MI SENSOR API

Version 2.05

REVISION HISTORY

Revision No.	Description	Date
2.03	• Initial release	11/08/2018
2.04	• Added MI_SNR_CustFunction api • Added MI_SNR_CUST_DIR_e • Added bEarlyInit to MI_SNR_PADInfo_t • Added Shutter/Gain to MI_SNR_PlaneInfo_t	11/08/2019
2.05	• Updated description of MI_SNR_SetFps and MI_SNR_GetFps	12/17/2019

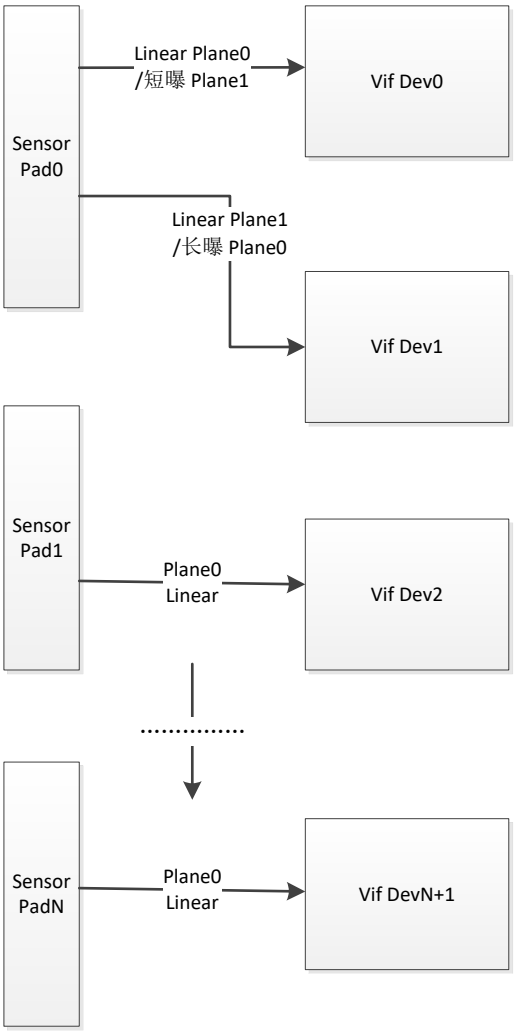
TABLE OF CONTENTS

1. 概述

1.1. 概述

SNR(sensor)模块实现获取摄像头接口信息、调整分辨率和帧率等功能。

1.2. 流程框图



1.3. 关键字说明

- Pad
  Sensor 硬件插口位置。

- Plane
  Pad 下的通道名称。

- Res
  Resolution 分辨率简称。

- Orien
  确定方向， 设置 sensor 水平和竖直方向镜像。

- VC
  Virtual Channel 虚拟通道。

2. API 参考

API名	功能
MI_SNR_Enable	设置Sensor使能
MI_SNR_Disable	设置Sensor失能
MI_SNR_GetPadInfo	获取Sensor设备信息
MI_SNR_GetPlaneInfo	获取Sensor通道信息
MI_SNR_GetFps	获取Sensor当前帧率
MI_SNR_SetFps	设置Sensor帧率
MI_SNR_GetBT656SrcType	获取BT656 Sensor源输入格式
MI_SNR_QueryResCount	获取Sensor支持分辨率的数量
MI_SNR_GetRes	获取索引对应的sensor分辨率
MI_SNR_GetCurRes	获取Sensor当前分辨率
MI_SNR_SetRes	设置Sensor分辨率
MI_SNR_GetOrien	获取Sensor翻转属性
MI_SNR_SetOrien	设置Sensor翻转
MI_SNR_SetPlaneMode	设置Sensor通道模式
MI_SNR_GetPlaneMode	获取设置的Sensor通道模式
MI_SNR_CustFunction	设置Sensor客制化功能

2.1. MI_SNR_Enable

➢ 功能

设置 SENSOR 对应设备的使能

➢ 语法

MI_S32 MI_SNR_Enable(MI_SNR_PAD_ID_e ePADId);

➢ 形参

参数名称	描述	输入/输出
ePADId	SENSOR 设备号。 取值范围：[0, MI_SNR_MAX_PAD_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

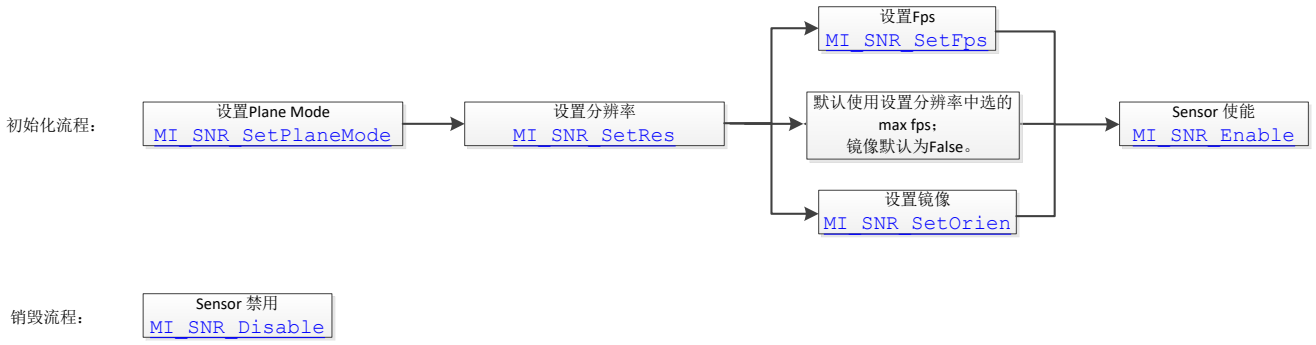
- 头文件：mi_sensor_datatype.h、mi_sensor.h
- 库文件：libmi_sensor.a

※ 注意

- 在调用前要保证 SENSOR设备处于未初始化状态。如果SENSOR设备已处于使能状态，可以使用MI_SNR_Disable来去初始化设备。
- enable 之前必须要设置 MI_SNR_SetPlaneMode 和 MI_SNR_SetRes。
- 由于MI_SNR模块没有与Dram交互， 所以不需要与后端模块进行sys 间的bind，数据流会自动流向MI_VIF。

➢ 举例

Sensor 初始化运行和退出范例如下：




```
MI_U32 u32ResCount =0;
MI_U8 u8ResIndex =0;
MI_U8 u8ChocieRes =0;
MI_SNR_PAD_ID_e eSnrPad= E_MI_SNR_PAD_ID_0;
MI_SNR_QueryResCount(eSnrPad, &u32ResCount);
for(u8ResIndex=0; u8ResIndex < u32ResCount; u8ResIndex++)
{
      MI_SNR_GetRes(E_MI_SNR_PAD_ID_0, u8ResIndex, &stRes);
      printf("index %d, Crop(%d,%d,%d,%d), outputsize(%d,%d), maxfps %d, minfps %d,
      ResDesc %s\n",u8ResIndex, stRes.stCropRect.u16X, stRes.stCropRect.u16Y,
      stRes.stCropRect.u16Width,stRes.stCropRect.u16Height,stRes.stOutputSize.u16Width,
      stRes.stOutputSize.u16Height, stRes.u32MaxFps,stRes.u32MinFps, stRes.strResDesc);
}

printf("select res\n");
scanf("%c", &select);

if(E_MI_VIF_HDR_TYPE_OFF== eHdrType)
{
      MI_SNR_SetPlaneMode(eSnrPad, FALSE);
}
else
{
      MI_SNR_SetPlaneMode(eSnrPad, TRUE);
}

MI_SNR_SetRes(eSnrPad,u8ResIdx);
MI_SNR_Enable(eSnrPad);

/*****************************/
/*   Exit call interface */
/*****************************/
MI_SNR_Disable(eSnrPad);
```

- 相关主题

  MI_SNR_Disable

2.2. MI_SNR_Disable

- 功能

  设置 SENSOR 对应设备失能。

- 语法

  MI_S32 MI_SNR_Disable(MI_SNR_PAD_ID_e ePADId);

- 形参

参数名称	描述	输入/输出
ePADId	SENSOR设备号。 取值范围：[0, MI_SNR_MAX_PAD_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_sensor_datatype.h、mi_sensor.h
  - 库文件：libmi_sensor.a

- 举例

  参考 MI_SNR_Enable 举例。

- 相关主题

  MI_SNR_Enable

2.3. MI_SNR_GetPadInfo

- 功能

  获取 SENSOR 设备信息。

- 语法

  MI_S32 MI_SNR_GetPadInfo(MI_SNR_PAD_ID_e ePADId, MI_SNR_PADInfo_t *pstPadInfo);

- 形参

参数名称	描述	输入/输出
ePADId	SENSOR设备号。 取值范围：[0, MI_SNR_MAX_PAD_NUM)。	输入
pstPadInfo	SENSOR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_sensor_datatype.h、mi_sensor.h
  - 库文件：libmi_sensor.a

- 相关主题

  无。

2.4. MI_SNR_GetPlaneInfo

- 功能

  获取 SENSOR 通道信息。

- 语法

  MI_S32 MI_SNR_GetPlaneInfo(MI_SNR_PAD_ID_e ePADId, MI_U32 u32PlaneID, MI_SNR_PlaneInfo_t *pstPlaneInfo);

- 形参

参数名称	描述	输入/输出
ePADId	SENSOR设备号。 取值范围：[0, MI_SNR_MAX_PAD_NUM)。	输入
u32PlaneID	SENSOR通道号。 取值范围：[0, MI_SNR_MAX_PLANE_NUM)。	输入

pstChnInfo	SENSOR通道信息。	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_sensor_datatype.h、mi_sensor.h
  - 库文件：libmi_sensor.a

- 相关主题

  无。

2.5. MI_SNR_GetFps

- 功能

  获取 SENSOR 帧率

- 语法

  MI_S32 MI_SNR_GetFps(MI_SNR_PAD_ID_e ePADId, MI_U32 *pFps);

- 形参

参数名称	描述	输入/输出
ePADId	SENSOR设备号。 取值范围：[0, MI_SNR_MAX_PAD_NUM)。	输入
pFps	帧率指针	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_sensor_datatype.h、mi_sensor.h
  - 库文件：libmi_sensor.a

※ 注意

获取到的 fps 范围为：
min*1000 < fps < max*1000： 精确到小数点后 3 位。

➢ 相关主题

MI_SNR_SetFps

2.6. MI_SNR_SetFps

➢ 功能

设置 SENSOR 帧率。

➢ 语法

MI_S32 MI_SNR_SetFps(MI_SNR_PAD_ID_e ePADId, MI_U32 *pFps);

➢

➢ 形参

参数名称	描述	输入/输出
ePADId	SENSOR设备号。 取值范围：[0, MI_SNR_MAX_PAD_NUM)。	输入
pFps	帧率指针	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_sensor_datatype.h、mi_sensor.h
- 库文件：libmi_sensor.a

※ 注意

- fps 有两个取值区间：
  min< fps < max： 精确到个位数
  min*1000 < fps < max*1000： 精确到小数点后 3 位。
- Fps 的最大/最小值为 MI_SNR_SetRes 时设置分辨率 Index 所对应的 max/min fps。

➢ 相关主题

MI_SNR_GetFps

2.7. MI_SNR_GetBT656SrcType

➢ 功能

获取 BT656 源输入格式。

➢ 语法

MI_S32 MI_SNR_GetBT656SrcType(MI_SNR_PAD_ID_e ePADId, MI_U32 u32PlaneID, MI_SNR_Anadec_SrcType_e *psttype);

➢ 形参

参数名称	描述	输入/输出
ePADId	SENSOR设备号。 取值范围：[0, MI_SNR_MAX_PAD_NUM)。	输入
u32PlaneID	SENSOR通道号。 取值范围：[0, MI_SNR_MAX_PLANE_NUM)。	输入
psttype	源输入格式	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_sensor_datatype.h、mi_sensor.h
- 库文件：libmi_sensor.a

※ 注意

在 BT656 sensor 上才可以用。

➢ 相关主题

MI_SNR_Anadec_SrcType_e

2.8. MI_SNR_QueryResCount

➢ 功能

获取SENSOR支持分辨率的数量。

➢ 语法

MI_S32 MI_SNR_QueryResCount(MI_SNR_PAD_ID_e ePADId,
MI_U32 *pu32ResCount);

➢ 形参

参数名称	描述	输入/输出
ePADId	SENSOR设备号。 取值范围：[0, MI_SNR_MAX_PAD_NUM)。	输入
*pu32ResCount	SENSOR设备支持的resolution 数量	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_sensor_datatype.h、mi_sensor.h
- 库文件：libmi_sensor.a

※ 注意

无。

➢ 举例

参考 MI_SNR_GetRes 举例

➢ 相关主题

MI_SNR_GetRes

2.9. MI_SNR_GetRes

- 功能

  获取 resolution 映射表中索引对应的分辨率。

- 语法

  MI_S32 MI_SNR_GetRes(MI_SNR_PAD_ID_e ePADId, MI_U8 u8ResIdx, MI_SNR_Res_t *pstRes);

➢ 形参

参数名称	描述	输入/输出
ePADId	SENSOR设备号。 取值范围：[0, MI_SNR_MAX_PAD_NUM)。	输入
u8ResIdx	分辨率映射表中的索引	输入
*pstRes	序号所对应的分辨率	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_sensor_datatype.h、mi_sensor.h
- 库文件：libmi_sensor.a

※ 注意

无

➢ 举例

获取 resolution 列表，以及选择对应分辨率设置范例如下：

```
MI_U32 u32ResCount =0;
MI_U8 u8ResIndex =0;
MI_U8 u8ChocieRes =0;
MI_SNR_QueryResCount(E_MI_SNR_PAD_ID_0, &u32ResCount);
for(u8ResIndex=0; u8ResIndex < u32ResCount; u8ResIndex++)
{
      MI_SNR_GetRes(E_MI_SNR_PAD_ID_0, u8ResIndex, &stRes);
      printf("index %d, Crop(%d,%d,%d,%d), outputsize(%d,%d), maxfps %d, minfps %d,
      ResDesc %s\n",u8ResIndex, stRes.stCropRect.u16X, stRes.stCropRect.u16Y,
      stRes.stCropRect.u16Width,stRes.stCropRect.u16Height,stRes.stOutputSize.u16Width,
stRes.stOutputSize.u16Height,stRes.u32MaxFps,stRes.u32MinFps,stRes.strResDesc);
}

printf("select res\n");
scanf("%c", &select);
MI_SNR_SetRes(E_MI_SNR_PAD_ID_0,u8ResIdx);
MI_SNR_GetCurRes(E_MI_SNR_PAD_ID_0, &u8ResIndex, &stRes);
```

➢ 相关主题

MI_SNR_QueryResCount
MI_SNR_Res_t

2.10. MI_SNR_GetCurRes

➢ 功能

获取 sensor 当前分辨率和在分辨率映射表中的位置。

➢ 语法

MI_S32 MI_SNR_GetCurRes(MI_SNR_PAD_ID_e ePADId, MI_U8 *pu8CurResIdx, MI_SNR_Res_t *pstCurRes);

➢ 形参

参数名称	描述	输入/输出
ePADId	SENSOR设备号。 取值范围：[0, MI_SNR_MAX_PAD_NUM)。	输入
*pu8CurResIdx	当前分辨率的索引	输出
*pstCurRes	当前分辨率信息	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_sensor_datatype.h、mi_sensor.h
- 库文件：libmi_sensor.a

※ 注意

无。

➢ 举例

参考 MI_SNR_GetRes 举例

➢ 相关主题

MI_SNR_Res_t

2.11. MI_SNR_SetRes

➢ 功能

设置 sensor 设备输出分辨率

➢ 语法

MI_S32 MI_SNR_SetRes(MI_SNR_PAD_ID_e ePADId, MI_U8 u8ResIdx);

➢ 形参

参数名称	描述	输入/输出
ePADId	SENSOR设备号。 取值范围：[0, MI_SNR_MAX_PAD_NUM)。	输入
u8ResIdx	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_sensor_datatype.h、mi_sensor.h
- 库文件：libmi_sensor.a

※ 注意

在 MI_SNR_Enable 之后不能单独 call 此 API，要用切换 Sensor resolution 的流程切换 Sensor 分辨率。

➢ 举例

参考 MI_SNR_GetRes 举例

➢ 相关主题

MI_SNR_GetRes
MI_SNR_Res_t

2.12. MI_SNR_SetOrien

➢ 功能

设置 sensor 图象翻转属性

➢ 语法

MI_S32 MI_SNR_SetOrien(MI_SNR_PAD_ID_e ePADId, MI_BOOL bMirror, MI_BOOL bFlip);

➢ 形参

参数名称	描述	输入/输出
ePADId	SENSOR设备号。 取值范围：[0, MI_SNR_MAX_PAD_NUM)。	输入
bMirror	使能竖直镜像翻转	输入
bFlip	使能水平镜像翻转	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_sensor.h
- 库文件：libmi_sensor.a

※ 注意

无。

➢ 举例

该 API 可以单独使用。

➢ 相关主题

MI_SNR_GetOrien

2.13. MI_SNR_GetOrien

➢ 功能

获取 sensor 图象翻转属性

➢ 语法

MI_S32 MI_SNR_GetOrien(MI_SNR_PAD_ID_e ePADId, MI_BOOL *pbMirror, MI_BOOL *pbFlip);

➢ 形参

参数名称	描述	输入/输出
ePADId	SENSOR设备号。 取值范围：[0, MI_SNR_MAX_PAD_NUM)。	输入
*pbMirror	使能竖直镜像翻转	输出
*pbFlip	使能水平镜像翻转	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_sensor.h
- 库文件：libmi_sensor.a

※ 注意

无。

➢ 举例

无。

➢ 相关主题

MI_SNR_SetOrien

2.14. MI_SNR_SetPlaneMode

- 功能

设置 sensor Plane 模式。

- 语法

MI_S32 MI_SNR_SetPlaneMode(MI_SNR_PAD_ID_e ePADId, MI_BOOL bEnable);

- 形参

参数名称	描述	输入/输出
ePADId	SENSOR设备号。 取值范围：[0, MI_SNR_MAX_PAD_NUM)。	输入
bEnable	开HDR 需要置为TRUE，否则为FALSE	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_sensor.h
- 库文件：libmi_sensor.a

※ 注意

plane 和 SensorPad 之间有两种关系， 当 PlaneMode 为 False 时，SensorPad 和 Plane 之间是一对一的关系;当 PlaneMode 为 True 时，SensorPad 一个对应多个 Plane。由于 Hdr Mode 时需要用到两个 Plane 分别接收长曝和短曝， 所以 Plane mode 应该设置为 True.

➢ 举例

```
if(E_MI_VIF_HDR_TYPE_OFF== eHdrType)
{
    MI_SNR_SetPlaneMode(eSnrPad, FALSE);
}
else
{
    MI_SNR_SetPlaneMode(eSnrPad, TRUE);
}
```

➢ 相关主题

MI_SNR_GetPlaneMode

2.15. MI_SNR_GetPlaneMode

➢ 功能

获取上层设置的 Sensor Plane 模式。

➢ 语法

MI_S32 MI_SNR_GetPlaneMode(MI_SNR_PAD_ID_e ePADId, MI_BOOL *pbEnable);

➢ 形参

参数名称	描述	输入/输出

ePADId	SENSOR设备号。 取值范围：[0, MI_SNR_MAX_PAD_NUM)。	输入
*pbEnable	开HDR 需要置为TRUE，否则为FALSE	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_sensor.h
- 库文件：libmi_sensor.a

※ 注意

无。

➢ 举例

无。

➢ 相关主题

MI_SNR_SetPlaneMode

2.16. MI_SNR_CustFunction

➢ 功能

设置 Sensor 客制化功能。例如需要对 sensor 寄存器需要进行读写，或者某些特殊 sensor 有传感器功能通过该 api 获取数据等。

➢ 语法

MI_S32 MI_SNR_CustFunction(MI_SNR_PAD_ID_e ePADId, MI_U32 u32CmdId, MI_U32 u32DataSize, void *pCustData, MI_SNR_CUST_DIR_e eDir);

➢ 形参

参数名称	描述	输入/输出
ePADId	SENSOR设备号。	输入

参数名称	描述	输入/输出
	取值范围：[0, MI_SNR_MAX_PAD_NUM)。	
u32CmdId	客制化功能ID	输入
u32DataSize	客制化功能数据 buffer size	输入
pCustData	客制化功能数据 buffer	输入
eDir	客制化数据类型	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_sensor.h
- 库文件：libmi_sensor.a

※ 注意

和 sensor driver 中的 pCus_sensor_CustDefineFunction api 接口对应

➢ 举例

对 sensor register 读写范例如下：

APP 中实现如下：

```
#define I2C_READ    (0x01)
#define I2C_WRITE   (0x02)

typedef struct stI2CRegData_s
{
    MI_U16 u16Reg;
    MI_U16 u16Data;
}stI2CRegData_t;

stI2CRegData_t stReadReg;
stI2CRegData_t stWriteReg;
MI_U16 u16DataSize=sizeof(stI2CRegData_t);
memset(&stReadReg, 0x0, sizeof(stI2CRegData_t));
memset(&stWriteReg, 0x0, sizeof(stI2CRegData_t));
```

```
stReadReg.u16Reg = 0x3007;
MI_SNR_CustFunction(E_MI_SNR_PAD_ID_0, I2C_READ, u16DataSize, &stReadReg,
E_MI_SNR_CUSTDATA_TO_USER);

stWriteReg.u16Reg = 0x3007;
stWriteReg.u16Data = 0x03;
MI_SNR_CustFunction(E_MI_SNR_PAD_ID_0, I2C_WRITE, u16DataSize, &stWriteReg,
E_MI_SNR_CUSTDATA_TO_DRIVER);
```

sensor driver 中实现如下函数：

```
#define I2C_READ    (0x01)
#define I2C_WRITE    (0x02)

typedef struct stI2CRegData_s
{
    MI_U16 u16Reg;
    MI_U16 u16Data;
}stI2CRegData_t;

static int pCus_sensor_CustDefineFunction(ms_cus_sensor *handle, u32 cmd_id, void *param)
{
    switch(cmd_id)
    {
        case I2C_READ:
            {
                stI2CRegData_t *pRegData = (stI2CRegData_t *)param;
                SensorReg_Read(pRegData->u16Reg, pRegData->u16Data);
            }
            break;
        case I2C_WRITE:
            {
                stI2CRegData_t *pRegData = (stI2CRegData_t *)param;
                SensorReg_Write(pRegData->u16Reg, pRegData->u16Data);
            }
            break;
        default:
            printk("cmdid %d, unknow \n");
            break
    }

    return SUCCESS;
}
```

- 相关主题

  MI_SNR_CUST_DIR_e

3. SENSOR 数据类型

视频输入相关数据类型定义如下：

MI_SNR_MAX_PADNUM	定义支持Sensor最大数量。
MI_SNR_MAX_PLANENUM	定义每一个Sensor Pad 支持通道数量
MI_SNR_PAD_ID_e	定义Sensor Pad枚举类型
MI_SNR_HDRSrc_e	定义Sensor HDR 通道序号
MI_SNR_HDRHWMode_e	定义 HDR 硬件设置模式
MI_SNR_Anadec_SrcType_e	定义Bt656 sensor 输入源格式
MI_SNR_Res_t	定义 Sensor分辨率属性
MI_SNR_AttrParallel_t	定义 Parallel Sensor属性
MI_SNR_MipiAttr_t	定义Mipi Sensor属性
MI_SNR_AttrBt656_t	定义Bt656 Sensor属性
MI_SNR_IntfAttr_u	定义Sensor接口联合体
MI_SNR_PADInfo_t	定义Sensor Pad信息
MI_SNR_PlaneInfo_t	定义Sensor 通道信息
MI_SNR_CUST_DIR_e	定义Sensor 客制化功能数据类型

3.1. MI_SNR_MAX_PADNUM

➢ 说明

定义支持 Sensor 接口最大数量。

➢ 定义

#define MI_SNR_MAX_PADNUM 4

※ 注意事项

无

➢ 相关数据类型及接口

无。

3.2. MI_SNR_MAX_PLANENUM

➢ 说明

定义每一个 Sensor Pad 支持通道数量。

➢ 定义

#define MI_SNR_MAX_PLANENUM 3

※ 注意事项

无。

➢ 相关数据类型及接口

无。

3.3. MI_SNR_PAD_ID_e

➢ 说明

Sensor Pad Id 枚举。

➢ 定义

```
typedef enum
{
    E_MI_SNR_PAD_ID_0 = 0,
    E_MI_SNR_PAD_ID_1 = 1,
    E_MI_SNR_PAD_ID_2 = 2,
    E_MI_SNR_PAD_ID_3 = 3,
    E_MI_SNR_PAD_ID_MAX = 3,
    E_MI_SNR_PAD_ID_NA = 0xFF,
} MI_SNR_PAD_ID_e;
```

※ 注意事项

和硬件上 Sensor Pad 接口相对应。

➢ 相关数据类型及接口

无。

3.4. MI_SNR_HDRSrc_e

➢ 说明

在 HdrMode 时，Plane 通道对应的 Virtual channel。

➢ 定义

```
typedef enum
{
    E_MI_SNR_HDR_SOURCE_VC0,
    E_MI_SNR_HDR_SOURCE_VC1,
    E_MI_SNR_HDR_SOURCE_VC2,
    E_MI_SNR_HDR_SOURCE_VC3,
    E_MI_SNR_HDR_SOURCE_MAX
} MI_SNR_HDRSrc_e;
```

※ 注意事项

无。

- 相关数据类型及接口

MI_SNR_PlaneInfo_t

3.5. MI_SNR_HDRHWMode_e

- 说明

定义 Sensor 硬件设置 HDR 模式。

- 定义

```
typedef enum
{
  E_MI_SNR_HDR_HW_MODE_NONE = 0,
  E_MI_SNR_HDR_HW_MODE_SONY_DOL = 1,
  E_MI_SNR_HDR_HW_MODE_DCG = 2,
  E_MI_SNR_HDR_HW_MODE_EMBEDDED_RAW8 = 3,
  E_MI_SNR_HDR_HW_MODE_EMBEDDED_RAW10 = 4,
  E_MI_SNR_HDR_HW_MODE_EMBEDDED_RAW12 = 5,
  E_MI_SNR_HDR_HW_MODE_EMBEDDED_RAW16 = 6, //Only for OV2718?
} MI_SNR_HDRHWMode_e;
```

- 成员

成员名称	描述
E_MI_SNR_HDR_HW_MODE_NONE	没有开HDR模式
E_MI_SNR_HDR_HW_MODE_SONY_DOL	Digital Overlap High Dynamic Range
E_MI_SNR_HDR_HW_MODE_DCG	双转换增益模式
E_MI_SNR_HDR_HW_MODE_EMBEDDED_RAW8	8bit 压缩模式
E_MI_SNR_HDR_HW_MODE_EMBEDDED_RAW10	10bit 压缩模式
E_MI_SNR_HDR_HW_MODE_EMBEDDED_RAW12	12bit 压缩模式
E_MI_SNR_HDR_HW_MODE_EMBEDDED_RAW16	16bit 压缩模式

※ 注意事项

无。

➢ 相关数据类型及接口

MI_SNR_MipiAttr_t

3.6. MI_SNR_Anadec_SrcType_e

➢ 说明

定义BT656 sensor 输入源格式。

➢ 定义

```
typedef enum
{
    E_MI_SNR_ANADEC_SRC_NO_READY = 0,
    E_MI_SNR_ANADEC_SRC_PAL,
    E_MI_SNR_ANADEC_SRC_NTSC,
    E_MI_SNR_ANADEC_SRC_HD,
    E_MI_SNR_ANADEC_SRC_FHD,
    E_MI_SNR_ANADEC_SRC_DISCNT,
    E_MI_SNR_ANADEC_SRC_NUM
} MI_SNR_Anadec_SrcType_e;
```

※ 注意事项

无。

➢ 相关数据类型及接口

MI_SNR_GetBT656SrcType

3.7. MI_SNR_Res_t

➢ 说明

定义 Sensor 分辨率属性

➢ 定义

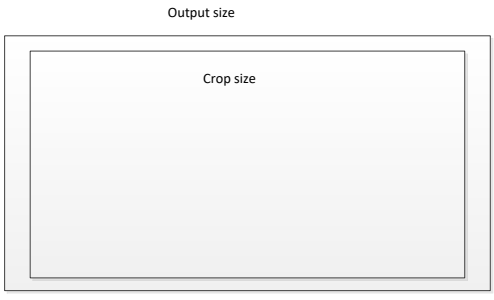
```
typedef struct MI_SNR_Res_s
{
 MI_SYS_WindowRect_t   stCropRect;
 MI_SYS_WindowSize_t   stOutputSize;   /**< Sensor actual output size */

 MI_U32 u32MaxFps;      /**< Max fps in this resolution */
 MI_U32 u32MinFps;      /**< Min fps in this resolution*/
 MI_S8 strResDesc[32];      // Need to put "HDR" here if the resolution is for HDR
} __attribute__((packed, aligned(4))) MI_SNR_Res_t;
```

➢ 成员

成员名称	描述
stCropRect	在output size上裁剪的区域
stOutputSize	Sensor 输出大小范围
u32MaxFps	当前分辨率下最大帧率
u32MinFps	当前分辨率下最小帧率
strResDesc	Resolution string

※ 注意事项

stOutputSize 为 Sensor 的原始宽高，stCropRect 为在原始图像上裁剪的大小，所以 stCropRect 为 Sensor 的实际输出区域。



➢ 相关数据类型及接口

MI_SNR_GetRes
MI_SNR_GetCurRes
MI_SNR_SetRes

3.8. MI_SNR_AttrParallel_t

➢ 说明

定义parallel sensor 属性。

➢ 定义

```
typedef struct MI_SNR_AttrParallel_s
{
    MI_VIF_SyncAttr_t stSyncAttr;
} MI_SNR_AttrParallel_t;
```

➢ 成员

成员名称	描述
stSyncAttr	同步信号属性

※

➢ 注意事项

无。

➢ 相关数据类型及接口

MI_SNR_IntfAttr_u

3.9. MI_SNR_MipiAttr_t

➢ 说明

定义Mipi sensor 属性。

➢ 定义

```
typedef   struct MI_SNR_MipiAttr_s
{
    MI_U32   u32LaneNum; // 几条信号同时传
    MI_U32   u32DataFormat;          //0: YUV 422 format. 1: RGB pattern.
    MI_VIF_DataYuvSeq_e      eDataYUVOrder;
    MI_U32u32HsyncMode;
    MI_U32   u32Sampling_delay;
    /** < MIPI start sampling delay */ /*bit 0~7: clk_skip_ns. bit 8~15: data_skip_ns*/
```

```
    MI_SNR_HDRHWMode_e    eHdrHWmode;
    MI_U32    u32Hdr_Virchn_num;
    MI_U32    u32Long_packet_type[2];
}MI_SNR_MipiAttr_t;
```

➢ 成员

成员名称	描述
u32LaneNum	支持同时传输数据的信号线数量
u32DataFormat	0: YUV 422 format. 1: RGB pattern
eDataYUVOrder	YUV 排列顺序
u32HsyncMode	0:同步前一条 1:同步后一条line的hsync 信号
u32Sampling_delay	延时跳过数据头部分
eHdrHWmode	Sensor支持的HDR mode
u32Hdr_Virchn_num	Sensor支持的HDR 虚拟通道数量
u32Long_packet_type[2]	Sensor支持的数据打包格式

※ 注意事项

无。

➢ 相关数据类型及接口

MI_SNR_IntfAttr_u

3.10. MI_SNR_AttrBt656_t

➢ 说明

定义BT656 sensor 属性。

➢ 定义

```
typedef struct    MI_SNR_AttrBt656_s
{
    MI_U32    u32Multiplex_num;
    MI_VIF_SyncAttr_t stSyncAttr;
    MI_VIF_ClkEdge_e      eClkEdge;
    MI_VIF_BitOrder_e     eBitSwap;
```

} MI_SNR_AttrBt656_t;

➢ 成员

成员名称	描述
u32Multiplex_num	复合模式的路数
stSyncAttr	同步信号属性
eClkEdge	采样时钟模式
eBitSwap	数据排列方向

※ 注意事项

无。

➢ 相关数据类型及接口

MI_SNR_IntfAttr_u

3.11. MI_SNR_IntfAttr_u

➢ 说明

定义sensor 接口类型联合体。

➢ 定义

```
typedef union {
  MI_SNR_AttrParallel_t   stParallelAttr;
  MI_SNR_MipiAttr_t  stMipiAttr;
  MI_SNR_AttrBt656_t      stBt656Attr;
} MI_SNR_IntfAttr_u;
```

➢ 成员

成员名称	描述
stParallelAttr	Parallel sensor 属性
stMipiAttr	Mipi sensor属性
stBt656Attr	Bt656 sensor属性

※ 注意事项

无。

➢ 相关数据类型及接口

MI_SNR_PADInfo_t

3.12. MI_SNR_PADInfo_t

➢ 说明

定义sensor Pad 信息属性。

➢ 定义

```
typedef struct MI_SNR_PADInfo_s
{
    MI_U32              u32PlaneCount;
    //It is different expo number for HDR. It is mux number for BT656. //??
    MI_VIF_IntfMode_e    eIntfMode;
    MI_VIF_HDRType_e     eHDRMode;
    MI_SNR_IntfAttr_u    unIntfAttr;
      MI_BOOL             bEarlyInit;
} MI_SNR_PADInfo_t;
```

➢ 成员

成员名称	描述
u32PlaneCount	BT656 sensor 代表最大复合路数 Mipi sensor 代表长短曝数量
eIntfMode	Sensor 接口枚举
eHDRMode	Hdr 模式
unIntfAttr	Sensor 接口属性联合体
bEarlyInit	Sensor 是否已经提前初始化

※ 注意事项

在 dualos 系统中 bEarlyInit 为 True， Pure Linux 下位 False。

➢ 相关数据类型及接口

MI_SNR_GetPadInfo

3.13. MI_SNR_PlaneInfo_t

- **说明**

定义sensor 通道 信息属性。

➢ 定义

```
typedef struct MI_SNR_PlaneInfo_s
{
    MI_U32                  u32PlaneID;// For HDR long/short exposure or BT656 channel 0~3
    MI_S8                   s8SensorName[32];
    MI_SYS_WindowRect_t     stCapRect;
    MI_SYS_BayerId_e        eBayerId;
    MI_SYS_DataPrecision_e  ePixPrecision;
    MI_SNR_HDRSrc_e         eHdrSrc;
    MI_U32                  u32ShutterUs;
    MI_U32                  u32SensorGainX1024;
    MI_U32                  u32CompGain;
} MI_SNR_PlaneInfo_t;
```

➢ 成员

成员名称	描述
u32PlaneID	通道代号
s8SensorName	Sensor name字符串
stCapRect	在sensor 数据上裁剪的位置
eBayerId	RGB 排列顺序
ePixPrecision	RGB 压缩模式
eHdrSrc	HDR 通道号
u32ShutterUs	Sensor Shutter
u32SensorGainX1024	Sensor Gain
u32CompGain	Sensor Compensate Gain

※ 注意事项

- 当mipi接口 不开Hdr时u32PlaneID = 0xff;开HDR时 u32PlaneID =0代表长曝光，u32PlaneID =1代表短曝光。
- 当BT656接口时，代表当前Plane在复合路数中的通道id。
- #define RGB_BAYER_PIXEL(BitMode, PixelID)
  (E_MI_SYS_PIXEL_FRAME_RGB_BAYER_BASE+
  BitMode*E_MI_SYS_PIXEL_BAYERID_MAX+ PixelID)。
- 通过sys 接口将sensor的eBayerId和ePixPrecision 转换成sys的pixel format，设置给后端

MI_VIF output和MI_VPE input。

- MI_SYS_PixelFormat_e ePixel = RGB_BAYER_PIXEL(ePixPrecision, eBayerId);

➢ 相关数据类型及接口

MI_SNR_GetPadInfo

3.14. MI_SNR_CUST_DIR_e

➢ 说明

定义Sensor 客制化功能数据类型。

➢ 定义

```
typedef enum
{
    E_MI_SNR_CUSTDATA_TO_DRIVER,
    E_MI_SNR_CUSTDATA_TO_USER,
    E_MI_SNR_CUSTDATA_MAX = E_MI_SNR_CUSTDATA_TO_USER,
} MI_SNR_CUST_DIR_e;
```

➢ 成员

成员名称	描述
E_MI_SNR_CUSTDATA_TO_DRIVER	客制化buffer 数据设置给Sensor Driver
E_MI_SNR_CUSTDATA_TO_USER	客制化buffer 数据从sensor 获取
E_MI_SNR_CUSTDATA_MAX	数据类型MAX 选项

※ 注意事项

无。

➢ 相关数据类型及接口

MI_SNR_CustFunction

4. SENSOR 错误码

视频输入 API 错误码如表 所示。

Sensor API 错误码

错误代码	宏定义	描述
0xA0032001	MI_ERR_SNR_INVALID_DEVID	设备号无效
0xA0032002	MI_ERR_SNR_INVALID_CHNID	通道号无效
0xA0032003	MI_ERR_SNR_INVALID_PARA	参数设置无效
0xA0032006	MI_ERR_SNR_INVALID_NULL_PTR	输入参数空指针错误
0xA0032007	MI_ERR_SNR_FAILED_NOTCONFIG	设备或通道属性未配置
0xA0108008	MI_ERR_SNR_NOT_SUPPORT	操作不支持
0xA0108009	MI_ERR_SNR_NOT_PERM	操作不允许
0xA0108010	MI_ERR_SNR_SYS_NOTREADY	系统未初始化
0xA0108012	MI_ERR_SNR_BUSY	系统忙
0xA0032080	MI_ERR_SNR_FAIL	端口无效