输入模式为亮度色度复合模式，分量模式为单分量。
E_MI_VIF_MODE_DIGITAL_CAMERA	输入数据的协议为 Digital camera 协议，端口数据输入模式为亮度色度复合模式，分量模式为单分量。
E_MI_VIF_MODE_BT1120_STANDARD	输入数据的协议符合标准 BT.1120 协议（BT.656+双分量），端口数据输入模式为亮度色度分离模式，分量模式为双分量。
E_MI_VIF_MODE_BT1120_INTERLEAVED	输入数据的协议符合 BT.1120 interleave 模式，端口数据输入模式为亮度色度分离模式，分量模式为双分量。
E_MI_VIF_MODE_MIPI	输入数据符合MIPI 协议

### ※ 注意事项

如果是 Sensor，则为 E\_MI\_VIF\_MODE\_DIGITAL\_CAMERA。

### ➤ 相关数据类型及接口

[MI\\_VIF\\_DevAttr\\_t](#)

[MI\\_VIF\\_SetDevAttr](#)

2.10. MI\_VIF\_WorkMode\_e

➤ 说明

定义视频设备的复合工作模式。

➤ 定义

```
typedef enum
{
    /* BT656 multiple ch mode */
    E_MI_VIF_WORK_MODE_1MULTIPLEX,
    E_MI_VIF_WORK_MODE_2MULTIPLEX,
    E_MI_VIF_WORK_MODE_4MULTIPLEX,

    /* RGB mode for MIPI/Parallel sensor */
    E_MI_VIF_WORK_MODE_RGB_REALTIME,
    E_MI_VIF_WORK_MODE_RGB_FRAMEMODE,
    E_MI_VIF_WORK_MODE_MAX
} MI_VIF_WorkMode_e;
```

➤ 成员

成员名称	描述
E_MI_VIF_WORK_MODE_1MULTIPLEX	1 路复合工作模式。
E_MI_VIF_WORK_MODE_2MULTIPLEX	2 路复合工作模式，输入数据的协议必须为标准 BT.656 协议。
E_MI_VIF_WORK_MODE_4MULTIPLEX	4 路复合工作模式，输入数据的协议必须为标准 BT.656 协议。
E_MI_VIF_WORK_MODE_RGB_REALTIME	RGB Realtime mode for MIPI/Parallel sensor
E_MI_VIF_WORK_MODE_RGB_FRAMEMODE,	RGB Framemode mode for MIPI/Parallel sensor

※ 注意事项

当该项设为 2 路或 4 路复合工作模式时，设备输入的协议必须是 BT.656 协议。1 路复合工作模式没有限制。

➤ 相关数据类型及接口

[MI\\_VIF\\_DevAttr\\_t](#)  
[MI\\_VIF\\_SetDevAttr](#)

## 2.11. MI\_VIF\_FrameRate\_e

➤ 说明

定义视频设备输出 fps 和输入 fps 的关系。

➤ 定义

```
typedef enum
{
    E_MI_VIF_FRAMERATE_FULL = 0,
    E_MI_VIF_FRAMERATE_HALF,
    E_MI_VIF_FRAMERATE_QUARTR,
    E_MI_VIF_FRAMERATE_OCTANT,
    E_MI_VIF_FRAMERATE_THREE_QUARTERS,
    E_MI_VIF_FRAMERATE_NUM,
} MI_VIF_FrameRate_e;
```

➤ 成员

成员名称	描述
E_MI_VIF_FRAMERATE_FULL	源和目标1:1输出。
E_MI_VIF_FRAMERATE_HALF	源和目标2:1输出。
E_MI_VIF_FRAMERATE_QUARTER	源和目标4:1输出。
E_MI_VIF_FRAMERATE_OCTANT	源和目标8:1输出。
E_MI_VIF_FRAMERATE_THREE_QUARTERS	源和目标4:3输出。

※ 注意事项

仅在 BT656multiple ch mode 下设置又效

➤ 相关数据类型及接口

```
MI\_VIF\_ChnPortAttr\_t
MI\_VIF\_SetChnPortAttr
```

## 2.12. MI\_VIF\_DataYuvSeq\_e

➤ 说明

定义视频设备接收的 YUV 数据的数据排列顺序。

➤ 定义

```
typedef enum
{
    E_MI_VIF_INPUT_DATA_VUVU = 0,
    E_MI_VIF_INPUT_DATA_UVUV,
    E_MI_VIF_INPUT_DATA_UYVY = 0,
    E_MI_VIF_INPUT_DATA_VYUY,
    E_MI_VIF_INPUT_DATA_YUYV,
    E_MI_VIF_INPUT_DATA_YVYU,
    E_MI_VIF_DATA_YUV_NUM
} MI_VIF_DataYuvSeq_e;
```

➤ 成员

成员名称	描述
E_MI_VIF_INPUT_DATA_VUVU	YUV 数据通过分离模式输入时，C 分量的输入排列顺序为 VUVU。
E_MI_VIF_INPUT_DATA_UVUV	YUV 数据通过分离模式输入时，C 分量的输入排列顺序为 UVUV。
E_MI_VIF_INPUT_DATA_UYVY	YUV 数据通过复合模式输入时，顺序为 UYVY。
E_MI_VIF_INPUT_DATA_VYUY	YUV 数据通过复合模式输入时，顺序为 VYUY。
E_MI_VIF_INPUT_DATA_YUYV	YUV 数据通过复合模式输入时，顺序为 YUYV。
E_MI_VIF_INPUT_DATA_YVYU	YUV 数据通过复合模式输入时，顺序为 YVYU。

※ 注意事项

无。

➤ 相关数据类型及接口

[MI\\_VIF\\_DevAttr\\_t](#)  
[MI\\_VIF\\_SetDevAttr](#)

## 2.13. MI\_VIF\_ClkEdge\_e

➤ 说明

定义视频设备接收的时钟类型。

➤ 定义

```
typedef enum
{
    E_MI_VIF_CLK_EDGE_SINGLE_UP = 0,
    E_MI_VIF_CLK_EDGE_SINGLE_DOWN,
    E_MI_VIF_CLK_EDGE_DOUBLE,
    E_MI_VIF_CLK_EDGE_NUM
} MI_VIF_ClkEdge_e;
```

➤ 成员

成员名称	描述
E_MI_VIF_CLK_EDGE_SINGLE_UP	时钟单沿模式，且 VIF 设备在上升沿采样。
E_MI_VIF_CLK_EDGE_SINGLE_DOWN	时钟单沿模式，且 VIF 设备在下降沿采样。
E_MI_VIF_CLK_EDGE_DOUBLE	前端送过来双沿数据时，VIF 进行双沿采样。

※ 注意事项

无。

➤ 相关数据类型及接口

[MI\\_VIF\\_DevAttr\\_t](#)  
[MI\\_VIF\\_SetDevAttr](#)

## 2.14. MI\_VIF\_BitOrder\_e

➤ 说明

Vif 设备的 data bit 顺序翻转设定

➤ 定义

```
typedef enum
{
    E_MI_VIF_BITORDER_NORMAL = 0,
    E_MI_VIF_BITORDER_REVERSED
} MI_VIF_BitOrder_e;
```

➤ 成员

成员名称	描述
E_MI_VIF_BITORDER_NORMAL	正常data bit排序
E_MI_VIF_BITORDER_REVERSED	逆序data bit排序

※ 注意事项

无。

➤ 相关数据类型及接口

[MI\\_VIF\\_DevAttr\\_t](#)  
[MI\\_VIF\\_SetDevAttr](#)

## 2.15. MI\_VIF\_HDRType\_e

➤ 说明

定义视频设备 HDR 类型。

➤ 定义

```
typedef enum
{
    E_MI_VIF_HDR_TYPE_OFF,
    E_MI_VIF_HDR_TYPE_VC,          //virtual channel mode HDR,vc0->long, vc1->short
    E_MI_VIF_HDR_TYPE_DOL,
    E_MI_VIF_HDR_TYPE_EMBEDDED,   //compressed HDR mode
    E_MI_VIF_HDR_TYPE_LI,         //Line interlace HDR
    E_MI_VIF_HDR_TYPE_MAX
} MI_VIF_HDRType_e;
```

➤ 成员

成员名称	描述
E_MI_VIF_HDR_TYPE_OFF	不开HDR
E_MI_VIF_HDR_TYPE_VC	virtual channel mode HDR,vc0->long, vc1->short
E_MI_VIF_HDR_TYPE_DOL	Digital Overlap High Dynamic Range
E_MI_VIF_HDR_TYPE_EMBEDDED	compressed HDR mode
E_MI_VIF_HDR_TYPE_LI	Line interlace HDR

※ 注意事项

目前只支持如下两种模式：  
E\_MI\_VIF\_HDR\_TYPE\_OFF  
E\_MI\_VIF\_HDR\_TYPE\_DOL

➤ 相关数据类型及接口

[MI\\_VIF\\_DevAttr\\_t](#)  
[MI\\_VIF\\_SetDevAttr](#)



## 2.16. MI\_VIF\_Polar\_e

➤ 说明

定义视频输入信号有效。

➤ 定义

```
typedef enum
{
    E_MI_VIF_PIN_POLAR_POS,
    E_MI_VIF_PIN_POLAR_NEG
} MI_VIF_Polar_e;
```

➤ 成员

成员名称	描述
E_MI_VIF_PIN_POLAR_POS	高电平信号有效
E_MI_VIF_PIN_POLAR_NEG	低电平信号有效

※ 注意事项

只有 parallel sensor 接口可以设置

➤ 相关数据类型及接口

[MI\\_VIF\\_SyncAttr\\_t](#)

## 2.17. MI\_VIF\_SyncAttr\_t

➤ 说明

同步信号属性设置。

➤ 定义

```
typedef struct MI_VIF_SyncAttr_s
{
    MI_VIF_Polar_e    eVsyncPolarity;
    MI_VIF_Polar_e    eHsyncPolarity;
    MI_VIF_Polar_e    ePclkPolarity;
    MI_U32            VsyncDelay;
    MI_U32            HsyncDelay;
    MI_U32            PclkDelay;
} MI_VIF_SyncAttr_t;
```

➤ 成员

成员名称	描述
eVsyncPolarity	垂直同步信号有效极性
eHsyncPolarity	水平同步信号有效极性
ePclkPolarity	Pixel Clock 有效极性
VsyncDelay	垂直同步信号延时时间
HsyncDelay	水平同步信号延时时间

PclkDelay	Pixel Clock 延时时间
-----------	------------------

※ 注意事项

只有 parallel sensor 接口可以设置

➤ 相关数据类型及接口

[MI\\_VIF\\_DevAttr\\_t](#)

## 2.18. MI\_VIF\_DevAttr\_t

➤ 说明

定义视频输入设备的属性。

➤ 定义

```
typedef struct MI_VIF_DevAttr_s
{
    MI\_VIF\_IntfMode\_e eIntfMode;
    MI\_VIF\_WorkMode\_e eWorkMode;
    MI\_VIF\_HDRTType\_e eHDRTType;
    MI\_VIF\_ClkEdge\_e eClkEdge;
    MI\_VIF\_DataYuvSeq\_e eDataSeq;
    MI\_VIF\_BitOrder\_e eBitOrder;
    /* adjust bit order layout */
    MI\_VIF\_SyncAttr\_t stSyncAttr;
} MI_VIF_DevAttr_t;
```

➤ 成员

成员名称	描述
eIntfMode	接口模式。
eWorkMode	工作模式。
eHDRTType	HDR类型
eClkEdge	时钟边沿模式（上升沿采样、下降沿采样、双沿采样）。
eDataSeq	输入数据顺序（仅支持 yuv 格式），DC 模式时必须配置，其它模式时无效。
eBitOrder	Vif的data线layout是正序还是逆序
stSyncAttr	同步信号属性

※ 注意事项

无

➤ 相关数据类型及接口

[MI\\_VIF\\_SetDevAttr](#)

2.19. MI\_VIF\_ChnPortAttr\_t

➤ 说明

定义 VIF 通道 Port 属性。

➤ 定义

```
typedef struct MI_VI_ChnPortAttr_s{  
    MI_SYS_WindowRect_t stCapRect;  
  
    MI_SYS_WindowRect_t stDestSize;  
  
    MI_SYS_FieldType_e   enCapSel;  
  
    MI_SYS_FrameScanMode_e nScanMode ;  
  
    MI_SYS_PixelFormat_e ePixFormat;  
  
    MI\_VI\_FrameRate\_e   eFrameRate;  
  
    MI_U32 u32FrameModeLineCount  
  
} MI_VI_ChnPortAttr_t;
```

➤ 成员

次 Port 仅仅支持设置 stDestSize , enDstFrameRate , 其他属性会被忽略。

成员名称	描述
stCapRect	捕获区域起始坐标(相对于设备图像的大小)与宽高
stDestSize	目标图像大小。必须配置，且大小不应该超出外围 ADC 输出图像的大小范围，否则可能导致VIF 硬件工作异常
eCapSel	帧场选择，只用于隔行模式，建议捕获单场时选择捕获底场。逐行模式时，该项必须设置为 E_MI_SYS_FIELDTYPE_BOTH。
eScanMode	输入扫描模式（逐行、隔行）
ePixFormat	像素存储格式支持 sp420 and sp422
eFrameRate	目标帧率和输入帧率的比值关系。如果不进行帧率控制，则该值设置为0。可以按照1:1, 2:1, 4:1, 8:1, 4:3等比例输出
u32FrameModeLineCount	通知下一级处理的时机，用于 E_MI_VIF_WORK_MODE_RGB_FRAMEMODE的时候

➤ 芯片差异

成员名称	
------	--

stDestSize	VIF 通道有缩放能力，可以水平缩小 2 倍 竖直缩小 2 倍。
------------	----------------------------------

➤ 相关数据类型及接口

[MI\\_VIF\\_SetChnPortAttr](#)

2.20. MI\_VIF\_ChnPortStat\_t

➤ 说明

VIF 通道信息结构体。

➤ 定义

```
typedef struct MI_VIF_ChnStat_s
{
    MI_BOOL bEnable; /* Whether this channel is enabled */
    MI_U32 u32IntCnt; /* The VIFdeo frame interrupt count */
    MI_U32 u32FrmRate; /* current frame rate */
    MI_U32 u32LostInt; /* The interrupt is received but nobody care */
    MI_U32 u32VbFail; /* VIFdeo buffer malloc failure */
    MI_U32 u32PicWidth; /* curren pic width */
    MI_U32 u32PicHeight; /* current pic height */
} MI_VIF_ChnPortStat_t;
```

➤ 成员

成员名称	描述
bEnable	通道是否使能。
u32IntCnt	中断计数。
u32FrmRate	每 10 秒的平均帧率，该值不一定精确。
u32LostInt	中断丢失计数。
u32VbFail	获取 VB 失败计数。
u32PicWidth	图像宽度。
u32PicHeight	图像高度。

※ 注意事项

- 结构体的中断计数，可用于无中断检测。
- 该结构体的帧率是每 10 秒钟的平均帧率，即 VIF 会每隔十秒统计一次平均帧率，该值并不精确。
- 如果查询到该结构体的中断丢失计数一直在增加，说明 VIF 工作出现异常。

➤ 相关数据类型及接口

无。

## 2.21. MI\_VIF\_SNRPad\_e

### ➤ 说明

定义 SensorPad Id。

### ➤ 定义

```
typedef enum
{
    E_MI_VIF_SNRPAD_NULL,
    E_MI_VIF_SNRPADID0,
    E_MI_VIF_SNRPADID1,
    E_MI_VIF_SNRPADID2,
    E_MI_VIF_SNRPADID3,
    E_MI_VIF_SNRPAD_NUM
}MI_VIF_SNRPad_e;
```

### ➤ 成员

成员名称	描述
E_MI_VIF_SNRPAD_NULL	不绑定SensorPad
E_MI_VIF_SNRPADID0	对应硬件设备Sensor0
E_MI_VIF_SNRPADID1	对应硬件设备Sensor1
E_MI_VIF_SNRPADID2	对应硬件设备Sensor2
E_MI_VIF_SNRPADID3	对应硬件设备Sensor3
E_MI_VIF_SNRPAD_NUM	超过最大Sensor Num

### ※ 注意事项

在默认情况下是 VIF Dev0 对应 Sensor0, Dev1 对应 Sensor1。

### ➤ 相关数据类型及接口

[MI\\_VIF\\_Dev2SnrPadMuxCfg\\_t](#)

## 2.22. MI\_VIF\_Dev2SnrPadMuxCfg\_t

### ➤ 说明

定义VIF 设备和SensorPad 绑定关系。

➤ 定义

```
typedef struct MI_VIF_VIFDev2SnrPadMuxConf_s
{
    MI\_VIF\_SNRPad\_e eSensorPadID;    //sensor Pad id
    MI_U32 u32PlaneID;               //For HDR, 1 is short exposure, 0 is long exposure ,
} MI_VIF_Dev2SnrPadMuxCfg_t;
```

➤ 成员

成员名称	描述
eSensorPadID	Sensor Pad Id
u32PlaneID	PlaneId, for HDR 1 长曝, 0 短曝, for liner 为0xff.

※ 注意事项

在默认情况下是 VIF Dev0 对应 Sensor0, Dev1 对应 Sensor1. 默认不调用该接口。

➤ 相关数据类型及接口

[MI\\_VIF\\_SetDev2SnrPadMux](#)



### 3. VIF 错误码

视频输入 API 错误码如表所示。

视频输入 API 错误码

错误代码	宏定义	描述
0xA0032001	MI_ERR_VIF_INVALID_DEVID	视频输入设备号无效
0xA0032002	MI_ERR_VIF_INVALID_CHNID	视频输入通道号无效
0xA0032003	MI_ERR_VIF_INVALID_PARA	视频输入参数设置无效
0xA0032006	MI_ERR_VIF_INVALID_NULL_PTR	输入参数空指针错误
0xA0032007	MI_ERR_VIF_FAILED_NOTCONFIG	视频设备或通道属性未配置
0xA0108008	MI_ERR_VIF_NOT_SUPPORT	操作不支持
0xA0108009	MI_ERR_VIF_NOT_PERM	操作不允许
0xA010800C	MI_ERR_VIF_NOMEM	分配内存失败
0xA010800E	MI_ERR_VIF_BUF_EMPTY	视频输入缓存为空
0xA010800F	MI_ERR_VIF_BUF_FULL	视频输入缓存为满
0xA0108010	MI_ERR_VIF_SYS_NOTREADY	视频输入系统未初始化
0xA0108012	MI_ERR_VIF_BUSY	视频输入系统忙
0xA0032080	MI_ERR_VIF_INVALID_PORTID	视频输入端口无效
0xA0032081	MI_ERR_VIF_FAILED_DEVNOTENABLE	视频输入设备未启用
0xA0032082	MI_ERR_VIF_FAILED_DEVNOTDISABLE	视频输入设备未禁用
0xA0032083	MI_ERR_VIF_FAILED_PORTNOTENABLE	视频输入通道未启用
0xA0032084	MI_ERR_VIF_FAILED_PORTNOTDISABLE	视频输入通道未禁用
0xA0032085	MI_ERR_VIF_CFG_TIMEOUT	视频配置属性超时
0xA0032087	MI_ERR_VIF_INVALID_WAYID	视频通路号无效
0xA0032088	MI_ERR_VIF_INVALID_PHYCHNID	视频物理通道号无效
0xA0032089	MI_ERR_VIF_FAILED_NOTBIND	视频通道未绑定
0xA003208A	MI_ERR_VIF_FAILED_BINDED	视频通道已绑定

## 4. 典型应用场景

典型场景	规格描述
16 路 D1 采集	Dev0、Dev1、Dev2、Dev3 各进 4D1 ( BT656/Auto) · 共 4x4 路 D1 Chn0 ~ 15 分别输出 1D1
16 路 960H 采集	Dev0、Dev1、Dev2、Dev3 各进 4 路 960H ( BT656/144Mhz) · 共 4x4 路 960H Chn0 ~ 15 分别输出 1 路 960H
8 路 720P/30fps 输入	4 个端口均配置为 BT656 · 2 路复合模式 · 共 4x2 路 720P Chn0/2/4/6/8/10/12/14 分别输出 1 路 720P ( 需要修改通道绑定关系 )
4 路 1080P/30fps 输入	Dev0 ~ 3 分别进 1 路 1080P ( 每个端口连 8bit 数据线 · BT656/148.5Mhz ) Chn0/4/8/12 分别输出 1 路 1080P
2 路 1080P/30fps 输入	Dev0、Dev2 各进 1 路 1080P ( 每个端口连 16bit 数据线 · BT656/148.5Mhz ) 。 Chn0/8/分别输出 1 路 1080P
8 路 1080P/15fps 输入双沿输入	Dev0 ~ 3 双沿采样 · 分别进 2 路 1080P · Chn0/2/4/6/8/10/12/14 分别输出 1 路 1080P ( 需要修改通道绑定关系 )