MI VIF API

Version 2.04

REVISION HISTORY

Revision No.	Description	Date
2.03	• Initial release	07/25/2018
2.04	• Updated for accuracy	06/13/2019

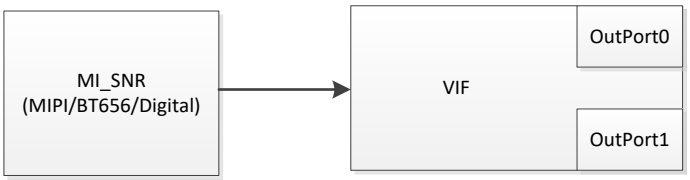
TABLE OF CONTENTS

1. 概述

1.1. 模块说明

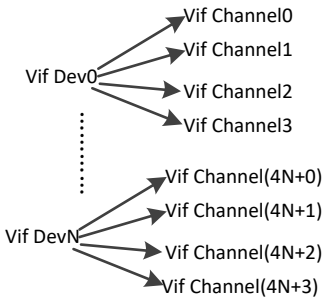
视频输入（VIF）实现启用视频输入设备、视频输入通道、绑定视频输入通道等功能。

1.2. 流程框图



Vif output port1 只有在 BT656 接口中用到， 并且 outport1 的 width/height 是 outport0 的一半.

MI_VIF device 和 channel 之间的映射关系如下：



328Q/329D/326D 和 336D/336Q/339G 芯片中 N max=3,
325/325DE/327DE 和 335/337DE 芯片中 N max=2
注：N 为 Device Id.

Mipi 接口中只能用到对应 VifDev 中的第一个 channel，BT656 接口复合模式下才会用到 VifDev 中的 multi channel.

1.3. 关键字说明

无。

2. API 参考

该功能模块提供以下 API:

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

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。	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

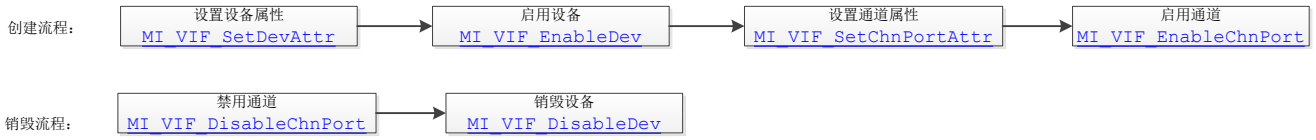
- 头文件：mi_vif_datatype.h、mi_vif.h
- 库文件：libmi_vif.a

※ 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 对接，支持的接口模式包括 BT.656、digital camera、MIPI sensor。具体属性意义参见 3.1 0 数据类型部分的说明。

➢ 举例

VIF 初始化启动及退出范例：




```
STCHECKRESULT(MI_SNR_GetPadInfo(eSnrPadId, &stPad0Info));
STCHECKRESULT(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);

/**********************************************
 init VIF
**********************************************/
MI_VIF_DevAttr_t stDevAttr;
memset(&stDevAttr, 0x0, sizeof(MI_VIF_DevAttr_t));

stDevAttr.eIntfMode = stPad0Info.eIntfMode;
stDevAttr.eWorkMode = pstVifDevAttr->eWorkMode;
stDevAttr.eHDRType = (MI_VIF_HDRType_e)pstVpeChnattr->eHdrType;
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;

STCHECKRESULT(MI_VIF_SetDevAttr(vifDev, &stDevAttr));
STCHECKRESULT(MI_VIF_EnableDev(vifDev));

MI_VIF_ChnPortAttr_t stVifPortInfo;
memset(&stVifPortInfo, 0, sizeof(MI_VIF_ChnPortAttr_t));
stVifPortInfo.stCapRect.u16X = stSnrPlane0Info.stCapRect.u16X;
stVifPortInfo.stCapRect.u16Y = stSnrPlane0Info.stCapRect.u16Y;
stVifPortInfo.stCapRect.u16Width =   stSnrPlane0Info.stCapRect.u16Width;
stVifPortInfo.stCapRect.u16Height = stSnrPlane0Info.stCapRect.u16Height;
stVifPortInfo.stDestSize.u16Width = u32CapWidth;
stVifPortInfo.stDestSize.u16Height = u32CapHeight;
stVifPortInfo.ePixFormat = ePixFormat;
```

```
//stVifPortInfo.u32FrameModeLineCount for lowlantancy mode

if(stDevAttr.eIntfMode == E_MI_VIF_MODE_BT656)
{
      stVifPortInfo.eFrameRate = E_MI_VIF_FRAMERATE_FULL;
      stVifPortInfo.eCapSel = E_MI_SYS_FIELDTYPE_BOTH;
      stVifPortInfo.eScanMode = E_MI_SYS_FRAME_SCAN_MODE_PROGRESSIVE;
}
STCHECKRESULT(MI_VIF_SetChnPortAttr(vifChn, vifPort, &stVifPortInfo));
STCHECKRESULT(MI_VIF_EnableChnPort(vifChn, vifPort));
/************************************************
call sys bind interface
************************************************/

/************************************************
exit call sys unbind interface
************************************************/
STCHECKRESULT(MI_VIF_DisableChnPort(vifChn, vifPort));
STCHECKRESULT(MI_VIF_DisableDev(vifDev));
```

- 相关主题

  MI_VIF_GetDevAttr

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。	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

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。	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

- 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

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。	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

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。	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

请参见 MI_VIF_SetDevAttr 的举例。

- 相关主题

MI_VIF_GetChnPortAttr

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。	输入
u32ChnPort	Port号。	输入
pstAttr	VIF 通道Port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

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

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。
  - 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

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，如果已经绑定后端，则后端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

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	输入
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。

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 Maping 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 Dev2 -> SensorPad1.如果要用到其它的 Vif Dev 或者 Sensor Pad 需要用 MI_VIF_SetDev2SnrPadMux api 设置对应连接关系。

➢ 举例

该 api 实现 Vif Dev0 bind SensorPad0, Dev1 bind SensorPad2, Dev2 bind SensorPad1绑定关系范例如下：

```
 MI_VIF_Dev2SnrPadMuxCfg_t stVifDev[4];
stVifDev[0].eSensorPadID = E_MI_VIF_SNRPAD_ID_0;
stVifDev[0].u32PlaneID = 0xff;//oxff liner mode, 0:长曝光，1:短曝光。
stVifDev[1].eSensorPadID = E_MI_VIF_SNRPAD_ID_2;
stVifDev[1].u32PlaneID = 0xff;
stVifDev[2].eSensorPadID = E_MI_VIF_SNRPAD_ID_1;
stVifDev[2].u32PlaneID = 0xff;
stVifDev[3].eSensorPadID = E_MI_VIF_SNRPAD_ID_3;
stVifDev[3].u32PlaneID = 0xff;
MI_VIF_SetDev2SnrPadMux(stVifDev, 4);
```

➢ 相关主题

无。

3. VIF 数据类型

视频输入相关数据类型定义如下：

MI_VIF_IntfMode_e	定义视频输入设备的接口模式
MI_VIF_WorkMode_e	定义视频设备的复合工作模式
MI_VIF_DataYuvSeq_e	定义视频设备接收的 YUV 数据的数据排列顺序
MI_VIF_ClkEdge_e	定义视频设备接收的时钟类型
MI_VIF_BitOrder_e	定义视频设备的data bit翻转顺序
MI_VIF_HDRType_e	定义视频设备HDR 类型
MI_VIF_Polar_e	定义信号极性
MI_VIF_SyncAttr_t	定义同步信号属性
MI_VIF_DevAttr_t	定义视频输入设备的属性。
MI_VIF_ChnPortAttr_t	定义 VIF 通道属性。
MI_VIF_ChnPortStat_t	定义VIF 通道信息结构体
MI_VIF_SNRPad_e	定义 SensorPad Id
MI_VIF_Dev2SnrPadMuxCfg_t	定义VIF 设备和SensorPad 绑定关系

3.1. 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

可以通过 MI_SNR_GetPadInfo 中的 eIntfMode 获取当前接口模式。

➢ 相关数据类型及接口

MI_VIF_DevAttr_t

MI_VIF_SetDevAttr

3.2. 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 1 路或 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。
- 当该项设为 RGB_REALTIME 时 后端只能绑定 MI_VPE 模块，且 MI_VPE 也要设置为 Realtime mode， 两个模块之间没有 Dram buffer，底层硬件直接相连，两个模块不支持分

时复用。

- 当该项设置为 RGB_FRAME 时,VIF 向 Dram 写出 buffer，送给后端模块。

➢ **相关数据类型及接口**

MI_VIF_DevAttr_t

MI_VIF_SetDevAttr

3.3. 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

3.4. 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

3.5. 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

3.6. 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

3.7. 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

hdr type 和 sensor 相关， 可以通过 MI_SNR_GetPadInfo 的 eHDRMode 获取当前 sensor 支

持的 HDR type。

➢ 相关数据类型及接口

MI_VIF_DevAttr_t

MI_VIF_SetDevAttr

3.8. 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

3.9. MI_VIF_SyncAttr_t

➢ 说明

同步信号属性设置。

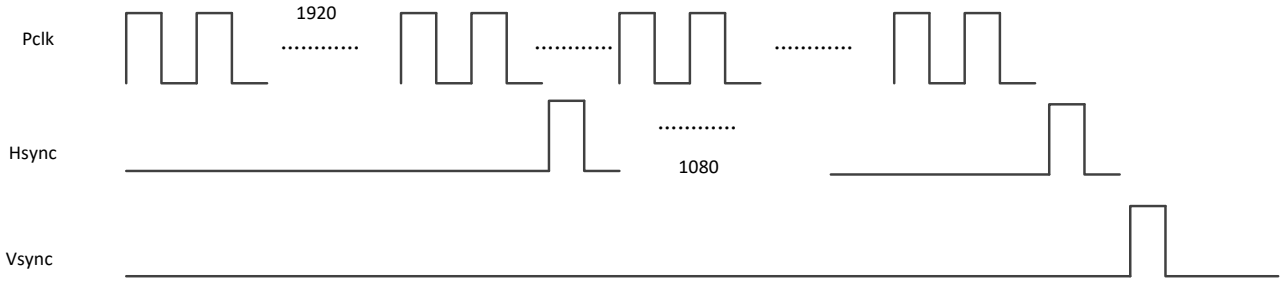
➢ 定义

```
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

以 1920x1080 分辨率， Pclk/Hsync/Vsync 都是高电平有效，波形对应关系如下：



Pclk: pixel clock，每收到一个 pixel，产生一个 clock。
Hsync:行同步，有效时收到的信号属于同一行。
Vsync:场同步，有效时收到的信号属于同一张 frame。

※ 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

3.10. 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_HDRType_e        eHDRType;
    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。
eHDRType	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

3.11. MI_VIF_ChnPortAttr_t

➢ 说明

定义 VIF 通道 Port 属性。

- 定义

```
typedef struct MI_VIF_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_VIF_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，在BT656格式下仅支持E_MI_SYS_PIXEL_FRAME_YUV422_YUYV。
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，仅在BT656接口中可用。
u32FrameModeLineCount	通知下一级处理的时机，用于E_MI_VIF_WORK_MODE_RGB_FRAMEMODE的时候

- 相关数据类型及接口

MI_VIF_SetChnPortAttr

3.12. 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; /* video 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。

3.13. 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, Dev2 对应 Sensor1。

参考 MI_SENSOR_API.doc, 1.2 章节框图。

➢ 相关数据类型及接口

MI_VIF_Dev2SnrPadMuxCfg_t

3.14. 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, Dev2 对应 Sensor1.默认不调用该接口。

➢ 相关数据类型及接口

MI_VIF_SetDev2SnrPadMux

4. 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