INGENIC®
T31 ISP API Reference

Date: 2022-04
Viewer: Jason Xu

北京君正集成电路股份有限公司
Ingenic Semiconductor Co., Ltd.

INGENIC®

T31 ISP API Reference

Release history

Date	Revision	Change
2022-04	1.0	First release

Disclaimer

Ingenic Semiconductor Co. Ltd

Add: Junzheng R&D Center, Phase II, Zhongguancun Software Park, Dongbeiwangxi Road, Haidian District, Beijing, China

Tel: 86-10-56345000

Fax: 86-10-56345001

Http: //www.ingenic.com

Content

1 ISP Module Overview

1.1 Module profile

ISP performs the effect processing of digital images through a series of digital image processing algorithms.It mainly includes 3A, bad point correction, denoising, strong light suppression, backlight compensation, color enhancement, lens shadow correction and other processing.

1.2 Key words

- ISP

  ISP: Image Signal Processor

- VIC

  VIC: VIDEO INPUT Controlor

- DVP

  DVP: Digital Video Port

- MIPI CSI

  CS: Camera Serial Interface

2 Driver Introduction

2.1 Driver introduction

T31 ISP Driver is located in: opensource/drivers/isp-t40/tx-isp-t31.

2.2 Driver Compiling and loading

2.2.1 Driver compilation

2.2.1.1 ISP driver compiling

1) Modify this macro definition ISVP_ENV_KERNEL_DIR, it is located in Makefile.
2) Compile: make clean; make.
3) Copy the tx-isp-t31.ko to rootfs.

2.2.1.2 Sensor driver compiling

1) sensor driver relies on kernel and ISP driver. Therefore, it is necessary to compile the kernel and isp driver before compiling sensor driver.

2) Access the sensor driver, modify ISVP_ENV_KERNEL_DIR and ISP_DRIVER_DIR accordingly.

3) Compile: make clean;make .

4) Copy the sensor_xxx.ko to rootfs.

2.2.2 Driver loading

2.2.2.1 ISP driver loading

The ISP driver registration provides the module_param parameter has the isp_clk parameter, its reference configuration:

For example, the 3M@25fps uses 125Mhz,

```
$ insmod tx-isp-t31.ko isp_clk=125000000
```

2.2.2.2 Sensor driver loading

Sensor driver registration provides multiple module_param configurable parameters,If the

product hardware follows the reference design for Insmod without the following parameters, use the default values.

A. Reset, power_down configuration: add parameters when loading modules, such as:

```
$ insmod sensor_xx.ko reset_gpio=18 pwdn_gpio=20
```

Where, the value of gpio is the GPIO number,The rule is:num = 32 * n + bit， for instance:The GPIO number of PA18 is 18，The GPIO number of PC20 is 84.

B. DVP data port configuration: add parameters when loading modules, such as:

```
$ insmod sensor_xx.ko sensor_gpio_func=1
```

Where, sensor_gpio_func is the configuration option of DVP Port.0：PA Low-10bit，1：PA High-10bit，2：PA 12bit.

C. The Sensor has a variety of data interfaces. Currently, DVP and MIPI CSI-2 interfaces are supported.However, some sensor may support both types of data interfaces. The module parameter data_interface is added to distinguish the two types of data interfaces. For example:

```
$ insmod sensor_xx.ko data_interface=1
```

Data_inteface indicates the sensor data interface configuration options 1: MIPI 2: DVP

D. In order to reduce power consumption, the parameter sensor_max_fps with the maximum frame rate is added to the sensor driver. Common sensor drivers support switching between 30/25fps and 15fps. Such as:

```
$ insmod sensor_xx.ko sensor_max_fps=15
```

Sensor_max_fps is the sensor frame rate configuration option. 15: 15 FPS. 25: 25 FPS. The default mode is 30fps or 25 FPS.

3 SDK Usage

3.1 IMP_ISP Unit of Image Signal processing

3.1.1 Function introduction

Image signal processing unit. It mainly includes Sensor registration, addition, image effect setting, mode switching, etc.

ISP module is independent of data flow, not Bind, only Sensor control and effect parameter setting.

3.1.2 Operating Procedure of Module

3.1.2.1 Init procedure

1) Create the ISP module.
2) Add a sensor, and the sensor driver has been added to the kernel before this operation.
3) Enables sensor, and now sensor starts transmitting images.
4) Ability to enable ISP tuning before you can call the ISP debugging interface.
5) Effect debugging.

3.1.2.2 Exit procedure

1) Exit the System-related configuration.
2) Close Sensor, Sensor stops transmitting the image, before which the corresponding channels on FrameSource must be closed.
3) Delete the Sensor, which must be closed before this operation.
4) Clean up the ISP module, and all Sensor s must be removed before this operation.
5) Close the ISP node.

3.2 API Reference

API Name	Function
IMP_ISP_Open	Open the ISP module
IMP_ISP_Close	Turn off the ISP module
IMP_ISP_AddSensor	Add a sensor to provide the data source to the ISP module
IMP_ISP_DelSensor	Delete a sensor
IMP_ISP_EnableSensor	Enables a sensor
IMP_ISP_DisableSensor	Turn off a sensor
IMP_ISP_Tuning_SetISPBypass	Is the ISP module a bypass
IMP_ISP_WDR_ENABLE	Enabling the ISP WDR
IMP_ISP_WDR_ENABLE_GET	Gets the I S P, the W D R mode
IMP_ISP_EnableTuning	Enables the ISP effect debugging function
IMP_ISP_DisableTuning	No ISP effect debugging function is enabled
IMP_ISP_SetSensorRegister	Set the value of a register
IMP_ISP_GetSensorRegister	Gets the value of the sensor for one register
IMP_ISP_Tuning_GetSensorAttr	Gets the fill parameter
IMP_ISP_Tuning_SetSensorFPS	Sets the camera output frame rate
IMP_ISP_Tuning_GetSensorFPS	Get the camera output frame rate
IMP_ISP_Tuning_SetVideoDrop	Set up the video loss function
IMP_ISP_Tuning_SetISPRunningMode	Set up the ISP working mode
IMP_ISP_Tuning_GetISPRunningMode	Get the ISP working mode
IMP_ISP_Tuning_GetAntiFlickerAttr	Set the ISP anti-flash attribute
IMP_ISP_Tuning_SetAntiFlickerAttr	Obtain the ISP anti-flash frequency attribute
IMP_ISP_Tuning_SetHVFLIP	Set the mode of the HV Flip
IMP_ISP_Tuning_GetHVFLIP	Get the pattern of the HV Flip
IMP_ISP_Tuning_SetBrightness	Set the ISP integrated effect picture brightness
IMP_ISP_Tuning_GetBrightness	Get the ISP comprehensive effect picture brightness
IMP_ISP_Tuning_SetISPRunningMode	Set the ISP integrated effect picture brightness
IMP_ISP_Tuning_GetISPRunningMode	Get the ISP comprehensive effect picture brightness
IMP_ISP_Tuning_SetContrast	Set the picture contrast of the ISP integrated effects
IMP_ISP_Tuning_GetContras	Obtain the ISP comprehensive effect picture

	contrast
IMP_ISP_Tuning_SetSharpness	Set the ISP comprehensive effect picture sharpness
IMP_ISP_Tuning_GetSharpness	Get the ISP composite effect picture sharpness
IMP_ISP_Tuning_SetSaturation	Set the ISP integrated effect picture saturation
IMP_ISP_Tuning_GetSaturation	Get the ISP comprehensive effect picture saturation
IMP_ISP_Tuning_SetBcshHue	Set the tone of the image
IMP_ISP_Tuning_GetBcshHue	Gets the tone value of the image
IMP_ISP_Tuning_SetModuleControl	Set up the bypass function of the various ISP modules
IMP_ISP_Tuning_GetModuleControl	Gets the bypass function of the ISP modules
IMP_ISP_Tuning_SetCCMAttr	Set the CCM properties
IMP_ISP_Tuning_GetCCMAttr	Gets the CCM properties
IMP_ISP_Tuning_SetAeWeight	Set the weights for the AE statistics area
IMP_ISP_Tuning_GetAeWeight	Weight was obtained for the AE statistical area
IMP_ISP_Tuning_SetAwbWeight	Set the weight for the AWB statistics area
IMP_ISP_Tuning_GetAwbWeight	Obets weights for the AWB statistical region
IMP_ISP_Tuning_SetAfWeight	Set the weights of the AF statistics area
IMP_ISP_Tuning_GetAfWeight	Weight weight the AF statistical area
IMP_ISP_Tuning_SetAutoZoom	Set the parameters for the autofocus function
IMP_ISP_Tuning_SetMask	Set the fill parameter
IMP_ISP_Tuning_GetMask	Gets the fill parameter

Table 3-1 T31 ISP API

IMP_ISP_Open

【Function】

Open isp module.

【Grammar】

int IMP_ISP_Open(void);

【Formal parameter】

None.

【Return value】

Return value	Describe
0	Success
None 0	Fail

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Create ISP module, ready to add sensor to ISP and turn on ISP effect debugging Function.
- Must be called before the sensor is added.

【Example】

None.

IMP_ISP_Close

【Function】

Close ISP module.

【Grammar】

int IMP_ISP_Close(void);

【Formal parameter】

None.

【Return value】

Return value	Describe
0	Success
None 0	Fail

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before you use this function, you must ensure all FrameSource and effect debug Functions are closed and all sensor s have been uninstalled.

【Example】

None.

IMP_ISP_AddSensor

【Function】

Add a sensor to provide data source to ISP module.

【Grammar】

int IMP_ISP_AddSensor(IMPSensorInfo *pinfo);

【Formal parameter】

Parameter name	Describe	Input/output
pinfo	Add an information pointer to the sensor	Input

【Return value】

Return value	Describe
0	Success
None 0	Fail

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before using this function, you must ensure that the camera driver is registered into the kernel.

【Example】

None.

IMP_ISP_DelSensor

【Function】

Delete a sensor.

【Grammar】

int IMP_ISP_DelSensor(IMPSensorInfo *pinfo);

【Formal parameter】

Parameter name	Describe	Input/output
pinfo	The information pointer of sensor needs to be added	Input

【Return value】

Return value	Describe
0	Success
None 0	Fail

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before using this function, you must ensure that the camera has stopped working, namely that the IMP_ISP_DisableSensor function is called.

【Example】

None.

IMP_ISP_EnableSensor

【Function】

使能一个 sensor.

【Grammar】

int IMP_ISP_EnableSensor(void);

【Formal parameter】

None.

【Return value】

Return value	Describe
0	Success
None 0	Fail

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before using this function, you must ensure that the camera has been added to the ISP module.

【Example】

None.

IMP_ISP_DisableSensor

【Function】

Disable a sensor.

【Grammar】

int IMP_ISP_DisableSensor(void);

【Formal parameter】

None.

【Return value】

Return value	Describe
0	Success
None 0	Fail

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before using this function, you must ensure that all FrameSource have stopped Output images and the effect debugging is not enabling.

【Example】

None.

IMP_ISP_Tuning_SetISPBypass

【Function】

Whether the ISP module is bypass.

【Grammar】

int IMP_ISP_Tuning_SetISPBypass(IMPISPTuningOpsMode enable);

【Formal parameter】

None.

【Return value】

Return value	Describe
0	Success
None 0	Fail

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- You must guarantee that the ISP module is turned on before using this function.

【Example】

None.

IMP_ISP_Tuning_WDR_ENABLE

【Function】

Enabling the ISP WDR.

【Grammar】

int IMP_ISP_WDR_ENABLE(IMPISPTuningOpsMode mode);

【Formal parameter】

Parameter name	Describe	Input/output
mode	ISP WDR pattern	Input

【Return value】

Return value	Describe
0	Success
None 0	Fail

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

IMP_ISP_WDR_ENABLE_GET

【Function】

Get the ISP WDR mode.

【Grammar】

int IMP_ISP_WDR_ENABLE_Get(IMPISPTuningOpsMode* mode);

【Formal parameter】

Parameter name	Describe	Input/output
mode	ISP WDR pattern	Input

【Return value】

Return value	Describe
0	Success
None 0	Fail

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

IMP_ISP_EnableTuning

【Function】

Enables the ISP effect to debug the Function.

【Grammar】

Int IMP_ISP_EnableTuning(void);

【Formal parameter】

None.

【Return value】

Return value	Describe
0	Success
None 0	Fail

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

IMP_ISP_DisableTuning

【Function】

不使能 ISP 效果调试 Function.

【Grammar】

int32_t IMP_ISP_DisableTuning(void);

【Formal parameter】

None.

【Return value】

Return value	Describe
0	Success
None 0	Fail

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before using this function, you must ensure that it is called before not enabling the sensor.

【Example】

None.

IMP_ISP_SetSensorRegister

【Function】

Set the value of the sensor for a single register.

【Grammar】

int IMP_ISP_SetSensorRegister(uint32_t reg, uint32_t value);

【Formal parameter】

Parameter name	Describe	Input/output
reg	Register address	Input
value	Register value	Input

【Return value】

Return value	Describe
0	Success
None 0	Fail

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before using this function, it must be guaranteed that the camera is already enabled.

【Example】

None.

IMP_ISP_GetSensorRegister

【Function】

Gets the value of the sensor for one register.

【Grammar】

int IMP_ISP_GetSensorRegister(uint32_t reg, uint32_t *value);

【Formal parameter】

Parameter name	Describe	Input/output
reg	Register address	Input
value	Register value	Output

【Return value】

Return value	Describe
0	Success
None 0	Fail

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before using this function, it must be guaranteed that the camera is already enabled.

【Example】

None.

IMP_ISP_Tuning_GetSensorAttr

【Function】

Gets the fill parameter.

【Grammar】

int IMP_ISP_Tuning_GetSensorAttr(IMPISPSENSORAttr *attr)；

【Formal parameter】

Parameter name	Describe	Input/output
attr	sensor property parameter	Output

【Return value】

Return value	Describe
0	Success
None 0	Fail

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

IMP_ISP_Tuning_SetSensorFPS

【Function】

Set the camera Output frame rate.

【Grammar】

int IMP_ISP_Tuning_SetSensorFPS(uint32_t fps_num, uint32_t fps_den);

【Formal parameter】

Parameter name	Describe	Input/output
fps_num	Molecular parameters for setting the frame rate	Input
fps_den	Set the denominator parameter for the frame rate	Input

【Return value】

Return value	Describe
0	Success
None 0	Fail

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before you use this function, IMP_ISP_EnableSensor and IMP_ISP_EnableTuning must be called.

【Example】

None.

IMP_ISP_Tuning_GetSensorFPS

【Function】

Get the camera Output frame rate.

【Grammar】

int IMP_ISP_Tuning_GetSensorAttr(IMPISPSENSORAttr *attr);

【Formal parameter】

Parameter name	Describe	Input/output
fps_num		Input
fps_den	Set the denominator parameter for the frame rate	Input

【Return value】

Return value	Describe
0	Success
None 0	Fail

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before you use this function, IMP_ISP_EnableSensor and IMP_ISP_EnableTuning must be called.

【Example】

None.

IMP_ISP_Tuning_SetVideoDrop

【Function】

Set video Loss Function. When the sensor and the motherboard connection line problems, the set callback function is executed.

【Grammar】

int IMP_ISP_Tuning_SetVideoDrop(void (*cb)(void))；

【Formal parameter】

Parameter name	Describe	Input/output
cb	callback function	Input

【Return value】

Return value	Describe
0	Success
None 0	Fail

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

IMP_ISP_Tuning_SetISPRunningMode

【Function】

Set the ISP working mode, normal mode or night vision mode or customization; the default is normal mode.

【Grammar】

int IMP_ISP_Tuning_SetISPRunningMode(IMPISPRunningMode mode);

【Formal parameter】

Parameter name	Describe	Input/output
mode	Run mode parameters	Input

【Return value】

Return value	Describe
0	Success
None 0	Fail

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

```
IMPISPRunningMode mode;
 if( it is during a night now){
    mode = IMPISP_RUNNING_MODE_NIGHT
}else{
    mode = IMPISP_RUNNING_MODE_DAY;
}
ret = IMP_ISP_Tuning_SetISPRunningMode(IMPVI_MAIN, &mode);
if(ret){
    IMP_LOG_ERR(TAG, "IMP_ISP_Tuning_SetISPRunningMode error !\n");
    return -1;
}
```

IMP_ISP_Tuning_GetISPRunningMode

【Function】

Get the ISP working mode.

【Grammar】

int IMP_ISP_Tuning_GetISPRunningMode(IMPISPRunningMode *pmode);

【Formal parameter】

Parameter name	Describe	Input/output
pmode	Operation parameter pointer	Input

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

IMP_ISP_Tuning_SetAntiFlickerAttr

【Function】

Set the ISP anti-flash attribute.

【Grammar】

int IMP_ISP_Tuning_SetAntiFlickerAttr(IMPISPAntiflickerAttr attr);

【Formal parameter】

Parameter name	Describe	Input/output
pattr	Set parameter values	Input

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before you use this function, I must ensure that the ISP effect debug Function is enabled.

【Example】

None.

IMP_ISP_Tuning_GetAntiFlickerAttr

【Function】

Obtain the ISP anti-flash frequency attribute.

【Grammar】

int IMP_ISP_Tuning_GetAntiFlickerAttr(IMPISPAntiflickerAttr *pattr);

【Formal parameter】

Parameter name	Describe	Input/output
pattr	Gets the parameter value pointer	Input

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before you use this function, I must ensure that the ISP effect debug Function is enabled.

【Example】

None.

IMP_ISP_Tuning_SetHVFLIP

【Function】

Set the mode of the HV Flip.

【Grammar】

int IMP_ISP_Tuning_SetHVFLIP(IMPISPHVFLIP hvflip);

【Formal parameter】

Parameter	Describe	Input/output

name		
hvflip	HV Flip pattern	Input

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

IMP_ISP_Tuning_GetHVFLIP

【Function】

Get the pattern of the HV Flip.

【Grammar】

int IMP_ISP_Tuning_GetHVFlip(IMPISPHVFLIP *hvflip);

【Formal parameter】

Parameter name	Describe	Input/output
hvflip	HV Flip pattern	Output

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

IMP_ISP_Tuning_SetBrightness

【Function】

Set the ISP integrated effect picture brightness.

【Grammar】

int IMP_ISP_Tuning_SetBrightness(unsigned char bright);

【Formal parameter】

Parameter name	Describe	Input/output
bright	Image brightness parameters	Input

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.
- The default value is 128, more than 128 increases brightness, less than 128 decreases brightness.

【Example】

None.

IMP_ISP_Tuning_GetBrightness

【Function】

Get the ISP comprehensive effect picture brightness.

【Grammar】

int32_t IMP_ISP_Tuning_GetBrightness(IMPVI_NUM num, unsigned char *bright);

【Formal parameter】

Parameter name	Describe	Input/output
bright	Image brightness parameter pointer	Input

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before you use this function, I must ensure that the ISP effect debug Function is enabled.
- The default value is 128, more than 128 increases brightness, less than 128 decreases brightness.

【Example】

None.

IMP_ISP_Tuning_SetContrast

【Function】

Set the ISP composite effect picture contrast.

【Grammar】

int IMP_ISP_Tuning_SetContrast(unsigned char contrast);

【Formal parameter】

Parameter name	Describe	Input/output
contrast	Picture contrast parameters	Input

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before using this Function, you must ensure that ISP effects debugging Function is enabled.
- The default value is 128, greater than 128 increases contrast, less than 128 decreases contrast.

【Example】

None.

IMP_ISP_Tuning_GetContrast

【Function】

Obtain the ISP comprehensive effect picture contrast.

【Grammar】

int IMP_ISP_Tuning_GetContrast(unsigned char *pcontrast);

【Formal parameter】

Parameter name	Describe	Input/output
contrast	Picture contrast parameter pointer	Input

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before using this Function, you must ensure that ISP effects debugging Function is enabled.
- The default value is 128, greater than 128 increases contrast, less than 128 decreases contrast.

【Example】

None.

IMP_ISP_Tuning_SetSharpness

【Function】

Set the ISP comprehensive effect picture sharpness.

【Grammar】

int IMP_ISP_Tuning_SetSharpness(unsigned char sharpness);

【Formal parameter】

Parameter name	Describe	Input/output

sharpness	Picture sharpness parameter values	Input

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before using this Function, you must ensure that ISP effects debugging Function is enabled.
- The default value is 128, greater than 128 increasing sharpness, and less than 128 decreasing sharpness.

【Example】

None.

IMP_ISP_Tuning_GetSharpness

【Function】

Get the ISP composite effect picture sharpness.

【Grammar】

int IMP_ISP_Tuning_GetSharpness(unsigned char *psharpness);

【Formal parameter】

Parameter name	Describe	Input/output
sharpness	Image sharpness parameter value pointer	Input

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before using this Function, you must ensure that ISP effects debugging Function is enabled.

- The default value is 128, greater than 128 increasing sharpness, and less than 128 decreasing sharpness.

【Example】

None.

IMP_ISP_Tuning_SetSaturation

【Function】

Set the ISP integrated effect picture saturation.

【Grammar】

int IMP_ISP_Tuning_SetSaturation(unsigned char sat);

【Formal parameter】

Parameter name	Describe	Input/output
sat	Picture saturation parameter values	Input

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before using this Function, you must ensure that ISP effects debugging Function is enabled.
- The default value is 128, greater than 128 increasing saturation and less than 128 decreasing saturation.

【Example】

None.

IMP_ISP_Tuning_GetSaturation

【Function】

Get the ISP comprehensive effect picture saturation.

【Grammar】

int IMP_ISP_Tuning_GetSaturation(unsigned char *psat);

【Formal parameter】

Parameter name	Describe	Input/output
sat	Image saturation parameter value pointer	Input

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before using this Function, you must ensure that ISP effects debugging Function is enabled.
- The default value is 128, greater than 128 increasing saturation and less than 128 decreasing saturation.

【Example】

None.

IMP_ISP_Tuning_SetBcshHue

【Function】

Sets the tone of the image.

【Grammar】

int IMP_ISP_Tuning_SetBcshHue(unsigned char hue);

【Formal parameter】

Parameter name	Describe	Input/output
hue	Tue reference value of the image	Input

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.
- The default value is 128, greater than 128 positive adjustment tone, less than 128 reverse adjustment tone, adjustment range 0 to 255.

【Example】

None.

IMP_ISP_Tuning_GetBcshHue

【Function】

Gets the tone value of the image.

【Grammar】

int IMP_ISP_Tuning_GetBcshHue(unsigned char *hue);

【Formal parameter】

Parameter name	Describe	Input/output
hue	Tue reference value pointer for the image	Output

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.
- The default value is 128, greater than 128 positive adjustment tone, less than 128 reverse adjustment tone, adjustment range 0 to 255.

【Example】

None.

IMP_ISP_Tuning_SetModuleControl

【Function】

Set the ISP each module bypassFunction.

【Grammar】

int IMP_ISP_Tuning_SetModuleControl(IMPISPModuleCtl *ispmodule);

【Formal parameter】

Parameter name	Describe	Input/output
ispmodule	ISP module by bypassFunction	Input

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

IMP_ISP_Tuning_GetModuleControl

【Function】

Get the ISP module bypassFunction.

【Grammar】

int IMP_ISP_Tuning_GetModuleControl(IMPISPModuleCtl *ispmodule);

【Formal parameter】

Parameter name	Describe	Input/output
ispmodule	ISP module bypassFunction	Output

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

IMP_ISP_Tuning_SetCCMAttr

【Function】

Set the CCM properties.

【Grammar】

int IMP_ISP_Tuning_SetCCMAttr(IMPISPCCMAttr *ccm);

【Formal parameter】

Parameter name	Describe	Input/output
csc	CSC attribute parameter	Input

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

IMP_ISP_Tuning_GetCCMAttr

【Function】

Gets the CCM properties.

【Grammar】

int IMP_ISP_Tuning_GetCCMAttr(IMPISPCCMAttr *ccm);

【Formal parameter】

Parameter name	Describe	Input/output
csc	CSC attribute parameter	Output

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

IMP_ISP_Tuning_SetAeWeight

【Function】

Set the weights for the AE statistics area.

【Grammar】

int IMP_ISP_Tuning_SetAeWeight(IMPISPWeight *ae_weight);

【Formal parameter】

Parameter name	Describe	Input/output
ae_weight	Weight information	Input

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

IMP_ISP_Tuning_GetAeWeight

【Function】

Gets the weights of AE statistical regions.

【Grammar】

int IMP_ISP_Tuning_GetAeWeight(IMPISPWeight *ae_weight);

【Formal parameter】

Parameter name	Describe	Input/output
ae_weight	Weight information	Output

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

IMP_ISP_Tuning_SetAwbWeight

【Function】

Set the weight for the AWB statistics area.

【Grammar】

int IMP_ISP_Tuning_SetAwbWeight(IMPISPWeight *awb_weight);

【Formal parameter】

Parameter name	Describe	Input/output
awb_weight	Regional weight information	Input

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

IMP_ISP_Tuning_GetAwbWeight

【Function】

Obtain the weights of AWB statistics regions.

【Grammar】

int IMP_ISP_Tuning_GetAwbWeight(IMPISPWeight *awb_weight);

【Formal parameter】

Parameter name	Describe	Input/output
awb_weight	Regional weight information	Output

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

IMP_ISP_Tuning_SetAfWeight

【Function】

Set the weights of the AF statistics area.

【Grammar】

int IMP_ISP_Tuning_SetAfWeight(IMPISPWeight *af_weigh);

【Formal parameter】

Parameter name	Describe	Input/output
af_weight	Regional weight information	Input

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

IMP_ISP_Tuning_GetAfWeight

【Function】

Get the weight of AF statistics area.

【Grammar】

int IMP_ISP_Tuning_GetAfWeight(IMPISPWeight *af_weight);

【Formal parameter】

Parameter name	Describe	Input/output
af_weight	Regional weight information	Output

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

IMP_ISP_Tuning_SetAutoZoom

【Function】

Set the properties of the autofocus Function.

【Grammar】

int IMP_ISP_Tuning_SetAutoZoom(IMPISPAutoZoom *ispautozoom);

【Formal parameter】

Parameter name	Describe	Input/output

ispautozoom	Properties of the autofocus Function	Input

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- Before using this function, you must ensure that IMP_ISP_EnableSensor is executed and returns Success.

【Example】

None.

IMP_ISP_Tuning_SetMask

【Function】

Set the fill parameter.

【Grammar】

int IMP_ISP_Tuning_SetMask(IMPISPMASKAttr *mask);

【Formal parameter】

Parameter name	Describe	Input/output
mask	Filling parameters	Input

【Return value】

Return value	Describe

0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

IMP_ISP_Tuning_GetMask

【Function】

Gets the fill parameter.

【Grammar】

int IMP_ISP_Tuning_GetMask(IMPISPMASKAttr *mask);

【Formal parameter】

Parameter name	Describe	Input/output
mask	Filling parameters	Output

【Return value】

Return value	Describe
0	Success.
None 0	Fail.

【Dependence】

Head file：imp_isp.h

Lib file：libimp.a / libimp.so

【NB】

- The IMP_ISP_EnableTuning has been called before using this function.

【Example】

None.

3.3 Data Type

The IMP_ISP related structure types are as follows:

Name	Definition
IMPSensorControlBusType	Camera control bus type enumeration
IMPI2Cinfo	Camera I2C attribute structure
IMPSPIInfo	Camera SPI attribute structure
IMPSensorInfo	Camera registration information structure
IMPISPTuningOpsType	ISPFunction pattern
IMPISPAntiflickerAttr	ISP anti-flash frequency attribute parameter structure
IMPISPRunningMode	ISP working mode configuration, normal mode or night vision mode
IMPISPAWBCluster	Cluster, mode white balance parameters
IMPISPTuningMode	Pattern Selection Options
IMPISPAEMin	AE Min
IMPISPHVFLIP	HVFlip pattern
IMPISPMASKAttr	Filling parameters
IMPISPSENSORAttr	The Sensor property parameter
IMPISPCCMAttr	ISP color matrix attribute
IMPISPAEAttr	ISP AE manual mode properties
IMPISPAEState	AE convergence correlation parameters
IMPISPScalerLv	Scale effect parameters
IMPISPAeInitAttr	AE properties of the Definition
IMPISPAwbAttr	AWB properties of the customer from the Definition automatic white balance library
IMPISPBlcAtt	Black-level-corrected Function properties
IMPISPWdrOutputMode	WDROutput pattern
IMPISPFrameDrop	Lost frame parameter

IMPSensorControlBusType

【Explain】

Camera control bus type enumeration.

【Definition】

```
typedef enum {
    TX_SENSOR_CONTROL_INTERFACE_I2C = 1,
    TX_SENSOR_CONTROL_INTERFACE_I2C,
} IMPSensorControlBusType;
```

【Member】

Member name	Describe
TX_SENSOR_CONTROL_INTERFACE_I2C	I2C control bus
TX_SENSOR_CONTROL_INTERFACE_I2C	SPI control bus

【NB】

None.

IMPI2CInfo

【Explain】

Camera control parameter configuration when the bus type is I2C.

【Definition】

```
typedef enum {
    IMPISP_TUNING_OPS_TYPE_AUTO,
    IMPISP_TUNING_OPS_TYPE_MANUAL,
    IMPISP_TUNING_OPS_TYPE_BUTT,
} IMPISPTuningOpsType;
typedef struct {
    char type[20];
    int addr;
    int i2c_adapter_id;
} IMPI2CInfo;
```

【Member】

Member name	Describe

type[20]	I2C device name must be consistent with the name variable in struct i2c_device_id in the camera driver
addr	I2C address
i2c_adapter_id	I2C controller

【NB】

- The I2C device name must be consistent with the name variable in struct i2c_device_id in the camera driver.

IMPSPIInfo

【Explain】

Camera control the parameter structures to be configured when the bus type is SPI.

【Definition】

```
typedef struct {
  char modalias[32];
        int bus_num;
} IMPSPIInfo;
```

【Member】

Member name	Describe
modalias[32]	The SPI device name must be consistent with the name variable in struct spi_device_id in the camera driver
bus_num	The SPI bus address

【NB】

- The SPI device name must be consistent with the name variable in struct spi_device_id in the camera driver .

IMPSensorInfo

【Explain】

Camera registration information structure.

【Definition】

```
typedef struct {
    char name[32];
    IMPSensorControlBusType cbus_type;
     union {
```

```
        IMPI2CInfo i2c;
        IMPSPIInfo spi;
    };
     unsigned short rst_gpio;
    unsigned short pwdn_gpio;
    unsigned short power_gpio;
} IMPSensorInfo;
```

【Member】

Member name	Describe
name[32]	Camera name
cbus_type	Camera control bus type
i2c	I2C bus information
spi	The SPI bus information
rst_gpio	The GPIO of the camera reset interface link, NB: This parameter is not enabled right now
pwdn_gpio	GPIO, NB of the camera power down interface link: This parameter is not enabled right now
power_gpio	GPIO, NB of the camera power interface link: This parameter is not enabled right now
sensor_id	Camera ID number
video_interface	Camera interface type
mclk	Camera clock source
default_boot	Camera initialization configuration selection

【NB】

- The value of default_boot is used in the sensor driver and you choose to use that configuration table. Use 0 by default; if sensor supports the dvp interface and the mipi interface, the default default_boot=0 is the dvp interface, and default_boot=1 is the mipi interface.

IMPISPTuningOpsMode

【Explain】

The ISPFunction switch.

【Definition】

```
typedef enum {
    IMPISP_TUNING_OPS_MODE_DISABLE,
```

```
    IMPISP_TUNING_OPS_MODE_ENABLE,
    IMPISP_TUNING_OPS_MODE_BUTT,
} IMPISPTuningOpsMode
```

【Member】

Member name	Describe
IMPISP_TUNING_OPS_MODE_DISABLE	Do not enable the module, the Function
IMPISP_TUNING_OPS_MODE_ENABLE	Enables the module, the Function
IMPISP_TUNING_OPS_MODE_BUTT	To judge the validity of the parameter, the parameter size must be less than this value

【NB】

None.

IMPISPTuningOpsType

【Explain】

ISPFunction selection switch.

【Definition】

```
typedef enum {
    IMPISP_TUNING_OPS_TYPE_AUTO,
    IMPISP_TUNING_OPS_TYPE_MANUAL,
  IMPISP_TUNING_OPS_TYPE_BUTT,
} IMPISPTuningOpsType;
```

【Member】

Member name	Describe
IMPISP_TUNING_OPS_MODE_DISABLE	The module operates in automatic mode
IMPISP_TUNING_OPS_MODE_ENABLE	This module operates in a manual mode
IMPISP_TUNING_OPS_MODE_BUTT	To judge the validity of the parameter, the parameter size must be less than this value

【NB】

None.

IMPISPAntiflickerAttr

【Explain】

ISP anti-flash frequency attribute parameter structure.

【Definition】

```
typedef enum {
    IMPISP_ANTIFLICKER_DISABLE,
    IMPISP_ANTIFLICKER_50HZ,
    IMPISP_ANTIFLICKER_60HZ,
    IMPISP_ANTIFLICKER_BUTT,
} IMPISPAntiflickerAttr;
```

【Member】

Member name	Describe
IMPISP_ANTIFLICKER_DISABLE	Do not enable the ISP anti-flash frequency Function
IMPISP_ANTIFLICKER_50HZ	Enable the ISP flash frequency resistance of Function, and set the frequency to 50HZ
IMPISP_ANTIFLICKER_50HZ	Enable the ISP anti-flash frequency Function, and set the frequency to 60HZ
IMPISP_ANTIFLICKER_BUTT	To judge the validity of the parameter, the parameter size must be less than this value

【NB】

None.

IMPISPRunningMode

【Explain】

ISP working mode configuration, normal mode or night vision mode.

【Definition】

```
typedef enum {
    IMPISP_RUNNING_MODE_DAY = 0,
    IMPISP_RUNNING_MODE_NIGHT = 1,
    IMPISP_RUNNING_MODE_BUTT,
} IMPISPRunningMode;
```

【Member】

Member name	Describe

IMPISP_RUNNING_MODE_DAY	normal mode
IMPISP_RUNNING_MODE_NIGHT	Night vision mode
IMPISP_RUNNING_MODE_BUTT	crest value

【NB】

None.

IMPISPAWBCluster

【Explain】

Cluster, mode white balance parameters.

【Definition】

```
typedef struct {
    IMPISPTuningOpsMode ClusterEn;
    IMPISPTuningOpsMode ToleranceEn;
    unsigned int tolerance_th;
    unsigned int awb_cluster[7];
}IMPISPAWBCluster;
```

【Member】

Member name	Describe
ClusterEn	Cluster, the mode white balance enables
ToleranceEn	The AWB convergence tolerance enables the
tolerance_th	AWB convergence tolerance threshold, with values ranging from 0 to 64
awb_cluster[7]	Cluster, mode white balance parameters

【NB】

None.

IMPISPTuningMode

【Explain】

Pattern Selection Options.

【Definition】

```
typedef enum {
```

```
    IMPISP_TUNING_MODE_AUTO,
    IMPISP_TUNING_MODE_MANUAL,
    IMPISP_TUNING_MODE_RANGE,
    IMPISP_TUNING_MODE_BUTT,
} IMPISPTuningMode;
```

【Member】

Member name	Describe
IMPISP_TUNING_MODE_AUTO	The module operates in an automatic mode
IMPISP_TUNING_MODE_MANUAL	This module operates in a manual mode
IMPISP_TUNING_MODE_RANGE	The module operates to set the range mode
IMPISP_TUNING_MODE_BUTT	To judge the validity of the parameter, the parameter size must be less than this value

【NB】

None.

IMPISPAEMin

【Explain】

AE Min.

【Definition】

```
typedef struct {
    unsigned int min_it;
    unsigned int min_again;
    unsigned int min_it_short;
    unsigned int min_again_short;
} IMPISPAEMin;
```

【Member】

Member name	Describe
min_it	AE minimal exposure
min_again	AE Minimum simulation gain
min_it_short	Minimum exposure of the AE short frames
min_again_short	Minimum simulation gain for AE short frames

【NB】

None.

IMPISPHVFLIP

【Explain】

HVFlip pattern.

【Definition】

```
typedef enum {
    IMPISP_FLIP_NORMAL_MODE = 0,
    IMPISP_FLIP_H_MODE = 1,
    IMPISP_FLIP_V_MODE = 2,
    IMPISP_FLIP_HV_MODE = 3,
    IMPISP_FLIP_MODE_BUTT,
} IMPISPHVFLIP;
```

【Member】

Member name	Describe
IMPISP_FLIP_NORMAL_MODE	normal mode
IMPISP_FLIP_H_MODE	Mirror mode
IMPISP_FLIP_V_MODE	Flip mode
IMPISP_FLIP_HV_MODE	Mirror and flip the mode

【NB】

None.

IMPISPMASKAttr

【Explain】

Filling parameters.

【Definition】

```
typedef struct {
    IMPISP_MASK_BLOCK_PAR chn0[4];
    IMPISP_MASK_BLOCK_PAR chn1[4];
    IMPISP_MASK_BLOCK_PAR chn2[4];
    IMPISP_MASK_TYPE mask_type;
} IMPISPMASKAttr;
```

【Member】

Member name	Describe
IMPISP_MASK_BLOCK_PAR chn0	Channel 0 fill parameter
IMPISP_MASK_BLOCK_PAR chn1	Channel 1 fill parameter
IMPISP_MASK_BLOCK_PAR chn2	Channel 3 fill parameters
IMPISP_MASK_TYPE mask_type	Fill the data type

【NB】

None.

IMPISPSENSORAttr

【Explain】

The Sensor property parameter.

【Definition】

```
typedef struct {
    unsigned int hts;
    unsigned int vts;
    unsigned int fps;
    unsigned int width;
    unsigned int height;
} IMPISPSENSORAttr;
```

【Member】

Member name	Describe
hts	sensor hts
vts	sensor vts
fps	The sensor frame rate
width	sensorOutput width
height	Height of the sensorOutput

【NB】

None.

IMPISPCCMAttr

【Explain】

The ISP color matrix attribute.

【Definition】

```
typedef struct {
    IMPISPTuningOpsMode ManualEn;
    IMPISPTuningOpsMode SatEn;
    float ColorMatrix[9];
} IMPISPCCMAttr;
```

【Member】

Member name	Describe
IMPISPTuningOpsMode ManualEn	Manual CCM enables the
IMPISPTuningOpsMode SatEn	Saturation enables in manual mode
ColorMatrix[9]	Color matrix

【NB】

None.

IMPISPAEAttr

【Explain】

The ISP AE manual mode properties.

【Definition】

```
typedef struct {
    /* AE manual mode properties for long frames in linear mode and WDR mode */
    IMPISPTuningOpsMode AeFreezenEn;
    IMPISPTuningOpsMode AeItManualEn;
    unsigned int AeIt;
    IMPISPTuningOpsMode AeAGainManualEn;
    unsigned int AeAGain;
    IMPISPTuningOpsMode AeDGainManualEn;
    unsigned int AeDGain;
    IMPISPTuningOpsMode AeIspDGainManualEn;
    unsigned int AeIspDGain;
```

```
        /*AE manual mode properties for short frames in WDR mode*/
        IMPISPTuningOpsMode AeWdrShortFreezenEn;
        IMPISPTuningOpsMode AeWdrShortItManualEn;
        unsigned int AeWdrShortIt;
    IMPISPTuningOpsMode AeWdrShortAGainManualEn;
        unsigned int AeWdrShortAGain;
        IMPISPTuningOpsMode AeWdrShortDGainManualEn;
        unsigned int AeWdrShortDGain;
        IMPISPTuningOpsMode AeWdrShortIspDGainManualEn;
        unsigned int AeWdrShortIspDGain;
    } IMPISPAEAttr;
```

【Member】

Member name	Describe
AeFreezenEn	The AE Freezen enables the
AeItManualEn	The AE Exposure manual mode enables the
AeIt	Exposure value in the AE manual mode, in the exposure row
AeAGainManualEn	The AE Sensor analog gain manual mode enables
AeAGain	AE Sensor simulated gain value in multiples x 1024
AeDGainManualEn	The AE Sensor Digital Gain manual mode enables the
AeDGain	AE Sensor digital gain value in multiple x 1024
AeIspDGainManualEn	The AE ISP Digital Gain manual mode enables the
AeIspDGain	AE ISP digital gain value, per unit multiple of x 1024
AeWdrShortFreezenEn	The AE Freezen enables the
AeWdrShortItManualEn	The AE Exposure manual mode enables the
AeWdrShortIt	Exposure value in the AE manual mode, in the exposure row
AeWdrShortAGainManualEn	The AE Sensor analog gain manual mode enables
AeWdrShortAGain	AE Sensor simulated gain value in multiples x 1024

AeWdrShortDGainManualEn	The AESensor Digital Gain manual mode enables the
AeWdrShortDGain	AESensor digital gain value in multiple x 1024
AeWdrShortIspDGainManualEn	The AE ISP Digital Gain manual mode enables the
AeWdrShortIspDGain	AE ISP digital gain value, per unit multiple of x 1024

【NB】

None.

IMPISPAEState

【Explain】

AE convergence correlation parameters.

【Definition】

```
typedef struct {
    bool stable;
    unsigned int target;
    unsigned int ae_mean;
}IMPISPAEState;
```

【Member】

Member name	Describe
bool stable	The AE convergence state, 1: represents stability 0: is converging
target	Current target brightness
ae_mean	After the superimposed weights, the current statistical mean of the AE

【NB】

None.

IMPISPScalerLv

【Explain】

Scale effect parameters.

【Definition】

```
typedef struct {
```

```
    unsigned char channel;
    IMPISPScalerMethod method;
    unsigned char level;
} IMPISPScalerLv;
```

【Member】

Member name	Describe
channel	channel 0~2
method	Scale method
level	Scale the clarity level, range 0~128

【NB】

None.

IMPISPAeInitAttr

【Explain】

AE properties of the Definition.

【Definition】

```
typedef struct {
    uint32_t change;
    enum isp_core_expr_unit AeIntegrationTimeUnit;
    uint32_t    AeIntegrationTime;
    uint32_t    AeAGain;
    uint32_t    AeDGain;
    uint32_t    AeIspDGain;
    uint32_t    AeShortIntegrationTime;
    uint32_t    AeShortAGain;
    uint32_t    AeShortDGain;
    uint32_t    AeShortIspDGain;
    uint32_t    luma;
    uint32_t    luma_scence;
} IMPISPAeAttr;
```

【Member】

Member name	Describe

change	Whether to update the AE parameter
AeIntegrationTimeUnit	AE exposure time unit
AeIntegrationTime	Exposure value of AE
AeAGain	AE Sensor simulated gain value in multiples x 1024
AeDGain	AESensor digital gain value in multiple x 1024
AeIspDGain	AE ISP digital gain value, per unit multiple of x 1024
AeShortIntegrationTime	Exposure values in the AE manual mode
AeShortAGain	AE Sensor simulated gain value in multiples x 1024
AeShortDGain	AE Sensor digital gain value in multiple x 1024
AeShortIspDGain	AE ISP digital gain value, per unit multiple of x 1024
luma	AELuma price
luma_scence	The AE Scene Luma value

【NB】

None.

IMPISPAwbAttr

【Explain】

The AWB properties of the customer from the Definition automatic white balance library.

【Definition】

```
typedef struct {
    uint32_t change;
    uint32_t r_gain;
    uint32_t b_gain;
    uint32_t ct;
} IMPISPAwbAttr;
```

【Member】

Member name	Describe
change	Whether to update the AWB parameter
r_gain	The AWB parameter is the r_gain
b_gain	The AWB parameter, b_gain
ct	Current color temperature

【NB】

None.

IMPISPBlcAttr

【Explain】

Black level correction Function property.

【Definition】

```
typedef struct {
    unsigned int black_level_r;
    unsigned int black_level_gr;
    unsigned int black_level_gb;
    unsigned int black_level_b;
    unsigned int black_level_ir;
} IMPISPBlcAttr;
```

【Member】

Member name	Describe
black_level_r	R channel
black_level_gr	GR channel
black_level_gr	GB channel
black_level_b	B channel
black_level_ir	IR channel

【NB】

None.

IMPISPWdrOutputMode

【Explain】

WDROutput pattern

【Definition】

```
typedef enum {
        IMPISP_WDR_OUTPUT_MODE_FUS_FRAME,
        IMPISP_WDR_OUTPUT_MODE_LONG_FRAME,
        IMPISP_WDR_OUTPUT_MODE_SHORT_FRAME,
IMPISP_WDR_OUTPUT_MODE_BUTT,
} IMPISPWdrOutputMode;
```

【Member】

Member name	Describe
IMPISP_WDR_OUTPUT_MODE_FUS_FRAME	mixed model
IMPISP_WDR_OUTPUT_MODE_LONG_FRAME	Long frame mode
IMPISP_WDR_OUTPUT_MODE_SHORT_FRAME	Short frame mode
IMPISP_WDR_OUTPUT_MODE_BUTT	To judge the validity of the parameter, the parameter size must be less than this value

【NB】

None.

IMPISPFrameDrop

【Explain】

Lost frame parameter.

【Definition】

```
typedef struct {
        IMPISPTuningOpsMode enable;
        uint8_t lsize;
        uint32_t fmark;
} IMPISPFrameDrop;
```

【Member】

Member name	Describe
IMPISPTuningOpsMode enable	The can mark
lsize	Total quantity (range: 0~31)
fmark	Bit flag (1Output, 0 missing)

【NB】

None.