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

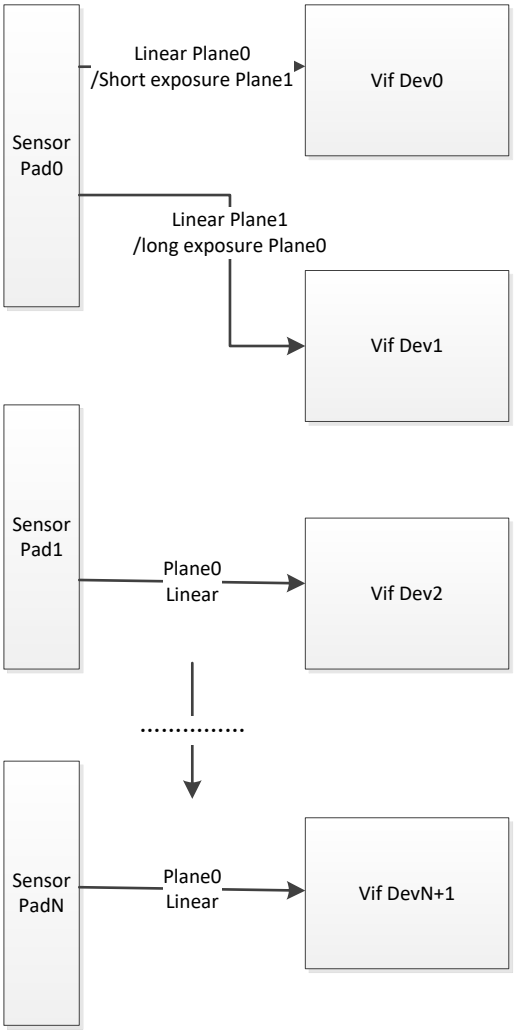
REVISION HISTORY i
TABLE OF CONTENTS ii
1. SCOPE 1
1.1. Module description 1
1.2. Flow chart 1
1.3. Keyword 2
2. API REFERENCE 3
2.1. MI_SNR_Enable 4
2.2. MI_SNR_Disable 5
2.3. MI_SNR_GetPadInfo 6
2.4. MI_SNR_GetPlaneInfo 6
2.5. MI_SNR_GetFps 7
2.6. MI_SNR_SetFps 8
2.7. MI_SNR_GetBT656SrcType 8
2.8. MI_SNR_QueryResCount 9
2.9. MI_SNR_GetRes 10
2.10. MI_SNR_GetCurRes 11
2.11. MI_SNR_SetRes 12
2.12. MI_SNR_SetOrien 12
2.13. MI_SNR_GetOrien 13
2.14. MI_SNR_SetPlaneMode 14
2.15. MI_SNR_GetPlaneMode 15
2.16. MI_SNR_CustFunction 16
3. SENSOR DATA TYPE 19
3.1. MI_SNR_MAX_PADNUM 20
3.2. MI_SNR_MAX_PLANENUM 20
3.3. MI_SNR_PAD_ID_e 20
3.4. MI_SNR_HDRSrc_e 21
3.5. MI_SNR_HDRHWMode_e 21
3.6. MI_SNR_Anadec_SrcType_e 22
3.7. MI_SNR_Res_t 23
3.8. MI_SNR_AttrParallel_t 24
3.9. MI_SNR_MipiAttr_t 24
3.10. MI_SNR_AttrBt656_t 25
3.11. MI_SNR_IntfAttr_u 26
3.12. MI_SNR_PADInfo_t 27
3.13. MI_SNR_PlaneInfo_t 27
3.14. MI_SNR_CUST_DIR_e 29
4. SENSOR ERROR CODES 30

1. SCOPE

1.1. Module Description

The SNR (sensor) module can obtain the camera interface information and adjust the resolution and frame rate.

1.2. Flow Chart



1.3. Keyword

- Pad
  Sensor hardware Jack location.

- Plane
  The name of the channel under the pad.

- Res
  Abbreviation for resolution.

- Orien
  Determine the direction, and set the sensor to mirror horizontally and vertically.

- VC
  Virtual Channel.

2. API REFERENCE

Name of API	Function
MI_SNR_Enable	Sensor Enable
MI_SNR_Disable	Sensor Disable
MI_SNR_GetPadInfo	Get Sensor pad information
MI_SNR_GetPlaneInfo	Get Sensor channel information
MI_SNR_GetFps	Get Sensor current frame rate
MI_SNR_SetFps	Set Sensor frame rate
MI_SNR_GetBT656SrcType	Get BT656 Sensor source input format
MI_SNR_QueryResCount	Get Sensor supported resolution count
MI_SNR_GetRes	Get sensor resolution by corresponding index
MI_SNR_GetCurRes	Get Sensor current resolution
MI_SNR_SetRes	Set Sensor resolution
MI_SNR_GetOrien	Get Sensor orientation attribute
MI_SNR_SetOrien	Set Sensor orientation
MI_SNR_SetPlaneMode	Set Sensor plane mode
MI_SNR_GetPlaneMode	Get Sensor plane mode
MI_SNR_CustFunction	Set sensor customization function

2.1. MI_SNR_Enable

- Function

  Set sensor corresponding pad enable

- Syntax

  MI_S32 MI_SNR_Enable(MI_SNR_PAD_ID_e ePADId);

- Parameter

Parameter Name	Description	Input/Output
ePADId	SENSOR Pad ID Range: [0, MI_SNR_MAX_PAD_NUM].	Input

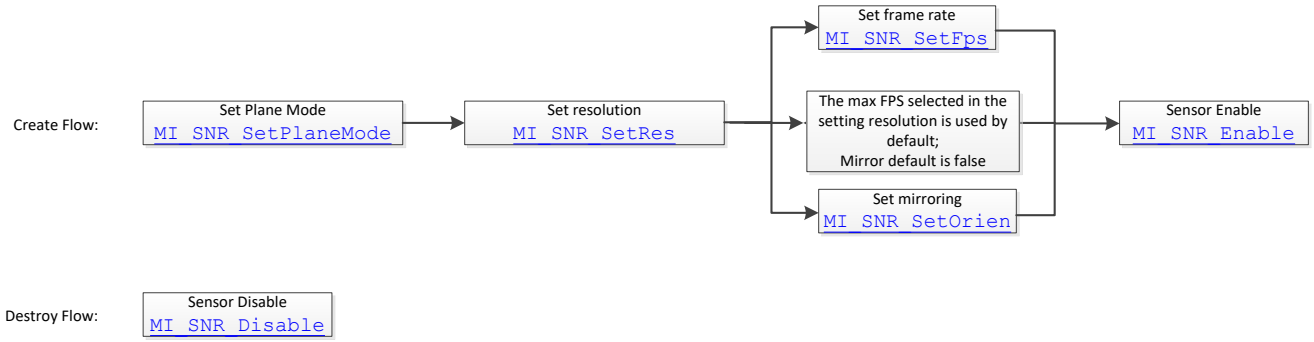
- Return Value
  - MI_OK: Successful
  - Non-zero: Failed, see error code for details

- Requirement
  - Header: mi_sensor_datatype.h, mi_sensor.h
  - Library: libmi_sensor.a

- Note
  - Before calling this function, ensure sensor pad is not initialized. If the sensor pad has been enabled already, use MI_SNR_Disable to deinitialize the pad.
  - Before enabling this API, MI_SNR_SetPlaneMode and MI_SNR_SetRes must be set.
  - Because the MI SNR module does not interact with the DRAM, there is no need to bind it with the backend module; the data flow will automatically go to the MI Vif.

- Example

  Examples of initialization and exit are as follows:



```
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);
```

- Related API

  MI_SNR_Disable

2.2. MI_SNR_Disable

- Function

  Set sensor corresponding pad disable

- Syntax

  MI_S32 MI_SNR_Disable(MI_SNR_PAD_ID_e ePADId);

- Parameter

Parameter Name	Description	Input/Output
ePADId	SENSOR Pad ID Range: [0, MI_SNR_MAX_PAD_NUM).	Input

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

- Requirement
  - Header: mi_sensor_datatype.h, mi_sensor.h
  - Library: libmi_sensor.a

- Example

  Refer to MI_SNR_Enable.

- Related API

  MI_SNR_Enable

2.3. MI_SNR_GetPadInfo

- Function

  Get sensor pad information

- Syntax

  MI_S32 MI_SNR_GetPadInfo(MI_SNR_PAD_ID_e ePADId, MI_SNR_PADInfo_t *pstPadInfo);

- Parameter

Parameter Name	Description	Input/Output
ePADId	SENSOR Pad ID Range: [0, MI_SNR_MAX_PAD_NUM).	Input
pstPadInfo	SENSOR pad attribute pointer	Output

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

- Requirement
  - Header: mi_sensor_datatype.h, mi_sensor.h
  - Library: libmi_sensor.a

- Related API

  N/A.

2.4. MI_SNR_GetPlaneInfo

- Function

  Get sensor plane information

- Syntax

  MI_S32 MI_SNR_GetPadInfo(MI_SNR_PAD_ID_e ePADId, MI_SNR_PlaneInfo_t *pstPadInfo);

- Parameter

Parameter Name	Description	Input/Output
ePADId	SENSOR Pad ID Range: [0, MI_SNR_MAX_PAD_NUM).	Input
u32PlaneID	SENSOR plane ID Range: [0, MI_SNR_MAX_PLANE_NUM).	Input
pstChnInfo	SENSOR plane information	Output

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

- Requirement
  - Header: mi_sensor_datatype.h, mi_sensor.h
  - Library: libmi_sensor.a

- Related API

  N/A.

2.5. MI_SNR_GetFps

- Function

  Get sensor frame rate

- Syntax

  MI_S32 MI_SNR_GetFps(MI_SNR_PAD_ID_e ePADId, MI_U32 *pFps);

- Parameter

Parameter Name	Description	Input/Output
ePADId	SENSOR Pad ID Range: [0, MI_SNR_MAX_PAD_NUM).	Input
pFps	Frame rate pointer	Output

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

- Requirement
  - Header: mi_sensor_datatype.h, mi_sensor.h
  - Library: libmi_sensor.a

➢ Note

The obtained FPS range is:
Min * 1000 < FPS < Max * 1000: accurate to 3 decimal places.

➢ Related API

MI_SNR_SetFps

2.6. MI_SNR_SetFps

➢ Function

Set sensor frame rate

➢ Syntax

MI_S32 MI_SNR_SetFps(MI_SNR_PAD_ID_e_ ePADId, MI_U32 *pFps);

➢ Parameter

Parameter Name	Description	Input/Output
ePADId	SENSOR Pad ID Range: [0, MI_SNR_MAX_PAD_NUM).	Input
pFps	Frame rate pointer	Output

➢ Return Value

- MI_OK: Successful
- Non-zero: Failed, see error code for details

➢ Requirement

- Header: mi_sensor_datatype.h, mi_sensor.h
- Library: libmi_sensor.a

➢ Note

FPS has two value ranges:
— Min < fps < max: Accurate to one digit
— min*1000 < fps < max*1000: Accurate to 3 decimal places
The maximum/minimum value of FPS is the max/min FPS corresponding to the resolution index when MI_SNR_SetRes is set.

➢ Related API

MI_SNR_GetFps

2.7. MI_SNR_GetBT656SrcType

➢ Function

Get BT656 source input format

- Syntax

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

- Parameter

Parameter Name	Description	Input/Output
ePADId	SENSOR Pad ID Range: [0, MI_SNR_MAX_PAD_NUM).	Input
u32PlaneID	SENSOR Plane ID Range: [0, MI_SNR_MAX_PLANE_NUM).	Input
psttype	Source input format	Output

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

- Requirement
  - Header: mi_sensor_datatype.h, mi_sensor.h
  - Library: libmi_sensor.a

- Note

  This function is applicable to BT656 sensor only.

- Related API

  MI_SNR_Anadec_SrcType_e

2.8. MI_SNR_QueryResCount

- Function

  Get sensor supported resolution count

- Syntax

  MI_S32 MI_SNR_QueryResCount(MI_SNR_PAD_ID_e ePADId, MI_U32 *pu32ResCount);

- Parameter

Parameter Name	Description	Input/Output
ePADId	SENSOR Pad ID Range: [0, MI_SNR_MAX_PAD_NUM).	Input
*pu32ResCount	SENSOR pad supported resolution count	Output

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

- Requirement
  - Header: mi_sensor_datatype.h, mi_sensor.h
  - Library: libmi_sensor.a

- Note

  N/A.

- Related API

  MI_SNR_GetRes

2.9. MI_SNR_GetRes

- Function

  Get the corresponding resolution from the index in resolution mapping table

- Syntax

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

- Parameter

Parameter Name	Description	Input/Output
ePADId	SENSOR Pad ID Range: [0, MI_SNR_MAX_PAD_NUM).	Input
u8ResIdx	Index in resolution mapping table	Input
*pstRes	Resolution corresponding to the serial number	Output

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

- Requirement
  - Header: mi_sensor_datatype.h, mi_sensor.h
  - Library: libmi_sensor.a

- Note

  N/A.

- Example

  Below is an example of obtaining resolution list and selecting corresponding resolution setting:

```
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);
```

- Related API

  MI_SNR_QueryResCount
  MI_SNR_Res_t

2.10. MI_SNR_GetCurRes

- Function

  Get sensor current resolution and its position in the resolution mapping table

- Syntax

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

- Parameter

Parameter Name	Description	Input/Output
ePADId	SENSOR Pad ID Range: [0, MI_SNR_MAX_PAD_NUM).	Input
*pu8CurResIdx	Current resolution index	Output
*pstCurRes	Current resolution information	Output

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

- Requirement
  - Header: mi_sensor_datatype.h, mi_sensor.h
  - Library: libmi_sensor.a

- Note

  N/A.

- Example

  Please refer to the example given in MI_SNR_GetRes

- Related API

  MI_SNR_Res_t

2.11. MI_SNR_SetRes

- Function

  Set sensor pad output resolution

- Syntax

  MI_S32 MI_SNR_SetRes(MI_SNR_PAD_ID_e ePADId, MI_U8 u8ResIdx);

- Parameter

Parameter Name	Description	Input/Output
stVifDevMap	Dev and SensorPad mapping relation	Input
u8Length	Dev Num	Input

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

- Requirement
  - Header: mi_sensor_datatype.h, mi_sensor.h
  - Library: libmi_sensor.a

- Note

  In default case, the relation between vif Dev and SensorPad is: vif Dev0 -> SensorPad0, vif Dev1 -> SensorPad1.

- Example

  Please refer to the example given in MI_SNR_GetRes

- Related API

  MI_SNR_GetRes
  MI_SNR_Res_t

2.12. MI_SNR_SetOrien

- Function

  Set sensor image orientation attribute

- Syntax

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

- Parameter

Parameter Name	Description	Input/Output
ePADId	SENSOR Pad ID Range: [0, MI_SNR_MAX_PAD_NUM).	Input
bMirror	Mirror orientation enable	Input
bFlip	Flip orientation enable	Input

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

- Requirement
  - Header: mi_sensor.h
  - Library: libmi_sensor.a

- Note

  N/A.

- Example

  The API can be used alone.

- Related API

  MI_SNR_GetOrien

2.13. MI_SNR_GetOrien

- Function

  Get sensor image orientation attribute

- Syntax

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

- Parameter

Parameter Name	Description	Input/Output
ePADId	SENSOR Pad ID Range: [0, MI_SNR_MAX_PAD_NUM).	Input
*pbMirror	Mirror orientation enable	Output
*pbFlip	Flip orientation enable	Output

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

- Requirement
  - Header: mi_sensor.h
  - Library: libmi_sensor.a

- Note

  N/A.

- Example

  N/A.

- Related API

  MI_SNR_SetOrien

2.14. MI_SNR_SetPlaneMode

- Function

  Set sensor plane mode

- Syntax

  MI_S32 MI_SNR_SetPlaneMode(MI_SNR_PAD_ID_e ePADId, MI_BOOL bEnable);

- Parameter

Parameter Name	Description	Input/Output
ePADId	SENSOR Pad ID Range: [0, MI_SNR_MAX_PAD_NUM).	Input
bEnable	Must be set to TRUE if HDR is in use, and FALSE otherwise	Input

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

- Requirement
  - Header: mi_sensor.h
  - Library: libmi_sensor.a

- Note

  There are two relationships between sensorpad and plane. When planemode is false, the relationship between sensorpad and plane is one-to-one; when planemode is true, one sensorpad corresponds to multiple planes. Since two planes are required to receive long exposure and short exposure in HDR mode, the plane mode should be set to true.

- Example

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

- Related API

MI_SNR_GetPlaneMode

2.15. MI_SNR_GetPlaneMode

- Function

Get upper layer sensor plane mode

- Syntax

MI_S32 MI_SNR_GetPlaneMode(MI_SNR_PAD_ID_e ePADId, MI_BOOL *pbEnable);

- Parameter

Parameter Name	Description	Input/Output
ePADId	SENSOR Pad ID Range: [0, MI_SNR_MAX_PAD_NUM).	Input
*pbEnable	Must be set to TRUE if HDR is in use, and FALSE otherwise	Output

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

- Requirement
  - Header: mi_sensor.h
  - Library: libmi_sensor.a

- Note

N/A.

- Example

N/A.

- Related API

MI_SNR_SetPlaneMode

2.16. MI_SNR_CustFunction

- Function

  Set sensor customization function. For example, enable sensor register read/write access, or allow some special sensors to have sensor functions to obtain data through the API.

- Syntax

  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);

- Parameter

Parameter Name	Description	Input/Output
ePADId	SENSOR Pad ID Range: [0, MI_SNR_MAX_PAD_NUM).	Input
u32CmdId	Customized function ID	Input
u32DataSize	Customized function data buffer size	Input
pCustData	Customized function data buffer	Input
eDir	Customized data type	Input

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

- Requirement
  - Header: mi_sensor.h
  - Library: libmi_sensor.a

- Note

  This API corresponds to the pCus_sensor_CustDefineFunction API interface in the sensor driver.

- Example

  An example of sensor register read/write is given below:

The implementation of app is as follows:

```
#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);
```

The following functions are implemented in the sensor driver:

```
#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;
}
```

- Related API

  MI_SNR_CUST_DIR_e

3. SENSOR DATA TYPE

The sensor related data types are shown in the table below:

MI_SNR_MAX_PADNUM	Define the maximum number of pads supported by the sensor
MI_SNR_MAX_PLANENUM	Define the number of planes supported by each Sensor Pad
MI_SNR_PAD_ID_e	Define Sensor Pad enumeration type
MI_SNR_HDRSrc_e	Define Sensor HDR plane number
MI_SNR_HDRHWMode_e	Define HDR hardware configuration mode
MI_SNR_Anadec_SrcType_e	Define BT656 sensor input source format
MI_SNR_Res_t	Define Sensor resolution attribute
MI_SNR_AttrParallel_t	Define Parallel Sensor attribute
MI_SNR_MipiAttr_t	Define MIPI Sensor attribute
MI_SNR_AttrBt656_t	Define BT656 Sensor attribute
MI_SNR_IntfAttr_u	Define Sensor interface mux
MI_SNR_PADInfo_t	Define Sensor Pad information
MI_SNR_PlaneInfo_t	Define Sensor plane information
MI_SNR_CUST_DIR_e	Define sensor customization function data type

3.1. MI_SNR_MAX_PADNUM

- Description

  Define the maximum number of pads supported by the sensor

- Definition

  #define MI_SNR_MAX_PADNUM 4

- Note

  N/A.

- Related Data Type and Interface

  N/A.

3.2. MI_SNR_MAX_PLANENUM

- Description

  Define the number of planes supported by each Sensor Pad

- Definition

  #define MI_SNR_MAX_PLANENUM 3

- Note

  N/A.

- Related Data Type and Interface

  N/A.

3.3. MI_SNR_PAD_ID_e

- Description

  Define Sensor Pad enumeration type

- Definition

```
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;
```

- Note

  This API corresponds to the sensor pad interface on the hardware.

- Related Data Type and Interface

  N/A.

3.4. MI_SNR_HDRSrc_e

- Description

  Define Sensor HDR plane number enumeration

- Definition

```
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;
```

- Note

  N/A.

- Related Data Type and Interface

  MI_SNR_PlaneInfo_t

3.5. MI_SNR_HDRHWMode_e

- Description

  Define HDR hardware configuration mode

- Definition

```
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;
```

- Member

Member	Description
E_MI_SNR_HDR_HW_MODE_NONE	HDR mode not enabled
E_MI_SNR_HDR_HW_MODE_SONY_DOL	Digital Overlap High Dynamic Range
E_MI_SNR_HDR_HW_MODE_DCG	Double conversion gain
E_MI_SNR_HDR_HW_MODE_EMBEDDED_RAW8	8-bit Compressed Mode
E_MI_SNR_HDR_HW_MODE_EMBEDDED_RAW10	10-bit Compressed Mode
E_MI_SNR_HDR_HW_MODE_EMBEDDED_RAW12	12-bit Compressed Mode
E_MI_SNR_HDR_HW_MODE_EMBEDDED_RAW16	16-bit Compressed Mode

- Note

  N/A.

- Related Data Type and Interface

  MI_SNR_MipiAttr_t

3.6. MI_SNR_Anadec_SrcType_e

- Description

  Define BT656 sensor input source format

- Definition

```
typedef enum
{
        E_MI_SNR_ANADEC_SRC_NO_READY = 0,
        E_MI_SNR_ANADEC_SRC_DISCNT,
        E_MI_SNR_ANADEC_SRC_PAL,
        E_MI_SNR_ANADEC_SRC_NTSC,
        E_MI_SNR_ANADEC_SRC_HD_25P,
        E_MI_SNR_ANADEC_SRC_HD_30P,
        E_MI_SNR_ANADEC_SRC_HD_50P,
        E_MI_SNR_ANADEC_SRC_HD_60P,
        E_MI_SNR_ANADEC_SRC_FHD_25P,
        E_MI_SNR_ANADEC_SRC_FHD_30P,
        E_MI_SNR_ANADEC_SRC_NUM
} MI_SNR_Anadec_SrcType_e;
```

- Member

Member	Description
E_MI_SNR_ANADEC_SRC_NO_READY	Input source not initialized or configured
E_MI_SNR_ANADEC_SRC_DISCNT	Input source disconnected
E_MI_SNR_ANADEC_SRC_PAL	Input source is PAL system
E_MI_SNR_ANADEC_SRC_NTSC	Input source is NTSC system

Member	Description
E_MI_SNR_ANADEC_SRC_HD_25P	Input source is HD 25p
E_MI_SNR_ANADEC_SRC_HD_30P	Input source is HD 30p
E_MI_SNR_ANADEC_SRC_HD_50P	Input source is HD 50p
E_MI_SNR_ANADEC_SRC_HD_60P	Input source is HD 60p
E_MI_SNR_ANADEC_SRC_FHD_25P	Input source is FHD 25p
E_MI_SNR_ANADEC_SRC_FHD_30P	Input source is FHD 30p

- Note

  N/A.

- Related Data Type and Interface

  MI_SNR_GetBT656SrcType

3.7. MI_SNR_Res_t

- Description

  Define Sensor resolution attribute

- Definition

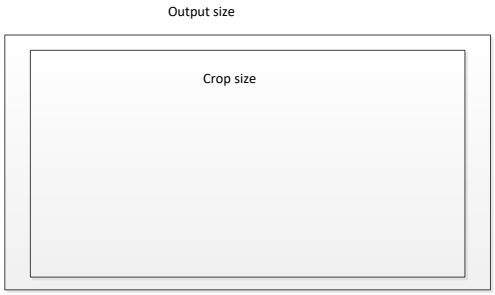
```
typedef struct MI_SNR_Res_s
{
    MI_SYS_WindowRect_t   stCropRect;
    MI_SYS_WindowSize_t   stOutputSize;   /**< Sensor actual output size */

    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;
```

- Member

Member	Description
stCropRect	Crop rectangle on the output size
stOutputSize	Sensor output size
u32MaxFps	Maximum frame rate under current resolution
u32MinFps	Minimum frame rate under current resolution
strResDesc	Resolution string

- Note

  Stoutputsize is the original width and height of sensor and stcroprect is the size cropped on the original image, so stcroprect is the actual output area of sensor.



- Related Data Type and Interface
  MI_SNR_GetRes
  MI_SNR_GetCurRes
  MI_SNR_SetRes

3.8. MI_SNR_AttrParallel_t

- Description
  Define Parallel Sensor attribute

- Definition

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

- Member

Member	Description
stSyncAttr	Parallel signal attribute

- Note
  N/A.

- Related Data Type and Interface
  MI_SNR_IntfAttr_u

3.9. MI_SNR_MipiAttr_t

- Description
  Define MIPI Sensor attribute

- Definition

```
typedef   struct MI_SNR_MipiAttr_s
{
    MI_U32   u32LaneNum; // multiple signals sent simultaneously
    MI_U32   u32DataFormat;       //0: YUV 422 format. 1: RGB pattern.
     MI_VIF_DataYuvSeq_e      eDataYUVOrder;
    MI_U32    u32HsyncMode; //hsync for previous or next line
    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];
// [0]Null [1]blinking [2]embedded [14]yuv422_8b [26]RAW8 [27]RAW10 [28]RAW12 [32]UD1
    [33]UD2 [34]UD3 [35]UD4 [36]UD5 [37]UD6 [38]UD7 [39]UD8
}MI_SNR_MipiAttr_t;
```

- Member

Member	Description
u32LaneNum	Number of lanes with support for simultaneous data transmission
u32DataFormat	0: YUV 422 format, 1: RGB pattern
eDataYUVOrder	YUV order
u32HsyncMode	Previous or next line hsync
u32Sampling_delay	Delay and skip header part
eHdrHWmode	Sensor supported HDR mode
u32Hdr_Virchn_num	Sensor supported HDR virtual channel number
u32Long_packet_type[2]	Sensor supported packet type

- Note

N/A.

- Related Data Type and Interface

MI_SNR_IntfAttr_u

3.10. MI_SNR_AttrBt656_t

- Description

Define BT656 sensor attribute

- Definition

```
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;
```

- Member

Member	Description
u32Multiplex_num	Number of lanes in Multiplex mode
stSyncAttr	Sync signal attribute
eClkEdge	Sampling clock mode
eBitSwap	Data orientation

- Note

N/A.

- Related Data Type and Interface

MI_SNR_IntfAttr_u

3.11. MI_SNR_IntfAttr_u

- Description

Define sensor interface type mux

- Definition

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

- Member

Member	Description
stParallelAttr	Parallel sensor attribute
stMipiAttr	MIPI sensor attribute
stBt656Attr	BT656 sensor attribute

- Note

N/A.

- Related Data Type and Interface
  MI_SNR_PADInfo_t

3.12. MI_SNR_PADInfo_t

- Description
  Define sensor pad information attribute

- Definition

```
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;
```

- Member

Member	Description
u32PlaneCount	Maximum mux plane count for BT656 sensor, and amount of long/short exposure for MIPI sensor
eIntfMode	Sensor interface enumeration
eHDRMode	HDR mode
unIntfAttr	Sensor interface attribute union
bEarlyInit	Whether the sensor has been initialized in advance

- Note
  In dual OS system, baearlyinit is true, and pure Linux is false.

- Related Data Type and Interface
  MI_SNR_GetPadInfo

3.13. MI_SNR_PlaneInfo_t

- Description
  Define sensor plane information attribute

- Definition

```
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;
```

- Member

Member	Description
u32PlaneID	Indicates whether the current plane applies long exposure or short exposure when HDR is turned on, and the current plane ID in mux plane when BT656 is turned on.
s8SensorName	Sensor name string
stCapRect	Crop position of sensor data
eBayerId	RGB order
ePixPrecision	RGB Compressed Mode
eHdrSrc	HDR channel number
u32ShutterUs	Sensor Shutter
u32SensorGainX1024	Sensor Gain
u32CompGain	Sensor Compensate Gain

- Note
  - When the MIPI interface is not enabled, u32planeid = 0xff. When HDR is enabled, u32planeid = 0 represents long exposure, and u32planeid = 1 represents short exposure.
  - When the BT656 interface is used, it represents the channel ID of the current plane in the composite path.
  - #define RGB_BAYER_PIXEL(BitMode, PixelID) (E_MI_SYS_PIXEL_FRAME_RGB_BAYER_BASE+ BitMode*E_MI_SYS_PIXEL_BAYERID_MAX+ PixelID).
  - Through the sys interface, the ebayerid and epixpecision of the sensor are converted into the pixel format of sys, which is set to the backend mi_vif output and mi_vpe input. MI_SYS_PixelFormat_e ePixel = RGB_BAYER_PIXEL(ePixPrecision, eBayerId)

- Related Data Type and Interface

  MI_SNR_GetPadInfo

3.14. MI_SNR_CUST_DIR_e

- Description

  Define sensor customization function data type.

- Definition

```
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;
```

- Member

Member	Description
E_MI_SNR_CUSTDATA_TO_DRIVER	Customized buffer data set to sensor driver
E_MI_SNR_CUSTDATA_TO_USER	Get customized buffer data from sensor
E_MI_SNR_CUSTDATA_MAX	Data type Max option

- Note

  N/A.

- Related Data Type and Interface

  MI_SNR_CustFunction

4. SENSOR ERROR CODES

The Sensor API error codes are listed in the table below:

Table 1: Sensor API Error Codes

Error Code	Macro Definition	Description
0xA0032001	MI_ERR_SNR_INVALID_DEVID	Invalid Device ID
0xA0032002	MI_ERR_SNR_INVALID_CHNID	Invalid channel number
0xA0032003	MI_ERR_SNR_INVALID_PARA	Invalid parameter setting
0xA0032006	MI_ERR_SNR_INVALID_NULL_PTR	Null pointer in input parameter
0xA0032007	MI_ERR_SNR_FAILED_NOTCONFIG	Pad or plane attribute not configured
0xA0108008	MI_ERR_SNR_NOT_SUPPORT	Unsupported operation
0xA0108009	MI_ERR_SNR_NOT_PERM	Operation not permitted
0xA0108010	MI_ERR_SNR_SYS_NOTREADY	System not initialized
0xA0108012	MI_ERR_SNR_BUSY	System busy
0xA0032080	MI_ERR_SNR_FAIL	Interface failed